CA Dataquery™ for CA Datacom®

Reference Guide

Version 14.02

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2015 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA products:

- CA Datacom®/DB
- CA Datacom®/AD
- CA Datacom® Datadictionary™ (Datadictionary)
- CA Datacom® SQL
- CA Dataquery™ for CA Datacom® (CA Dataquery)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction 15

Terminology 15
Reading Syntax Diagrams 16
- Statement Without Parameters 16
- Statement with Required Parameters 17
- Delimiters Around Parameters 17
- Choice of Required Parameters 17
- Default Value for a Required Parameter 18
- Optional Parameter 18
- Choice of Optional Parameters 18
- Repeatable Variable Parameter 19
- Separator with Repeatable Variable and Delimiter 19
- Optional Repeatable Parameters 20
- Default Value for a Parameter 20
- Variables Representing Several Parameters 20

Listing Libraries for CA Datacom Products 21

Chapter 2: How to Use Panels 23

Chapter 3: Panels 27

(DQAA0) Directory of Queries and Terms 27
(DQAB0) Directory of Queries and Terms 30
(DQAD0) LIST COMMAND Prompt (SQL Mode) 32
(DQAF0) Directory of Tables 34
(DQAK0) Directory Selection (SQL) 36
(DQAZ0) Bulletin Board 38
(DQA10) LIST Command 38
(DQA30) Directory of Queries and Terms 40
(DQA40) Directory Selection 42
(DQA50) Directory of Queries (Associate User) 44
(DQA70) Directory of Saved Sets 46
(DQA80) Save Active Found Set 48
(DQA90) Extract Active Found Set 50
(DQBA0) Column Display 51
(DQBB0) Column Display 53
(DQBC0) Extended Column Display 55

(DQBD0) Extended Key Display 59
(DQB00) DISPLAY Command 62
(DQB10) Keys and Columns Display 64
(DQB20) Column Display 66
(DQB40) Text Display (Column) 68
(DQB50) Key Display 69
(DQB70) Text Display (Key) 71
(DQB80) Text Display (Table) 73
(DQB90) Column Display (from Editor) 74
(DQDA0) Directory of Tables (SQL Mode) 76
(DQDL0) Define Variable List 78
(DQDP0) Define Descriptive Text 81
(DQDR0) Define Variable Range 83
(DQDV0) Define Variable 85
(DQD00) EDIT Command 88
(DQD10) Editor 90
(DQD30) Query Template 94
(DQD40) Query Template 96
(DQD50) Query Template (SQL Mode) 98
(DQD60) Editor (For Wide Screens) 100
(DQEA0) Query/Dialog Print Output (Wide Screens) 104
(DQEC0) FIND Statistics 106
(DQED0) Column Value Display 109
(DQEF0) Query Statistics 111
(DQED0) Column Value Display (Wide Screens) 114
(DQEG0) Table Maintenance 116
(DQEI0) Table Maintenance 118
(DQEL0) Dialog or PROC Value List 120
(DQEN0) Batch Execution 123
(DQEO0) Totaling Options 126
(DQER0) Dialog or PROC Range 128
(DQEV0) Dialog Validation 130
(DQEX0) Prompt Panel (Dialog or PROC Execution) 132
(DQE10) Online Execution 135
(DQE20) Online Execution (Associate User) 137
(DQE50) Data Selection Status 140
(DQE60) Online Execution SQL Query 141
(DQE70) Online Execution SQL Query (Associate User) 143
(DQE90) Query/Dialog Print Output 145
(DQFA0) Report Format - Function Control Break 147
(DQFC0) Report Format - Query Display 149
(DQF30) Report Format - Report Title 150

(DQF40) Report Format - Control Breaks 152
(DQF50) Report Format - Column Functions 154
(DQF60) Report Format - Column Headings 156
(DQF70) Report Format - Column Format 158
(DQF80) Report Format - Function Legends 160
(DQF90) Report Format - Function Format 163
(DQHA0) Select Columns for Calculation 165
(DQHB0) Define the Arithmetic Expression 168
(DQHD0) SORT Criteria Definition 171
(DQHE0) Row Ordering 173
(DQHF0) Report Definition 175
(DQHG0) Report Column Definition 177
(DQHI0) Column Definition 179
(DQHJ0) Function Control Break Selection 182
(DQHK0) Report Column Heading Definition 184
(DQHL0) Report Column Format Definition 187
(DQHM0) Guided Query Display 189
(DQHN0) Guided Query Display 191
(DQHO0) Column Function Legends 193
(DQHP0) Function Result Print Format 196
(DQHQ0) Multiple Occurrence Selection 198
(DQHR0) Extended Column Display 201
(DQHS0) Guided Extended Key 204
(DQHT0) Primary Table Selection 207
(DQHU0) Secondary Table Selection 210
(DQHV0) Guided Query Creation: Option Selection 213
(DQH10) Query Identification 214
(DQH40) Primary Table: Key Selection 216
(DQH50) Secondary Table: Key Selection 218
(DQH60) Selection Criteria: Designate Columns 220
(DQH70) Selection Criteria: Define Condition 223
(DQH80) Selection Criteria: Multiple Conditions 226
(DQH90) Define Temporary Results 229
(DQIA0) Define Expression: Select Columns 231
(DQIB0) Define Expression: Write Statement 233
(DQIE0) Ordering Rows: PRIORITY 236
(DQIG0) Output Row: Column Selection 238
(DQII0) Output Row: Definition 240
(DQIR0) Guided Extended Column 243
(DQIT0) Join Condition: Define Relation 245
(DQIU0) Grouping Rows: Priority 247
(DQIW0) Join Conditions: Multiple Conditions 249

(DQI40) Join Condition: Primary Table 251
(DQI50) Join Condition: Secondary Table 253
(DQI60) Search Conditions: Designate Columns 255
(DQI70) Search Conditions: Define Condition 257
(DQI80) Search Conditions: Multiple Conditions 260
(DQKL0) User Profile 263
(DQKM0) Hardcopy Print Options 266
(DQK20) Group Display 269
(DQMI0) Estimated Statistics 270
(DQM10) List of Help Topics 272
(DQM20) Primary Table Selection 273
(DQM30) Secondary Table Selection 275
(DQTA0) Drop Table Confirmation 277
(DQTB0) STORE Command Prompt 278
(DQT10) Directory of Tables 279
(DQT20) Display Table 281
(DQT30) Create Table 284
(DQT40) Maintain Rows 286
(DQT50) Insert Rows 289
(DQT60) Data Display 292
(DQZ10) Signon 295
(DQZ20) SignOff 296
(DQZ50) Main Menu (SQL Mode) 297
(DQZ60) Main Menu (DQL Mode) 299

Chapter 4: Editor Commands 301

Summary of All Editor Commands 302
/ and / - Begin and End Comments 305
CH (Change) 306
C (Copy) 309
D (Delete) 315
GC (Mass Change) 318
I (Insert) 320
NE (Locate Text) 323
PR (Locate Previous) 325
M (Move) 327
R (Repeat) 333
Scroll 336
Shift 340
SP (Split) 343
X (Cancel) 344

Chapter 5: Operation Commands 347

+nnnn - Scroll Forward to a Page 350
-nnnn - Scroll Backward to a Page 352
ADMIN - Display Administrative Menu 354
AUTHID - Change SQL Authorization ID 355
BOTTOM - Display Last Page 357
CREATE - Display Editor Creation Panel 358
DETAIL - Display Detail Report 360
DIRECTORY - Display Directory Selection Panel 361
DISPLAY - Display Keys and Columns 363
DRAW - Create a Simple Query 364
DQL - Change to DQL Mode 366
EDIT - Display a Member 367
EXECUTE - Begin Online Execution 369
EXTRACT - Extract Found Data 371
FORMAT - Change SQL Format 373
GUIDE - Access Guided Query 374
HELP - View Help Topics 376
KEEP - Save Found Set Location 377
LIST - Display a Directory 379
MENU - Display the Main Menu 382
MSG - Send a Message 384
NO-DETAIL - Suppress Detail Lines 385
NO-TOTALS - Suppress Report Totals 386
NOWRAP - Full Width Report 388
OFF - Sign Off 389
PDB - Access Personal Database Facility 391
PFn - Execute a PF Key Function 392
PROFILE - Display Your User Profile 394
SQL - Change to SQL Mode 395
STALL - Produce Additional Statistics 397
STATS - Display Execution Statistics 399
STORE - Create a Personal Table from Output 400
SUBMIT - Begin Batch Execution 402
TIME - Display Time and Date 404
TOP - Display First Screen 405
TOTALS - Display DQL Mode PRINT Totals Only 406
WHEN-ONLY - Show Only Calculation Results 408
WRAP - Wrap Report Lines 409

Chapter 6: DQL Statements 411

Performance Considerations 412
COUNT Statement 413
DELETE Statement 415
DISPLAY Statement 415
- Basic Syntax 417
- Alternate Output Headings 419

ERASE Statement 421
FIND Statement 421
INSERT Statement 423
PICTURE Clause 424
- Replacement Characters 426
- Insertion Characters 427
- Signs and Symbols 427
- Examples 428
- Sample DISPLAY Query 428
- Sample PRINT Query 430

PRINT Statement 431
- Basic Syntax Diagram 433
- Headings Syntax Diagram 436
- Totals Syntax Diagram 438

Relationship Clauses 444
- Simple Relationship with a Common Key 446
- Complex Relationships - No Common Key 449
- Outer Joins 455
- Types of Outer Joins and Disjunctions 456
- Adding Selection Criteria 462
- Equijoin Examples 463
- Combining Relationship Types 465

SET Statements 471
- Numeric Column Syntax Diagram 472
- Arithmetic Result Syntax Diagram 473
- Numeric Literal Syntax Diagram 479
- Mathematical Functions in SET 480
- Using Multiple SET Statements 482

SORT Statements 483
- Sorting Groups Within Groups 488
- Designating Control Breaks 489

UPDATE Statement 491
WHEN/DO Statements 492
- Specifying Functions at Control Breaks 493

End of Report 497
Page Break 501
Specifying Multiple DO Statements 502
WITH Clauses 504
WITH Clause Operand1 506
WITH Clause Comparison Operators 507
WITH Clause Operand2 509
Multiple WITH Clauses 511
Evaluation of Logical Expressions 512

Chapter 7: SQL Statements 515

SQL Overview 515
SQL Summary 515
Other Acceptable SQL Keywords 516
ALTER TABLE 517
Description 519
Processing 524
Assignment Statement 527
CALL/EXECUTE PROCEDURE 527
CASE Statement 529
COMMENT ON 529
Description 530
CREATE INDEX 532
Description 533
Processing 534
Example 534
CREATE PROCEDURE 535
External Procedures 536
SQL Procedures 537
CREATE PROCEDURE Syntax and Description 538
CREATE RULE 592
CREATE SCHEMA 593
CREATE SYNONYM 593
Description 594
Example 594
CREATE TABLE 594
Description 596
Privileges 597
Column Definition 597
Column Constraint Definition 599
Table Constraint Definition 602

Referential Constraint Definition 604
Data Types 609
CREATE TRIGGER/RULE 616
CREATE VIEW 620
Privileges 620
Description 621
Processing 622
DELETE 624
Searched DELETE 625
Description 626
Example 1 626
Example 2 627
DROP 627
Description 630
EXECUTE PROCEDURE 631
GRANT 631
Plan Security 632
Description of Plan Security Diagram 632
Description of Non-Plan Security Diagram 634
IF-THEN Statement 636
INSERT 636
Description 638
Rules for Inserting 639
Processing 639
Example 1 640
Example 2 640
Example 3 640
ITERATE Statement 640
LEAVE Statement 640
REPEAT-UNTIL Statement 641
REVOKE 641
Plan Security 642
Description of Plan Security Diagram 642
Description of Non-Plan Security Diagram 643
SELECT 646
Subselect 647
Full-Select Statement 654
Description 655
Duplicate Rows 655
Rules for Columns 655
Example 1 656
Example 2 656

Select-Statement 656
ORDER BY Clause 658
UPDATE 660
Searched UPDATE 661
WHILE Statement 663

Appendix A: Limitations 665

Personal Tables 665
SQL Mode Limitations 665
DQL Mode Limitations 666
Internal CA Dataquery Table 666
Syntax Limitations 667
DQL Mode Nesting Limitations 669
DRAW Command 669
Guided Query Limitations 670
Dialogs 670
Maximums for Both Modes 670
Miscellaneous 671
Batch CA Dataquery 671

Appendix B: Reserved and Ignored Words 673

CA Dataquery Reserved Words 674
SQL Reserved Words 676

Appendix C: Data Types 679

DQL Mode Data Types 679
Personal Database Data Types 680

Chapter 1: Introduction

The purpose of this guide is to provide information needed by the conventional end user to operate CA Dataquery in SQL Mode or DQL Mode.

Audience

This book is a reference manual for the CA Dataquery end user. Use it in your day-to-day work with CA Dataquery when you want to look up details about panel use, the CA Dataquery Editor, operation commands, or query syntax.

If you are a new CA Dataquery user or are unfamiliar with this version of CA Dataquery, we recommend starting with the companion to this book, the *CA Dataquery User Guide*. For information about DQL syntax, see the chapters on query syntax in this manual. For details about SQL syntax in queries, see SQL Statements or the *CA Datacom/DB SQL User Guide*.

If you are also authorized as a CA Dataquery Administrator, see the *CA Dataquery Administrator Guide* for information about CA Dataquery Administrator tasks, commands, concepts, and panels.

Terminology

Some terms in this manual are being used in a special context.

z/OS

The term *region* is used throughout this manual to represent *address space* for z/OS environments.

z/VSE

The term *region* is used throughout this manual to represent *partition* for z/VSE environments.

Reading Syntax Diagrams

Syntax diagrams are used to illustrate the format of statements and some basic language elements. Read syntax diagrams from left to right and top to bottom.

The following terminology, symbols, and concepts are used in syntax diagrams:

- Keywords appear in uppercase letters, for example, COMMAND or PARM. These words must be entered exactly as shown.
- Variables appear in italicized lowercase letters, for example, *variable*.
- Required keywords and variables appear on a main line.
- Optional keywords and variables appear below a main line.
- Default keywords and variables appear above a main line.
- Double arrowheads pointing to the right indicate the beginning of a statement.
- Double arrowheads pointing to each other indicate the end of a statement.
- Single arrowheads pointing to the right indicate a portion of a statement, or that the statement continues in another diagram.
- Punctuation marks or arithmetic symbols that are shown with a keyword or variable must be entered as part of the statement or command. Punctuation marks and arithmetic symbols can include the following:

,	comma	>	greater than symbol
.	period	<-	less than symbol
(	open parenthesis	=	equal sign
)	close parenthesis	¬	not sign
+	addition	-	subtraction
*	multiplication	/	division

Statement Without Parameters

The following is a diagram of a statement without parameters:

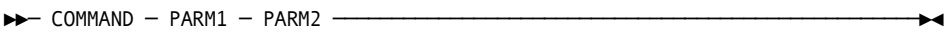
For this statement, you must write the following:

```
COMMAND
```

Statement with Required Parameters

Required parameters appear on the same horizontal line, the main path of the diagram, as the command or statement. The parameters must be separated by one or more blanks.

The following is a diagram of a statement with required parameters:



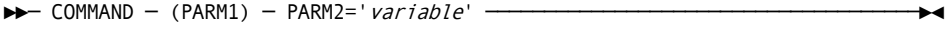
You must write the following:

```
COMMAND PARM1 PARM2
```

Delimiters Around Parameters

Delimiters, such as parentheses, around parameters or clauses must be included.

The following is a diagram of a statement with delimiters around parameters:



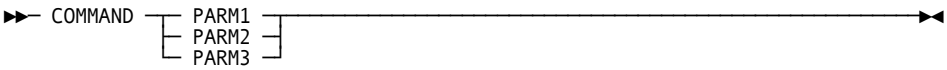
If the word *variable* is a valid entry, you must write the following:

```
COMMAND (PARM1) PARM2='variable'
```

Choice of Required Parameters

When you see a vertical list of parameters as shown in the following example, you must choose one of the parameters. This indicates that one entry is required, and only one of the displayed parameters is allowed in the statement.

The following is a diagram of a statement with a choice of required parameters:



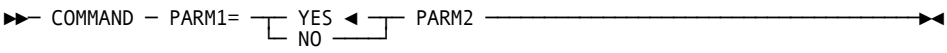
You can choose one of the parameters from the vertical list, such as in the following examples:

```
COMMAND PARM1
COMMAND PARM2
COMMAND PARM3
```

Default Value for a Required Parameter

When a required parameter in a syntax diagram has a default value, the default value appears above the main line, and it indicates the value for the parameter if the command is not specified. If you specify the command, you must code the parameter and specify one of the displayed values.

The following is a diagram of a statement with a default value for a required parameter:



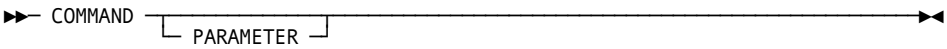
If you specify the command, you must write one of the following:

```
COMMAND PARM1=NO PARM2
COMMAND PARM1=YES PARM2
```

Optional Parameter

A single optional parameter appears below the horizontal line that marks the main path.

The following is a diagram of a statement with an optional parameter:



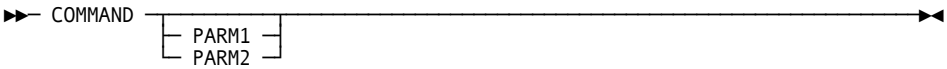
You can choose (or not) to use the optional parameter, as shown in the following examples:

```
COMMAND
COMMAND PARAMETER
```

Choice of Optional Parameters

If you have a choice of more than one optional parameter, the parameters appear in a vertical list below the main path.

The following is a diagram of a statement with a choice of optional parameters:



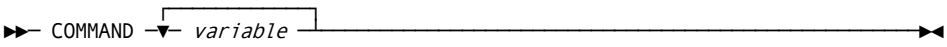
You can choose any of the parameters from the vertical list, or you can write the statement without an optional parameter, such as in the following examples:

```
COMMAND
COMMAND PARM1
COMMAND PARM2
```

Repeatable Variable Parameter

In some statements, you can specify a single parameter more than once. A repeat symbol indicates that you can specify multiple parameters.

The following is a diagram of a statement with a repeatable variable parameter:



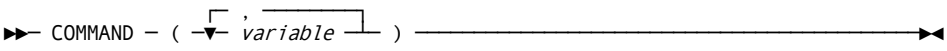
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply, but it is also on the main path, which means that you are required to specify at least one entry. The repeat symbol indicates that you can specify a parameter more than once. Assume that you have three values named VALUEX, VALUEY, and VALUEZ for the variable. The following are some of the statements you might write:

```
COMMAND VALUEX
COMMAND VALUEX VALUEY
COMMAND VALUEX VALUEX VALUEZ
```

Separator with Repeatable Variable and Delimiter

If the repeat symbol contains punctuation such as a comma, you must separate multiple parameters with the punctuation. The following diagram includes the repeat symbol, a comma, and parentheses:

The following is a diagram of a statement with a separator with a repeatable variable and a delimiter:



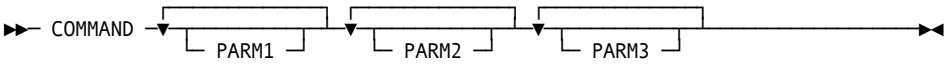
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply. It is also on the main path, which means that you must specify at least one entry. The repeat symbol indicates that you can specify more than one variable and that you must separate the entries with commas. The parentheses indicate that the group of entries must be enclosed within parentheses. Assume that you have three values named VALUEA, VALUEB, and VALUEC for the variable.

The following are some of the statements you can write:

```
COMMAND (VALUEC)
COMMAND (VALUEB,VALUEC)
COMMAND (VALUEB,VALUEA)
COMMAND (VALUEA,VALUEB,VALUEC)
```

Optional Repeatable Parameters

The following diagram shows a list of parameters with the repeat symbol for optional repeatable parameters:



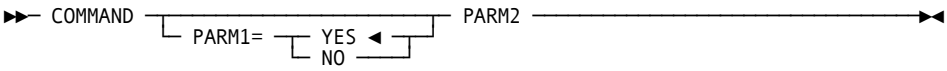
The following are some of the statements you can write:

```
COMMAND PARM1
COMMAND PARM1 PARM2 PARM3
COMMAND PARM1 PARM1 PARM3
```

Default Value for a Parameter

The placement of YES in the following diagram indicates that it is the default value for the parameter. If you do not include the parameter when you write the statement, the result is the same as if you had actually specified the parameter with the default value.

The following is a diagram of a statement with a default value for an optional parameter:

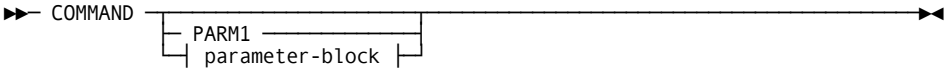


For this command, COMMAND PARM2 is the equivalent of COMMAND PARM1=YES PARM2.

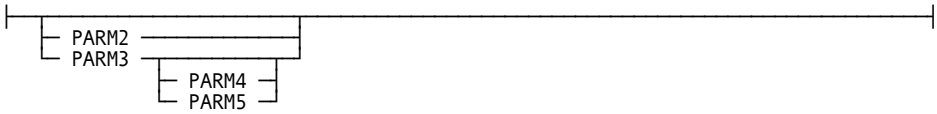
Variables Representing Several Parameters

In some syntax diagrams, a set of several parameters is represented by a single reference.

The following is a diagram of a statement with variables representing several parameters:



Expansion of parameter-block



The *parameter-block* can be displayed in a separate syntax diagram.

Choices you can make from this syntax diagram therefore include, but are not limited to, the following:

```
COMMAND PARM1
COMMAND PARM3
COMMAND PARM3 PARM4
```

Note: Before you can specify PARM4 or PARM5 in this command, you must specify PARM3.

Listing Libraries for CA Datacom Products

Guidelines to assist you in preparing your JCL are provided in this manual. The sample code provided in this document is intended for use as a reference aid only and no warranty of any kind is made as to completeness or correctness for your specific installation.

Samples for JCL and programs are provided in the install library (in z/OS, the default name for this library is CABDMAC). In z/VSE, sample PROCs are provided that allow you to make use of parameter substitution. You can copy and modify these samples for your specific requirements.

Any JOB statements should be coded to your site standards and specifications. All data set names and library names should be specified with the correct names for the installation at your site. In many examples, a REGION= or SIZE= parameter is displayed in an EXEC statement. The value displayed should be adequate in most instances, but you can adjust the value to your specific needs.

The libraries listed for searching must include the following in the order shown:

1. User libraries (*hlq*.CUSLIB) you may have defined for specially assembled and linked tables, such as DBMSTLST, DBSIDPR, DDSRTLM, DQSYSTBL, or User Requirements Tables
2. CA Datacom base libraries (*hlq*.CABDLOAD): CA Datacom/DB, CA Datacom Datadictionary, CA Dataquery for CA Datacom, SQL
3. CA IPC libraries (*hlq*.CAVQLOAD)

4. CA Common Services for z/OS base libraries (*hlq*.CAW0LOAD)
5. Libraries for additional products, such as CA Datacom CICS Services, CA Datacom VSAM Transparency, CA Ideal, and so on

CA Dataquery for CA Datacom users also need the following libraries and data sets for the following specific functions:

- The z/OS data set DQOUT or the z/VSE data set DQOUTD is used only if the DQBATCH execution uses the EXPORT function.
- In z/OS, running deferred queries with separate JCL members in batch requires, in addition to the SYSIN statement DEFER, the inclusion of a DD statement for the internal reader used by VPE. This DD statement should be:

```
//IRDR DD     SYSOUT=(A,INTRDR)
```

Chapter 2: How to Use Panels

This chapter presents basic operational instructions for each CA Dataquery for CA Datacom panel that allows an entry or selection from you. The panels appear in order by the panel number.

If your company customizes CA Dataquery for CA Datacom panels with the Language Maintenance Facility, the panels you see on your terminal can vary from the examples shown in this chapter. However, the purpose and operation of the panel, its fields, and its PF keys cannot be changed. To look up a panel by its CA-assigned panel name, see the index.

To look up panel usage information on panels used by the CA Dataquery for CA Datacom Administrator, please see the *CA Dataquery for CA Datacom Administration Guide*.

Formats

Every CA Dataquery for CA Datacom menu and panel has the same basic format. Match the numbers on the sample menus and panels on the facing page to their explanations on this chart.

Parts of a Menu or Panel:

Key	Name	Description
1	Command line	Use this line to enter one or more valid CA Dataquery commands.
2	Message line	Displays error messages, status information, or brief instructions.
3	Separator line	Displays a panel number for use in translating panels with the Language Maintenance Facility or for reference when requesting help.
4	Prefix	Preceded by a separator line and identifies the panel.

Key	Name	Description
5	Body	**Menu** - Contains list of options and selection field for typing an option number. **Selection panel** - Contains lists of items from which you make selections by positioning the cursor on your choices and pressing a key. Selections can be headed by lines of operation instructions. **Input panel** - Contains fields for entry or an editor area for writing a Standard Query. First lines can also contain operation instructions.
6	PF Key menu	Displays list of valid PF keys.

The Language Maintenance Facility can be used by authorized personnel at your site to customize the language or terminology of any panel. See the information about the Language Maintenance Facility in the *CA Dataquery for CA Datacom Administration Guide*.

Menu

```
1 =>

2

3 ------------------------------------------------------------------------DQZ60

4 DATAQUERY: MAIN MENU

5 ENTER THE NUMBER OF THE DESIRED FUNCTION ===> _

  1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
  2.  CREATE              - Query, Dialog or Term creation
  3.  GUIDE               - Structured query creation
  4.  PDB                 - List, create and maintain personal tables
  5.  ADMINISTRATION      - DATAQUERY system management
  6.  HELP                - Display Help Information
  7.  OFF                 - DATAQUERY session termination
```

Selection Panel

```
1 =>

2 Place the cursor on the desired name and press the appropriate PFkey
 ------------------------------------------------------------------------------DQA30
4 DATAQUERY: DIRECTORY OF QUERIES                START WITH: __________
 ------------------------------------------------------------------------------
5 QUERY NAME | TYPE | CREATED    |   USED   |     DESCRIPTION
 ------------------------------------------------------------------------------
   A-CUST-CITY     | DIALOG | 04/06/95 | 06/10/95 | SELECT CUSTOMER
   A-GUIDED-QUERY  | QUERY  | 03/19/95 | 06/10/95 | A SAMPLE OF DQ 4.0
   A-SAMPLE-QUERY  | QUERY  | 03/19/95 | 06/11/95 | A SAMPLE PAYROLL QUERY
   CAI-CUST-ZIP    | DIALOG | 03/14/95 | 06/09/95 | SELECT CUSTOMER
   CAI-CUST-TERMS  | QUERY  | 03/12/95 | 06/04/95 | SELECT CUSTOMER
   CAI-CUST-SHIP   | DIALOG | 04/01/95 | 06/10/95 | SELECT DALLAS CUSTOMERS
   CAI-DIALO-ORDNO | DIALOG | 04/13/95 | 06/11/95 | PRINT SAMPLE DATABASE
   CAI-DISPLAY-DT  | DIALOG | 04/13/95 | 06/08/95 | DISP SAMPLE DATABASE



6 ------------------------------------------------------------------------ ==>
 <PF1> HELP         <PF2>  RETURN        <PF3>  EXECUTE     <PF4>  EDIT
 <PF5> NOT USED     <PF6>  DELETE        <PF7>  BACKWARD    <PF8>  FORWARD
 <PF9> SUBMIT       <PF10> EXTENDED DEF  <PF11> NOT USED    <PF12> RIGHT
```

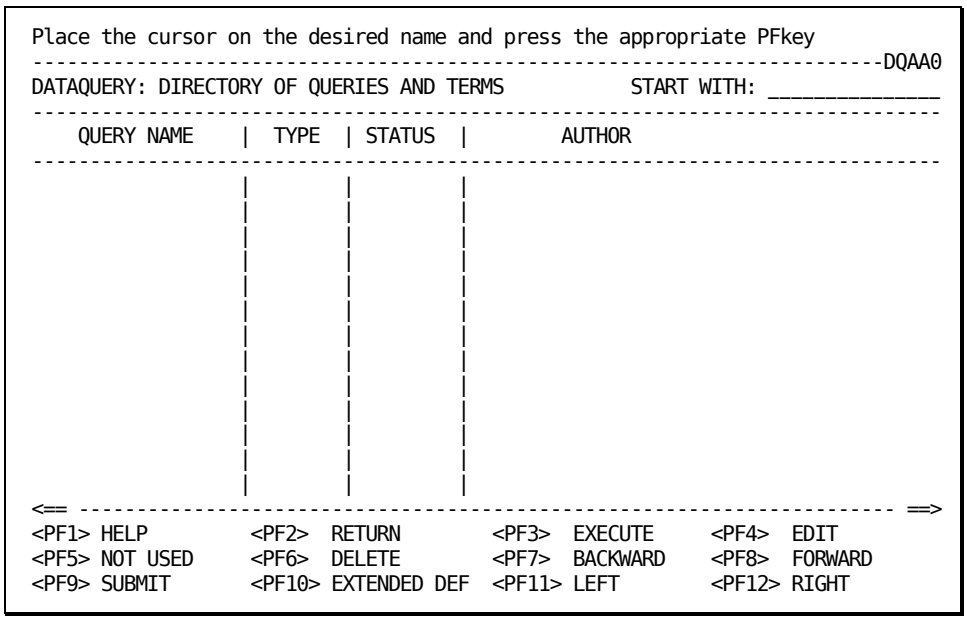
Input Panel

```
1 =>

2
 ------------------------------------------------------------------------------DQH10
4 DATAQUERY: QUERY IDENTIFICATION
 ------------------------------------------------------------------------------
5
 ENTER A UNIQUE NAME FOR THE NEW QUERY ==>   _______________

 SELECT THE ACCESSIBILITY LEVEL FOR YOUR QUERY BY PLACING ANY CHARACTER
 NEXT TO YOUR CHOICE.

    _   PUBLIC
   X   PRIVATE

 ENTER A DESCRIPTION FOR THE NEW QUERY
   ==> ____________________________________________________________


6 ------------------------------------------------------------------------------
 <PF1> HELP         <PF2> RETURN        <PF3> NOT USED      <PF4>  CONTINUE
```

Chapter 3: Panels

The following pages contain reference information about CA Dataquery panels. Look up an individual panel by its panel number, such as DQAA0.

(DQAA0) Directory of Queries and Terms



Description

This panel displays a directory of queries, terms, or dialogs you are allowed to access. The type (query, term, or dialog) and status (public, private) of the displayed members are determined by your directory selection or the command you entered.

The panel displays the type, status and author for the listed members. You can execute, delete, or edit a query or term by placing the cursor on the line which contains the member name and pressing the PF key which initiates the desired function. Scrolling right displays a new panel (AB0) listing each member's group assignment. Scrolling left displays a new panel (A30) listing each member's creation date, date last used, and description.

Action

From the directory, display scroll the display forward, backward, right, or left using PF keys 7, 8, 11 and 12 until you locate the member name you want to execute, delete, or edit. Move your cursor down to the line that contains the member name. Press the PF key to perform the desired function.

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for a discussion of each column on this panel.

QUERY NAME

Name assigned query or term at creation.

TYPE

Query, Dialog, and Term are valid types.

STATUS

Public or private.

AUTHOR

User ID of the person who created the member.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> EXECUTE

Begins execution of member selected by cursor position.

<PF4> EDIT

Displays member selected by cursor on Editor panel.

<PF6> DELETE

Deletes member selected by cursor position if you are the author.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

<PF9> SUBMIT

Begins batch execution of member selected by cursor position.

<PF10> EXTENDED DEF

Displays extended definition of member selected by cursor position.

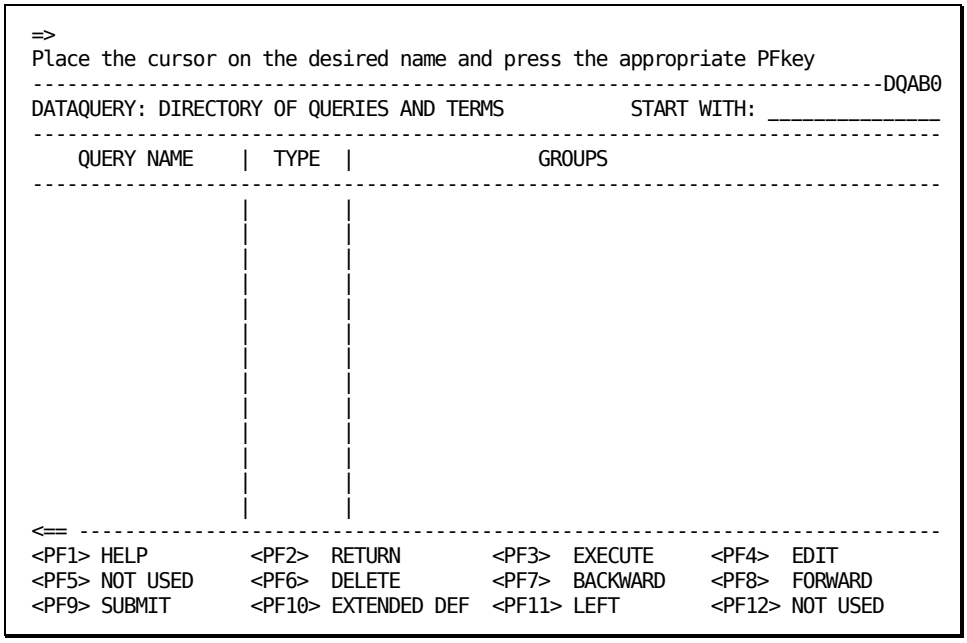
<PF11> LEFT

Displays creation date, date used, and description.

<PF12> RIGHT

Displays group assignments.

(DQAB0) Directory of Queries and Terms



Description

This panel displays a directory of queries, term, or dialogs you are allowed to access. The type (query, term, or dialog) and status (public, private) of the displayed members are determined by your directory selection or by entering a command.

The panel displays the types and group assignments for the listed members. You can select a member by placing the cursor on the line which contains the member name and pressing the PF Key which initiates the desired function.

Action

From the directory display, scroll the display forward, backward, or left using PF keys 7, 8, and 11 until you locate the member name you want to execute, delete, or edit. Move your cursor down to the line that contains the member name. Press the PF key to perform the desired function.

Entry Fields

Refer to the following for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for a discussion of each column on this panel.

QUERY NAME

Name assigned query, dialog or term at creation.

TYPE

Query, Dialog, and Term are valid types.

GROUP

The group to which the member has been assigned, if any.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF3> EXECUTE

Begins execution of member selected by cursor position.

<PF2> RETURN

Exits the current panel.

<PF4> EDIT

Displays member selected by cursor on Editor panel.

<PF6> DELETE

Deletes member selected by cursor position if you are the author.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

<PF9> SUBMIT

Begins batch execution of member selected by cursor position.

<PF10> EXTENDED DEF

Displays extended definition of member selected by cursor position.

<PF11> LEFT

Displays member types and descriptions.

(DQAD0) LIST COMMAND Prompt (SQL Mode)

```
=>

-------------------------------------------------------------------------------DQAD0
DATAQUERY:  LIST COMMAND

LIST ALL     PRIVATE                          A
     (1)       (2)                           (3)
-------------------------------------------------------------------------------
(1) Select a        (2) Select the type       (3) Enter 1 to 32 letters of the
    Directory Type:     of Query Library:         alphabet to indicate the
                                                  beginning of the display.
     ALL (Queries       PRIVATE
       and dialogs)     PUBLIC
     QUERIES            BOTH (Public and private)
     DIALOGS            A User (Enter ID)
     TABLES




-------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

Description

Entering LIST on the command line and pressing Enter displays the List Command prompt panel for SQL Mode. It prompts you for the directory type, library type (if a query or dialog), and a list starting position. You can enter LIST followed by operands and immediately display a directory of queries, dialogs, or tables. See Operation Commands for more information about the LIST command.

Action

Accept the default entries or enter a valid item from the list below each field and press Enter.

Entry Fields

Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits.

(DQAF0) Directory of Tables

```
=>
Place the cursor on the desired name and press the appropriate PFkey
-------------------------------------------------------------------------------DQAF0
DATAQUERY:  DIRECTORY OF TABLES            DATADICTIONARY BASE ID: 444
                                  START WITH: _____________________________
-------------------------------------------------------------------------------
      TABLE  NAME               |STATUS|     DESCRIPTION
-------------------------------------------------------------------------------
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
                                |      |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN       <PF3> DISPLAY COLUMNS <PF4> DISPLAY KEYS
<PF5> DISPLAY ALL   <PF6> DISPLAY TEXT <PF7> BACKWARD        <PF8> FORWARD
```

Description

This panel displays a list of database tables you are authorized to access with a query or dialog.

Action

View the list to determine which tables you want to search with your query or dialog. Note whether the table is open (O), as you can only query an open table. To display a list of columns or keys or the description of any table, position the cursor on its name and press the appropriate PF key.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

DATADICTIONARY BASE ID

The database ID of the data dictionary containing the tables you can access.

Entry Fields

Refer to the following for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME

Lists tables you are authorized to access.

STATUS

Lists status of each table:

O

Open and can be queried.

C

Closed and cannot be queried.

D

Disabled. Must be enabled in the Datadictionary before it can be accessed.

DESCRIPTION

A description of the table.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY COLUMNS

Displays a list of columns for the table indicated by cursor position.

<PF4> DISPLAY KEYS

Displays a list of keys for the table indicated by cursor position.

<PF5> DISPLAY ALL

Displays a list of keys and columns for the table indicated by the cursor position.

<PF6> DISPLAY TEXT

Displays the textual description for the table indicated by cursor position.

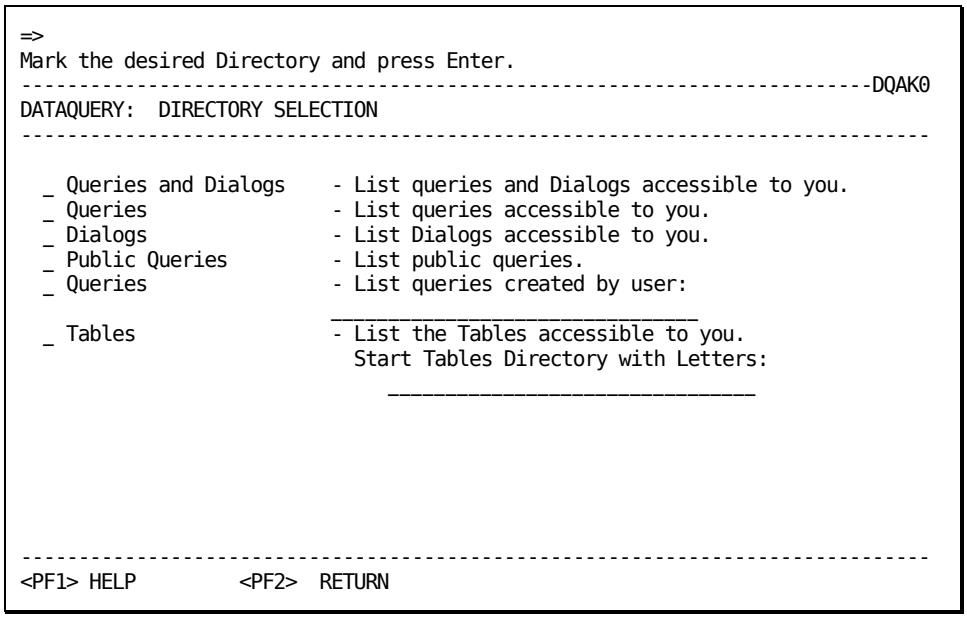
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQAK0) Directory Selection (SQL)



Description

In SQL Mode, when you select DIRECTORIES from the CA Dataquery Main Menu or enter DIRECTORY on the command line and press Enter, you see the Directory Selection panel that lets you specify the kind of directory you want CA Dataquery to display. You can display directories of query libraries and authorized database tables.

Action

Tell CA Dataquery which directory you want to see by placing a character in the field preceding its name and pressing Enter. For detailed information about each directory panel and its operation, see each panel by name or number in this chapter.

Select

Make selections on this panel according to the following.

Queries and Dialogs

Lists your private queries and dialogs and all public queries you can access.

Queries

Lists your private queries and all public queries you can access.

Dialogs

Lists your private dialogs and all public dialogs you can access.

Public Queries

Lists all public queries and dialogs you can access.

Queries

Lists all queries and dialogs created by user named in field indicated by underscores.

Tables

Lists all database tables you are authorized to access.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

(DQAZ0) Bulletin Board

```
=>
Press the Enter key to continue.
--------------------------------------------------------------------------------DQAZ0
DATAQUERY: BULLETIN BOARD
--------------------------------------------------------------------------------
```

Description

CA Dataquery offers an optional bulletin board which can be used by the CA Dataquery Administrator at your site to post notices for CA Dataquery users. If messages are currently posted, the Bulletin Board panel appears after you sign on to CA Dataquery.

Action

Read the message, if any, and press the Enter key to display the first CA Dataquery panel for your authorization level. Or, enter a valid command on the command line to proceed directly to an authorized function.

(DQA10) LIST Command

```
=>

--------------------------------------------------------------------------------DQA10
DATAQUERY:  LIST COMMAND

LIST ALL     BOTH                            A
     (1)       (2)                          (3)
------------------------------------------------------------------------------
(1) Select a           (2) Select the type   (3) Enter 1 to 32 letters of the
    Directory Type:        of Query or Term      alphabet to indicate the
                            Library:             beginning of the display.
     ALL (Queries,                               Not valid for LIST SETS.
     terms, and             PRIVATE
     dialogs)               PUBLIC
     QUERIES                   BOTH (Public and private)
     TERMS                      A User      (Enter ID)
     DIALOGS
     TABLES
     SETS


--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

Description

Entering LIST on the command line and pressing Enter displays the List Command prompt panel for DQL Mode. It prompts you for the directory type, library type (if a query, dialog or term), and a list starting position. You can enter LIST followed by operands and immediately display a directory of queries, terms, dialogs, tables, or saved sets. See Operation Commands for more information about the LIST command.

Action

Accept the default entries or enter a valid item from the list below each field and press Enter.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
(1)	Yes	ALL, QUERIES, TERMS, DIALOGS, TABLES, SETS	ALL
(2)	No	PUBLIC, PRIVATE, BOTH, A User	BOTH
(3)	No	1-32 alpha characters	A

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

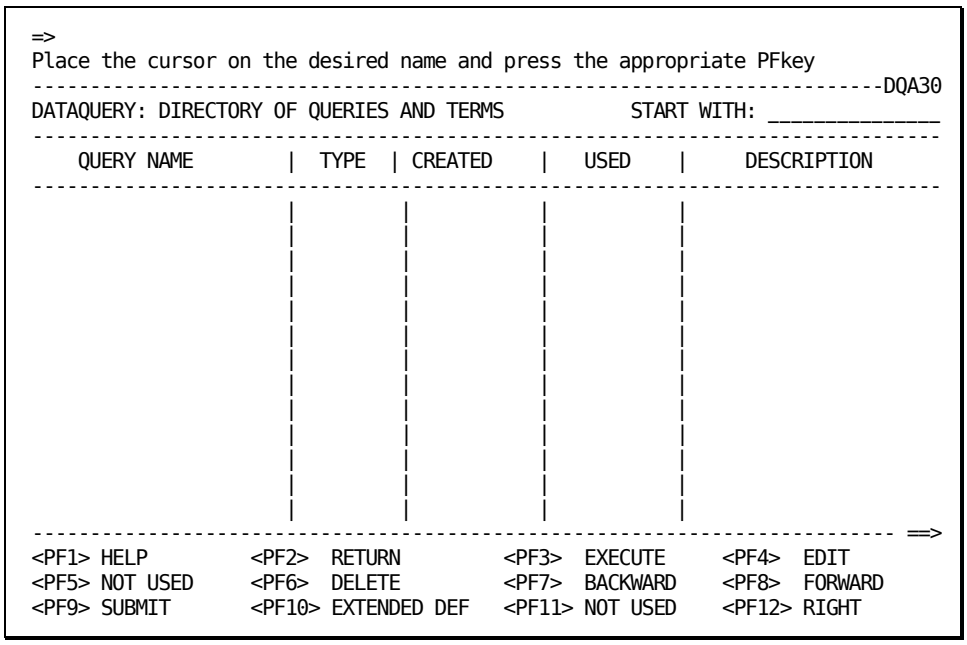
<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits.

(DQA30) Directory of Queries and Terms



Description

This panel displays a directory of queries, term, or dialogs you are allowed to access.

The panel displays the type, date created, date used, and description for the listed members. You select a member by placing the cursor on the line which contains the member name and pressing the PF key which initiates the desired function. Scrolling right on this panel displays a new panel (DQAA0) that lists each member's status and author.

Action

From the directory display, scroll the display forward and backward using PF keys 7 and 8 until you locate the member name you want to select. Move your cursor down to the line that contains the member name. Press the PF key to perform the desired function.

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for an explanation of each column on this panel.

QUERY NAME

Name assigned query or term at creation.

TYPE

Query, Dialog, and Term are valid types.

CREATED

Date of creation.

USED

Date of last access.

DESCRIPTION

Description attached at creation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> EXECUTE

Executes member selected by cursor position.

<PF4> EDIT

Displays member selected by cursor on Editor panel.

<PF6> DELETE

Deletes member selected by cursor position if you are the author.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

<PF9> SUBMIT

Begins batch execution of member selected by cursor position.

<PF10> EXTENDED DEF

Displays extended definition of member selected by cursor position.

<PF12> RIGHT

Displays authors.

(DQA40) Directory Selection

```
=>
Mark the desired Directory and press Enter.
--------------------------------------------------------------------------------DQA40
DATAQUERY:   DIRECTORY SELECTION
--------------------------------------------------------------------------------

  _ Queries and Terms        - List all queries and terms accessible to you.
  _ Queries Only             - List queries accessible to you.
  _ Terms Only               - List terms accessible to you.
  _ Dialogs                  - List Dialogs accessible to you.
  _ Public Queries           - List public queries.
  _ Queries and Terms        - List queries and terms created by user:
                             ____________________________

  _ Tables                   - List the tables accessible to you.
                               Start Table Directory with Letter:
                             _________________________________
  _ Saved Sets               - List the saved sets.




--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

Description

When you select DIRECTORIES from the DQL Mode Main Menu or enter DIRECTORY on the command line and press Enter, you see the Directory Selection panel that lets you specify the kind of directory you want CA Dataquery to display. You can display directories of query libraries, authorized database tables and sets of saved query output (*saved sets*).

Action

Tell CA Dataquery which directory you want to see by placing a character in the field preceding its name and pressing Enter. For detailed information about each directory panel and its operation, see each panel by name or number in this chapter.

Select

Make selections on this panel according to the following.

Queries and Terms

Lists your private queries, dialogs and terms and all public queries and terms you can access.

Queries Only

Lists your private queries and dialogs and all public queries and dialogs you can access.

Terms Only

Lists your private terms and all public terms you can access.

Dialogs

Lists your private dialogs and all public dialogs you can access.

Public Queries

Lists all public queries and dialogs you can access.

Queries and Terms

Lists all queries, dialogs and terms created by user named in field indicated by underscores.

Tables

Lists all database tables you are authorized to access.

Saved Sets

Lists all found sets saved by you.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits.

(DQA50) Directory of Queries (Associate User)

```
=>
Place the cursor on the desired name and press the appropriate PFkey
-------------------------------------------------------------------------------DQA50
DATAQUERY: DIRECTORY OF QUERIES                     START WITH: _______________
-------------------------------------------------------------------------------
       QUERY NAME          |   TYPE   |   USED   |     DESCRIPTION
-------------------------------------------------------------------------------
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
                           |          |          |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> SIGN-OFF       <PF3> EXECUTE     <PF4> SUBMIT
<PF5> NOT USED      <PF6> NOT USED       <PF7> BACKWARD    <PF8> FORWARD
```

Description

If you are an associate user this panel is the first panel you see after pressing Enter on the Bulletin Board. This list allows you to select a query or dialog for submission or execution.

Action

After CA Dataquery displays the directory, you can scroll the list of queries and dialogs backward or forward by using PF keys 7 and 8. Once you have located the query or dialog you want to submit or execute, move your cursor down to the line which contains the query or dialog name. If you want to execute that query or dialog online press, <PF3> EXECUTE. To submit in batch, press <PF4> SUBMIT.

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for a discussion of each column on this panel.

QUERY NAME

Name assigned the query or dialog at creation.

TYPE

Query or dialog.

USED

Last date the query or dialog was used.

DESCRIPTION

Description assigned the query or dialog at creation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays information about the current panel.

<PF2> SIGN-OFF

Signs off CA Dataquery.

<PF3> EXECUTE

Displays online execution panel for the selected query or dialog.

<PF4> SUBMIT

Displays batch execution panel for the selected query or dialog.

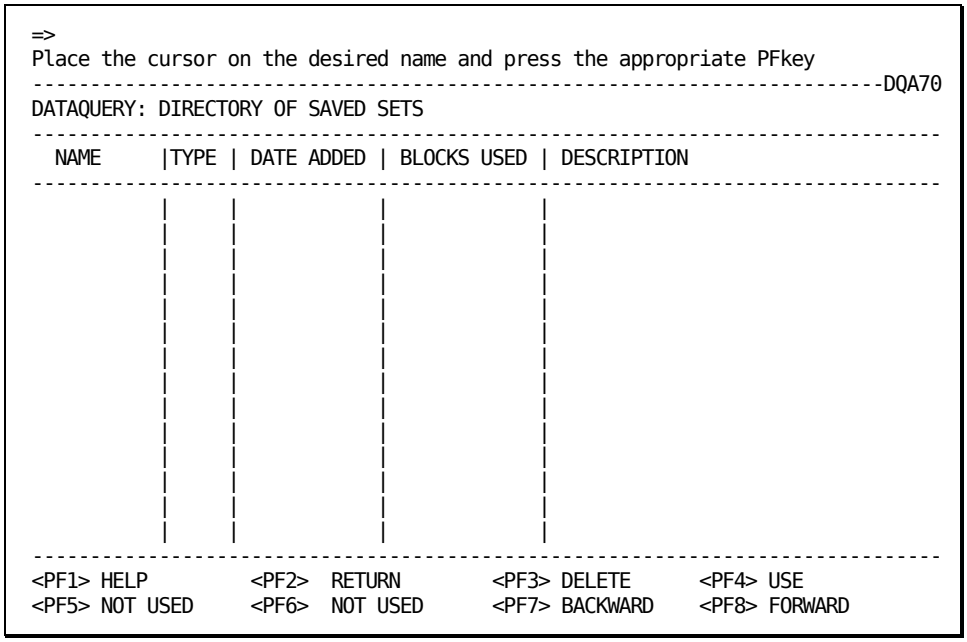
<PF7> BACKWARD

Scrolls backward one page.

<PF8> FORWARD

Scrolls forward one page.

(DQA70) Directory of Saved Sets



Description

This DQL Mode panel lists saved sets and extracted sets by name, type, description, date and size. Display the directory when you want to use a saved set or when you want to delete a saved or extracted set.

Action

To display your saved sets, use the LIST command or select Saved Sets from the Directories Menu. CA Dataquery then displays all your saved sets and any saved sets you have extracted. You can then select a saved set for use or deletion by moving the cursor to the line which contains the saved set name and pressing the appropriate PF key.

Columns

Refer to the following for an explanation of each column on this panel.

NAME

Name you assigned a saved or extracted set.

TYPE

K for saved set. E for extracted set.

DATE ADDED

Date you saved or extracted the set.

BLOCKS USED

Number of physical blocks occupied by the set. Your CA Dataquery Administrator controls the number of blocks allotted to each user for maintaining sets.

DESCRIPTION

Description you attached when you saved the set.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits.

<PF3> DELETE

Deletes the set indicated by your cursor.

<PF4> USE

Activates the set indicated by your cursor position.

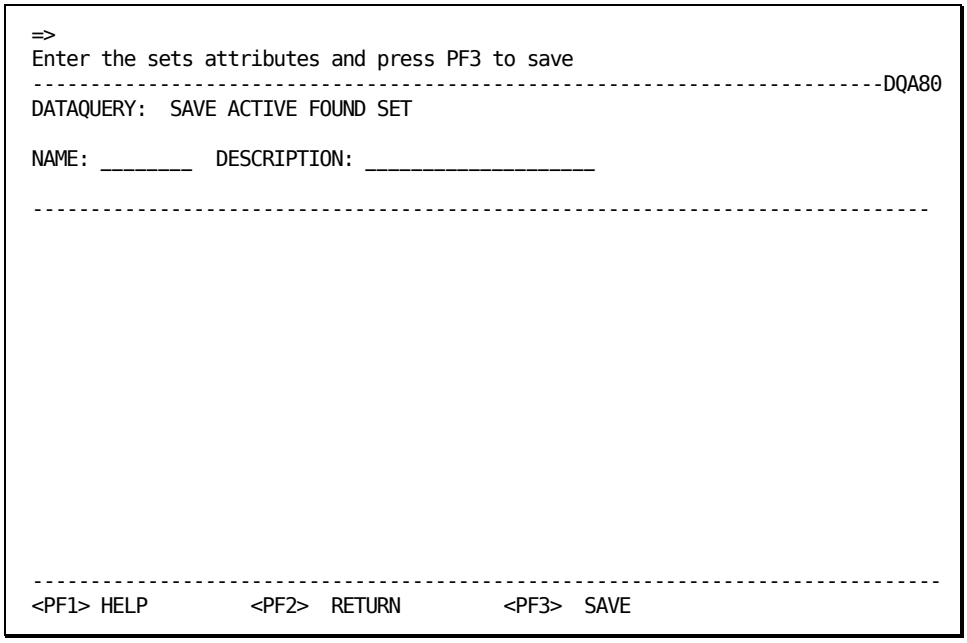
<PF7> BACKWARD

Scrolls back to previous panel, if any.

<PF8> FORWARD

Scrolls to next panel, if any.

(DQA80) Save Active Found Set



Description

When you enter KEEP on the command line in DQL Mode and press ENTER, CA Dataquery displays the Save Active Found Set panel. Use the panel to save the active found set to speed processing time for future query executions.

For more information about sets, see the *CA Dataquery User Guide*. For more information about the KEEP command, see Operation Commands.

Action

Enter a set name and description and press <PF3> SAVE to save the found set. Only one saved set is permitted per active found set.

Note: A query must have first been executed for an active found set to exist.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
NAME:	Y	Any eight characters beginning with a letter.	No default
DESCRIPTION:	Y	1-20 characters.	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

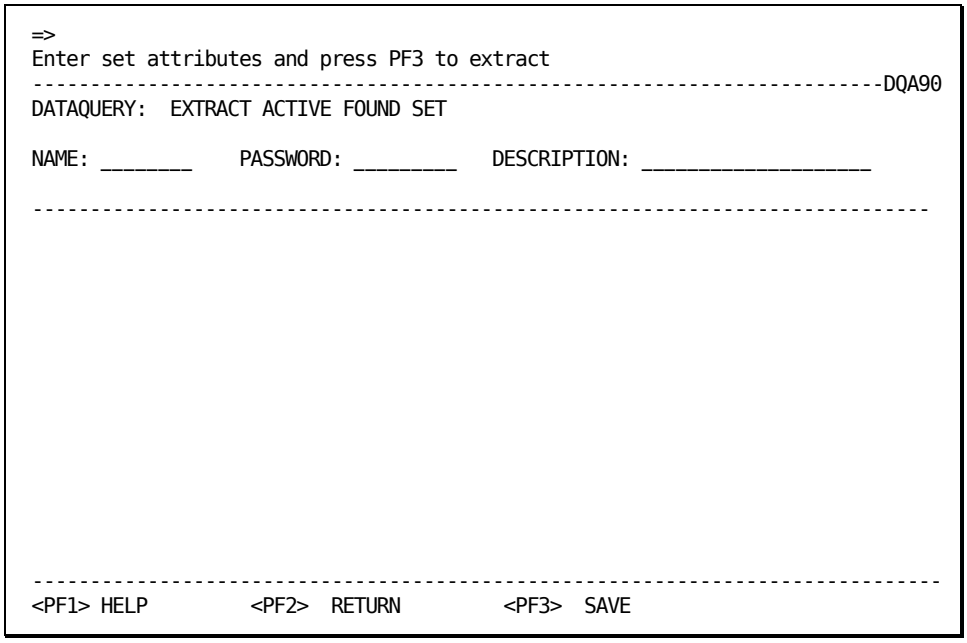
<PF2> RETURN

Exits panel.

<PF3> SAVE

Saves the found set with the name and description entered.

(DQA90) Extract Active Found Set



Description

Use this DQL Mode panel to create a set of data that can be accessed by non-CA Dataquery programs. To display this panel, type **EXTRACT** in the command line of any panel. For more information about extracting active found sets, see Operation Commands.

Action

Fill in this panel by entering a name for the extracted set. Type a password and description if desired. Press <PF3> SAVE to save the set.

Note: A query must have first been executed for an active found set to exist.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
NAME:	Y	Enter the set name. Must begin with alpha character.	No default

Field	Required?	Valid Entries	Default
PASSWORD:	N	Enter a 1-9 character password.	No default
DESCRIPTION:	N	Enter a 1-20 character description.	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits.

<PF3> SAVE

Saves the extracted set.

(DQBA0) Column Display

```
=>
Place the cursor on the desired name and press the appropriate PFkey.
-------------------------------------------------------------------------------DQBA0
DATAQUERY:  COLUMN DISPLAY                    DICTIONARY DATABASE ID:
-------------------------------------------------------------------------------
 TABLE NAME:
-------------------------------------------------------------------------------
 SYNONYM FOR:
-------------------------------------------------------------------------------
           COLUMN NAME                |             DESCRIPTION
-------------------------------------------------------------------------------
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
                                      |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2>  RETURN    <PF3> NOT USED       <PF4> NOT USED
<PF5> EXTENDED DEFN  <PF6>  TEXT      <PF7> BACKWARD       <PF8> FORWARD
```

Description

This SQL Mode panel displays the names and descriptions of columns in the table you selected. Also displayed is the name of the table for which this table is a synonym.

Action

No action is expected. You can press a PF key to return or for additional information, or enter a command on the command line.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

DICTIONARY DATABASE ID:

Names your Datadictionary database ID.

TABLE NAME:

Names the table you selected.

SYNONYM FOR:

Names the table for which this table is a synonym.

Columns

Refer to the following for an explanation of each column on this panel.

COLUMN NAME

Lists columns defined for this table.

DESCRIPTION

Lists descriptions, if any, for columns in this table.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF5> EXTENDED DEFN

Displays additional information about a column.

<PF6> TEXT

Displays text information about the table.

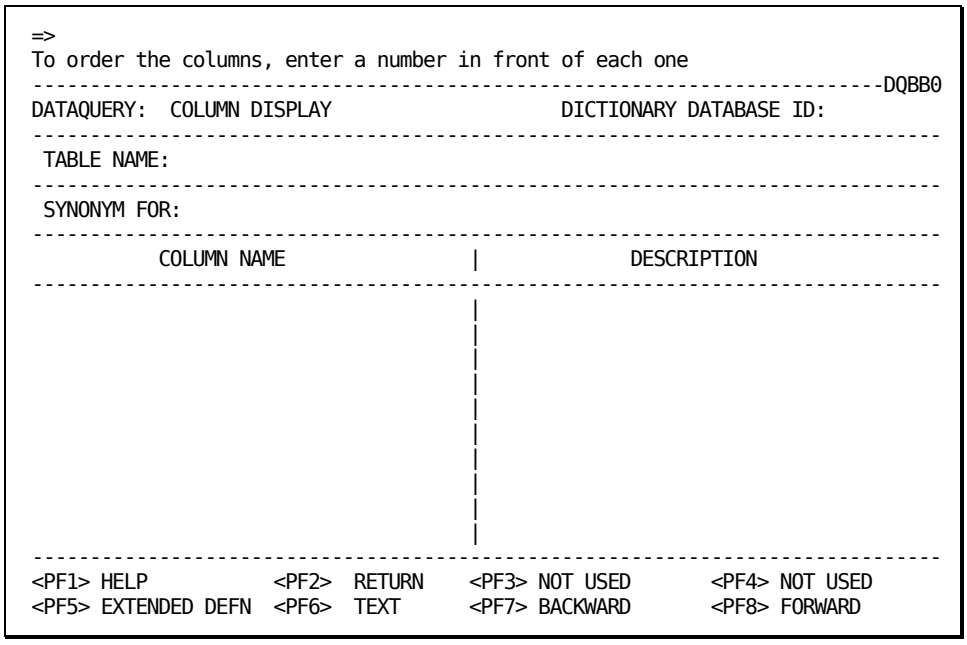
<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQBB0) Column Display



Description

This SQL Mode panel displays the names and descriptions of columns in the table you selected. Also displayed is the name of the table for which this table is a synonym.

Action

No action is expected. You can press a PF key to return or for additional information or enter a command on the command line.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

DICTIONARY DATABASE ID:

Names your Datadictionary database ID.

TABLE NAME:

Names the table you selected.

SYNONYM FOR:

Names the table for which this table is a synonym.

Columns

Refer to the following for an explanation of each column on this panel.

COLUMN NAME

Lists columns defined for this table.

DESCRIPTION

Lists descriptions, if any, for columns in this table.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF5> EXTENDED DEFN

Displays additional information about a column.

<PF6> TEXT

Displays text information about the table.

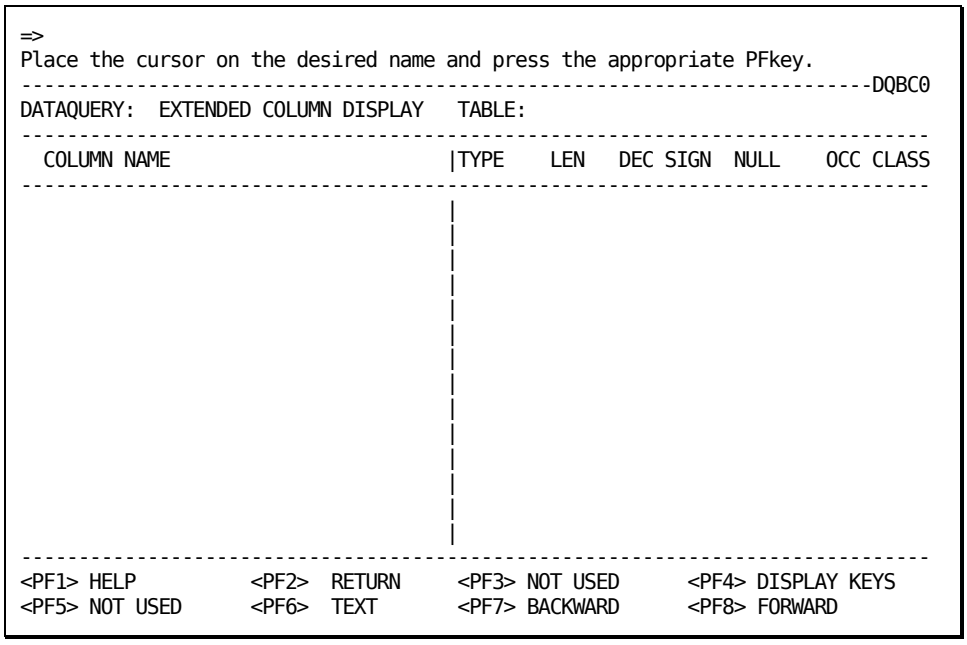
<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQBC0) Extended Column Display



Description

This panel displays additional attributes not shown on the COLUMN DISPLAY panel. Display this panel by selecting the EXTENDED DEF PF key on the COLUMN DISPLAY panel (DQB20).

Action

When this panel is displayed, you can view the attributes of the column or use the cursor position to select a column.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

TABLE:

Name of the table whose columns are listed on the panel.

Columns

Refer to the following for explanations of the columns on this panel.

COLUMN NAME

Names of the columns for the table you selected.

TYPE

Data types of each column:

C

Character

E

Extended floating-point

G

Graphics

H

Hex

K

Kanji

L

Long floating-point

N

Numeric (including binary)

S

Short floating-point

T

Bit for PL/I

Y

Double character

Z

Double character mixed

LEN

Number of positions.

DEC

Number of positions after the decimal.

SIGN

If a numeric column, Y (has a + or - sign) or N (is unsigned).

NULL

You may assign null values to this field.

OCC

Number of occurrences. Contains a 1 if not a repeating column.

CLASS

C

Compound for group columns

S

Simple for single columns

F

Filler for place-holder columns

Numeric

Packed, Zoned and Halfword Binary types and lengths are reported as numeric and may not indicate an appropriate match for RELATED BY statements.

DATE/TIME

When creating queries, do not include columns defined as data types DATE, TIME, or TIMESTAMP. These data types are reserved for SQL Mode queries. These data types are listed as character types on Extended Column Display panels because SQL Mode queries return character-type data.

To test a column that is suspected of being one of these types before selecting it for a DQL Mode query, follow these steps:

1. Look at the length of the column. DATE length is always 10, TIME is always 8, and TIMESTAMP is always 26.
2. If the data type is suspect, create a query that finds one row and PRINT the output.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> DISPLAY KEYS

Displays the KEY DISPLAY panel (DQB50).

<PF6> TEXT

Displays text description on TEXT DISPLAY (COLUMN) panel (DQB40).

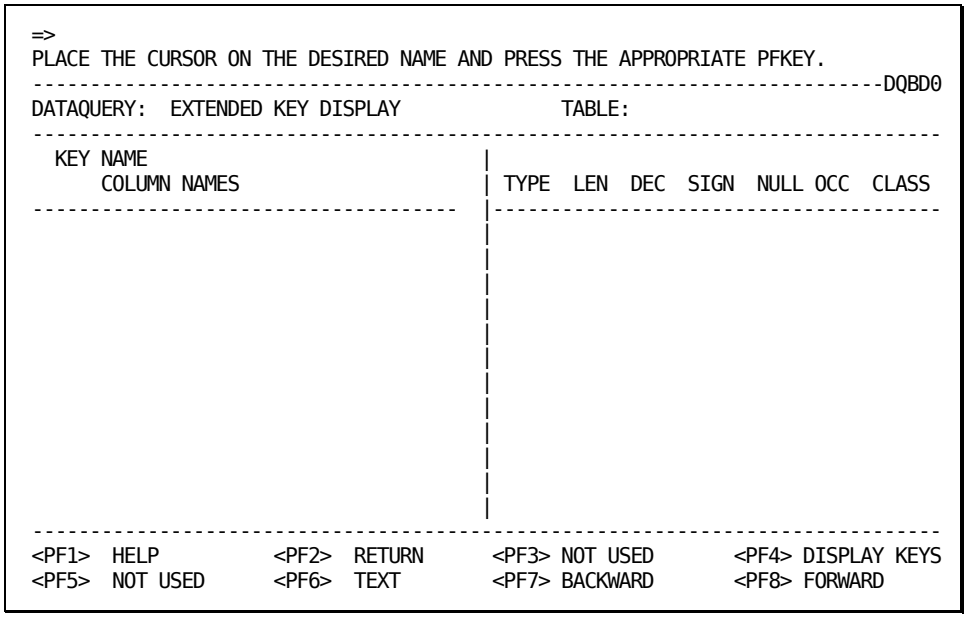
<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQBD0) Extended Key Display



Description

This panel displays descriptions of keys shown on the KEY DISPLAY panel (DQB50). Display this panel by selecting the EXTENDED DEF PF key on any panels where it is displayed as an option.

Action

When this panel is displayed, you can view the descriptions of the key or use the cursor position to select a key.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

TABLE:

Name of the table whose keys and columns are listed on the panel.

Columns

Refer to the following for explanations of the columns on this panel.

KEY NAME/ COLUMN NAMES

Names of the keys and columns for the table you selected.

TYPE

Data types for each column:

C

Character

E

Extended floating-point

G

Graphics

H

Hex

K

Kanji

L

Long floating-point

N

Numeric (including binary)

S

Short floating-point

T

Bit for PL/I

Y

Double character

Z

Double character mixed

LEN

Number of positions.

DEC

Number of positions after the decimal.

SIGN

If a numeric key/column, Y (has a + or - sign) or N (is unsigned).

NULL

You may assign null values.

OCC

Number of occurrences. If 1, it is not a repeating key/column.

CLASS

C

Compound for group keys/columns

S

Simple for single keys/columns

F

Filler for place-holder keys/columns

Numeric

Packed, Zoned and Halfword Binary types and lengths are reported as numeric and may not indicate an appropriate match for RELATED BY statements.

DATE/TIME

When creating queries, do not include columns defined as data types DATE, TIME, or TIMESTAMP. These data types are reserved for SQL Mode queries. These data types are listed as character types on Extended Column Display panels because SQL Mode queries return character-type data.

To test a column that is suspected of being one of these types before selecting it for a DQL Mode query, follow these steps:

1. Look at the length of the column. DATE length is always 10, TIME is always 8, and TIMESTAMP is always 26.
2. If the data type is suspect, create a query that finds one row and PRINT the output.

Note: For details about data types, see SQL Statements (see page 515) or see the *CA Datacom/DB SQL User Guide*.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> DISPLAY KEYS

Displays the COLUMN DISPLAY panel (DQB20).

<PF6> TEXT

Displays text description.

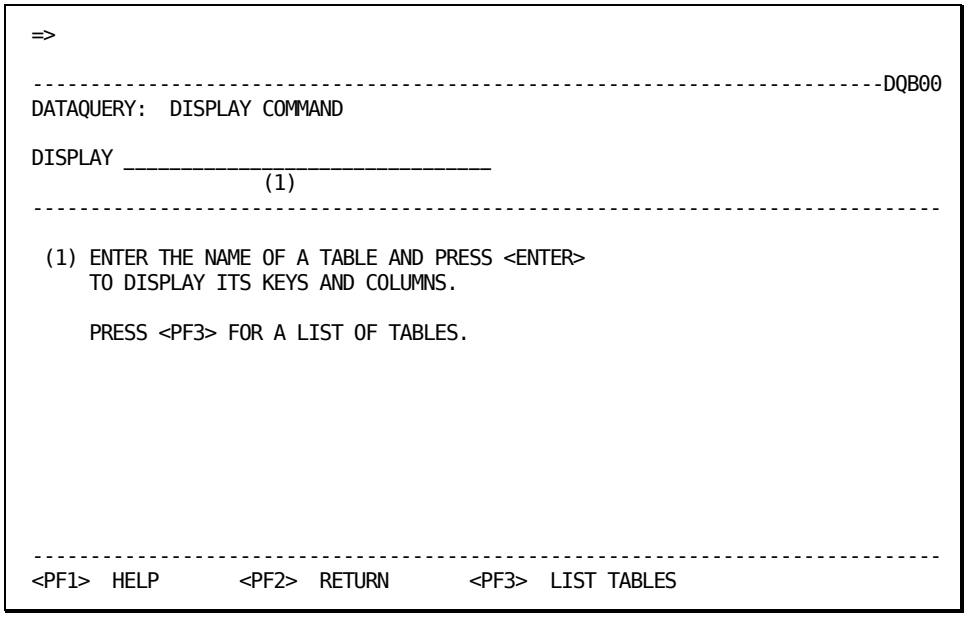
<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQB00) DISPLAY Command



Description

This panel prompts you for a table name. You get this panel by entering DISPLAY in the command line without a table name following the command.

Action

To select a table, either enter a valid table name, or press <PF3> to display a list of table names and then type **DISPLAY table-name** in the command line.

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
(1)	Y	A valid table name	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> LIST TABLES

Displays a list of all database tables you are authorized to access.

(DQB10) Keys and Columns Display

```
=>
PLACE THE CURSOR ON THE DESIRED NAME AND PRESS THE APPROPRIATE PFKEY.
-------------------------------------------------------------------------------DQB10
DATAQUERY:  KEYS AND COLUMNS DISPLAY               DICTIONARY DATABASE ID:
-------------------------------------------------------------------------------
 TABLE NAME:                                       DB NAME:       DB ID:
-------------------------------------------------------------------------------
  KEY NAMES                                       |  DESCRIPTION
      COLUMN NAMES                                |
-------------------------------------------------------------------------------
key-name                                          |
high-order-column-name                            |
key-name                                          |
key-name                                          |
    column-name                                   |
    column-name                                   |
    column-name                                   |
                                                  |
                                                  |
                                                  |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2>  RETURN    <PF3> DISPLAY COLUMNS   <PF4> DISPLAY KEYS
<PF5> NOT USED      <PF6>  TEXT      <PF7> BACKWARD          <PF8> FORWARD
```

Description

This panel displays the names and descriptions of keys and columns. You can get this panel by using the DISPLAY ALL PF key on panels where it is shown as an option. You can also type **DISPLAY table-name** in the command line to display this panel.

Action

You can view names of keys and columns for a specific table. To find a key name, use a name that is not indented. To find a column name, choose an indented object that is not also un-indented.

Display Fields

Refer to the following for explanations of the display fields at the top of this panel that can contain variable data.

DICTIONARY DATABASE ID:

The DATACOM-ID of the current Datadictionary database containing the definitions for this table.

TABLE NAME:

Name of the table whose key and column names are displayed on this panel.

DB NAME:

The DATACOM-ID of the table whose key and column names are displayed on this panel.

DB ID:

The DATACOM-ID of the database containing the table whose key and column names are displayed on this panel.

Columns

Refer to the following for a discussion of each column on this panel.

KEY NAMES/ COLUMN NAMES

Lists key and first column name of each key. Other column names are indented.

DESCRIPTION

Lists description, if any, of key or column named at left.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> DISPLAY COLUMNS

Displays the COLUMN DISPLAY panel (DQB20).

<PF4> DISPLAY KEYS

Displays the KEY DISPLAY panel (DQB50).

<PF6> TEXT

Displays the TEXT DISPLAY (Key) panel (DQB70) for the selected key or column.

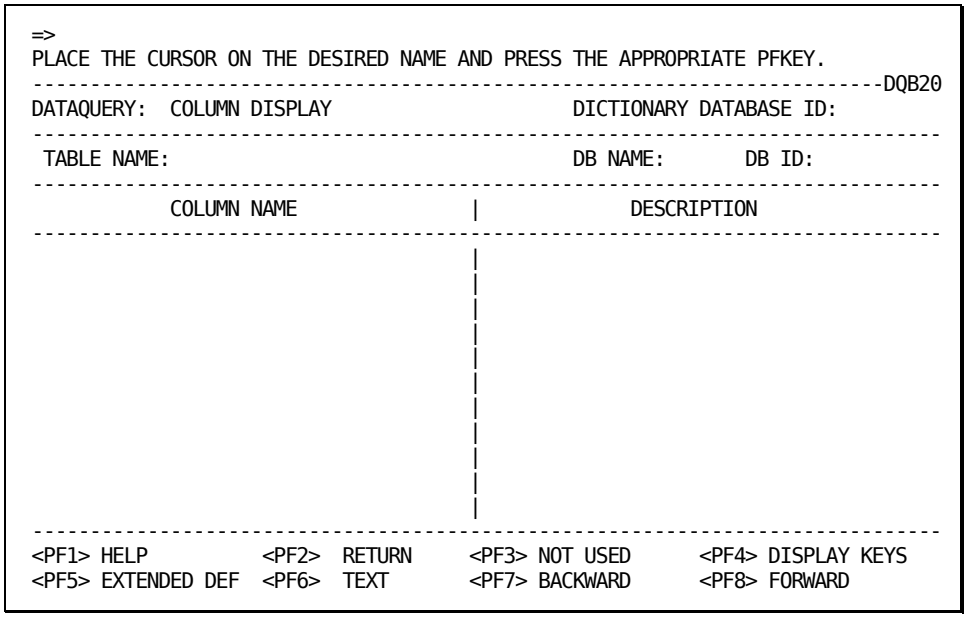
<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQB20) Column Display



Description

This panel displays the names and descriptions of columns. You can get this panel by using the DISPLAY COLUMNS PF key on any panel where it is shown as an option. This is not the same panel which is displayed when you choose the DISPLAY COLUMNS option from the CA Dataquery Editor. That panel is COLUMN DISPLAY (DQB90).

Action

You can view the names of the columns or select a column by cursor position. Use the DISPLAY table-name command if you want to know the name of the high-order column.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

DICTIONARY DATABASE ID:

The DATACOM-ID of the current Datadictionary database containing the definitions for this table.

TABLE NAME:

Name of the table whose key and column names are displayed on this panel.

DB NAME:

The DATACOM-ID of the table whose key and column names are displayed on this panel.

DB ID:

The DATACOM-ID of the database containing the table whose key and column names are displayed on this panel.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Lists column names for the table.

DESCRIPTION

Lists description, if any, of column named at left.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> DISPLAY KEYS

Displays the KEY DISPLAY panel.

<PF5> EXTENDED DEF

Displays the EXTENDED DEFINITION panel.

<PF6> TEXT

Displays text panel for the selected column.

<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQB40) Text Display (Column)

```
=>
USE PF7/PF8 TO SCROLL THE TEXT IF THERE ARE MULTIPLE PAGES, USE PF2 TO RETURN
--------------------------------------------------------------------------------DQB40
DATAQUERY:  TEXT DISPLAY
--------------------------------------------------------------------------------
COLUMN:                                  TABLE:
--------------------------------------------------------------------------------
ALTERNATE HEADING:                          PICTURE:


--------------------------------------------------------------------------------









--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN      <PF3>  NOT USED     <PF4>  NOT USED
<PF5>  NOT USED   <PF6>  NOT USED    <PF7>  BACKWARD     <PF8>  FORWARD
```

Description

This panel displays the text description (if any exists) for a column. You can get this panel by using the TEXT PF key on the COLUMN DISPLAY or EXTENDED COLUMN DISPLAY panels.

Action

View the alternate heading and edit pattern, if any, for the selected column and press a PF key to continue processing.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

COLUMN:

Name of the column selected for display.

TABLE:

Name of the table containing the column selected for display.

ALTERNATE HEADING:

An alternate name for the column.

PICTURE:

The assigned edit pattern that controls how numbers are displayed, if any.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

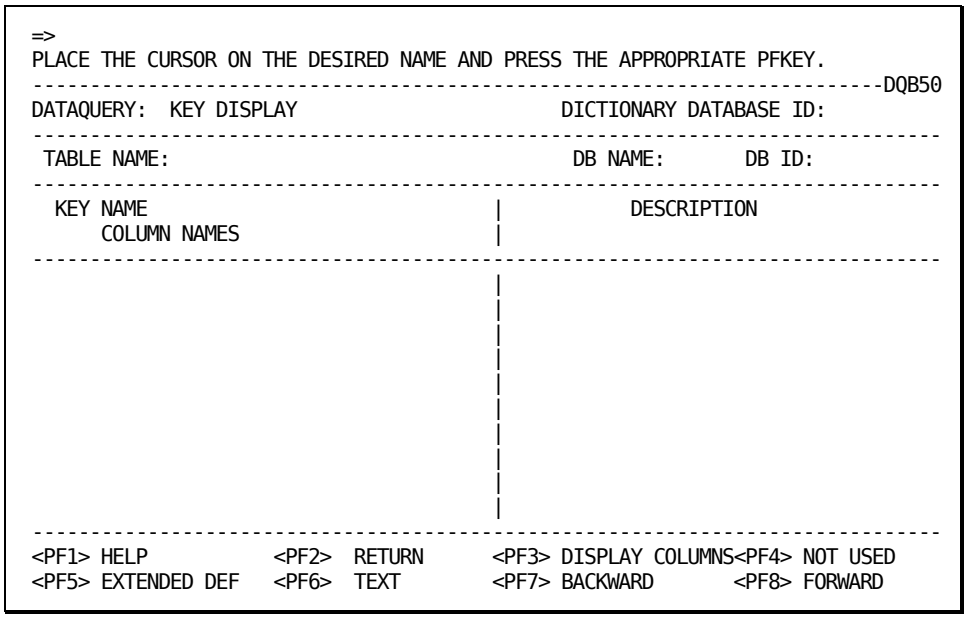
<PF7> BACKWARD

Scrolls the text backward.

<PF8> FORWARD

Scrolls the text forward.

(DQB50) Key Display



Description

This panel displays the names and descriptions of keys and the columns in each key. You get this panel by using the DISPLAY KEYS PF key on any panel where it is shown as an option.

Action

You can view the name of the key, then position the cursor on a key or column and press a PF key to continue.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

DICTIONARY DATABASE ID:

The DATACOM-ID of the current Datadictionary database containing the definitions for this table.

TABLE NAME:

Name of the table whose key and column names are displayed on this panel.

DB NAME:

The DATACOM-ID of the table whose key and column names are displayed on this panel.

DB ID:

The DATACOM-ID of the database containing the table whose key and column names are displayed on this panel.

Columns

Refer to the following for a discussion of each column on this panel.

KEY NAME/ COLUMN NAMES

Lists key and column names for the table. Column names are indented.

DESCRIPTION

Lists description, if any, of key or column named at left.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> DISPLAY COLUMNS

Displays the COLUMN DISPLAY panel.

<PF5> EXTENDED DEF

Displays the EXTENDED DISPLAY panel for the selected key or column.

<PF6> TEXT

Displays text description for the selected key or column.

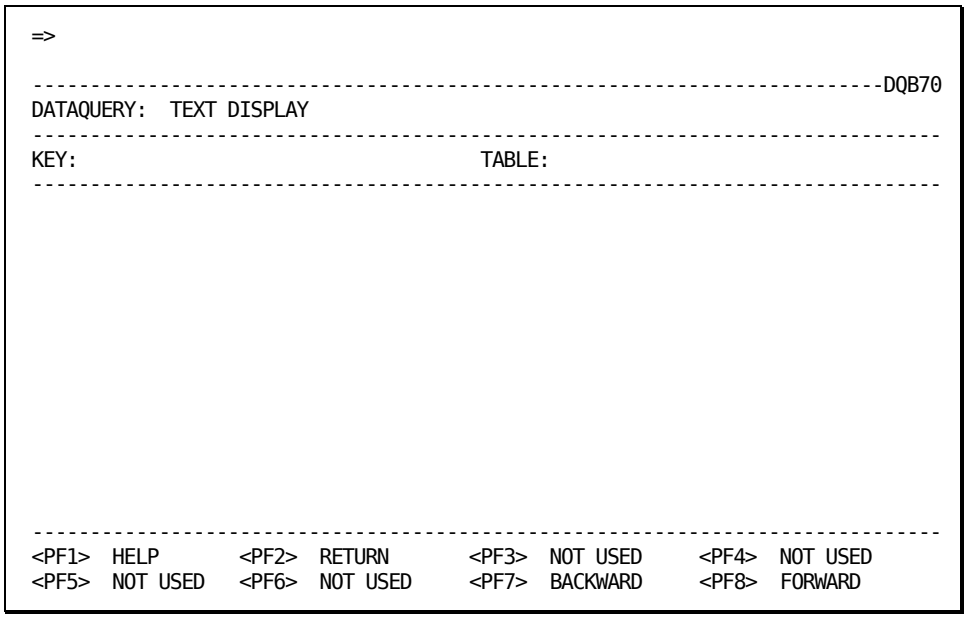
<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQB70) Text Display (Key)



Description

This panel displays the text description (if any exists) for a key. You can get this panel by using the TEXT PF key on any panel where it is shown as an option.

Action

You can view the text, if any, for the selected key.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

KEY:

Name for the key whose text display you requested.

TABLE:

Name for the table whose key text display you requested.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

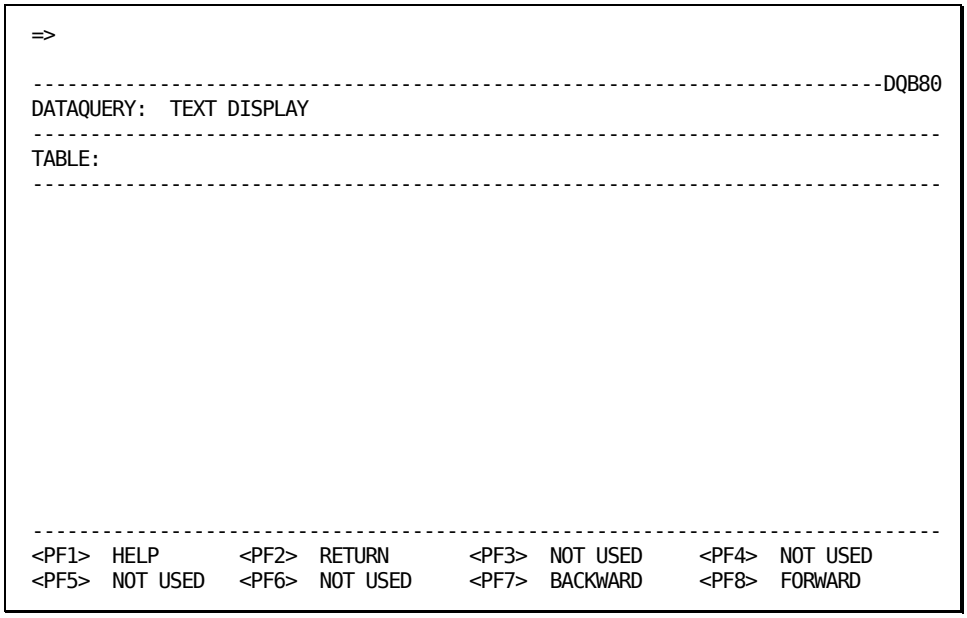
<PF7> BACKWARD

Scrolls the text backward.

<PF8> FORWARD

Scrolls the text forward.

(DQB80) Text Display (Table)



Description

This panel displays the text description, if available, for a database table. You can get this panel by using the TEXT PF key on the LIST TABLES panel.

Action

View the text description for the selected table and then press <PF2> RETURN to resume processing.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

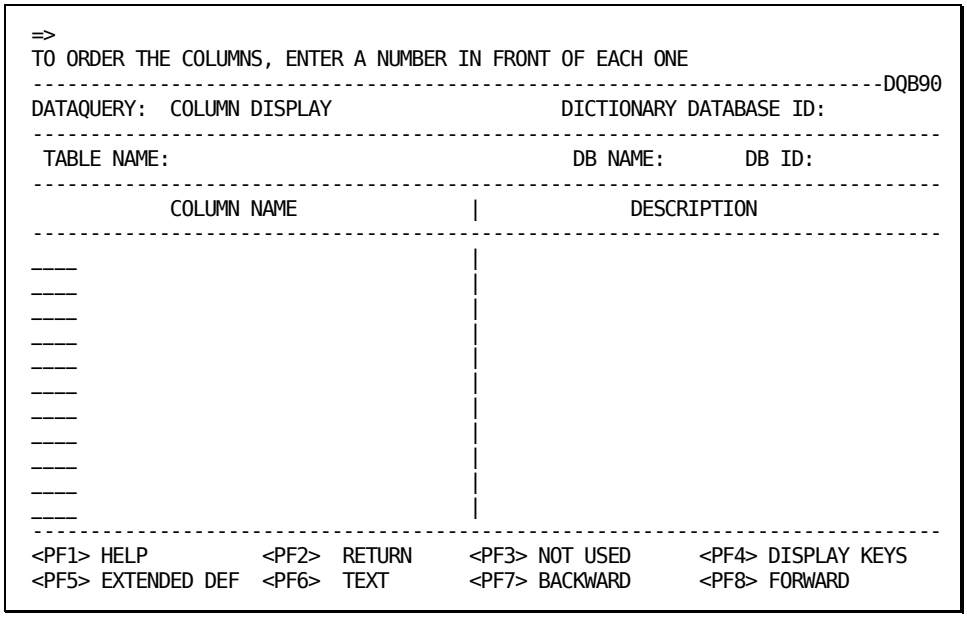
<PF7> BACKWARD

Scrolls the text backward.

<PF8> FORWARD

Scrolls the text forward.

(DQB90) Column Display (from Editor)



Description

This panel is displayed when you press the DISPLAY COLUMNS PF key from the Editor panel. This panel allows input or numbers to the left of the displayed column names. The numbers you enter indicate that the column name is to be copied back to the query text being edited and the desired order for the selected columns within the query text.

Anyone authorized to edit a query can use this panel.

Action

Enter a number to indicate list order in the selection column for each column name you want displayed on the Editor panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

DICTIONARY DATABASE ID:

The DATACOM-ID of the current Datadictionary database containing the definitions for this table.

TABLE NAME:

Name of the table whose column names are displayed on this panel.

DB NAME:

The DATACOM-ID of the table whose column names are displayed on this panel.

DB ID:

The DATACOM-ID of the database containing the table whose column names are displayed on this panel.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Column names for the table (indented).

DESCRIPTION

Lists the description, if any, of key or column named at left.

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
___	N	A one- or two-digit number to show the order of appearance.	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Editor panel with any selected column names.

<PF4> DISPLAY KEYS

Displays keys for the current table.

<PF5> EXTENDED DEF

Displays the extended definitions of listed columns.

<PF6> TEXT

Displays query text.

<PF7> BACKWARD

Scrolls to previous panel, if any.

<PF8> FORWARD

Scrolls to next panel, if any.

(DQDA0) Directory of Tables (SQL Mode)

```
=>
Place the cursor on the desired name and press the appropriate PFkey
-------------------------------------------------------------------------------DQDA0
DATAQUERY:  DIRECTORY OF TABLES   START WITH: _____________________________
-------------------------------------------------------------------------------
      OBJECT NAME                 |     DESCRIPTION
-------------------------------------------------------------------------------
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
                                  |
-------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN       <PF3> DISPLAY COLUMNS <PF4> NOT USED
<PF5> NOT USED     <PF6> DISPLAY TEXT <PF7> BACKWARD        <PF8> FORWARD
```

Description

This SQL Mode panel displays a list of database tables you are authorized to access with a query or dialog.

Action

View the list to determine which tables you want to search with your query or dialog. To display a list of columns or the description of any table, position the cursor on its name and press the appropriate PF key.

Entry Fields

Refer to the following for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	First 1-15 letters of an item name to move to the top of the display. Name may be partial or complete.	No default

Columns

Refer to the following for a discussion of each column on this panel.

OBJECT NAME

Lists tables you are authorized to access.

DESCRIPTION

A description of the table.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY COLUMNS

Displays a list of columns for the table indicated by cursor position.

<PF6> DISPLAY TEXT

Displays the textual description for the table indicated by cursor position.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQDL0) Define Variable List

```
=>
ENTER THE VALID VALUES FOR THE VARIABLE BELOW
--------------------------------------------------------------------------------DQDL0
DATAQUERY:  DEFINE VARIABLE LIST
--------------------------------------------------------------------------------
NAME
VARIABLE ID:              ?
PROMPT FOR VARIABLE:
--------------------------------------------------------------------------------









--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  DELETE      <PF4>  SAVE
<PF5>  NOT USED     <PF6>  NOT USED     <PF7>  BACKWARD    <PF8>  FORWARD
```

Description

The optional Define Variable List panel permits you to provide a list of valid substitutes for a character or alphabetic dialog or JCL PROC variable that you are creating. Twenty-five fields are available for listing substitutes for the current variable definition. An overview on completing this panel appears in the *CA Dataquery User Guide*.

Action

Enter a list of valid values for this variable, including the default for the variable. Save the list and return to the Define Variable panel.

Sample

The following sample shows a typical Define Variable List panel containing information in the display fields. The entry fields' lengths are identical to the default variable length, although the values you enter need not be the full length.

```
=>
ENTER THE VALID VALUES FOR THE VARIABLE BELOW
-------------------------------------------------------------------------------DQDL0
DATAQUERY:  DEFINE VARIABLE LIST
--------------------------------------------------------------------------------
NAME                    AAA-SAMP-DIALOG
VARIABLE ID:            1 ?
PROMPT FOR VARIABLE:
TYPE THE NAME OF THE CITY FOR WHICH YOU WANT A COMPLETE LIST OF CUSTOMERS


'DALLAS         '
_________________
_________________
_________________
_________________
_________________
_________________
_________________
_________________
--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  DELETE      <PF4>  SAVE
<PF5>  NOT USED     <PF6>  NOT USED     <PF7>  BACKWARD    <PF8>  FORWARD
```

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

NAME

Name of the dialog or JCL PROC.

VARIABLE ID:

Number you assigned to this variable, followed by the site dialog symbol (a question mark in the example).

PROMPT FOR VARIABLE:

Prompt you wrote to appear when this variable is shown on the prompt panel during dialog/PROC execution. Maximum length is 77 characters.

default variable

The default appears after the prompt. (DALLAS is the prompt in this sample.) Maximum length is 77 characters.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
_____	N	All valid values using any character except underscores and using apostrophes where required. If alphabetic type, only letters A-Z are allowed.	Site-specific default variable

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Define Variable panel.

<PF3> DELETE

Deletes the list definition.

<PF4> SAVE

Saves the list definition.

<PF7> BACKWARD

Scrolls the variable list backward.

<PF8> FORWARD

Scrolls the variable list forward.

(DQDP0) Define Descriptive Text

```
=>

--------------------------------------------------------------------------------DQDP0
DATAQUERY:    DEFINE DESCRIPTIVE TEXT        NAME:
--------------------------------------------------------------------------------
DESCRIPTION:
STATUS:
--------------------------------------------------------------------------------
ENTER THE DESCRIPTIVE TEXT TO BE USED AS A TITLE AND SAVE IT

_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________









--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN      <PF3> CONTINUE        <PF4> SAVE
<PF5> NOT USED    <PF6> NOT USED    <PF7> LAST VARIABLE   <PF8> FIRST VARIABLE
```

Description

The Define Descriptive Text panel allows you to explain the purpose and operation of the dialog or JCL PROC you create. This purpose and description will appear on the prompt panel when execution is initiated. Four lines are available for entering a complete description. An overview on using this panel appears in the *CA Dataquery User Guide*.

Action

Type a description that tells a prospective user anything needed to make a decision about using this dialog or JCL PROC. Save the description with <PF4> or save the description and go on to the next definition panel with <PF3>.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

NAME:

Name you assigned the dialog or JCL PROC on the Editor panel.

DESCRIPTION:

Description you assigned the dialog or JCL PROC on the Editor panel.

STATUS:

- COMPLETE - The description and all variable definitions have been saved for this dialog or JCL PROC.
- INCOMPLETE - The description or a variable definition has not been saved for this dialog or JCL PROC, or there are definitions saved for variables that have been changed or removed from the Editor panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Four blank lines	Y	Any characters except underscores.	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the CA Dataquery Editor.

<PF3> CONTINUE

Saves text entered and continues to next definition panel.

<PF4> SAVE

Saves text.

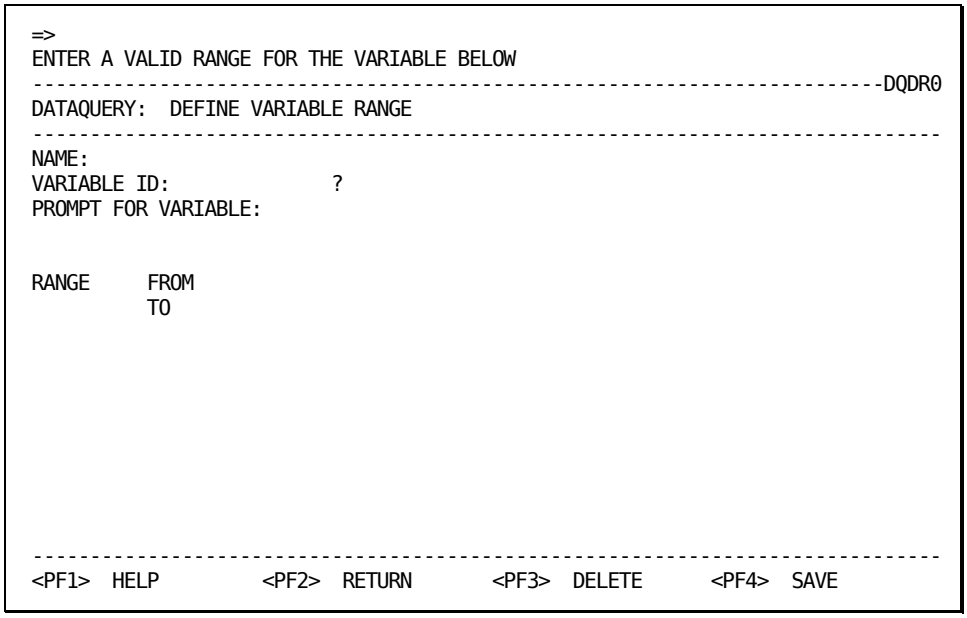
<PF7> LAST VARIABLE

Displays Define Variable (DV0) panel for last variable appearing in the dialog or JCL PROC.

<PF8> FIRST VARIABLE

Displays the Define Variable (DV0) panel for the first variable appearing in the dialog or JCL PROC.

(DQDR0) Define Variable Range



Description

The optional Define Variable Range panel appears after you save a dialog or JCL PROC variable whose type you defined as numeric and press <PF5> RANGE/LIST. It allows you to describe the lower and upper values of a range of numbers that can be entered as substitutes for the variable. An overview of defining this panel appears in the *CA Dataquery User Guide*.

Action

Enter numbers for the low and high ends of the valid range you want to define for this variable. You can enter up to 18 digits, a sign (+ or -_), and a decimal point. Save the range definition and return to the Define Variable panel for this variable.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
FROM	Y	Low value	0
TO	Y	High value	0

Sample

The following sample shows you how a typical panel looks when it contains information in the display fields.

```
=>
ENTER A VALID RANGE FOR THE VARIABLE BELOW
--------------------------------------------------------------------------------DQDR0
DATAQUERY:  DEFINE VARIABLE RANGE
--------------------------------------------------------------------------------
NAME:                   AAA-SAMP-DIALOG
VARIABLE ID:            2 ?
PROMPT FOR VARIABLE:
ENTER A PRODUCT CODE IN THE DEFINED RANGE OR ACCEPT THE 75042 DEFAULT.

RANGE     FROM        75000
          TO          75999









--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  DELETE      <PF4>  SAVE
```

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

NAME:

Name you gave this dialog or JCL PROC.

VARIABLE ID:

The two-digit number you entered to identify this variable on the Editor panel.

PROMPT FOR VARIABLE:

Prompt assigned to the variable with the Define Variable panel.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Define Variable (DQDV0) panel for this variable.

<PF3> DELETE

Deletes a saved range.

<PF4> SAVE

Saves the range definition.

(DQDV0) Define Variable

```
=>
ENTER THE REQUIRED DATA AND PRESS THE APPROPRIATE PFKEY.
--------------------------------------------------------------------------------DQDV0
DATAQUERY:  DEFINE VARIABLE                    NAME:
--------------------------------------------------------------------------------
DESCRIPTION:
STATUS:




VARIABLE ID:               ?
ENTER THE PROMPT TO BE DISPLAYED FOR THIS VARIABLE:
_______________________________________________________________________________
VARIABLE TYPE:            _    C - CHARACTER    N - NUMERIC    A - ALPHABETIC
VARIABLE LENGTH:
VARIABLE DEFAULT VALUE:
_______________________________________________________________________________

--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> CONTINUE       <PF4> SAVE
<PF5> RANGE/LIST  <PF6> DELETE       <PF7> PREV VARIABLE  <PF8> NEXT VARIABLE
```

Description

The required Define Variable panel appears for each variable you enter on the Editor panel. It contains information about the variable you entered and provides two fields that you must complete: a prompt that appears during execution and the data type code that tells CA Dataquery what kind of data will appear as a result of executing the query with this variable. An overview on completing this panel appears in the *CA Dataquery User Guide*.

Action

Type a prompt that appears when this variable can be substituted during execution or validation. Choose a data type you want to allow for the substitute. Save the definition and go on to the next panel.

Sample

The following sample shows how a typical Define Variable panel appears containing information in the display fields.

```
=>
Enter the required data and press the appropriate PFkey.
------------------------------------------------------------------------------DQDV0
DATAQUERY:  DEFINE VARIABLE                      NAME:   AAA-SAMP-DIALOG
------------------------------------------------------------------------------
DESCRIPTION:      SELECT CUSTOMER BY CITY
STATUS:           COMPLETE

WITH CITY = 1?'DALLAS         '
WITH PROD = 2?75042
PRINT       TITLE1 'SAMPLE ORDER ENTRY DATABASE'

VARIABLE ID:             2 ?
ENTER THE PROMPT TO BE DISPLAYED FOR THIS VARIABLE:
_______________________________________________________________________________
VARIABLE TYPE:           _     C - CHARACTER    N - NUMERIC    A - ALPHABETIC
VARIABLE LENGTH:         05
VARIABLE DEFAULT VALUE:
75042


------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> CONTINUE       <PF4> SAVE
<PF5> RANGE/LIST  <PF6> DELETE       <PF7> PREV VARIABLE  <PF8> NEXT VARIABLE
```

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

NAME:

Name you assigned the dialog or JCL PROC on the Editor panel.

DESCRIPTION:

Description you assigned the dialog or JCL PROC on the Editor panel.

STATUS:

Status options are:

- COMPLETE - The description and all variable definitions have been saved for this dialog or JCL PROC.
- INCOMPLETE - The description or a variable definition has not been saved for this dialog or JCL PROC, or there are definitions saved for variables that have been changed or removed from the Editor panel. Below the Status field, you see a replica of the variables as they appear on the Editor panel.

VARIABLE ID:

Two-digit ID assigned to the variable on the Editor panel.

VARIABLE LENGTH:

Length of the default plus any dialog fill characters. This length may differ from the actual variable length if you have changed the variable on the Editor panel since the last time you saved this variable's definition. If so, delete the definition so that this display field contains the correct information. The maximum length of the variable is 77 characters.

VARIABLE DEFAULT VALUE:

Characters entered on the Editor panel following the two-digit variable ID and dialog symbol and excluding any dialog fill characters.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Enter the prompt to be displayed for this variable:	Y	Any characters	No default
VARIABLE TYPE:	Y	C - CHARACTER N - NUMERIC A - ALPHABETIC	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the CA Dataquery Editor (DQD10).

<PF3> CONTINUE

Saves this definition and continues to the next step.

<PF4> SAVE

Saves this definition.

<PF5> RANGE/LIST

Displays Define Variable Range panel if variable is numeric, Define Variable List panel if variable is character type.

<PF6> DELETE

Deletes any saved definition for this variable.

<PF7> PREV VARIABLE

Displays the Definition panel for the previous variable.

<PF8> NEXT VARIABLE

Saves the Definition panel for the next variable.

(DQD00) EDIT Command

```
=>

--------------------------------------------------------------------------------DQD00
DATAQUERY:  EDIT COMMAND

EDIT ACTIVE-QUERY
           (1)
--------------------------------------------------------------------------------

 (1) ENTER 'ACTIVE-QUERY' TO DISPLAY QUERY CURRENTLY IN USE
     OR ENTER THE NAME OF ANY QUERY, TERM, OR DIALOG.
     PRESS <ENTER> TO DISPLAY IT FOR EDITING.

     PRESS <PF3> FOR A LIST OF QUERIES.









--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> LIST QUERIES
```

Description

Use EDIT when you want to copy any query you can access or change any query you created. You can enter EDIT * to change the active query. If you enter EDIT without an operand and press Enter, CA Dataquery presents the Edit Command prompt panel. From this panel, you can see a list of all queries, dialogs and terms that you can access. See Operation Commands for details on using the Edit Command.

Action

Enter the name of the item you want to edit or press <PF3> to display a list from which you can select the name of the item you want to display on the CA Dataquery Editor panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
(1)	Y	Name of existing authorized query, dialog, or term.	ACTIVE-QUERY

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

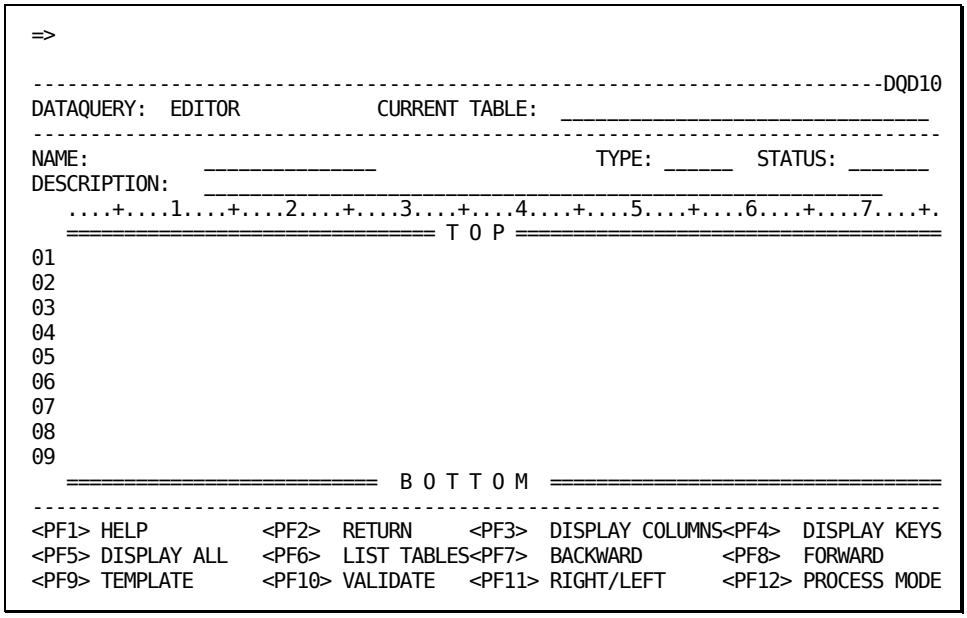
<PF2> RETURN

Exits panel.

<PF3> LIST QUERIES

Displays a list of queries, terms, and dialogs.

(DQD10) Editor



Description

The Editor Panel appears when you select CREATE from the Main Menu, when you press an EDIT PF key during directory display, or when you use the EDIT command to display an existing query, dialog, or term. When you select CREATE (or enter the CREATE command), the Editor panel appears with the Create Mode PF keys displayed at the bottom.

When you edit (with a PF key or EDIT command), the Editor panel appears with the Process Mode PF keys displayed at the bottom and the query, dialog, or term displayed in the editing area. Use the Editor Panel to create, copy or modify a query, term, or dialog. (Users with CA Dataquery Administrator authorization also use it to create, copy or modify JCL and other member types.)

The CA Dataquery Editor is available to you for typing, inserting, moving and deleting lines between the TOP and BOTTOM lines.

In Create Mode, the PF keys allow you to display helpful information or switch to Process Mode where you use the PF keys to save, validate, process, or delete the current query or term, or to use a query as a query dialog. You can change modes with <PF12>.

Sample

Complete the fields at the top of this panel. Then write or change a query or a term in the area enclosed by the TOP and BOTTOM lines. You can only change queries, dialogs, and terms that you own. However, you can copy an existing public query by editing and renaming it. To create or edit a query, type DQL or press <PF9> TEMPLATE in Create Mode to select statement templates for your query and type valid values in place of template text (like ROW-NAME).

See Editor Commands for detailed instructions on using the Editor commands. See the chapters on query syntax for detailed instructions on using each DQL statement. Save your entries in Process Mode by using <PF4> SAVE or <PF9> UPDATE if you are changing an existing member.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
CURRENT TABLE:	N	Name of a table you can access	No default
NAME:	Y	Any characters except blanks	No default
TYPE:	Y	Query, Dialog, or Term	Query
STATUS:	Y	Private or Public	Private
DESCRIPTION:	N	1 to 60 characters	No default
Line numbers	N	Editor line commands See Editor Commands.	No default

CA Dataquery Administrators can enter other types in the TYPE field. When they do, a different PF key menu appears from the one shown in this document.

Note: For details on using the Editor to create JCL and other members, see the *CA Dataquery Administrator Guide*.

PF Keys Create Mode

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> DISPLAY COLUMNS

Displays a list of all columns for the current table.

<PF4> DISPLAY KEYS

Displays a list of all keys for the current table.

<PF5> DISPLAY ALL

Displays a list of all keys and columns in the current table.

<PF6> LIST TABLES

Displays a list of all tables your query can access.

<PF7> BACKWARD

Displays the previous panel of a partially shown panel.

<PF8> FORWARD

Displays the next panel of a partially shown panel.

<PF9> TEMPLATE

Displays template for selecting CA Dataquery language statements.

<PF10> VALIDATE

Screens the current entries on the Editor panel for errors.

<PF11> RIGHT/LEFT

Moves current panel 4 columns left or right on 80-column terminal screens.

<PF12> PROCESS MODE

Displays PF keys for processing query or term.

PF Keys Process Mode

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays information about the current panel.

<PF2>RETURN

Redisplays previous panel.

<PF3> EXECUTE

Executes the query online.

<PF4> SAVE

Saves the current query or term in the appropriate library.

<PF5> DIALOG DEF

Defines dialog variables if variables have been entered and TYPE is *dialog*.

<PF6> DELETE

Deletes the last version saved, if you own the member.

<PF7> BACKWARD

Displays the previous panel of a partially shown panel.

<PF8> FORWARD

Displays the next panel of a partially shown panel.

<PF9> UPDATE

Applies any changes made to the member, if you own it.

<PF10> VALIDATE

Has CA Dataquery check the entries for errors.

<PF11> RIGHT/LEFT

Moves the panel 4 columns to the left or right, if an 80-column terminal screen.

<PF12> CREATE MODE

Displays PF keys for displaying table information or template of DQL statements.

(DQD30) Query Template

```
=>
Enter the number of times each statement is to be included in the query
-------------------------------------------------------------------------------DQD30
DATAQUERY:  QUERY TEMPLATE
--------------------------------------------------------------------------------
Lower case words represent entries which must be replaced by the user.
Parentheses indicate optional entries: these may be included by removing the
parentheses only or deleted by removing both the parentheses and data.
--------------------------------------------------------------------------------
_  FIND (count)  table-name     ROWS
_  WITH selection-criteria
_  RELATED BY key-name
       TO (FIRST) table-name
_  RELATED BY link-column
       VIA key-name column-name TO table-name
_  SET result         (13.2)
       = numeric-column-name arithmetic-expression numeric-literal
_  SORT table-name        BY column-name          (DOWN)


--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  NOT USED    <PF4>  NOT USED
<PF5>  NOT USED     <PF6>  NOT USED     <PF7>  BACKWARD    <PF8>  FORWARD
```

Description

The Query Template panel presents a list of CA Dataquery statements and syntax in the order in which they should appear in a query. Each statement contains DQL words in uppercase letters (like FIND and WIDTH) and lowercase letters (like table-name) to indicate where you need to provide information. You can copy one or more statements to the Editor or simply use the panel as a reference for query syntax.

On a terminal screen with 43 lines, the remainder of the statement templates available to you also appear on this panel. On a terminal screen with fewer lines, the remaining statement templates appear on DQD40 when you use <PF8> FORWARD while displaying this panel.

Action

Enter a number or a character in the field next to any statements you want to use in your query. The number represents the number of times you want the statement template to appear on the Editor panel in the position indicated by the cursor when you press <PF9> TEMPLATE during Editor display. Entering a character means you want to return to the Editor panel with one copy of the statement template. You can select all statements at once, return to the Query Template panel to copy one or more statements at a time, or return without copying any statements.

Scroll forward with <PF8> FORWARD to see additional statement templates if your terminal screen is smaller than 43 lines.

Template Text

Each statement template contains DQL Language words in uppercase letters (like FIND and WIDTH) and words you type over on the Editor panel in lowercase letters (like (count) and table-name).

Entry Fields

Refer to the following chart for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
__	N	A number showing the number of times to appear on the Editor or any character to select once.	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Editor panel with any selected statements.

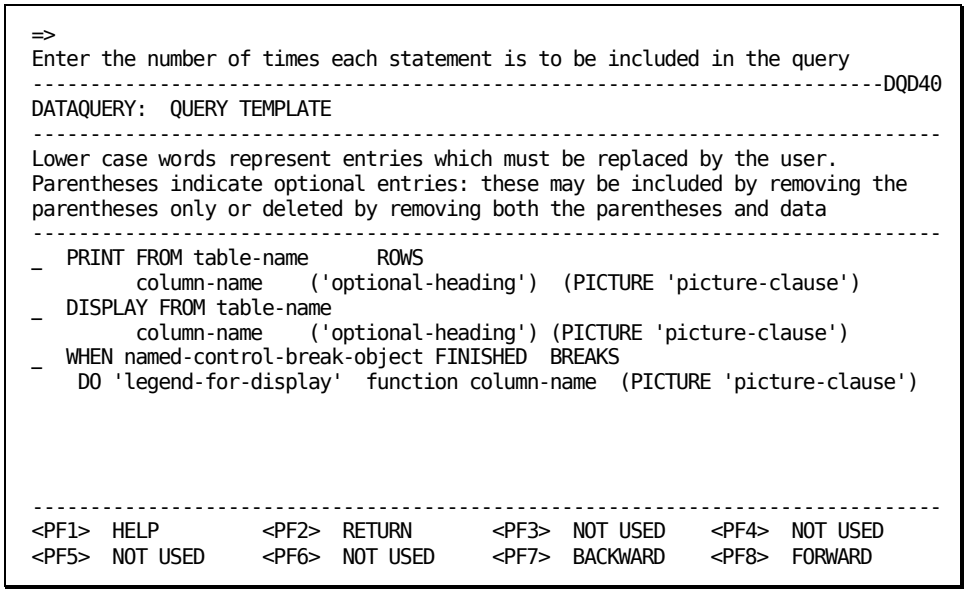
<PF7> BACKWARD

Scrolls backward one panel.

<PF8> FORWARD

Scrolls remaining statement templates (DQD40) if terminal screen is shorter than 43 lines.

(DQD40) Query Template



Description

This continuation of the query template appearing on DQD30 presents a list of remaining CA Dataquery statements and syntax in the order in which they should appear in a query (following the statement templates on DQD30). Each statement contains DQL words in uppercase letters (like PRINT and FROM) and words in lowercase letters (like table-name) that you type over. You can copy one or more statements to the Editor or simply use the panel as a reference for query syntax.

Action

Enter a number in the field next to any statements you want to use in your query. The number represents the number of times you want the statement template to appear on the Editor panel in the position indicated by the cursor when you press <PF9> TEMPLATE during Editor display. You can select all statements at once, return to the Query Template panel to copy one or more statements at a time, or return without copying any statements.

Scroll backward with <PF7> BACKWARD to see previous statement templates on DQD30.

Template Text

Each statement template contains DQL Language words in uppercase letters (like PRINT and FROM) and words you type over on the Editor panel in lowercase letters (like table-name).

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
_____	N	A number showing the number of times to appear on the Editor, or any character to select one copy.	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Editor panel with any selected statements.

<PF7> BACKWARD

Views previous statement templates (DQD30) if terminal screen is shorter than 43 lines.

<PF8> FORWARD

Scrolls forward to see additional templates if terminal screen is shorter than 43 lines.

(DQD50) Query Template (SQL Mode)

```
=>
Place a nonblank beside each statement that is to be included in the query
-------------------------------------------------------------------------------DQD50
DATAQUERY:  QUERY TEMPLATE
--------------------------------------------------------------------------------
Lower case words represent entries which must be replaced by the user.
Parentheses indicate optional entries: these may be included by removing the
parentheses only or deleted by removing both the parentheses and data.
--------------------------------------------------------------------------------
_  SELECT column1, column2, count(column3)
_  FROM table1, table2
_  WHERE table1.column1 = table2.column2 and column3 > 0
_  GROUP BY column1, column2
_  ORDER BY column1









--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  NOT USED    <PF4>  NOT USED
```

Description

The SQL Mode Query Template panel presents a list of SQL syntax in the order in which they should appear in a query. You can copy one or more statement templates to the Editor or simply use the panel as a reference for query syntax.

On a terminal screen with 43 lines, the remainder of the statement templates available to you also appear on this panel. On a terminal screen with fewer lines, the remaining statement templates appear on DQD40 when you use <PF8> FORWARD while displaying this panel.

Action

Enter a number or a character in the field next to any statements you want to use in your query. The number represents the number of times you want the statement template to appear on the Editor panel in the position indicated by the cursor when you pressed <PF9> TEMPLATE during Editor display. Entering a character means you want to return to the Editor panel with one copy of the statement template. You can select all statements at once, return to the Query Template panel to copy one or more statements at a time, or return without copying any statements.

Scroll forward with <PF8> FORWARD to see additional statement templates if your terminal screen is smaller than 43 lines.

Template Text

Each statement template contains DQL words in uppercase letters (like SELECT and FROM) and words you type over on the Editor panel in lowercase letters (like column1).

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
__	N	A number showing number of times to appear on Editor or any character to select once.	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

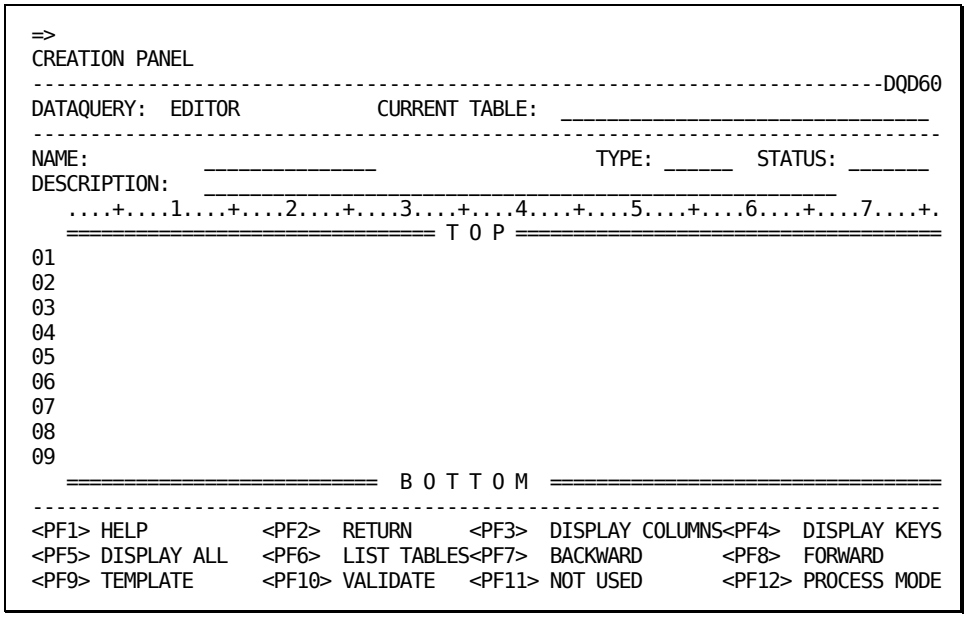
<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Editor panel with any selected statements.

(DQD60) Editor (For Wide Screens)



Description

The Editor Panel appears when you select CREATE from the Main Menu, when you press an EDIT PF key during directory display, or when you use the EDIT command to display an existing query, dialog, or term. When you select CREATE (or enter the CREATE command), the Editor panel appears with the Create Mode PF keys displayed at the bottom.

When you edit (with a PF key or EDIT command), the Editor panel appears with the Process Mode PF keys displayed at the bottom and the query, dialog, or term displayed in the editing area. Use the Editor Panel to create, copy or modify a query, term, or dialog. (Users with CA Dataquery Administrator authorization also use it to create, copy or modify conditions and JCL.)

The CA Dataquery Editor is available to you for typing, inserting, moving and deleting lines between the TOP and BOTTOM lines.

In Create Mode, the PF keys allow you to display helpful information or switch to Process Mode where you use the PF keys to save, validate, process, or delete the current query or term, or to use a query as a query dialog. You can change modes with <PF12>.

Sample

Complete the fields at the top of this panel. Then write or change a query or a term in the area enclosed by the TOP and BOTTOM lines. You may change only queries, dialogs, and terms that you own. However, you can copy an existing public query by editing and renaming it.

To create or edit a query, type DQL or press <PF9> TEMPLATE in Create Mode to select statement templates for your query and type valid values in place of template text (like ROW-NAME).

See Editor Commands for detailed instructions on using the Editor commands. See the chapters on query syntax for detailed instructions on using each DQL statement.

Save your entries in Process Mode by using <PF4> SAVE or <PF9> UPDATE if changing an existing member.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
CURRENT TABLE:	N	Name of a table you can access.	No default
NAME:	Y	1 to 15 characters beginning with a letter.	No default
TYPE:	Y	Query, Term, or Dialog. CA Dataquery Administrators can enter other types in the TYPE field. When they do, a different PF key menu appears from the one shown in this document. For details on using the Editor to create JCL and other members, see the *CA Dataquery Administrator Guide*.	No default
STATUS:	Y	Private or Public.	Private
DESCRIPTION:	N	1 to 60 characters.	No default
Line numbers	N	Editor line commands. See Editor Commands.	No default

PF Keys Create Mode

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> DISPLAY COLUMNS

Displays a list of all columns for the current table.

<PF4> DISPLAY KEYS

Displays a list of all keys for the current table.

<PF5> DISPLAY ALL

Displays a list of all keys and columns in the current table.

<PF6> LIST TABLES

Displays a list of all tables your query can access.

<PF7> BACKWARD

Displays the previous panel of a partially shown panel.

<PF8> FORWARD

Displays the next panel of a partially shown panel.

<PF9> TEMPLATE

Displays template for selecting CA Dataquery language statements.

<PF10> VALIDATE

Screens the current entries on the Editor panel for errors.

<PF12> PROCESS MODE

Displays PF keys for processing query or term.

PF Keys Process Mode

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays information about the current panel.

<PF2> RETURN

Redisplays previous panel.

<PF3> EXECUTE

Executes the query online.

<PF4> SAVE

Saves the current query or term in the appropriate library.

<PF5> DIALOG DEF

Defines dialog variables if variables have been entered and TYPE is *dialog*.

<PF6> DELETE

Deletes the last version saved, if you own the member.

<PF7> BACKWARD

Displays the previous panel of a partially shown panel.

<PF8> FORWARD

Displays the next panel of a partially shown panel.

<PF9> UPDATE

Applies any changes made to the member, if you own it.

<PF10> VALIDATE

Has CA Dataquery check the entries for errors.

<PF11> RIGHT/LEFT

Moves the panel 4 columns to the left or right, if an 80-column terminal screen.

<PF12> CREATE MODE

Displays PF keys for displaying table information or template of DQL statements.

(DQEA0) Query/Dialog Print Output (Wide Screens)

```
 =>
07/13/10                          DATAQUERY                          PAGE 1
11:31:20                         TOTAL SALES                       NO-TOTALS

  SLMN-ID  NAME                                YTD-SALES
  -------  ------------------------------  -------------
  00760    HARTFORD IRON WORKS INC             7,600.00
  23160    UNIVERSAL AIRWORKS                231,600.00
  00795    NATIONAL HARRIS CORPORATION         7,950.00
  34222    CANNON TOOLS CO                 3,322,123.00
  23615    M.A.C. SAVINGS                    236,150.00
  11400    MALIRY ENTERTAINMENT INDUSTRY     114,000.00
  25155    CHESTERSON-KIDD INC               251,550.00
  00655    WEST LIFE INSURANCE                 6,550.00
  09755    NETTLETON AIRCRAFT                 97,550.00
  10595    JOHNSON MULTIFOODS                105,950.00
  10420    MARK & MARK INTERNATIONAL         104,200.00
  05730    BELL-BAKER POWER & SERVICE         57,300.00


  ------------------------------------------------------------------------
 <PF1>  HELP      <PF2>  RETURN   <PF3>  TOTALS ONLY  <PF4>  DETAIL
 <PF5>  NO TOTALS <PF6>  STATS    <PF7>  BACKWARD     <PF8>  FORWARD
 <PF9> NOT USED   <PF10> NOT USED <PF11>  LEFT        <PF12> RIGHT
```

Description

This panel displays the output of your query in a columnar format.

Action

Scroll through the output or change modes (detail/totals), display query statistics, or return to online execution. There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

Date

System date.

Time

System time.

Page

Page segment number (vertical) and letter (horizontal).

Title line 1

Title used by your system.

Title line 2

Specified in the query with TITLE1 keyword.

Title line 3

Specified in the query with TITLE2 keyword.

Column headings

Alternate headings specified by the query, alternate headings, or column names.

Column data

Values from each row.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> TOTALS ONLY

Changes report format to TOTALS ONLY.

<PF4> DETAIL

Changes report format to DETAIL.

<PF5> NO TOTALS

Changes report format to NO TOTALS.

<PF6> STATS

Displays statistics for this query.

<PF7> BACKWARD

Displays previous page of current row, if multiple pages.

<PF8> FORWARD

Displays next page of current row, if multiple pages.

<PF11> LEFT

Displays preceding horizontal page segment, if any, indicated by <= on bottom separator line.

<PF12> RIGHT

Displays next horizontal page segment, if any, indicated by => on bottom separator line.

(DQEC0) FIND Statistics

```
=>
Current Dataquery FIND/SELECT Statistics.
-------------------------------------------------------------------------------DQEC0
DATAQUERY:  FIND STATISTICS                        QUERY NAME: PROGRAMS
-------------------------------------------------------------------------------

TNM-BID  STRG  Q RELS RELO     I/O   E-I/O  E-ROWS    CARD RQAS      KEY F-BUF
-------- ----- - ---- ---- ------- ------- ------- ------- ---- -------- -----
PGM    2   336 N                5     N/A     N/A     N/A   72




-------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> OVERALL STATS
```

Description

The FIND/SELECT Statistics panel appears when you display query output on your video screen, press <PF6> STATS, or enter the STALL command. When you press <PF3> OVERALL STATS after the FIND Statistics panel (DQEF0) appears, further statistical information about query processing appears on DQEC0.

Action

Note the query statistics if a problem occurs during query execution and be prepared to report the information to your CA Dataquery Administrator for help with problem resolution.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

TNM-BID

Table name and database ID.

STRG

Length of permanent storage for this table, including the request area, RQA size, nonqualifying with list, and overhead. CA Dataquery and the Compound Boolean Selection Facility of CA Datacom/DB dynamically handle the search optimization. CA Dataquery uses predicates (a WITH in a DQL Mode query FIND statement) to build a Compound Boolean Selection Request Qualification Area (RQA), converting the selection criteria in a query's statements into the Compound Boolean Selection format.

NQ

Nonqualifying predicates.

RELS

Related subject, column, key or value.

RELO

Related object, column, or key.

I/O

Number of I/Os for this execution.

E-I/O and E-ROWS

Estimated I/O from Compound Boolean Selection estimate calls for join optimization. E-I/O and E-ROWS apply to a single table. The values are calculated using the E-I/O and E-ROWS values for each table, the relative cardinalities of the tables, whether the related objects are indexed or not, and the number of rows requested in the query. The numbers shown are normally the values received from the Compound Boolean Selection Facility only for the first table listed on DQEC0. The other tables have been adjusted based on relative cardinality, and so forth.

CARD

The number of rows on the table, if joining. The relative cardinality of the tables figures into join optimization.

RQAS

Shows whether a query is nearing the 9999 RQA limitation.

KEY

Means one of two things, depending on whether the name is followed by an R or a G.

R

Indicates that Compound Boolean Selection used this key because it is a related object in the table and the ratio of table cardinalities indicated that the key may optimize the join.

G

Indicates that Compound Boolean Selection suggested using the key if executing in batch and only for the primary table in the search.

F-BUF

Storage used for buffering for nonqualifying predicates or cross-joined WITH statements.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

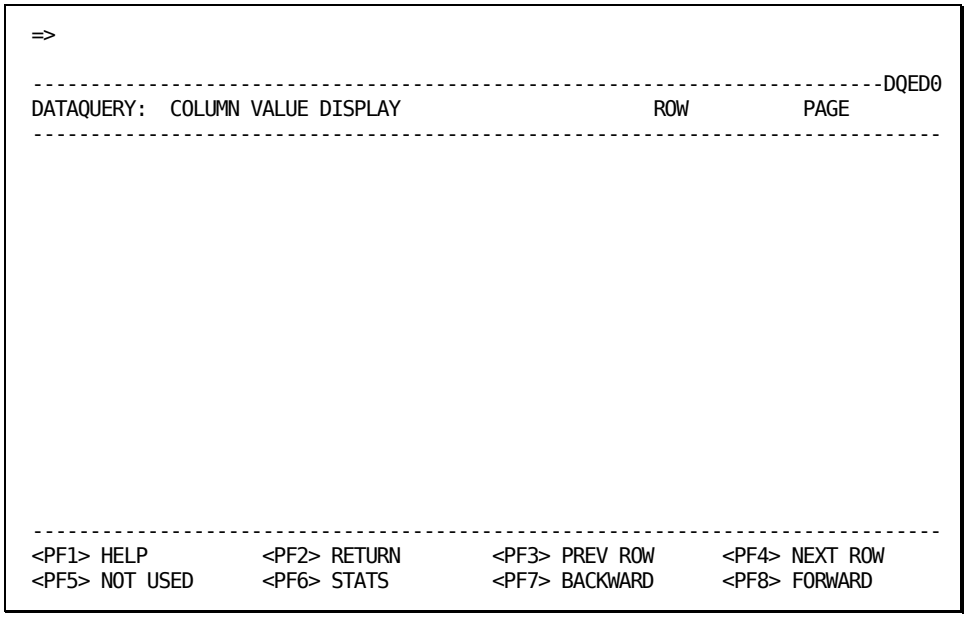
<PF2> RETURN

Exits panel.

<PF3> OVERALL STATS

Views Query Statistics panel (DQEF0).

(DQED0) Column Value Display



Description

The Column Value Display panel appears once for every row found by a query or dialog whose output is list format on the screen. It contains the data requested by the query for one row.

Action

Scroll through the Column Value Display panels for the query output, using PF keys, the +nnnn and -nnnn commands, or enter a row number to display a specific row (nn). There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

ROW

Row's sequence number in the found set.

PAGE

Page number of the current rows. Page number will always be one unless the output of a single row extends beyond one panel.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> PREV ROW

Displays first page of a previous row, if any.

<PF4> NEXT ROW

Displays first page of next row, if any.

<PF6> STATS

Displays statistics for this query.

<PF7> BACKWARD

Displays previous page of current row, if multiple pages. If none, displays first page of a previous row, if any.

<PF8> FORWARD

Displays next page of current row, if multiple pages. If none, displays first page of next row, if any.

(DQEF0) Query Statistics

```
=>
CURRENT DATAQUERY FIND/SELECT STATISTICS.
--------------------------------------------------------------------------DQEF0
DATAQUERY:  FIND STATISTICS                        QUERY NAME:
--------------------------------------------------------------------------------

     NUMBER REQUESTED:                             COMPLETION DATE:
     NUMBER FOUND:                                 COMPLETION TIME:

     FIND TERMINATED BECAUSE:

                                OVERALL    OPTIMIZATION     SEARCH
                                -------    ------------    -------
     ELAPSED TIME (SECONDS):
     I/O EVENTS:
     SELFR TOTAL:
     SELNR TOTAL:
     TOTAL BYTES:
     TOTAL JOINS:
     VALID JOINS:
     ESTIMATED I/O:
     ESTIMATED ROWS:
--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN
```

Description

The Query Statistics panel appears when you display query output on your video screen and press <PF6> STATS. It also appears at the front of query output routed to the printer, if your user profile is set up to print it automatically. It contains information about how your query processed.

Note: Total Joins, Valid Joins, Estimated I/O, Estimated Rows, and <PF3> OVERALL STATS appear only if the STALL command has been issued during this session. See (DQEC0) FIND Statistics (see page 106) and STALL - Produce Additional Statistics for more information.

Action

Note the query statistics if a problem occurs during query execution and be prepared to report the information to your CA Dataquery Administrator for help with problem resolution.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name of the query to which these statistics apply.

NUMBER REQUESTED:

Number of rows requested in your query.

NUMBER FOUND:

Number of rows found in the database.

COMPLETION DATE:

Date that the row search for this query ended.

COMPLETION TIME:

Time that the row search for this query ended.

FIND TERMINATED BECAUSE:

NORMAL END OF SEARCH WAS REACHED-- The FIND process completed normally.

SITE I/O LIMIT (DQOPTLST.SRCHLIM) EXCEEDED -- the FIND exceeded The value set for the I/O limit in the System Option Table parameter SRCHLIM=. Either reduce the number of rows which CA Dataquery is to FIND, or ask your CA Dataquery Administrator to increase the value in the SRCHLIM parameter so that your query can finish the FIND processing.

MAX WORK TABLE BLOCKS EXCEEDED -- Exceeds the FNDBLKS parameter in the System Option Table or your row. See the CA Dataquery Administrator.

MAX ELAPSED TIME EXCEEDED -- The maximum time allowed for the processing of a query FIND has been exceeded. Either simplify the FIND so that the FIND processing time is shorter or ask the CA Dataquery Database Administrator to increase the maximum time allowed.

FIND CANCELED FROM REQUESTING TERMINAL -- This query has been canceled from the requesting user's terminal.

FIND CANCELED FROM ANOTHER TERMINAL -- This query was canceled from a terminal that is not the requesting user's terminal.

ELAPSED TIME (SECONDS):

The overall time elapsed in seconds to process the FIND statement.

OPTIMIZATION -- The process of determining the search strategy for searching the database efficiently.

SEARCH -- The actual search time against the database for the FIND statement.

I/O EVENTS:

OVERALL -- The overall number of CA Datacom/DB physical I/O events for the processing of the FIND statement.

OPTIMIZATION -- The number of CA Datacom/DB physical I/O events used for determining the database search strategy.

SEARCH -- The number of CA Datacom/DB physical I/O events used for the search of the database by the FIND statement.

SELFR TOTAL:

OVERALL -- The overall number of CA Datacom/DB Compound Boolean Selection SELFR (select first) commands that were issued to execute the FIND statement.

OPTIMIZATION -- The number of CA Datacom/DB Compound Boolean Selection SELFR (select first) commands that were issued for determining the database search strategy.

SEARCH -- The number of CA Datacom/DB Compound Boolean Selection SELFR (select first) commands that were issued for the search of the database by the FIND statement.

SELNR TOTAL:

OVERALL -- The overall number of CA Datacom/DB Compound Boolean Selection SELNR (select next) commands that were issued to execute the FIND statement.

OPTIMIZATION -- The number of CA Datacom/DB Compound Boolean Selection SELNR (select next) commands that were issued for determining the database search strategy.

SEARCH -- The number of CA Datacom/DB Compound Boolean Selection SELNR (select next) commands that were issued for the search of the database by the FIND statement.

TOTAL BYTES:

OVERALL -- The overall number of bytes that were required to hold the internal control blocks that CA Dataquery built to process the FIND statement.

OPTIMIZATION -- The number of bytes that were required to hold the internal control blocks that CA Dataquery built for determining the data base strategy.

SEARCH -- The number of bytes that were required to hold the internal control block that CA Dataquery built for the search of the database by the FIND statement.

The following statistics appear on this panel if the STALL command has been used:

TOTAL JOINS:

The number of times tables were joined during query processing.

VALID JOINS:

The number of joins that could be used. Valid joins never exceed total joins. The more tables there are involved, the smaller the percentage of joins that are valid. CA Dataquery considers every possible join and chooses the optimum combination.

ESTIMATED I/O:

An estimate from the lowest cost join.

ESTIMATED ROWS:

An estimate from the lowest cost join.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

(DQED0) Column Value Display (Wide Screens)

```
=>

------------------------------------------------------------------------------DQED0
DATAQUERY:  COLUMN VALUE DISPLAY                      ROW          PAGE
------------------------------------------------------------------------------









------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> PREV ROW      <PF4> NEXT ROW
<PF5> NOT USED      <PF6> STATS         <PF7> BACKWARD      <PF8> FORWARD
```

Description

The Column Value Display panel appears once for every row found by a query or dialog whose output is list format on the screen. It contains the data requested by the query for one row.

Action

Scroll through the Column Value Display panels for the query output, using PF keys, the +nnnn and -nnnn commands, or enter a row number to display a specific row (nn). There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

ROW

Row's sequence number in the found set.

PAGE

Page number of the current rows. Page number will always be 1 unless the output of a single row extends beyond one screen.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> PREV ROW

Displays first page of a previous row, if any.

<PF4> NEXT ROW

Displays first page of next row, if any.

<PF6> STATS

Displays statistics for this query.

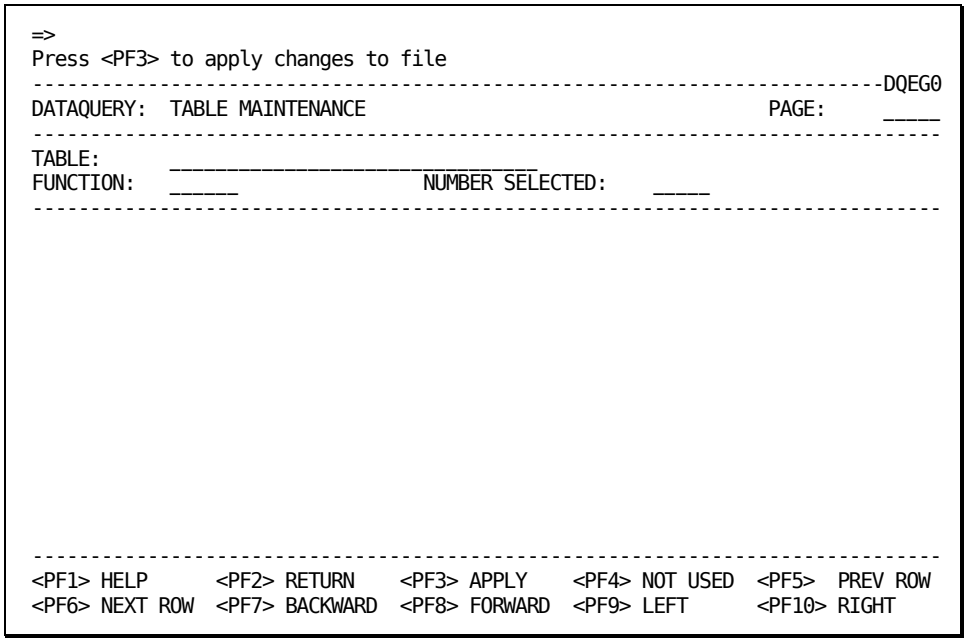
<PF7> BACKWARD

Displays previous page of current row, if multiple pages. If none, displays first page of a previous row, if any.

<PF8> FORWARD

Displays next page of current row, if multiple pages. If none, displays first page of next row, if any.

(DQEG0) Table Maintenance



Description

The Table Maintenance panel appears whenever a query containing an INSERT statement is executed. It displays the rows to be added, updated, or deleted, allowing you to verify the changes to the table before they occur. INSERT statements require special authorization. Their use is documented in the *CA Dataquery Administrator Guide*.

Action

View the displayed row to verify that the changes should be applied to the table. You may use the <PF7> and <PF8> keys to scroll the list backward and forward. If the changes should be applied to the table, press the <PF3> key, otherwise the changes will not be applied. Press the <PF2> key to return to the previous panel, or press the Clear key to return to the Main Menu.

Display Fields

The fields on this panel are for display only. No data can be entered. Refer to the following for explanations of the display fields intended to contain variable data.

PAGE:

Page number of the displayed row.

TABLE:

Names the table to which changes will be applied.

FUNCTION:

INSERT.

NUMBER SELECTED:

Always one row.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> APPLY

Applies the changes to the table.

<PF5> PREV ROW

No effect.

<PF6> NEXT ROW

No effect.

<PF7> BACKWARD

Scrolls display backward one page.

<PF8> FORWARD

Scrolls display forward one page.

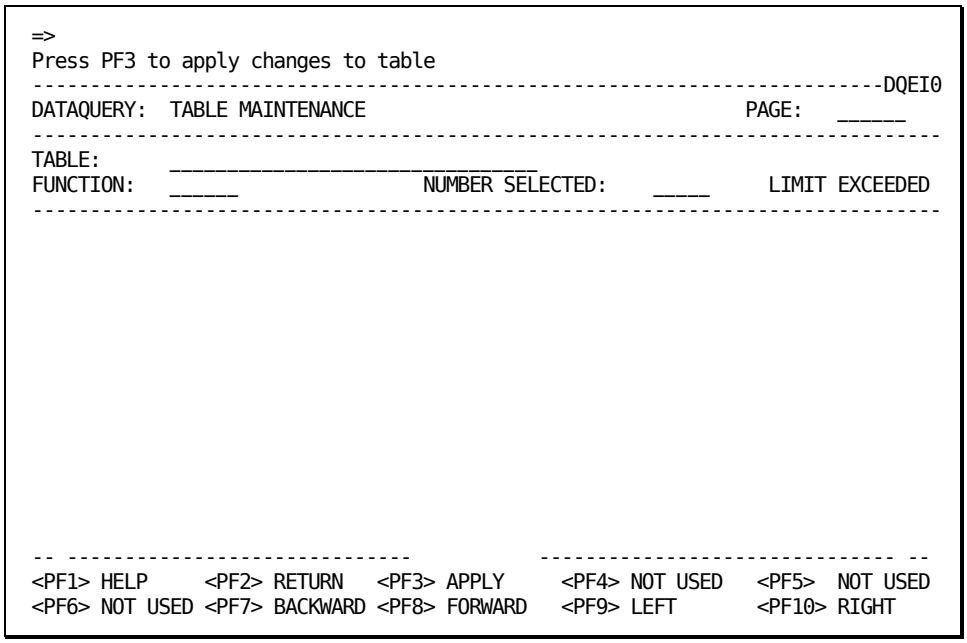
<PF9> LEFT

Scrolls display left.

<PF10> RIGHT

Scrolls display right.

(DQEI0) Table Maintenance



Description

The Table Maintenance panel appears whenever a query containing an UPDATE or ERASE statement is executed. It displays the rows to be added, updated, or deleted, allowing you to verify the changes to the table before they occur. UPDATE and ERASE statements require special authorization. Their use is documented in the *CA Dataquery Administrator Guide*.

Action

View the displayed rows to verify that the changes should be applied to the table. You may use the <PF7> and <PF8> keys to scroll the list backward and forward. If the changes should be applied to the table, press the <PF3> key otherwise the changes will not be applied. Press the <PF2> key to return to the previous panel, or press the CLEAR key to return to the Main Menu.

Display Fields

The fields on this panel are for display only. No data can be entered. Refer to the following for explanations of the display fields intended to contain variable data.

PAGE:

Page number of the displayed rows.

TABLE NAME:

Names the table to which changes will be applied.

FUNCTION:

UPDATE or ERASE.

NUMBER SELECTED:

Number of rows selected for maintenance in the table named at the top of the panel.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> APPLY

Applies the changes to the table.

<PF7> BACKWARD

Scrolls display backward one page.

<PF8> FORWARD

Scrolls display forward one page.

<PF9> LEFT

Scrolls display left.

<PF10> RIGHT

Scrolls display right.

(DQEL0) Dialog or PROC Value List

```
=>

--------------------------------------------------------------------------------DQEL0

CHOOSE A VALUE FROM THE LIST BELOW VIA CURSOR POSITION AND PRESS <PF2>.
PROMPT:









--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  NOT USED    <PF4>  NOT USED
<PF5>  NOT USED     <PF6>  NOT USED     <PF7>  BACKWARD    <PF8>  FORWARD
```

Description

When you press <PF5> RANGE/LIST with the cursor on a variable during display of a dialog or JCL prompt panel (DQEX0), CA Dataquery displays this panel, if a list of variables was defined for the default variable.

The first line following the word PROMPT contains the prompt defined by the dialog or PROC author for the current variable. Below it are listed valid variables. An overview of the operation of this panel appears in the *CA Dataquery User Guide*.

Action

You can select a variable from the list by positioning the cursor on it and pressing <PF2> RETURN to return to the prompt panel.

Sample

The following sample shows a typical panel containing information in the display fields.

```
=>

--------------------------------------------------------------------------DQEL0

CHOOSE A VALUE FROM THE LIST BELOW VIA CURSOR POSITION AND PRESS <PF2>.
PROMPT:
TYPE THE NAME OF THE CITY FOR WHICH YOU WANT A COMPLETE LIST OF CUSTOMERS.

  'DALLAS          '
  'ATLANTA         '
  'NEW YORK        '
  'LOS ANGLES      '
  'SEATTLE         '
  'BOSTON          '
  'MIAMI           '
  'HOUSTON         '
  'COPENHAGEN      '
  'LONDON          '
  'AMSTERDAM       '
  'BRUSSELS        '
  'HONG KONG       '
--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  NOT USED    <PF4>  NOT USED
<PF5>  NOT USED     <PF6>  NOT USED     <PF7>  BACKWARD    <PF8>  FORWARD
```

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

PROMPT:

The prompt written by the dialog or JCL PROC author for this variable.

Valid variables

Defined variables that can be selected (city names in the above sample).

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF7> BACKWARD

Displays a previous panel of a multi-screen panel.

<PF8> FORWARD

Displays next panel of a multi-screen panel.

(DQEN0) Batch Execution

```
=>
-------------------------------------------------------------------------------DQEN0
DATAQUERY:  BATCH EXECUTION
--------------------------------------------------------------------------------
Enter name of query to submit:           ACTIVE-QUERY
Select the type of execution:            X  Immediate
                                         _  Defer execution until time __ : __
Enter the name of the JCL member to use: $$DQJCL
Enter non-blank to use JCL for deferred:  _
Select the report type:
          X  Detail and totals           _  When/do column functions only
          _  Detail only (no totals)     _  No detail (totals and when/do)
          _  Totals only (summary)       _  Suppress report
To export print data to a sequential file, select output record type:
          _  Variable comma separated    _  Fixed-length record
For variable, enter name of output set   ________________
For variable, select output type:        _  Detail
                                         _  Totals
Select the output file device type:      _  Tape
                                         _  Disk

--------------------------------------------------------------------------------
<PF1> HELP      <PF2>  RETURN      <PF3>  SUBMIT
```

Description

When you press a SUBMIT PF key from a panel or enter the SUBMIT command, CA Dataquery displays the Batch Execution panel. You complete entries on this panel and press a PF key to continue execution. CA Dataquery presents a prompt panel if the query is a dialog and also presents another prompt panel if the JCL requires additional specifications. For information about responding to these prompt panels, see the *CA Dataquery User Guide*.

Action

You can leave the default entries on the screen or change them if necessary. Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character to select an option. Be sure the panel contains an entry or selection for each field on the screen.

Entry Fields

Refer to the following table for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Item	Entry	Explanation
Enter name of query to submit:	ACTIVE-QUERY	Query previously selected from Directory of Queries or query just executed.
	Type any valid query name	Type over ACTIVE-QUERY with a name.
Select the type of execution:	Immediate	Execution begins at next system opportunity.
	Defer execution until time __ : __	Computer operator submits query for execution after the time specified. Time is expressed in *hours:minutes* using military time. For example, 3:00 PM is 15:00.
Enter the name of the JCL member to use:	A valid JCL member name	Enter the name of a valid JCL member, if authorized.
Enter non-blank to use JCL for deferred:	Enter a non-blank character	Enter any non-blank character to use deferred JCL. *Defer execution until time* __:__ must also be selected.
Select the report type:	Detail and totals	Display the whole report.
	Detail only (no totals)	Shows detail lines, no WHEN/DO results and no totals.
	Totals only (summary)	Show only totals and WHEN/DO results.
	WHEN/DO column functions only	Suppress all contents except DQL Mode WHEN/DO results.
	No detail (totals and when/do)	Suppress detail lines.
	Suppress report	Do not produce report. Use when exporting.
To export print data to a sequential file, select output record type:	Variable comma separated	Data in character format with data fields separated by commas. Character fields will be in quotes with trailing blanks truncated. Numeric fields will have leading zeros truncated. Rows will vary in length.
	Fixed-length record	Data will be exported in same type and length as retrieved from the database table. All rows will be the same length.
For variable, enter name of output set	Any valid name	Obtain a valid set name from the CA Dataquery Administrator.

Item	Entry	Explanation
For variable, select output type:	Choose one or more: ■ Detail ■ Totals ■ Tape ■ Disk	See "Select the report type."
Select the output file device type:	Choose one: ■ Tape ■ Disk	DOS only

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> SUBMIT

Submits the job for batch execution. May display a prompt panel if executing dialog and a prompt panel if using JCL PROC.

(DQEO0) Totaling Options

```
=>
ENTER THE DESIRED OPTIONS AND PRESS THE APPROPRIATE PFKEY
-------------------------------------------------------------------------------DQEO0
DATAQUERY:  TOTALING OPTIONS
--------------------------------------------------------------------------------

QUERY NAME =>  ACTIVE-QUERY

TOTALING OPTIONS

  _  DETAIL AND TOTALS             - PRODUCE REPORT WITH DETAIL AND TOTALS
  _  DETAIL ONLY                   - SUPPRESS ANY REQUESTED TOTALING
  _  TOTALS ONLY                   - PRODUCE THE REPORT WITH TOTALS ONLY
  _  TOTALS BY NAME                - ONLY USE THE ENTERED CONTROL-BREAK NAME
                                     ____________________________________
  _  WHEN/DO RESULTS ONLY          - SUPPRESS ALL EXCEPT WHEN/DO RESULTS
  _  NO DETAIL                     - SUPPRESS DETAIL LINES on REPORT




--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN     <PF3> EXECUTE
```

Description

When a DQL query with totals is executed with the video terminal as the destination, you can execute the query and change the totaling options as many times as you like when you view the report. With any report, if the query producing it is constructed to produce totals, you can select a valid totaling option for the current execution.

Action

Enter a character in the field next to the option you want for this execution. Press <PF3> EXECUTE to continue online execution.

Entry Fields

Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character beside the selection of your choice. Be sure the panel contains an entry or selection for each category on the screen.

QUERY NAME =>

The name of the query or dialog named on the execution panel.

TOTALING OPTIONS

DETAIL AND TOTALS

All data and totals specified.

DETAIL ONLY

Suppress totals specified in query.

TOTALS ONLY

Do not show detailed data.

TOTALS BY NAME

Display totals only when named column data changes.

WHEN/DO RESULTS ONLY

Display only WHEN/DO results.

NO DETAIL

Display no detail lines, only WHEN/DO results or query totals.

When the Totaling Options panel is complete, you press a PF key to tell CA Dataquery what to do next.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help for panel.

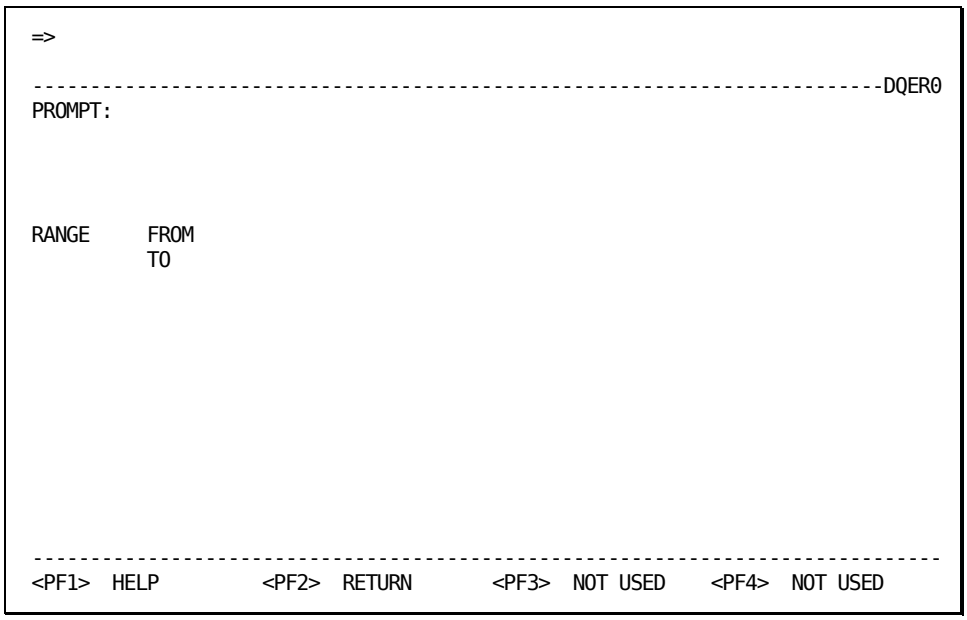
<PF2> RETURN

Redisplays Online Execution panel. Retains selections.

<PF3> EXECUTE

1. Begins execution process.
2. If dialog, displays prompter panel for selection.
3. Displays execution messages.
4. Displays report if destination is video terminal.

(DQER0) Dialog or PROC Range



Description

If you press <PF5> RANGE/LIST with the cursor on a numeric variable, CA Dataquery presents a range of valid numbers if one was defined by the query/JCL author. The cursor appears in a field in which you can type a number that falls within the range.

Action

Enter a number in the field containing the cursor that falls within the range and press <PF2> RETURN to return to the list of variables.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
At cursor position	N	Any number within range	Shown on variable list

Sample

The following sample shows a typical panel containing information in the display fields.

```
=>

--------------------------------------------------------------------------------DQER0
PROMPT:
ENTER A PRODUCT CODE IN THE RANGE DEFINED OR ACCEPT THE 75042 DEFAULT.



RANGE     FROM        75000
          TO          75999

















--------------------------------------------------------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  NOT USED    <PF4>  NOT USED
```

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

PROMPT:

Prompt written by the dialog or JCL PROC author appears.

FROM

The low value in the range of valid values.

TO

The high value in the range of valid values.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

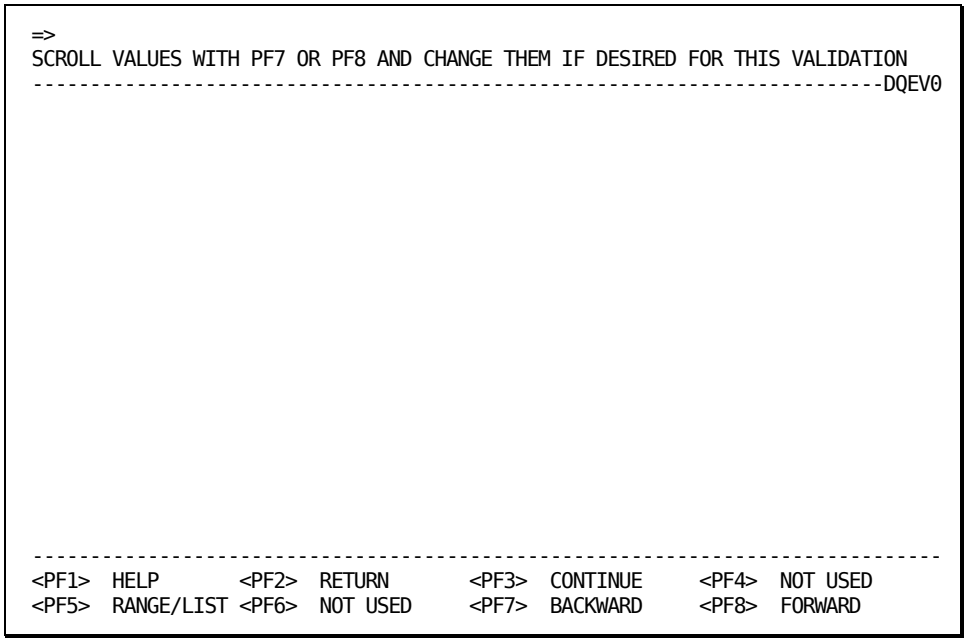
<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits list of variables.

(DQEV0) Dialog Validation



Description

Before a dialog can be executed, you must view the panels that a user executing the panels will see and decide whether they are what you intended. CA Dataquery wants you to try the dialog yourself before you make it available to others. When you press the VALIDATE <PF> key after defining the dialog, you see this prompt panel (EV0) that you use to test the dialog. More information is available on this subject in the *CA Dataquery User Guide*.

Action

Test the new dialog by entering all the values in a list, if one was defined for a character or alphabetic variable. Also, enter a variety of values for numeric values. Then <PF2> RETURN to return to the Editor panel. Look for a message saying that validation was successful.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

Description

A description of up to four lines written by the author describing the dialog/JCL PROC at the top of the screen.

Prompt 1

Following the description, a prompt written by the dialog or JCL PROC author, providing information about the following variable.

Default 1

Default variable entered on Editor panel. Additional prompts and variables can appear if defined by the author.

Sample

The following sample shows how a typical panel looks when it contains information in the display fields.

```
=>
SCROLL VALUES WITH PF7 OR PF8 AND CHANGE THEM IF DESIRED FOR THIS VALIDATION
--------------------------------------------------------------------------DQEV0
          PRODUCE A CUSTOMER REPORT FOR A SELECTED CITY.

        PROVIDE THE NAME OF THE CITY DESIRED AS DIRECTED.


TYPE THE NAME OF THE CITY FOR WHICH YOU WANT A COMPLETE LIST OF CUSTOMERS.
'DALLAS         '

ENTER A PRODUCT CODE IN THE RANGE DEFINED OR ACCEPT THE 75042 DEFAULT.
75042









- LAST PAGE ------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  CONTINUE     <PF4>  NOT USED
<PF5>  RANGE/LIST <PF6>  NOT USED     <PF7>  BACKWARD     <PF8>  FORWARD
```

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> CONTINUE

Continues with the validation process.

<PF5> RANGE/LIST

Displays a range panel or list panel for the variable at cursor position, if a range or list was defined.

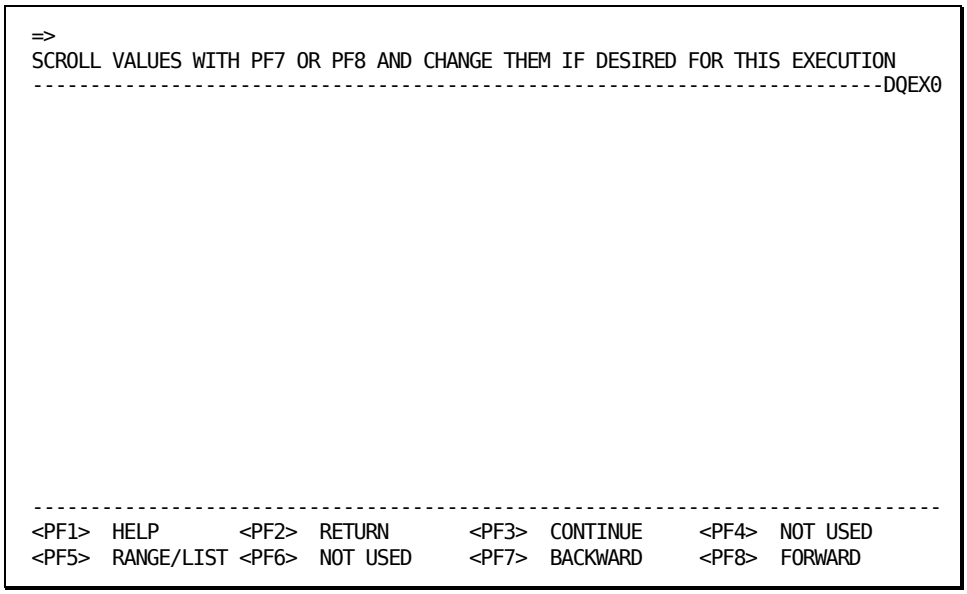
<PF7> BACKWARD

Displays a previous screen of a multi-screen panel.

<PF8> FORWARD

Displays next screen of a multi-screen panel.

(DQEX0) Prompt Panel (Dialog or PROC Execution)



Description

The Prompt Panel appears when a dialog is executed or if a JCL PROC is named on the Batch Execution panel. Variables can be replaced for the current dialog or PROC. The text and fields that appear are written by the dialog or PROC's creator.

Action

You type in a valid substitute for any variable, or press <PF5> RANGE/LIST for a list or range of valid substitutes. More information about responding to dialog and JCL prompts is available in the *CA Dataquery User Guide*.

Sample

The following sample shows how a typical panel looks when it contains information in the display fields.

```
=>
SCROLL VALUES WITH PF7 OR PF8 AND CHANGE THEM IF DESIRED FOR THIS VALIDATION
--------------------------------------------------------------------------DQEX0
Description:

              PRODUCE A CUSTOMER REPORT FOR A SELECTED CITY.

            PROVIDE THE NAME OF THE CITY DESIRED AS DIRECTED.


TYPE THE NAME OF THE CITY FOR WHICH YOU WANT A COMPLETE LIST OF CUSTOMERS.
'DALLAS          '

ENTER A PRODUCT CODE IN THE RANGE DEFINED OR ACCEPT THE 75042 DEFAULT.
75042




- LAST PAGE ------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN      <PF3>  CONTINUE     <PF4>  NOT USED
<PF5>  RANGE/LIST <PF6>  NOT USED    <PF7>  BACKWARD     <PF8>  FORWARD
```

Display Areas

Refer to the following chart for explanations of the display areas intended to contain variable data.

Area	Location	Explanation
Description:	Top	A description of up to four lines written by the author describing the dialog/JCL PROC at the top of the screen.

Area	Location	Explanation
Prompts	One line each	After the description, a prompt for each variable written by the dialog or JCL PROC author, providing information about the following variable.
Default variables	One line each	Variables entered on Editor panel (DALLAS and 75042 in the preceding example).

The default shown after any prompt will be used in execution if not typed over or replaced with a selection using <PF5> RANGE/LIST.

Additional prompts and variables can appear if defined by the author.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to execution panel.

<PF3> CONTINUE

Continues with the execution process.

<PF5> RANGE/LIST

Displays a list or numeric range for the current variable designated by cursor position, if one was defined.

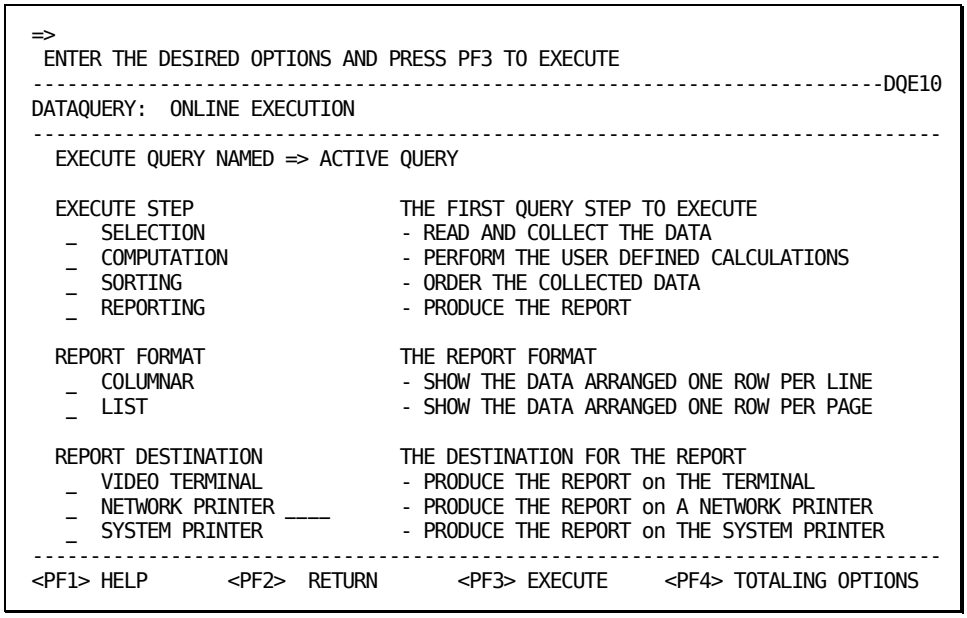
<PF7> BACKWARD

Scrolls backward through previous screens, if any.

<PF8> FORWARD

Scrolls forward through additional screens, if any.

(DQE10) Online Execution



Description

The Online Execution panel lets you request immediate use of system resources to query the database table and return information in the form of a report. You can enter a query name or execute the ACTIVE-QUERY. You can also specify a beginning execution point (for re-executions), a report format, and a destination.

Action

Leave the default entries on the panel or change them as needed. Follow these steps:

1. Verify that the query named is the one you want to execute.
2. Decide which query step to use for execution to begin. Only select a step that is appropriate for the query.
3. Select a report format. Leaving both options blank tells CA Dataquery to use the format specified in the query.
4. Select the output destination.
5. To view or change totaling options, display the Totaling Options panel with <PF4> TOTALING OPTIONS.
6. Press <PF3> EXECUTE.

Entry Fields

Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character beside the selection of your choice. Be sure the panel contains an entry or selection for each category on the screen.

EXECUTE QUERY NAMED =>

Use ACTIVE QUERY for query previously selected from Directory of Queries or query just executed.

Use any valid query name; type over ACTIVE-QUERY with a name.

EXECUTE STEP

(Choose one.)

SELECTION

First execution step. (FIND) Do not select unless FIND statement is changed or query is no longer active.

COMPUTATION

Re-execute beginning with math functions, if any. (SET)

SORTING

Re-execute beginning with sorting functions, if any. (SORT)

REPORTING

Re-execute beginning with report production. (PRINT/DISPLAY)

REPORT FORMAT

COLUMNAR

Rows of data with column headings.

LIST

Each row appears on a separate screen. (Not available if routing output to network printer.)

REPORT DESTINATION

VIDEO TERMINAL

Report appears on screen.

NETWORK PRINTER _____

Hardcopy appears at printer designated in _____ unless List option chosen.

SYSTEM PRINTER

Hardcopy appears at system printer.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help for panel.

<PF2> RETURN

Redisplays Directory of Queries panel.

<PF3> EXECUTE

1. Begins execution process.
2. If dialog, displays prompt panel for selection.
3. Displays execution messages.
4. Displays report if destination is video terminal.

<PF4> TOTALING OPTIONS

Displays Totaling Options panel for changing options.

(DQE20) Online Execution (Associate User)

```
=>
 ENTER THE DESIRED OPTIONS AND PRESS PF3 TO EXECUTE
--------------------------------------------------------------------------------DQE20
DATAQUERY:  ONLINE EXECUTION
--------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  REPORT FORMAT                THE REPORT FORMAT
   _  COLUMNAR                 - SHOW THE DATA ARRANGED ONE ROW PER LINE
   _  LIST                     - SHOW THE DATA ARRANGED ONE ROW PER PAGE

  REPORT DESTINATION           THE DESTINATION FOR THE REPORT
   _  VIDEO TERMINAL           - PRODUCE THE REPORT on THE TERMINAL
   _  NETWORK PRINTER  ____    - PRODUCE THE REPORT on A NETWORK PRINTER
   _  SYSTEM PRINTER           - PRODUCE THE REPORT on THE SYSTEM PRINTER









--------------------------------------------------------------------------------
<PF1> HELP       <PF2>  RETURN       <PF3> EXECUTE     <PF4> TOTALING OPTIONS
```

Description

The Online Execution panel lets you request immediate use of system resources to query the database table and return information in the form of a report. You can enter a query name or execute the ACTIVE-QUERY. You can also specify a report format and a destination.

Action

Leave the default entries on the panel or change them as needed. Follow these steps:

1. Verify that the query named is the one you want to execute.
2. Select a report format. Leaving both options blank tells CA Dataquery to use the format specified in the query.
3. Select the output destination.
4. To view or change totaling options, display the Totaling Options panel with <PF4> TOTALING OPTIONS.
5. Press <PF3> EXECUTE.

Entry Fields

Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character beside the selection of your choice. Be sure the panel contains an entry or selection for each category on the screen.

EXECUTE QUERY NAMED

Use ACTIVE-QUERY for query previously selected from Directory of Queries or query just executed.

Use any valid query name; type over ACTIVE- QUERY with a name.

REPORT FORMAT

COLUMNAR:

Rows of data with column headings

LIST:

Each row appears on separate page. (Not available if routing output to a network or system printer.)

DESTINATION

(You can choose one printer in addition to screen display.)

VIDEO TERMINAL

Report appears on screen.

NETWORK PRINTER

Hardcopy appears at designated printer ID. (Not valid for LIST.)

SYSTEM PRINTERR

Hardcopy appears at system printer.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help for panel

<PF2> RETURN

Redisplays Online Execution panel. Retains selections.

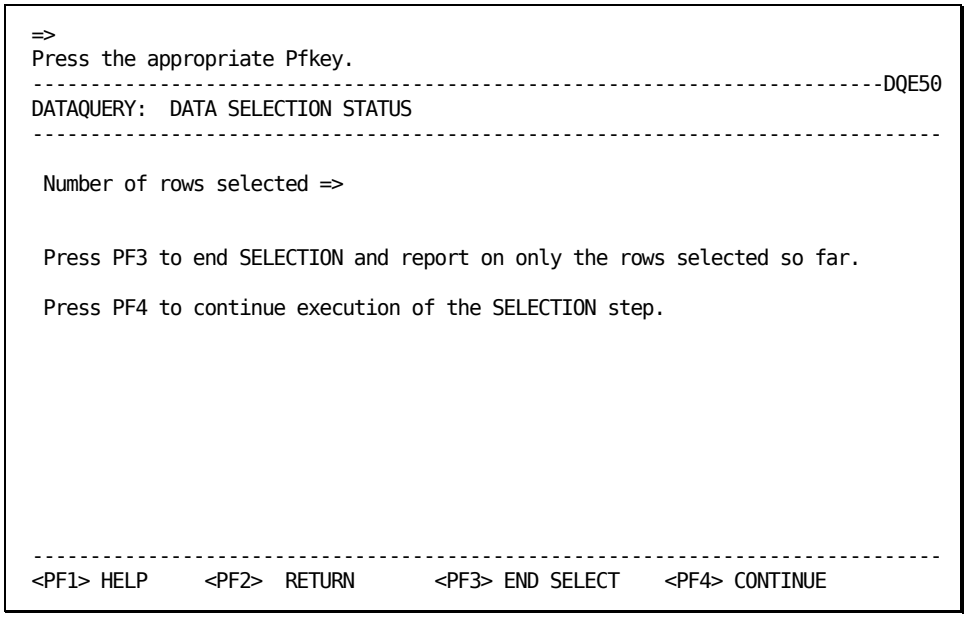
<PF3> EXECUTE

1. Begins execution process.
2. If dialog, displays prompt panel for selection.
3. Displays execution messages.
4. Displays report if destination is video terminal.

<PF4> TOTALING OPTIONS

Displays Totaling Options panel for changing options.

(DQE50) Data Selection Status



Description

Depending on the amount of system resources allotted to you for reading database tables with any one query, CA Dataquery may interrupt processing your query or dialog's FIND statement with the Data Selection Status panel. This panel tells you how many rows have been found so far and allows you to either report on the rows already found or continue processing.

Action

Press an appropriate PF key to indicate your choices.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

Number of rows selected =>

Number of rows found by the query or dialog when CA Dataquery interrupted processing.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> END SELECT

Discontinues row selection and produces report based on records found so far.

<PF4> CONTINUE

Continues processing FIND statement to select all rows.

(DQE60) Online Execution SQL Query

```
=>
 Enter the desired options and press PF3 to execute
--------------------------------------------------------------------------------DQE60
DATAQUERY:  ONLINE EXECUTION SQL QUERY
--------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  EXECUTE STEP                 The first query step to execute
   _  SELECTION                - Read and collect the data
   _  REPORTING                - Produce the report

  REPORT DESTINATION           The destination for the report
   _  VIDEO TERMINAL           - Produce the report on the terminal
   _  NETWORK PRINTER ____     - Produce the report on a network printer
   _  SYSTEM PRINTER           - Produce the report on the system printer









--------------------------------------------------------------------------------
<PF1> HELP       <PF2> RETURN       <PF3> EXECUTE       <PF4> FORMAT REPORT
```

Description

The SQL Mode Online Execution SQL Query panel for users lets you request immediate use of system resources to query the database table and return information in the form of a report. You can enter a query name or execute the ACTIVE QUERY. You can also specify a beginning execution point (for re-executions), a report format, and a destination.

Action

Leave the default entries on the panel or change them as needed. Follow these steps:

1. Verify that the query named is the one you want to execute.
2. Decide which query step to use for execution to begin. Only select a step that is appropriate for the query.
3. Press <PF4> FORMAT REPORT if you need to change the format specifications for this execution.
4. Select the output destination.
5. Press <PF3> EXECUTE.

Entry Fields

Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character beside the selection of your choice. Be sure the panel contains an entry or selection for each category on the screen.

Field	Valid Entries	Explanation
EXECUTE QUERY NAMED	ACTIVE-QUERY	Query most recently used.
	Any valid query name	Type over ACTIVE-QUERY with a name.
EXECUTE STEP (Choose one.)	SELECTION	First execution step (FIND). Do not select unless query is changed or is no longer active.
	REPORTING	Re-execute beginning with report production.
REPORT DESTINATION (video and/or printer)	VIDEO TERMINAL	Report appears on screen.
	NETWORK PRINTER	Hardcopy appears at printer designated (unless LIST option selected).
	SYSTEM PRINTER	Hardcopy appears at system printer.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help for panel

<PF2> RETURN

Redisplays Directory of Queries panel

<PF3> EXECUTE

1. Begins execution process.
2. If dialog, displays prompt panel for selection.
3. Displays execution messages.
4. Displays report if destination is video terminal.

<PF4> FORMAT REPORT

Displays panels for changing report specifications.

(DQE70) Online Execution SQL Query (Associate User)

```
=>
 Enter the desired options and press PF3 to execute
--------------------------------------------------------------------------------DQE70
DATAQUERY:  ONLINE EXECUTION SQL QUERY
--------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  REPORT DESTINATION             The destination for the report
    _   VIDEO TERMINAL           - Produce the report on the terminal
    _   NETWORK PRINTER ____     - Produce the report on a network printer
    _   SYSTEM PRINTER           - Produce the report on the system printer









--------------------------------------------------------------------------------
<PF1> HELP       <PF2> RETURN       <PF3> EXECUTE       <PF4> FORMAT REPORT
```

Description

The SQL Mode Online Execution panel for associate users lets you request immediate use of system resources to query the database table and return information in the form of a report. You can enter a query name or execute the ACTIVE-QUERY.

Action

Leave the default entries on the panel or change them as needed. Follow these steps:

1. Verify that the query named is the one you want to execute.
2. Press <PF4> FORMAT REPORT if you need to change the format specifications for this execution.

3. Select the output destination.
4. Press <PF3> EXECUTE.

Entry Fields

Use the space bar to remove entries and the keyboard keys to type different entries. Enter any character beside the selection of your choice. Be sure the panel contains an entry or selection for each category on the screen.

Field	Valid Entries	Explanation
EXECUTE QUERY NAMED	ACTIVE-QUERY	Query most recently used.
	Any valid query name	Type over ACTIVE-QUERY with a name.
REPORT DESTINATION (video and/or printer)	VIDEO TERMINAL	Report appears on screen.
	NETWORK PRINTER	Hardcopy appears at printer designated (unless LIST option selected).
	SYSTEM PRINTER	Hardcopy appears at system printer.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help for panel.

<PF2> RETURN

Redisplays Directory of Queries panel.

<PF3> EXECUTE

1. Begins execution process.
2. If dialog, displays prompt panel for selection.
3. Displays execution messages.
4. Displays report if destination is video terminal.

<PF4> FORMAT REPORT

Displays panels for changing report specifications.

(DQE90) Query/Dialog Print Output

```
 =>
07/13/10                          DATAQUERY                          PAGE 1
11:31:20                        TOTAL SALES                       NO-TOTALS

  SLMN-ID  NAME                                 YTD-SALES
  -------  -----------------------------  -------------
  00760    HARTFORD IRON WORKS INC              7,600.00
  23160    UNIVERSAL AIRWORKS                 231,600.00
  00795    NATIONAL HARRIS CORPORATION          7,950.00
  34222    CANNON TOOLS CO                3,322,123.00
  23615    M.A.C. SAVINGS                     236,150.00
  11400    MALIRY ENTERTAINMENT INDUSTRY      114,000.00
  25155    CHESTERSON-KIDD INC                251,550.00
  00655    WEST LIFE INSURANCE                  6,550.00
  09755    NETTLETON AIRCRAFT                  97,550.00
  10595    JOHNSON MULTIFOODS                 105,950.00
  10420    MARK & MARK INTERNATIONAL          104,200.00
  05730    BELL-BAKER POWER & SERVICE          57,300.00


 ------------------------------------------------------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12>RIGHT
```

Description

This panel displays the output of your query in a columnar format.

Action

Scroll through the output or change modes (detail/totals), display query statistics, or return to online execution.

There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

Date

System date.

Time

System time.

Page

Page segment number (vertical) and letter (horizontal).

Title line 1

Title used by your system.

Title line 2

Specified in the query with TITLE1 keyword.

Title line 3

Specified in the query with TITLE2 keyword.

Column headings

Alternate headings specified by the query, alternate headings, or Datadictionary field names.

Column data

Values from each row for the column identified with the column heading.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> TOTALS ONLY

Changes report format to TOTALS ONLY.

<PF4> DETAIL

Changes report format to DETAIL.

<PF5> NO TOTALS

Changes report format to NO TOTALS.

<PF6> STATS

Displays statistics for this query.

<PF7> BACKWARD

Displays previous page of current row, if any.

<PF8> FORWARD

Displays next page of current row, if any.

<PF11> LEFT

Displays preceding horizontal page segment, if a <= appears on bottom separator line.

<PF12> RIGHT

Displays next horizontal page segment, if a => appears on bottom separator line.

(DQFA0) Report Format - Function Control Break

```
=>

-------------------------------------------------------------------------------DQFA0
DATAQUERY:  REPORT FORMAT-FUNCTION CONTROL BREAK   QUERY NAME: _______________
-------------------------------------------------------------------------------
Select the control break level at which the  _______ function for the
_________________________________ column is to be displayed.
The appropriate level may be indicated by placing any character next to the
desired names.
-------------------------------------------------------------------------------
SEL|  CONTROL BREAK COLUMN NAME       |
-------------------------------------------------------------------------------
   |                                  |
   |                                  |
   |                                  |
   |                                  |
   |                                  |
   |                                  |
   |                                  |
   |                                  |
   |                                  |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED       <PF6> NOT USED       <PF7> BACKWARD   <PF8> FORWARD
```

Description

This panel allows you to select a control break for calculating selected functions for a report created by executing an SQL query. This panel appears for each function you selected for your report. The function will be calculated and the results will be displayed whenever the value of the specified control break changes.

You may choose the end of the report as a control break which means that the function will be calculated for the entire column and will appear at the end of the report.

Action

Select the control breaks or the end of report by entering a character in the SELECT column beside the column name. Then press <PF4> or ENTER to go to the next report definition step.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SEL	N	Any character	No default

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Displays the next reporting panel.

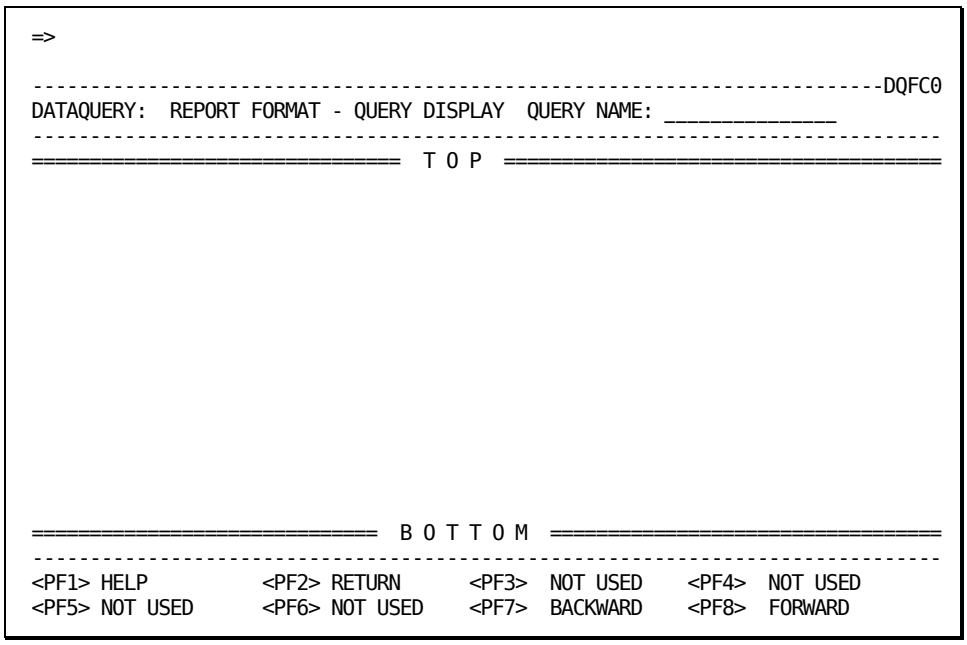
<PF7> BACKWARD

Displays previous page of current row, if any.

<PF8> FORWARD

Displays next page of current row, if any.

(DQFC0) Report Format - Query Display



Description

This panel appears when <PF6> DISPLAY is pressed during SQL Mode report formatting. It displays the query as it appears at the time you press <PF6>. The query shown, whether complete or incomplete, resides in the library you selected when you named the query (private or public). You may exit at any time, and edit the query by selecting it from the library and editing it on the Editor panel.

Action

View the query as an aid to making formatting decisions.

There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name of this query.

Query display area

Contains the query for which you are formatting a report.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF7> BACKWARD

Displays previous page, if multiple pages.

<PF8> FORWARD

Displays next page, if multiple pages.

(DQF30) Report Format - Report Title

```
=>

--------------------------------------------------------------------------------DQF30
DATAQUERY:  REPORT FORMAT - REPORT TITLE      QUERY NAME:    TESTEXEC
--------------------------------------------------------------------------------

Enter up to two lines for a
report title:      => SEE SQL REPORT PANELS
                   =>

Select the desired report format:    X  Columnar format: one row per line.
                                     _  List format: one row per page.









--------------------------------------------------------------------------------
<PF1> HELP       <PF2> RETURN       <PF3> DELETE FORMAT      <PF4> CONTINUE
```

Description

This SQL Mode report formatting panel is not required. However, if you want to assign a different title to your report or display your data one row per page instead of in columns, complete this panel.

Action

Complete the columns or accept all defaults. Then press Enter or <PF4> CONTINUE to go on to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name of this query.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Report title	N	1-55 alphanumeric characters each line	None
Columnar Format	N	Any character	X
List Format	N	Any character	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Sort Criteria Definition (DQHD0).

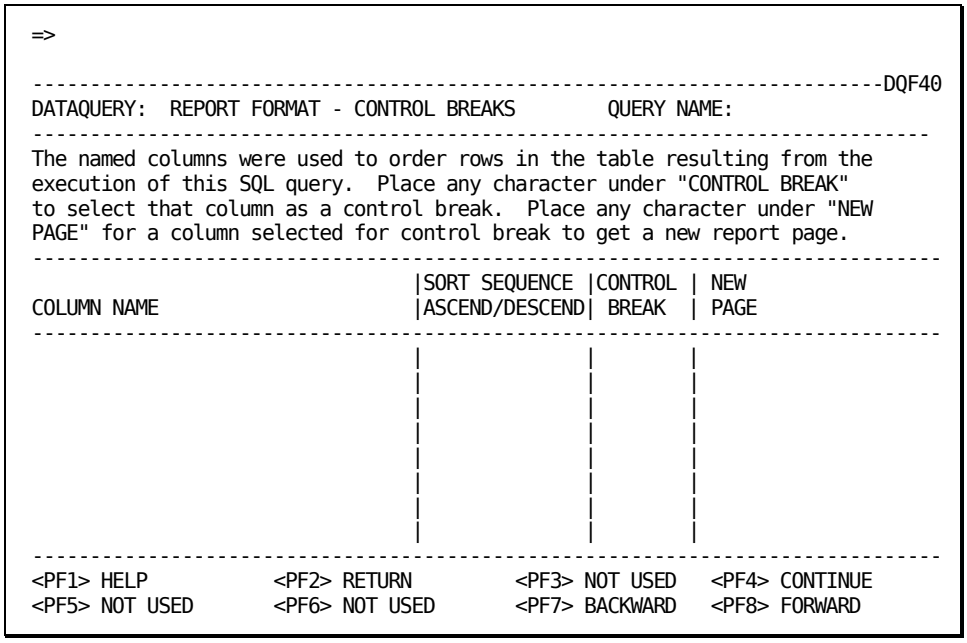
<PF3> DELETE FORMAT

Deletes this type of format.

<PF4> CONTINUE

Displays the next panel when this one is complete.

(DQF40) Report Format - Control Breaks



Description

This SQL Mode report formatting panel allows you to designate a column named in an ORDER BY statement as a control break during report generation.

Action

Specify a sorting sequence and whether you want the column to be a control break or generate a new page.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name of this query.

COLUMN NAME

The named columns used to order rows in the table.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SORT SEQUENCE ASCEND/DESCEND	N	Any character (under ASCEND or DESCEND)	Ascending
CONTROL BREAK	N	Any character	None
NEW PAGE	N	Any character	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Goes to the next report formatting panel.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQF50) Report Format - Column Functions

```
=>
 Press <PF1> for help with defining the report
-------------------------------------------------------------------------------DQF50
DATAQUERY:  REPORT FORMAT - COLUMN FUNCTIONS  QUERY NAME:    TESTEXEC
-------------------------------------------------------------------------------
Specify the order in which the columns are to appear on the report by
inserting an order number below.  Column functions may be selected by placing
any character under the corresponding functions.
-------------------------------------------------------------------------------
NUMBER| COLUMN NAME                      | AVG | MAX | MIN | SUM | CNT | TOT
-------------------------------------------------------------------------------
 01   | ID                               |     |     |     |     |     |
 02   | NAME                             |     |     |     |     |     |
 03   | DEPT                             |     |     |     |     |     |
 04   | JOB                              |     |     |     |     |     |
 05   | YEARS                            |     |     |     |     |     |
 06   | SALARY                           |     |     |     |     |     |
 07   | COMM                             |     |     |     |     |     |
      |                                  |     |     |     |     |     |
      |                                  |     |     |     |     |     |
      |                                  |     |     |     |     |     |
- LAST PAGE ----------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> NOT USED      <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode report formatting panel can be accepted with the defaults CA Dataquery assigns, resulting in report data printed in columns in the same order in which it was selected. However, you may want to change the order in which columns are printed. You may also want to request that mathematical functions be performed on some or all numeric columns. If so, you will select math functions for columns. At the end of the report, or at control breaks that you can specify later, you can obtain the results of the functions and specify a special legend to identify them on the report.

Action

Number the columns in the order wanted in the Order Number column. Enter any character in a column function column you want to perform on a particular column. Press ENTER or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name assigned the query you are defining.

COLUMN NAME

Names of the columns you selected.

Columns

Refer to the following for a discussion of each column on this panel.

NUMBER

Numbers appearing in this column are assigned automatically by CA Dataquery. Change the numbers as needed by typing over the current number to indicate the order of columns on the report.

AVG

Enter a character in this column next to any column or key for which you want CA Dataquery to produce an average at control break or at the end of the report.

MAX

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the maximum amount found at control break or at the end of the report.

MIN

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the minimum amount found at control break or at the end of the report.

SUM

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a total at control break or at the end of the report.

CNT

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a count of the data items in this column at control break or at the end of the report.

TOT

Enter a character in this column next to any column or key for which you want CA Dataquery to produce totals at control breaks and at the end of the report, and for which you do not need to assign legends different from those assigned automatically by CA Dataquery.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Definition (DQHG0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQF60) Report Format - Column Headings

```
=>

-------------------------------------------------------------------------------DQF60
DATAQUERY:  REPORT FORMAT - COLUMN HEADINGS     QUERY NAME: TESTEXEC
-------------------------------------------------------------------------------
You may enter alternate headings for any column on the report.
Defaults are provided and can be modified.
-------------------------------------------------------------------------------
COLUMN: ID                              LINE1: ID
                                        LINE2:

COLUMN: NAME                            LINE1: NAME
                                        LINE2:

COLUMN: DEPT                            LINE1: DEPT
                                        LINE2:

COLUMN: JOB                             LINE1: JOB
                                        LINE2:


-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> NOT USED      <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode report formatting panel lets you define your own titles for each column printed in your query output. If you do not define a title, CA Dataquery supplies one using the Datadictionary alternate heading. If no Datadictionary alternate heading exists, the column name is used. The default names are shown in the LINE1: input columns for each column selected for the report.

Action

Enter a one or two line title for each column heading in the columns provided for each column, typing over the default column names, or press Enter or <PF4> CONTINUE to go on to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

COLUMN:

Contains the name of a column you selected to appear in the report. Temporary results are not shown since you assign them a name yourself when you create them.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the TAB and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
LINE1:	N	1-32 characters except slash (/)	Alternate Datadictionary heading or field name
LINE2:	N	1-32 characters except slash (/)	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Column Definition (DQHI0) panel.

<PF4> CONTINUE

Displays the next panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays the next panel, if any.

(DQF70) Report Format - Column Format

```
=>

--------------------------------------------------------------------------------DQF70
DATAQUERY:  REPORT FORMAT - COLUMN FORMAT       QUERY NAME: _______________
--------------------------------------------------------------------------------
Select the type of editing desired for each numeric column on your report.
Place any character under the number that represents the edit format needed.
Defaults are provided and can be modified.
  Column 1:  12345.678  Column 3: 0,012,345.678  Column 5: E12,345.68
  Column 2: 12,345.678  Column 4:   $12,345.68   Use only for euro support
--------------------------------------------------------------------------------
COLUMN NAME                      |   1   |   2   |   3   |   4   |   5
--------------------------------------------------------------------------------
 AVERAGE WEEKLY EXPENSES         |   _   |   _   |   _   |   _   |   _
 AVERAGE WEEKLY INCOME           |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
                                 |   _   |   _   |   _   |   _   |   _
--------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> NOT USED       <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode report formatting panel appears if any numeric column is reported in the query output. It allows you to select a format for the appearance of the numeric data or accept the default.

Action

Enter a character in the column next to each column for an alternate edit pattern that you want to use and space over the default selection or press Enter or <PF4> CONTINUE to accept the default.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name assigned this query.

COLUMN NAME

The table and column names for each numeric column selected for the report.

Columns

Refer to the following for a discussion of each column on this panel.

Column 1

Suppress leading zeros, no commas, decimal places as required to display numeric column.

Column 2

Suppress leading zeros, use commas and decimal places as required.

Column 3

Show leading zeros, use commas and decimal places as required.

Column 4

Using floating dollar sign, commas, and two decimal places. Numbers are rounded upwards to the nearest cent.

Column 5

Use the Datadictionary edit pattern assigned to the column/key, if one exists. The text display for a column shows the Datadictionary edit pattern, if one exists.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Heading Definition (DQHK0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQF80) Report Format - Function Legends

```
=>

-------------------------------------------------------------------------------DQF80
DATAQUERY:  REPORT FORMAT - FUNCTION LEGENDS       QUERY NAME: _______________
-------------------------------------------------------------------------------
Enter a legend to be displayed with the result of the indicated function
during control break processing.
Defaults are provided and can be modified.
-------------------------------------------------------------------------------
FNC | COLUMN NAME                      | LEGEND
-------------------------------------------------------------------------------
AVG | AVERAGE MONTHLY EXPENSES         |
    |                                  |
MIN | MINIMUM EXPENDITURES             |
    |                                  |
MAX | MAXIMUM EXPENDITURES             |
    |                                  |
    |                                  |
    |                                  |
    |                                  |
    |                                  |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> NOT USED       <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode report formatting panel lists each column function specified for this report. Here, you can assign a legend to the printed results of each column function. The legend you define appears next to the function results when they appear on the query output.

Action

Accept the default legend or type a new one over it. You can use a heading substitution string in the Legend field. If you want the current value of the control break printed within this legend, you can include a site-specified heading substitution string like && in the legend. See the System Profile panel in Topic Help to learn what characters are used at your site as a heading substitution string. Press Enter or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

COLUMN NAME

Column names for each numeric column selected for the

Columns

Refer to the following for explanations of the columns on this panel.

FNC

Function assigned to the column listed at right.

AVG

Average value of the column at calculation.

MAX

Highest value found in the column at calculation.

MIN

Lowest value found in the column at calculation.

CNT

Number of times this column was printed at calculation.

SUM

The sum of the values in this column at calculation.

COLUMN NAME

Names of columns to which you assigned a column function to be calculated at control breaks or end of report.

LEGEND

Enter a 1- to 38-character legend to be printed when functions are calculated and results are printed. You can use a heading substitution string in the Legend field. If you want the current value of the control break printed within this legend, you can include a site-specified heading substitution string like && in the legend. See the System Profile panel in Topic Help to learn what characters are used at your site as a heading substitution string.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Format Definition (DQHL0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel.

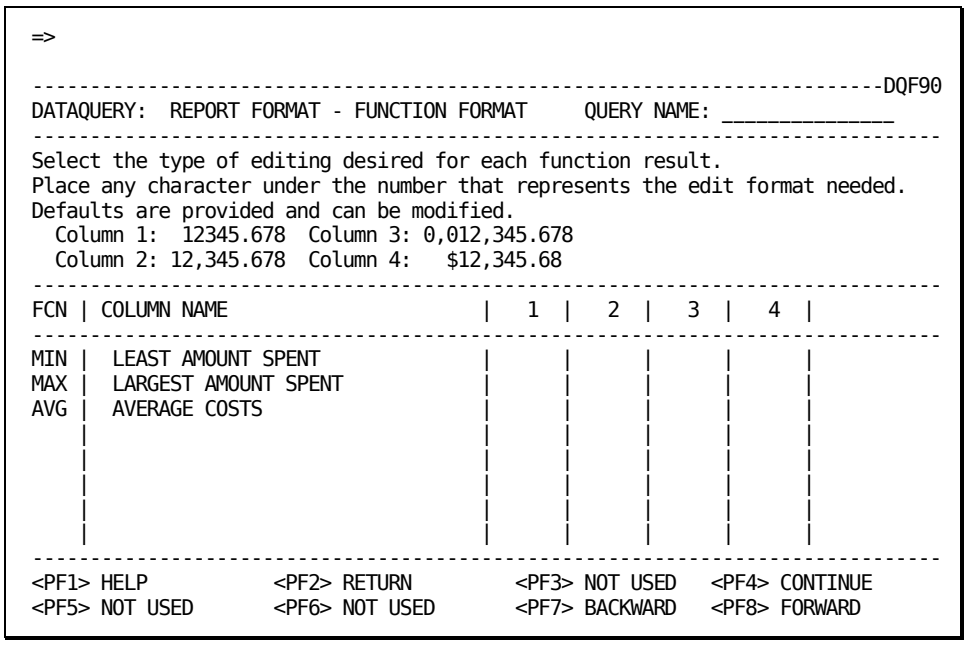
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQF90) Report Format - Function Format



Description

This SQL Mode report formatting panel permits you to change the display characteristics (numeric format or edit pattern) of any numeric data that appears in the columns that display the result of calculations selected as column functions. The default is indicated by an X in a column next to each function selection.

Action

Select a numeric format for each column function result by placing a character in the numbered column on this panel that represents the format you want the calculation result to have in the output.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

COLUMN NAME

Column names for each numeric column selected for the report.

Columns

Refer to the following for explanations of the columns on this panel.

FCN

Function assigned to the column listed at right.

AVG

Average value of the column at calculation.

MAX

Highest value in the column at calculation.

MIN

Lowest value in the column at calculation.

CNT

Number of times this column was printed at calculation.

SUM

Sum of values in this column at calculation.

COLUMN NAME: Name of each column to which you assigned a column function to be calculated at control breaks or report end.

Column 1

No leading zeros, no commas, decimal places as required to display number.

Column 2

No leading zeros, comma separators, decimal places as required to display number.

Column 3

Leading zeros, comma separators, decimal places as required to display number.

Column 4

Floating dollar sign ($), comma separators, two decimal places.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel or step.

<PF4> CONTINUE

Accepts the current entries and displays the next panel. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF7> BACKWARD

Displays the previous panel, if any.

<PF8> FORWARD

Displays the next panel, if any.

(DQHA0) Select Columns for Calculation

```
=>

--------------------------------------------------------------------------------DQHA0
DATAQUERY:  SELECT COLUMNS FOR CALCULATION         QUERY NAME: _______________
--------------------------------------------------------------------------------
SELECT THE COLUMN(S) YOU WISH TO USE IN CALCULATING THE VALUE OF THE TEMPORARY
RESULT ___total amt_____________________________  WITH PRECISION _2___ .
PLACE ANY CHARACTER NEXT TO THE DESIRED NAME(S).
--------------------------------------------------------------------------------
SELECT|  TABLE NAME/COLUMN NAME              |  DESCRIPTION
--------------------------------------------------------------------------------
  X   |  Quantity                           |
      |  Item Description                   |
      |  Suppliers                          |
  X   |  Price                              |
      |                                     |
      |                                     |
      |                                     |
      |                                     |
      |                                     |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, completion of the Guided Query Select Columns for Calculation panel is not required for each Temporary Result Column named. Use it only if your arithmetic expression will include data found during query execution in a column.

With it, you tell CA Dataquery which columns you will use in an arithmetic expression to create the results you want. You can use up to ten columns. You will also be allowed to include literal values in your calculation.

Note: For more information about defining arithmetic expressions, see the chapters on query syntax. An introduction to this panel appears in the *CA Dataquery User Guide*.

Action

Select a column from the list of numeric columns for the files you selected that you want to be used in defining an arithmetic expression by placing a character in the select column next to the column names. Press Enter or <PF4> CONTINUE to go to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name assigned this query with the first Guided panel.

RESULT

Name you assigned to the temporary result field you are currently defining.

WITH PRECISION

Precision you assigned the temporary result column you are currently defining, or the default (13.2) if none was assigned.

Columns

TABLE NAME/ COLUMN NAME

Numeric fields referenced in the query that may be used in the calculation that defines the temporary result column contents.

DESCRIPTION

The Datadictionary description of each column listed.

Entry Fields

Refer to the following chart for valid entries in the column on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SELECT	Y	Any character	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Define Temporary Result (DQH90) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHB0) Define the Arithmetic Expression

```
=>
Press <PF1> for help with writing the arithmetic expression
-------------------------------------------------------------------------------DQHB0
DATAQUERY:  DEFINE THE ARITHMETIC EXPRESSION       QUERY NAME: _______________
-------------------------------------------------------------------------------
Write an arithmetic expression to be used in calculating the value of
temporary result  _____________________________  with precision _____.
Use the column numbers, F01, F02, etc., numeric literals, or a combination of
both and arithmetic operators.
The default calculation appears below. You may type over it to change it.
=> ____________________________________________________________________________
-------------------------------------------------------------------------------
NUMBER| TABLE NAME/COLUMN NAME               |  DESCRIPTION
-------------------------------------------------------------------------------
      |                                      |
      |                                      |
      |                                      |
      |                                      |
      |                                      |
      |                                      |
      |                                      |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, the Guided Query Define the Arithmetic Expression panel is required only if you named a Temporary Result. One panel appears for each Temporary Result you named. With it, you type an algebraic equation that may be simple or complex, can contain numbers like F01 and F02 that represent column names, or literal values. An introduction to this panel appears in the *CA Dataquery User Guide*. You can define any arithmetic expression on this panel that could be defined in a DQL SET statement.

Action

Accept the default expression or enter a valid calculation whose result will provide the value of the temporary result for each row for which the calculation is made. You can write over the default expression. For more information about arithmetic expressions in temporary result calculations, see the chapters on query syntax.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned to this query.

Result

Name you assigned to the temporary result field you are currently defining.

With Precision

Precision you assigned the temporary result column you are currently defining, or 13.2 default precision.

Columns

Refer to the following for a discussion of each column on this panel.

NUMBER

Number CA Dataquery has assigned each numeric column you selected for use in structuring your calculation.

TABLE NAME/ COLUMN NAME

Names of the columns you selected for use in structuring your calculation, along with their table names.

DESCRIPTION

Datadictionary description of each column you selected for use in structuring your calculation.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
default calculation expression	Y	Column numbers, numeric literals, arithmetic operators, parentheses	Addition of all columns

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Select Columns for Calculation (DQHA0).

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

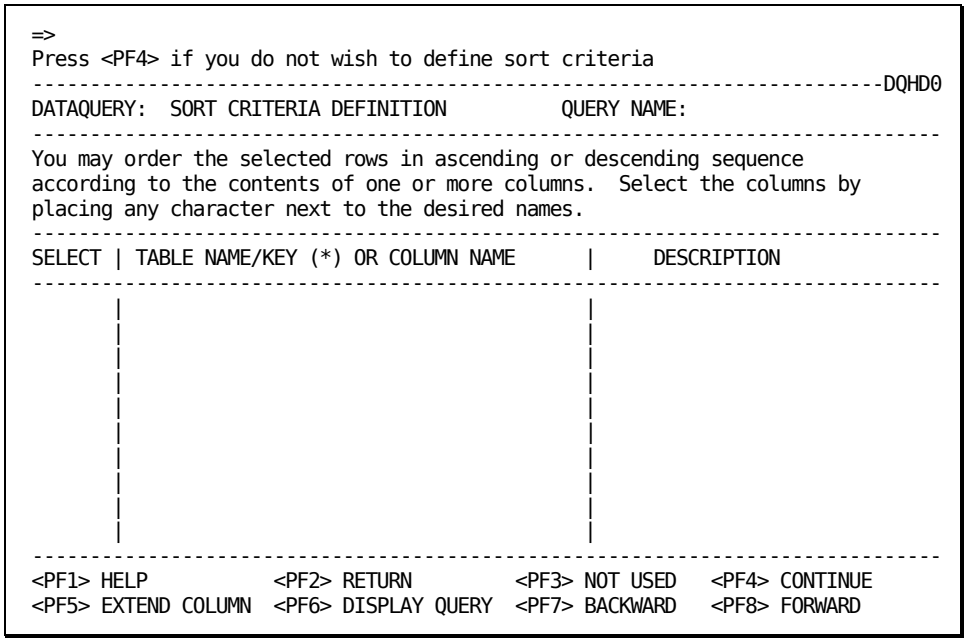
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHD0) SORT Criteria Definition



Description

In DQL Mode, the optional Guided Query Sort Criteria Definition panel allows you to order rows in your report in ascending or descending sequence according to the contents of one or more database or temporary results. With this capability, you can group rows together for purposes of reporting. Please see the *CA Dataquery User Guide* for an introduction to using this panel.

Action

Select from one to ten columns or keys whose values you want sorted to determine the order of rows in your query output. Place any character in the Select column next to your choices and press <PF4> CONTINUE or Enter to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME/ KEY(*) OR COLUMN NAME

The names of columns or keys in the tables you selected for this query. If a name is asterisked, it is a key. Keys are preferred over columns for faster processing.

DESCRIPTION

The Datadictionary description assigned each column.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SELECT	Yes, if you want to sort	Any character	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Define Temporary Result (DQH90) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHE0) Row Ordering

```
=>

--------------------------------------------------------------------------------DQHE0
DATAQUERY:  ROW ORDERING                       QUERY NAME:
--------------------------------------------------------------------------------
If you chose more than 1 sort column, number them 1, 2, etc. to show their
sort priority from major to minor. Defaults are provided and can be modified.
Sort sequence, control breaks, and paging control may be selected
by placing any character in the corresponding columns.
--------------------------------------------------------------------------------
ORDER | TABLE NAME/                            |SORT SEQUENCE |CONTROL | NEW
NUMBER|  KEY OR COLUMN NAME                    |ASCEND/DESCEND| BREAK  | PAGE
--------------------------------------------------------------------------------
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
      |                                        |              |        |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, the Guided Query Row Ordering panel appears when you select a sort column. On it are listed the columns selected in the order in which they were listed on the previous panel. Their order on the Row Ordering panel indicates sorting priority from major to minor. You can change the priority.

This panel also allows you to specify whether you want to sort from lowest to highest or highest to lowest value. Especially important, this panel lets you name the sort columns, allow control breaks, and start a new report page for each logical group created by sorting. An introduction to this panel appears in the *CA Dataquery User Guide*.

Action

Accept the default sorting order or change the numbers to reflect the sorting priority for columns you selected with the previous panel. Indicate ascending or descending order for each column and, if you want a control break taken when the values change for any column, enter a character in the control break column for the columns.

If you want the output to begin a new page when the value in a sort column changes, place a character in the New Page column next to the column name. Press Enter or <PF4> CONTINUE to go on.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

ORDER NUMBER

Enter a new number in this column for one or more columns.

TABLE NAME/KEY OR COLUMN NAME

Names of columns or keys selected as sort columns and their table names.

SORT SEQUENCE

ASCEND/DESCEND

For each column/key name, place a character in the ASCEND/ column under ascend if you want to sort from lowest to highest, under descend if you want to sort from highest to lowest.

CONTROL BREAK

For each column/key name, place a character in this column if you want totaling or calculations to be performed when the value (contents) of a column changes.

NEW PAGE

For each column/key name, place a character in this column if you want the report to begin a new page when the contents of the column/key changes.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Sort Criteria Definition (DQHD0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHF0) Report Definition

```
=>

--------------------------------------------------------------------------------DQHF0
DATAQUERY:  REPORT DEFINITION                QUERY NAME:
--------------------------------------------------------------------------------

Enter up to two lines for a
report title:      =>
                   =>

Select the desired report format: X  Columnar format: one output row per line.
                                  _  List format: one output row per page.

Enter the number of rows to be
processed for the report:      => ALL









--------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN         <PF3> NOT USED     <PF4> CONTINUE
<PF5> NOT USED     <PF6> DISPLAY QUERY  <PF7> NOT USED     <PF8> NOT USED
```

Description

In DQL Mode, the Guided Query Report Definition Panel is not required. However, if you want to assign a title to your report, display your data one row per page instead of in columns, or select fewer than all rows for the report, you will want to make some entries on this panel.

Action

Complete the columns or accept all defaults. Then press Enter or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Enter up to two lines for a report title:	N	1-55 alphanumeric characters each line	None
Columnar format:	N	Any character	X
List format:	N	Any character	None
number of rows	N	Any number 1-99999	All

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Sort Criteria Definition (DQHD0).

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF6> DISPLAY QUERY

Displays the query created so far.

(DQHG0) Report Column Definition

```
=>

--------------------------------------------------------------------------------DQHG0
DATAQUERY:  REPORT COLUMN DEFINITION          QUERY NAME:
--------------------------------------------------------------------------------
Select the keys or columns to be included on the report by placing any
character next to the desired names.
--------------------------------------------------------------------------------
SELECT|TABLE/KEY(*) OR COLUMN NAME             |     DESCRIPTION
--------------------------------------------------------------------------------
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
      |                                       |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, the required Guided Query Report Column Definition panel allows you to choose a number of the keys and columns from the tables you selected for your query, depending on how complex your query has become.

If you plan to create your own titles and reformat numeric data in a later step, your query could become quite lengthy and exceed the maximum line limit of the Editor established at your site. (See your CA Dataquery Administrator regarding line limits.)

If you have displayed your query with each step, you have an idea of how many lines it occupies. The contents of each key or column selected, including temporary result columns, can be printed on the report. An introduction to this panel appears in the *CA Dataquery User Guide*.

Action

Select at least one key or column whose values are to print on the query output. Enter any character in the Select column next to your selection or selections and press Enter or <PF4> CONTINUE to go on to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

TABLE/KEY(*) OR COLUMN NAME

Lists all columns and keys in tables chosen along with table names.

DESCRIPTION

Datadictionary description of each key and column listed, if descriptions are defined.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SELECT	Yes, at least one	Any character	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Definition (DQHF0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

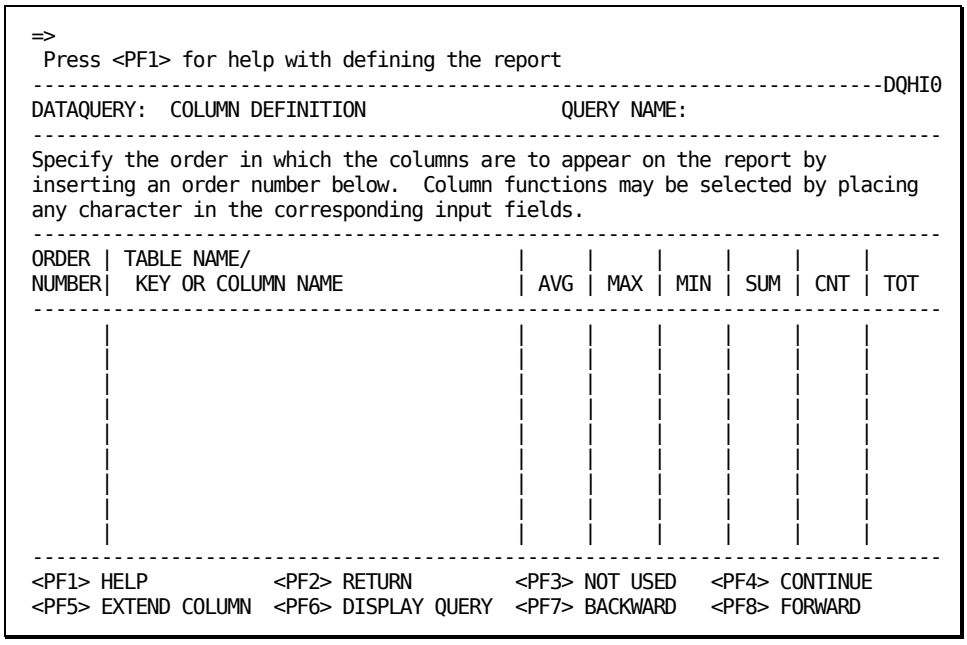
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHI0) Column Definition



Description

In DQL Mode, the optional Guided Query Column Definition panel can be accepted with the defaults CA Dataquery assigns, resulting in report data printed in columns in the same order in which it was selected. However, you may want to change the order in which columns are printed.

You may also want to request that mathematical functions be performed on some or all numeric columns. If so, you will select math functions for columns. At the end of the report, or at control breaks that you can specify later, you can obtain the results of the functions and specify a special legend to identify them on the report.

Refer to the *CA Dataquery User Guide* for a complete explanation of each available column function and an introduction to using this panel.

Action

Number the columns in the order wanted in the Order Number column. Enter any character in a column function column you want to perform on a particular column. Press Enter or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name assigned the query you are defining.

TABLE NAME/ KEY OR COLUMN NAME

Lists all columns and keys in tables chosen along with table names.

Columns

Refer to the following for a discussion of each column on this panel.

ORDER NUMBER

Numbers appearing in this column are assigned automatically by CA Dataquery. Change the numbers as needed by typing over the current number to indicate the order of columns on the report.

AVG

Enter a character in this column next to any column or key for which you want CA Dataquery to produce an average at control break or at the end of the report.

MAX

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the maximum amount found at control break or at the end of the report.

MIN

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the minimum amount found at control break or at the end of the report.

SUM

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a total at control break or at the end of the report.

CNT

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a count of the data items in this column at control break or at the end of the report.

TOT

Enter a character in this column next to any column or key for which you want CA Dataquery to produce totals at control breaks and at the end of the report and for which you do not need to assign legends different from those assigned automatically by CA Dataquery.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Definition (DQHG0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

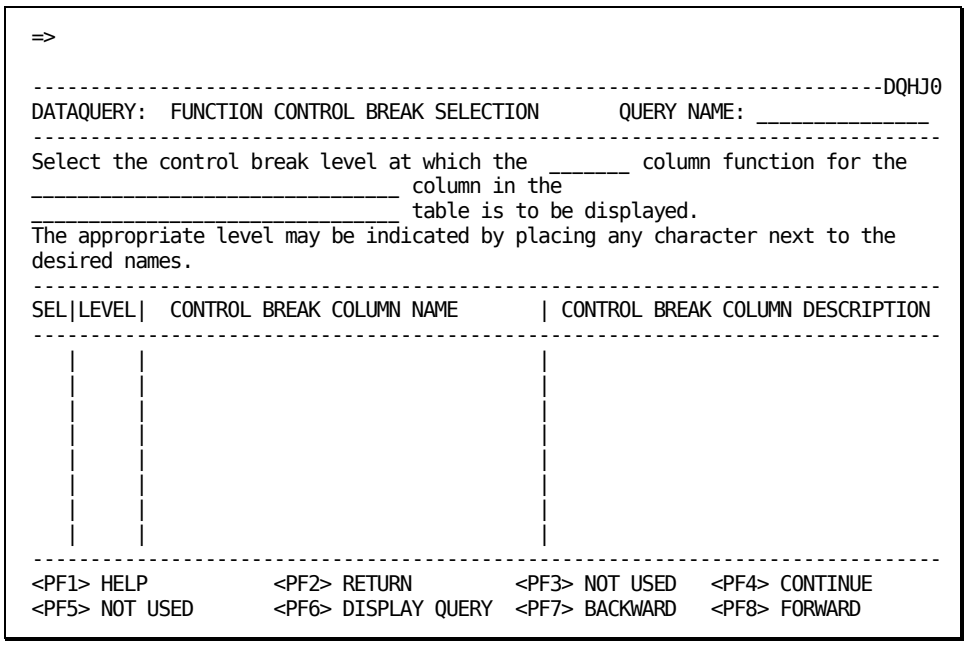
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHJ0) Function Control Break Selection



Description

In DQL Mode, the optional Guided Query Function Control Break Selection panel allows you to tell CA Dataquery when to print the results of a function (shown on panel) you selected for the column named on this panel.

One Function Control Break Selection panel appears for each function selected for each column. You can choose to produce function results at each control break (when the value in the column changes) or you can accept the default and print the result at the end of the report.

If you specified more than one sort column with the SORT CRITERIA DEFINITION (DQHD0) panel, and assigned control breaks to those columns with the ROW ORDERING (DQHE0) panel, each has a *level* or priority that appears on this panel. End of report priority is 000. You select the control breaks at which you want this function performed.

Action

Choose one or more control breaks or end of report and press ENTER or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

____ column function

Function selected for this column on the Column Definition panel (DQHI0). Can be AVG, MAX, MIN, SUM or CNT.

_____ column

Name of the column/key to which this column function was designated for production at control break or end of report.

_____ table

Name of the database table containing the column/key for this column function.

Columns

Refer to the following for a discussion of each column on this panel.

SEL

Place a character next to the levels at which you want the current column function performed.

LEVEL

You assigned a sort priority on the Row Ordering (DQHE0) panel for each sort column you selected for this query. Each has a priority number or level with 001 indicating the major sort column. Level 000 is assigned to the end of the report.

CONTROL BREAK/COLUMN NAME

This column lists the table and column name for each sort column you selected for this query.

CONTROL BREAK/COLUMN DESCRIPTION

The Datadictionary description, if any, attached to the column listed at the left. Also lists the END OF REPORT notation for the 000 level selection option.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHK0) Report Column Heading Definition

```
=>

--------------------------------------------------------------------------------DQHK0
DATAQUERY:  REPORT COLUMN HEADING DEFINITION    QUERY NAME: _______________
--------------------------------------------------------------------------------
You may enter alternate headings for any column on the report.
Defaults are provided and can be modified.
--------------------------------------------------------------------------------
COLUMN:                                 LINE1:
TABLE :                                 LINE2:

COLUMN:                                 LINE1:
TABLE :                                 LINE2:

COLUMN:                                 LINE1:
TABLE :                                 LINE2:

COLUMN:                                 LINE1:
TABLE :                                 LINE2:

COLUMN:                                 LINE1:
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, the optional Guided Query Report Column Heading Definition panel lets you define your own titles for each column or key printed in your query output. If you do not define a title, CA Dataquery supplies one using the Datadictionary alternate heading.

If no Datadictionary alternate heading exists, the column name is used. The default names are shown in the LINE1: input columns for each column selected for the report.

Action

Enter a one or two line title for each column heading in the columns provided for each column. Type over the default column names, or press Enter or <PF4> CONTINUE to go on to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

COLUMN:

Each COLUMN display column contains the name of a key or column you selected to appear in the report. Temporary results are not shown since you assign them a name yourself when you create them.

TABLE:

Each TABLE display column contains the table name of the key or column paired with it in the COLUMN display column. If it is a temporary result, no table name is listed. Instead, you see the legend *temporary result.*

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
LINE1:	N	1-32 characters except slash (/)	Alternate Datadictionary heading or field name

Field	Required?	Valid Entries	Default
LINE2:	N	1-32 characters except slash (/)	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Column Definition (DQHI0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

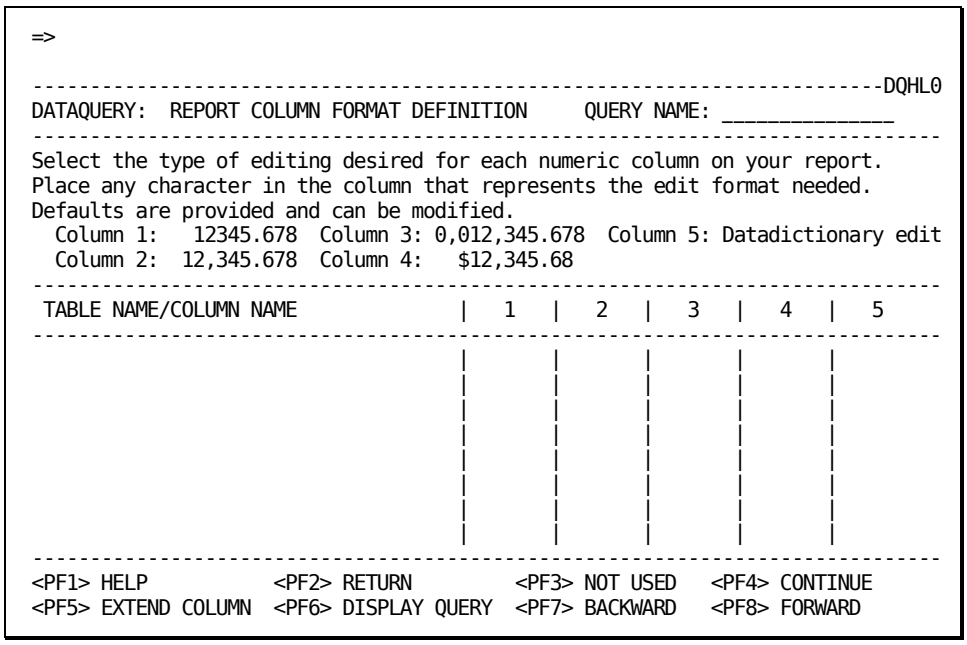
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHL0) Report Column Format Definition



Description

In DQL Mode, the optional Guided Query Report Column Format Definition panel appears if any numeric column is reported in the query output. It allows you to select a format for the appearance of the numeric data or accept the default.

Action

Enter a character in the column next to each column for an alternate edit pattern that you want to use and space over the default selection or press Enter or <PF4> CONTINUE to accept the default.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

TABLE NAME/ COLUMN NAME

The table and column names for each numeric column you selected for the report.

Columns

Refer to the following for a discussion of each column on this panel.

Column 1

Suppress leading zeros, no commas, decimal places as required to display numeric column.

Column 2

Suppress leading zeros, use commas and decimal places as required.

Column 3

Show leading zeros, use commas and decimal places as required.

Column 4

Using floating dollar sign, commas, and two decimal places.

Column 5

Use the Datadictionary edit pattern assigned to the column/key, if one exists. The text display for a column shows the Datadictionary edit pattern, if one exists.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Heading Definition (DQHK0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected.

<PF6> DISPLAY QUERY

Displays the query created so far.

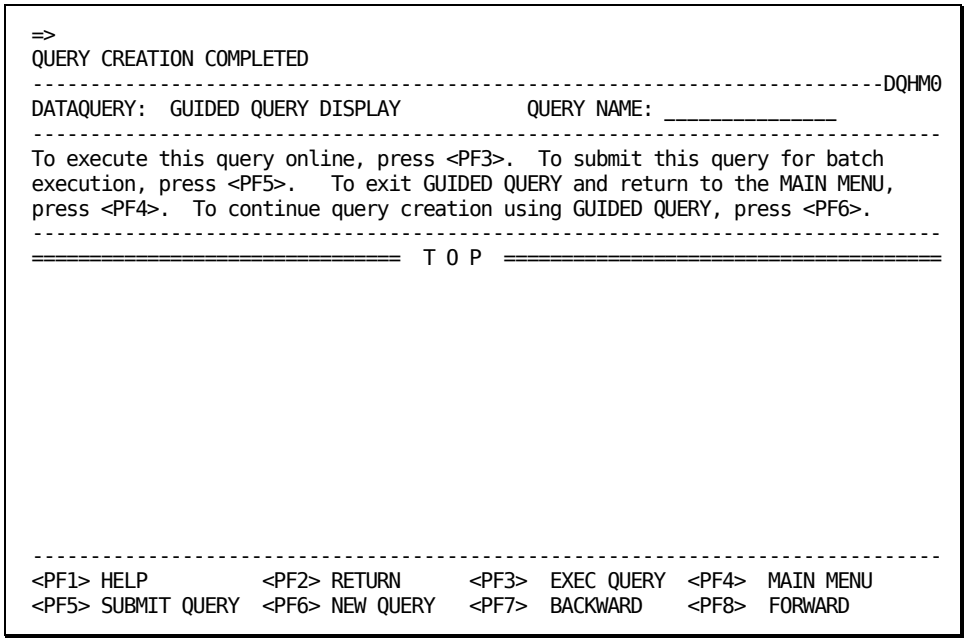
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHM0) Guided Query Display



Description

The Guided Query Display (DQHM0) panel appears when you complete the last Guided Query Creation function panel. It displays the query as it appears with all the specifications you entered translated into DQL statements. The query shown, whether complete or incomplete, resides in the library you selected when you named the query (private or public).

You can, at any time, exit Guided Query and edit the query by selecting it from the library and editing it on the Editor panel. However, you cannot return to Guided Query Creation to resume creation of this query after exiting Guided Query.

Action

Use the display as a learning tool to observe how the Guided Query function uses DQL to construct a query from the entries and selections you make.

There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Query display area

Contains the query built for you by CA Dataquery containing all of the specifications you entered on the Guided Query panels.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> EXEC QUERY

Executes the query.

<PF4> MAIN MENU

Displays the next Guided Query panel.

<PF5> SUBMIT QUERY

Submits query for batch execution.

<PF6> NEW QUERY

Begins creating another query with the Guided Query Creation function.

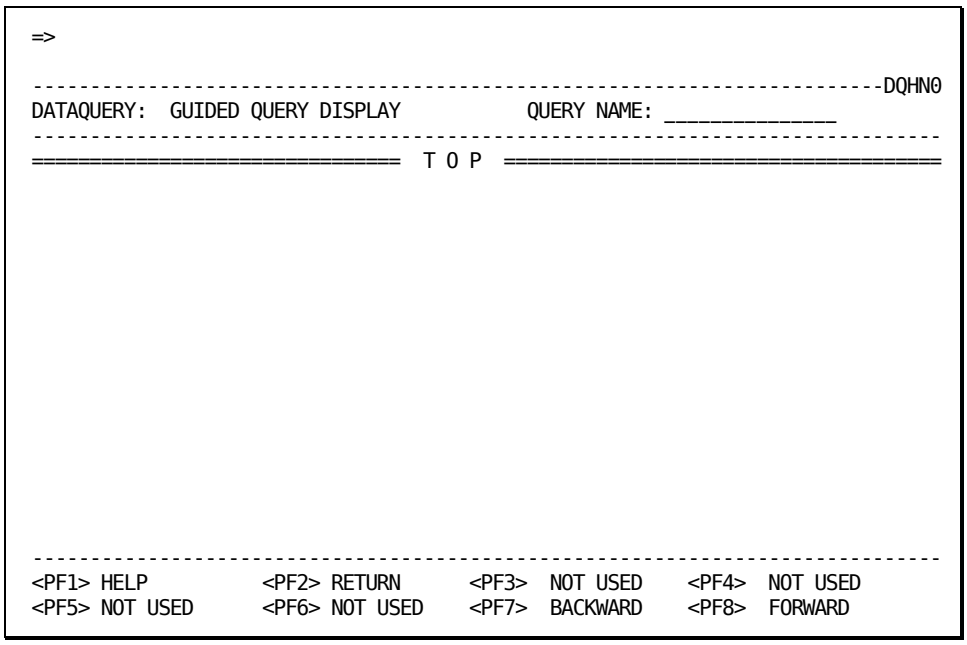
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHN0) Guided Query Display



Description

The Guided Query Display panel appears when <PF6> DISPLAY is pressed during display of any Guided Query Creation function panel after a selection is made on the Primary Table Selection panel. It displays the query as it appears at the time you press <PF6>. The query shown, whether complete or incomplete, resides in the library you selected when you named the query (private or public).

You can at any time, exit Guided Query and edit the query by selecting it from the library and editing it on the Editor panel. However, you cannot return to Guided Query Creation to resume creation of this query after exiting Guided Query.

Action

Use the display as a learning tool to observe how the Guided Query function uses DQL to construct a query from the entries and selections you make.

There are no fields on this panel for user entries.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Query display area

Contains the query built for you by CA Dataquery containing all of the specifications you entered so far on the Guided Query panels.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

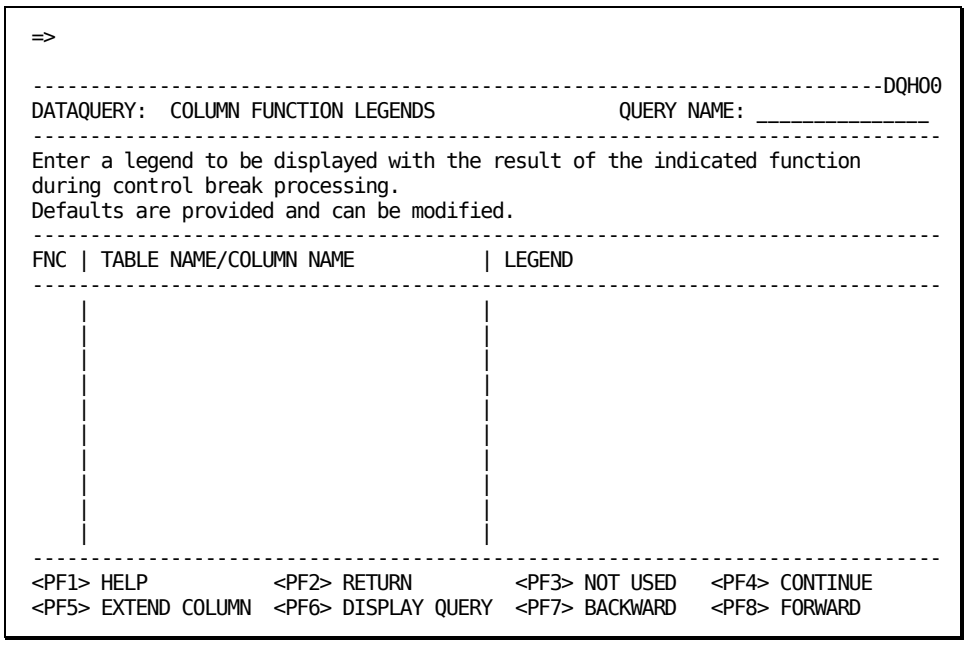
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHO0) Column Function Legends



Description

In DQL Mode, the optional Guided Query Column Function Legends panel lists each column function you specified with the Column Definition (DQHI0) panel. This panel allows you to assign a legend to the printed results of each column function. The legend you define appears next to the function results when they appear on the query output. An introduction to this panel appears in the *CA Dataquery User Guide*.

Action

Accept the default legend or type a new one over it. You can use a heading substitution string in the Legend field. If you want the current value of the control break printed within this legend, you can include a site-specified heading substitution string like && in the legend. See the System Profile panel in Topic Help to learn what characters are used at your site as a heading substitution string. Press Enter or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

Columns

Refer to the following for explanations of the columns on this panel.

FNC

Function assigned to the column listed at right.

AVG

Average value of the column at calculation.

MAX

Highest value found in the column at calculation.

MIN

Lowest value found in the column at calculation.

CNT

Number of times this column was printed at calculation.

SUM

The sum of the values in this column at calculation.

TABLE NAME/ COLUMN NAME

Names of columns to which you assigned a column function to be calculated at control breaks or end of report.

LEGEND

Enter a 1- to 38-character legend to be printed when functions are calculated and results are printed. You can use a heading substitution string in the Legend field. If you want the current value of the control break printed within this legend, you can include a site-specified heading substitution string like && in the legend. See the System Profile panel in Topic Help to learn what characters are used at your site as a heading substitution string.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Report Column Format Definition (DQHL0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel.

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQHP0) Function Result Print Format

```
=>

--------------------------------------------------------------------------------DQHP0
DATAQUERY:  FUNCTION RESULT PRINT FORMAT        QUERY NAME: _______________
--------------------------------------------------------------------------------
Select the type of editing desired for each function result.
Place any character in the column that represents the edit format needed.
Defaults are provided and can be modified.
   Column 1 =     12345.678              Column 3 =0,012,345.678
   Column 2 =    12,345.678              Column 4 =  $12,345.67
-------------------------------------------------------------------------
FCN | TABLE NAME/COLUMN NAME               |   1  |   2  |   3  |   4  |
-------------------------------------------------------------------------
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
    |                                      |      |      |      |      |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, this Guided Query panel permits you to change the display characteristics (numeric format or edit pattern) of any numeric data that appears in the columns that display the result of calculations selected as column functions. The default is indicated by an X in a column next to each function selection.

Action

Select a numeric format for each column function result by placing a character in the numbered column on this panel that represents the format you want the calculation result to have in the output.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

Columns

Refer to the following for explanations of the columns on this panel.

FCN

Function assigned to the column listed at right.

AVG

Average value of the column at calculation.

MAX

Highest value in the column at calculation.

MIN

Lowest value in the column at calculation.

CNT

Number of times this column was printed at calculation.

SUM

Sum of values in this column at calculation.

TABLE NAME/ COLUMN NAME

Name of each column to which you assigned a column function to be calculated at control breaks or report end.

Column 1

No leading zeros, no commas, decimal places as required to display number.

Column 2

No leading zeros, comma separators, decimal places as required to display number.

Column 3

Leading zeros, comma separators, decimal places as required to display number.

Column 4

Floating dollar sign ($), comma separators, two decimal places.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel or step.

<PF4> CONTINUE

Accepts the current entries and displays the next panel. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the panel that provides additional information about a column definition for the column indicated by the cursor.

<PF6> DISPLAY QUERY

Displays the query so far.

<PF7> BACKWARD

Displays the previous panel, if any.

<PF8> FORWARD

Displays the next panel, if any.

(DQHQ0) Multiple Occurrence Selection

```
=>

--------------------------------------------------------------------------------DQHQ0
DATAQUERY:  MULTIPLE OCCURRENCE SELECTION          QUERY NAME: _______________
--------------------------------------------------------------------------------
The column you have selected has multiple occurrences.  You may select the
individual occurrences to be displayed on the report by placing the occurrence
numbers in the fields provided below.
--------------------------------------------------------------------------------
  TABLE NAME:                                      COLUMN LENGTH:
  COLUMN NAME:                                     COLUMN TYPE  :
  DESCRIPTION:
  NUMBER OF OCCURRENCES:
--------------------------------------------------------------------------------









--------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN        <PF3> NOT USED      <PF4>  CONTINUE
```

Description

In DQL Mode, the Guided Query Multiple Occurrence Selection panel appears for each column selected during Guided Query creation that is a repeating column.

Action

Type numbers of the occurrences you want to use in the columns indicated by cursor position. Press the Tab key to move the cursor from one column to the next. The number of columns available for entry equals the number of occurrences shown in the NUMBER OF OCCURRENCES display, to a maximum of 96. Press <PF4> CONTINUE or Enter to go on to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

TABLE NAME:

Name of table containing the repeating column.

COLUMN NAME:

Name of the repeating column whose occurrences you are to select for the report.

DESCRIPTION:

Datadictionary description of the column.

NUMBER OF OCCURRENCES:

Total number of occurrences of the named column.

COLUMN LENGTH:

Number of positions in the column.

COLUMN TYPE:

One of the following:

C

Character

N

Numeric

S

Short floating-point

L

Long floating-point

E

Extended floating-point

Y

Double character

Z

Double character mixed

K

Kanji

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
At cursor position	N	1- to 5-digit number representing the number of occurrences shown in the NUMBER OF OCCURRENCES display.	00001

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

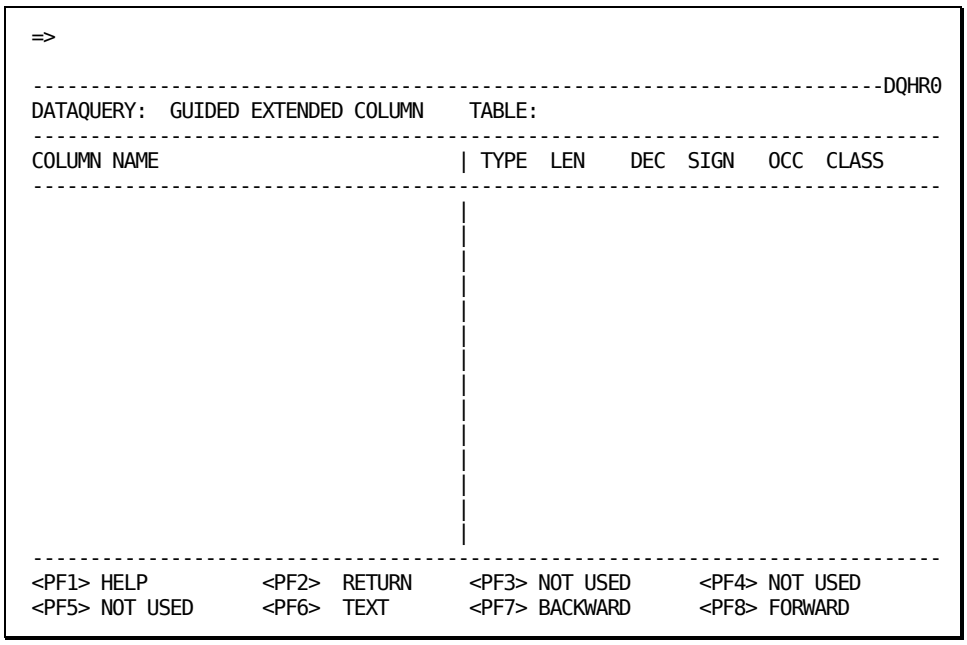
<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Goes to the next Guided Query Creation function panel.

(DQHR0) Extended Column Display



Description

In DQL Mode, this panel displays additional Datadictionary attributes not shown on the Guided Query list of column names. Display this panel by selecting the EXTENDED DEF PF key where provided in Guided Query.

Action

When this panel is displayed, you can view the Datadictionary attributes of the column.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

TABLE:

Datadictionary name of the table whose columns are listed on the panel.

Columns

Refer to the following for explanations of the columns on this panel.

COLUMN NAME

Datadictionary name of the column for the table you selected.

TYPE

One of the following:

C

Character

E

Extended floating-point

G

Graphics

H

Hex

K

Kanji

L

Long floating-point

N

Numeric (including binary)

S

Short floating-point

T

Bit for PL/1

Y

Double character

Z

Double character mixed

LEN

Number of positions.

DEC

Number of positions after the decimal.

SIGN

If a numeric column, Y (has a + or - sign) or N (is unsigned).

OCC

Number of occurrences. 1 indicates column is not repeating.

CLASS

Type of column:

C

Compound for group columns.

S

Simple for single columns.

F

Filler for place-holder columns.

Reported as Numeric

Packed, zoned, and half word binary types and lengths are reported as numeric and may not indicate an appropriate match for RELATED BY statements.

DATE, TIME, or TIMESTAMP

When creating queries, do not include columns defined as data types DATE, TIME, or TIMESTAMP. These data types are reserved for SQL Mode queries. These data types are listed as character types on Extended Column Display panels because SQL Mode queries return character-type data.

To test a column that is suspected of being one of these types before selecting it for a DQL Mode query, follow these steps.

1. Look at the length of the column. DATE length is always 10, TIME is always 8, and TIMESTAMP is always 26.
2. If the data type is suspect, create a query that finds one row and PRINT the output.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF6> TEXT

Displays text from Datadictionary on TEXT DISPLAY (COLUMN) (DQB40) panel.

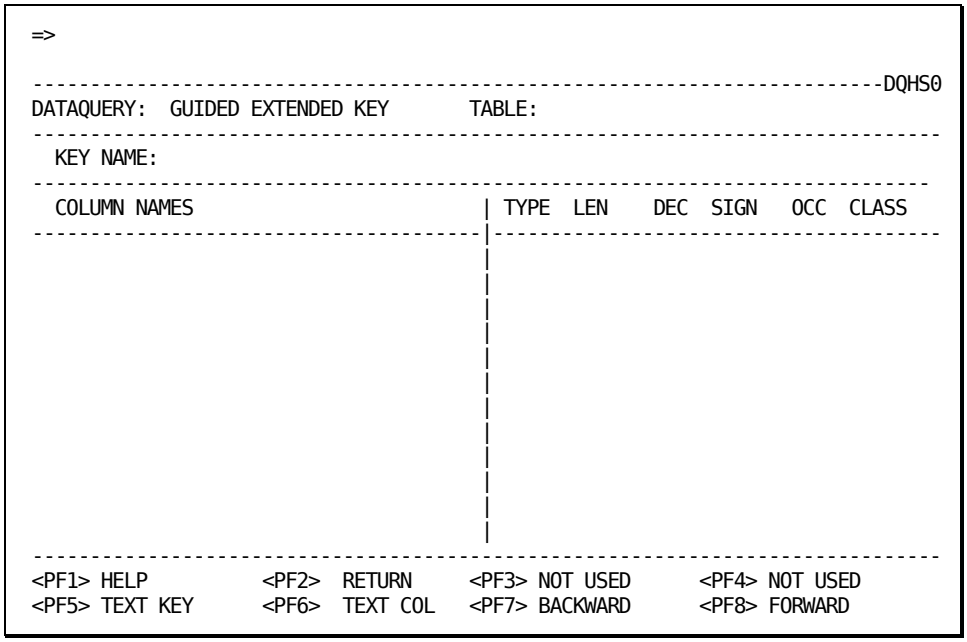
<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQHS0) Guided Extended Key



Description

In DQL Mode, this panel displays the Datadictionary description of any key shown on the Guided Query list containing key names. Display this panel by selecting the EXTENDED DEF PF key on panels where it is displayed as an option.

Action

When this panel is displayed, you can view the Datadictionary descriptions of the key.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE:

DATADICTIONARY name of the table whose keys and columns are listed on the panel.

KEY NAME:

Datadictionary description of the key name you have selected.

Columns

Refer to the following for explanations of the columns on this panel.

COLUMN NAMES

Datadictionary name of the key and columns in the key you selected.

TYPE

One of the following:

C

Character

N

Numeric (including binary)

E

Extended floating-point

S

Short floating-point

G

Graphics

T

Bit for PL/1

H

Hex

Y

Double character

K

Kanji

Z

Double character mixed

L

Long floating-point

LEN

Number of positions.

DEC

Number of positions after the decimal.

SIGN

If a numeric key/column, + or - sign.

OCC

Number of occurrences, if a repeating key/column.

CLASS

Type of column:

C

Compound for group keys/columns.

S

Simple for single keys/columns.

F

Filler for place-holder keys/columns.

Reported as Numeric

Packed, zoned, and half word binary types and lengths are reported as numeric and may not indicate an appropriate match for RELATED BY statements.

DATE, TIME, or TIMESTAMP

When creating queries, do not include columns defined as data types DATE, TIME, or TIMESTAMP. These data types are reserved for SQL Mode queries. These data types are listed as character types on Extended Column Display panels because SQL Mode queries return character-type data.

To test a column that is suspected of being one of these types before selecting it for a DQL Mode query, follow these steps.

1. Look at the length of the column. DATE length is always 10, TIME is always 8, and TIMESTAMP is always 26.
2. If the data type is suspect, create a query that finds one row and PRINT the output.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information.

<PF2> RETURN

Exits the current panel.

<PF5> TEXT KEY

Displays text for the key.

<PF6> TEXT COL

Displays Datadictionary text.

<PF7> BACKWARD

Scrolls the list backward.

<PF8> FORWARD

Scrolls the list forward.

(DQHT0) Primary Table Selection

```
=>

-------------------------------------------------------------------------------DQHT0
DATAQUERY:  PRIMARY TABLE SELECTION              QUERY NAME:  ________________
-------------------------------------------------------------------------------
Select one table from which you want information by placing any character
next to the desired table name.   START WITH:  ____________________________
-------------------------------------------------------------------------------
SELECT|    TABLE NAME                    |STATUS|     DESCRIPTION
-------------------------------------------------------------------------------
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
   _  |                                  |      |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> DISPLAY TEXT <PF4> CONTINUE
<PF5> DISPLAY TABLE <PF6> DISPLAY QUERY  <PF7> BACKWARD     <PF8> FORWARD
```

Description

In DQL Mode, the Guided Query Creation function required Primary Table Selection panel tells CA Dataquery which table to search (first if you want to search two tables) for the data you want. CA Dataquery reads the primary table in its search. It should contain all of the data you want if it is to be the only table read. An introduction to using this panel appears in the *CA Dataquery User Guide*.

Action

Place a character next to the name of the table you want the query to search and then press Enter or <PF4> CONTINUE.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

QUERY NAME:

Name you assigned the query.

Entry Fields

Refer to the following for valid entries in the field on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	1-15 character panel name.	No default

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to one database table name to select the secondary table you want your query to search.

TABLE NAME

Lists all database tables you are authorized to access. List can be continued on additional panels that you can display by scrolling forward with <PF8>.

STATUS

Current database status of each table listed:

O

Open and ready for use.

C

Closed and cannot be used at this time.

D

Disabled and cannot be used.

DESCRIPTION

Description of each table, if any.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY TEXT

Displays the text description of a table selected by cursor position.

<PF4> CONTINUE

Displays the next panel and saves this selection. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> DISPLAY TABLE

Displays the keys and columns for the table selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query so far.

<PF7> BACKWARD

Scrolls back to see previous panels of multi-screen panel.

<PF8> FORWARD

Scrolls forward to see next panel, if a multi-screen panel.

(DQHU0) Secondary Table Selection

```
=>
 Press <PF4> if information from a secondary table is not desired
------------------------------------------------------------------------------DQHU0
DATAQUERY:  SECONDARY TABLE SELECTION           QUERY NAME:  _______________
------------------------------------------------------------------------------
You can use information from the primary table ___________________________
to join with related information from another table. To join two tables,
select the second table by placing any character next to the desired name.

Retrieve the first related record (bypass other occurrences)?   Yes _  No X
------------------------------------------------------------------------------
SELECT|    TABLE NAME                   |STATUS|     DESCRIPTION
------------------------------------------------------------------------------
      |                                 |      |
      |                                 |      |
      |                                 |      |
      |                                 |      |
      |                                 |      |
      |                                 |      |
      |                                 |      |
      |                                 |      |
------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> DISPLAY TEXT <PF4> CONTINUE
<PF5> DISPLAY TABLE <PF6> DISPLAY QUERY  <PF7> BACKWARD     <PF8> FORWARD
```

Description

In the DQL Mode Guided Query Creation function, you can join your primary table (Primary Table Selection panel, DQH20) to a second table if they share a key or column. Then, you can report on data that is contained in both tables. Use this panel to select an optional secondary table for joining.

Action

Place a character next to the name of the table you want the query to search and then press Enter or <PF4> CONTINUE.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned the query.

primary table

Table selected first with DQH20 as the primary search table.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to one database table name to select the secondary table you want your query to search.

TABLE NAME

Lists all database tables you are authorized to access. List can be continued on additional panels that you can display by scrolling forward with <PF8>.

STATUS

Current database status of each table listed:

O

Open and ready for use.

C

Closed and cannot be used at this time.

D

Disabled and cannot be used.

DESCRIPTION

Description of each table, if any.

Entry Fields

Refer to the following for valid entries in the "Retrieve the first related row (bypass other occurrences)?" line. See the chapters on query syntax for more information about multiple occurrences.

Y

Place a character here and remove the X from NO if you want CA Dataquery to retrieve only the first related row when joining tables rather than all related rows to establish a relationship.

N

Leave X as the entry to indicate that CA Dataquery should retrieve all related rows in the secondary table to establish a relationship with the primary table.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY TEXT

Displays text description, if it exists, of a table selected by cursor position.

<PF4> CONTINUE

Displays the next panel. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> DISPLAY TABLE

Displays the keys and columns for the table selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query so far.

<PF7> BACKWARD

Scrolls back to see previous panels of multi-screen panel.

<PF8> FORWARD

Scrolls forward to see next panel, if a multi-screen panel.

(DQHV0) Guided Query Creation: Option Selection

```
=>
-------------------------------------------------------------------------------DQHV0
DATAQUERY:   GUIDED QUERY CREATION:  OPTION SELECTION
-------------------------------------------------------------------------------
When you create a query with Guided Query, a basic query will be generated as
the default.  You can override this default and customize your query by
selecting one or more of the extended options listed below.

To create a basic query that selects all rows from a single table, presenting
the selected rows in a random sequence, press <PF4>.

To select options to add to the basic query, place any character beside the
options of your choice and press <PF4>.

   _  Specify the order in which the retrieved information will be presented

   _  Limit the output report size by specifying selection criteria

   _  Create your own columns for the output report

   _  Retrieve and combine information from two tables

-------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN       <PF3> NOT USED       <PF4>  CONTINUE
```

Description

This panel appears after you select GUIDE from the Main Menu or issue the GUIDE command. It allows you to specify the type of query you want to create. The specifications you enter determine which panels appear during Guided Query operation.

Action

To create the simplest query, make no entries and press <PF4> CONTINUE. To add features to the query, enter a character in the field preceding one or more options and press <PF4> CONTINUE.

Select

Refer to the following for explanations of the selections available on this panel.

Specify the order . . .

Sort report data.

Limit the output . . .

Request specific rows to be selected.

Create your own columns . . .

Create data for the report by requesting calculations to be made.

Retrieve and combine . . .

Join two tables for data selection.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Displays first Guided Query input panel.

(DQH10) Query Identification

```
=>

-------------------------------------------------------------------------------DQH10
DATAQUERY:   QUERY IDENTIFICATION
-------------------------------------------------------------------------------

Enter a unique name for the new query ==>   _______________

Select the accessibility level for your query by placing any character
next to your choice.

   _  PUBLIC
   X  PRIVATE

Enter a description for the new query
  ==> ____________________________________________________________


-------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN        <PF3> NOT USED      <PF4>  CONTINUE
```

Description

The Guided Query Creation function Query Identification (DQH10) panel names a query, tells CA Dataquery which library to keep it in, and provides a short description for the library directory.

When you complete this panel and press Enter, the query resides in that library unless you delete it during a directory display. An introduction to completing this panel appears in the *CA Dataquery User Guide*.

Action

Complete the fields and press Enter or <PF4> to go to the next Guided Query panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Name for the new query	Y	Up to 15 characters starting with a letter.	None
PUBLIC/PRIVATE	Y	Choose public if you will not change the query.	Private
Description	N	Any characters.	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to CA Dataquery Main Menu.

<PF4> CONTINUE

Displays the next panel in the Guided Query Creation function.

(DQH40) Primary Table: Key Selection

```
=>

--------------------------------------------------------------------------------DQH40
DATAQUERY:  PRIMARY TABLE:  KEY SELECTION         QUERY NAME: _______________
--------------------------------------------------------------------------------
The primary table ___________________________________ can be joined with the
secondary table _____________________________________ by matching identical values
for a key or column appearing in both tables. Select the key or column to be
used from the primary table by placing any character next to its name.
--------------------------------------------------------------------------------
SELECT| KEY(*) AND COLUMN NAMES            |  DESCRIPTION
--------------------------------------------------------------------------------
      |                                    |
      |                                    |
      |                                    |
      |                                    |
      |                                    |
      |                                    |
      |                                    |
      |                                    |
      |                                    |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, you must complete this Guided Query panel if you selected a primary table and a secondary table for your query. The panel displays a list of keys and columns from which you can choose one that will contain a value that will match a value in a key or column of the secondary table. These matching values will join rows from each table for the query's search for data.

Action

Type a character in the Select column next to the name of the key or column to be used for joining with the secondary table, key or column.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

primary table_

The table you selected first.

secondary table_

The table you selected for CA Dataquery to read second.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the name of a key or column in the primary table whose values will match values in a key or column you will choose for the secondary table.

KEY(*) AND COLUMN NAMES

Keys and columns in the primary table. Scroll forward if list extends beyond the current panel.

DESCRIPTION

Description, if any, for each key and column.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

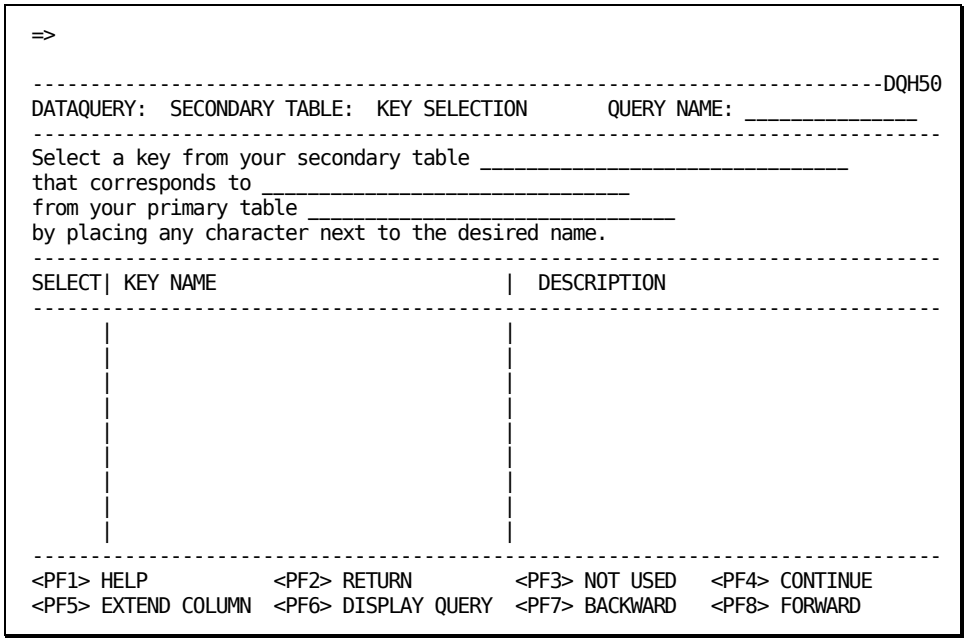
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQH50) Secondary Table: Key Selection



Description

In DQL Mode, you must complete this Guided Query panel if you selected a primary table and a secondary table for your query. The panel displays a list of keys and columns from which you can choose one that will contain a value that will match a value in a key or column of the primary table. These matching values will join rows from each table for the query's search for data. The list contains only valid keys that match the selected column or key from the primary table in length and type.

Action

Type a character in the Select column next to the name of the key or column to be used for joining with the primary table key or column.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

secondary table ___

The table you selected as the second one to be read in joining tables.

corresponds to _____

Key or column you selected from the primary table whose value will match values in the key you will select from the secondary table key names listed on this panel.

primary table __

The table you selected as the first one to be read in joining tables.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the name of a key in the secondary table whose values will match values in the key or column selected for joining in the primary table.

KEY NAME

Lists all keys in the secondary table. Scroll forward to see any additional names.

DESCRIPTION

Description, if any, of each key.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQH60) Selection Criteria: Designate Columns

```
=>
Press <PF4> if you do not wish to specify further selection conditions
-------------------------------------------------------------------------------DQH60
DATAQUERY: SELECTION CRITERIA: DESIGNATE COLUMNS   QUERY NAME: _______________
-------------------------------------------------------------------------------
Specify columns that will be evaluated for record selection by placing
any character next to the desired names.
-------------------------------------------------------------------------------
SELECT| KEY(*) AND COLUMN NAMES            |  DESCRIPTION
-------------------------------------------------------------------------------
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
      |                                   |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

In DQL Mode, the optional Guided Query Selection Criteria: Designate Columns panel allows you to begin narrowing the query's search for data. You can select up to 15 columns that contain data for which you can search, by stating that you want to see a range of data, an exact match, and so forth. You use this panel to begin specifying search criteria that tell CA Dataquery to read only rows that contain data you specify. You begin defining the selection criteria by naming the columns with this panel.

For instance, you can indicate that you only want rows that contain specific values in a particular column, like CITY = DALLAS. With this panel you would select CITY and on a following panel you would specify that CITY should contain DALLAS for a row to be selected. An introduction to using this panel appears in the *CA Dataquery User Guide*.

Action

Enter a character in the Select column next to any columns you want to use in specifying selection criteria for this query. Then press Enter or <PF4> CONTINUE to go to the next panel.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

QUERY NAME:

Name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the names of any columns for which you want to specify qualifying values.

KEY(*) AND COLUMN NAMES

Lists the names of all keys and columns in the tables you selected for this query. If two tables were selected, the primary table keys and columns are listed first, followed by the secondary keys and columns. Scroll forward to see additional panels if the list extends beyond the current panel.

DESCRIPTION

Description, if any, of each key and column listed.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection (DQH20) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column/definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQH70) Selection Criteria: Define Condition

```
=>

--------------------------------------------------------------------------------DQH70
DATAQUERY: SELECTION CRITERIA: DEFINE CONDITION    QUERY NAME: _______________
--------------------------------------------------------------------------------
Define the condition by selecting a comparison operator and a value to compare
with  _____________________________  in  _____________________________

Select one comparison operator:

_ Equal (EQ)       _ Greater than (GT)             _ Less than (LT)
_ Not equal (NE)   _ Greater than or equal (GTE)   _ Less than or equal (LTE)
_ Containing

Enter values for comparison with _____________________________

_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________

--------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> REPEAT COLUMN <PF4> CONTINUE
<PF5> EXTEND COLUMN <PF6> DISPLAY QUERY  <PF7> NOT USED      <PF8> NOT USED
```

Description

In DQL Mode, the optional Guided Query Selection Criteria: Define Condition panel appears for every column selected as a Selection Criteria column in the previous step. With it, you describe the data you want to be searched for in a column to qualify a row for inclusion in your report.

In a query, you can specify a selection criteria clause with the format *WITH operand1* operator *operand2.* You selected *operand1* with the Selection Criteria: Designate Columns (DQH60) panel. With this panel, you select the operator and *operand2*. The *operand2* parameter can be a column or key name, a value, or a logical expression that follows the rules for specifying *operand2* found in the chapters on query syntax. An introduction to using this panel appears in the *CA Dataquery User Guide*.

Action

Choose a comparison operator from the panel by entering a character next to the one that indicates how you want to compare the found column values to the values you enter at the bottom of the panel. You can choose only one operator.

Your entries on the blank lines provided must follow the rules for specifying comparisons in DQL WITH clauses found in the chapters on query syntax. When the panel is complete and correct, press Enter or <PF4> CONTINUE to go to the next step.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

compare with __

The key or column selected for comparison of the values you enter on this panel.

in ___

The name of the table containing the selected column.

Select

Make selections on this panel according to the following.

Equal (EQ)

Data in the column must be equal to the value you enter below for the row to be selected.

Greater than (GT)

Data in the column must be greater than the value you enter below for the row to be selected.

Less than (LT)

Data in the column named must be less than the value you enter below for the row to be selected.

Not equal (NE)

Data in the column named must not match the value you enter below for the row to be selected.

Greater than or equal (GTE)

Data in the column must be greater than or equal to the value you enter below for the row to be selected.

Less than or equal (LTE)

Data in the column named must be less than or equal to the value you enter below for the row to be selected.

Containing

Data in the column named must contain the value you enter below in the positions indicated by the value statement.

Entering Comparison Values

On the lines following "Enter values for comparison with", enter any valid comparison value, following the rules for specifying *operand2* of CA Dataquery selection criteria in the chapters on query syntax.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Selection Criteria: Designate Columns (DQH60) panel.

<PF3> REPEAT COLUMN

Lets you use the same column again in another comparison.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete.

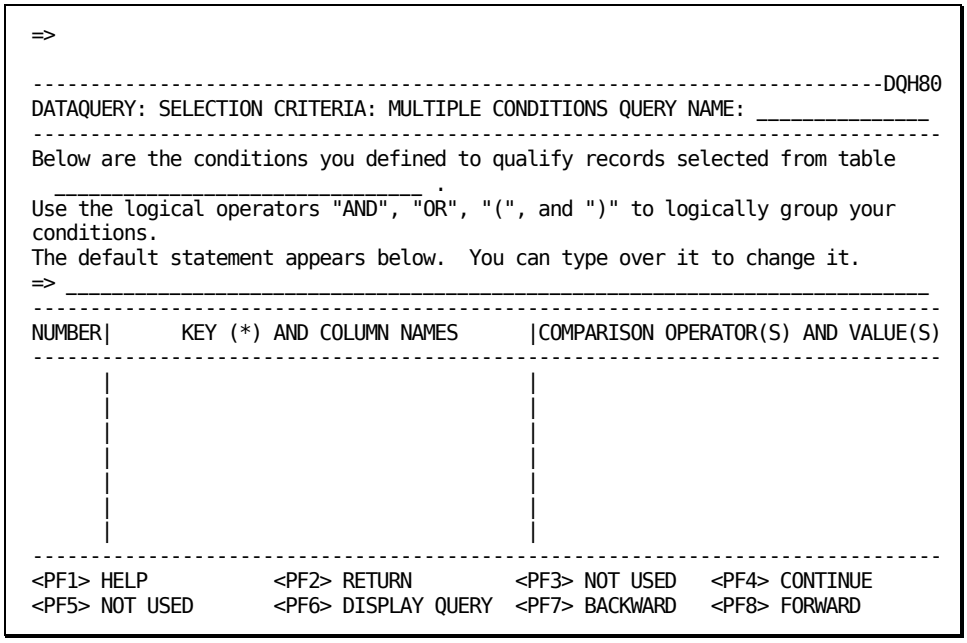
<PF5> EXTEND COLUMN

Displays the extended column definition for key or column.

<PF6> DISPLAY QUERY

Displays the query created so far.

(DQH80) Selection Criteria: Multiple Conditions



Description

This DQL Mode Guided Query panel appears when you select more than one column and condition per table for narrowing your query's search for data. You select columns that contain data you want to use in qualifying the search with a previous panel, Selection Criteria: Designate Columns. Then you specify a qualifying condition for each column with the Selection Criteria: Define Condition panel that appears for each column you name.

When those conditions are defined, this Multiple Conditions panel appears to allow you to group the conditions into a logical expression using AND, OR, and parentheses ( ) and the number assigned to each condition. CA Dataquery always *ands* each condition sequentially as the default expression.

Action

Type over the default logical expression with your own expression, using the numbers provided for each condition. Or, accept the default statement by pressing <PF4> CONTINUE.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

table_____

Name of the table containing the search column.

Columns

Refer to the following for a discussion of each column on this panel.

NUMBER

Number CA Dataquery assigned a column and its condition.

KEY (*) AND COLUMN NAMES

Names of columns/keys selected for qualifying the search for data.

COMPARISON OPERATOR(S) AND VALUE(S)

Conditions constructed for each column.

Entry Field

The following describes how you can use the entry field that contains the default statement.

default statement

A default logical expression combining all conditions by adding them together. You can write over this expression to tell CA Dataquery how to apply the qualifying conditions in the search for data.

Sample

```
=>

-------------------------------------------------------------------------------DQH80
DATAQUERY: SELECTION CRITERIA:MULTIPLE CONDITIONS QUERY NAME: MULTI-2
-------------------------------------------------------------------------------
Below are the conditions you defined to qualify rows selected from table
  CAI-RCPTS-REC                    .
Use the logical operators "AND", "OR", "(", and ")" to logically group your
conditions.
The default statement appears below.  You can type over it to change it.
=> (2 and 3) or (1 and 4)
-------------------------------------------------------------------------------
NUMBER|      KEY (*) AND COLUMN NAMES      |COMPARISON OPERATOR(S) AND VALUE(S)
-------------------------------------------------------------------------------
  01  |   RECEIPT-ID                       | GT     1
  02  |   REC-QTY                          | GTE    1
  03  |   ITM-ID                           | EQ     999
  04  |   REC-YR                           | EQ     86
      |                                    |
      |                                    |
      |                                    |
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

The preceding example tells CA Dataquery to retrieve only data from the CAI-RCPTS-REC table rows that contain a REC-QTY greater than or equal to 1 **AND** an ITM-ID equal to 999 **OR** rows that contain a RECEIPT-ID greater than 1 **AND** a REC-YR equal to 86.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Goes to the next Guided Query Creation step.

<PF6> DISPLAY QUERY

Displays the query created so far.

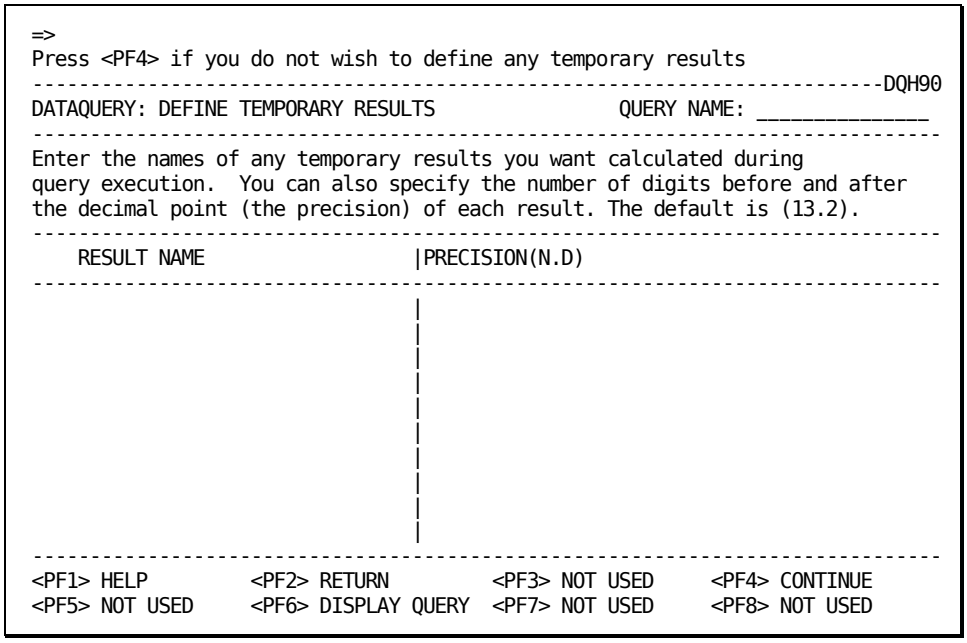
<PF7> BACKWARD

Scrolls to the previous panel, if any.

<PF8> FORWARD

Scrolls to the next panel, if any.

(DQH90) Define Temporary Results



Description

In DQL Mode, the optional Guided Query Define Temporary Results panel lets you name and define a length for as many as ten columns you want to create for your report that are not contained in the tables you selected.

You set up new columns when you want to know the results of calculating the contents of columns that do exist on the database. An introduction to this panel appears in the *CA Dataquery User Guide*.

Action

Enter a name for each new result you want to create and use the default precision or define a new precision, if known. Press Enter or <PF4> CONTINUE to go on to the next panel.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

RESULT NAME

Enter a 1- to 32-character name for a temporary that you want to create to contain the result of a calculation you will define on a later panel.

PRECISION(N.D)

Enter nothing to accept the default of 13 positions before the decimal point and two decimal positions (13.2) or enter the number of positions the result should have before and after the decimal point. For example, enter **6.3** to specify a result with six numbers and three decimal positions. N + D must be less than or equal to 18.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Selection Criteria: Designate Columns (DQH60) panel.

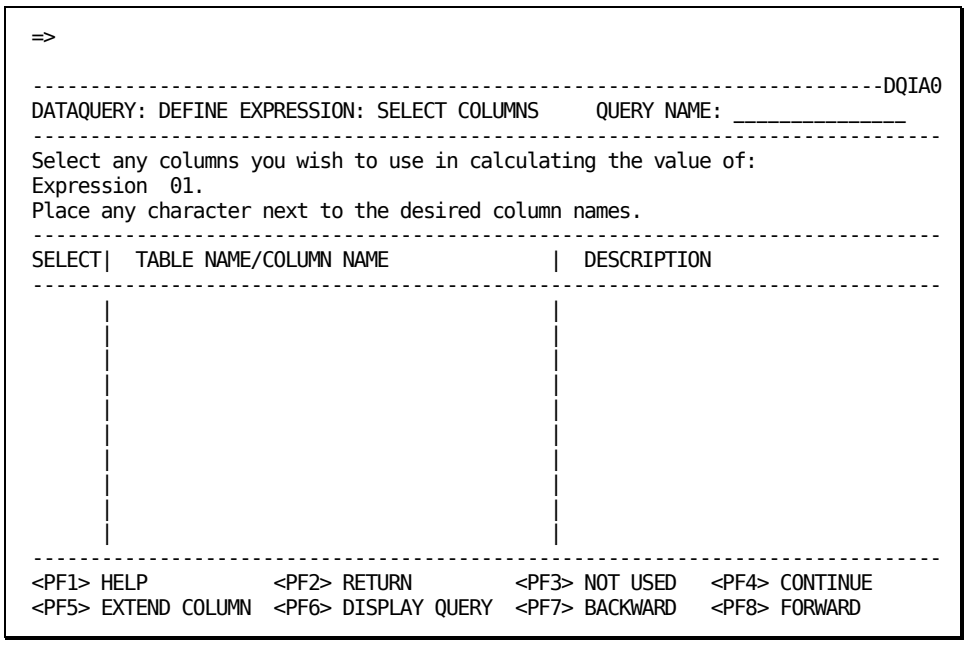
<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete.

<PF6> DISPLAY QUERY

Displays the query created so far.

(DQIA0) Define Expression: Select Columns



Description

This SQL Mode Guided Query panel helps you select columns for use in defining an expression for your query. Completion of the Define Expression: Select Columns panel is not required for each Result named. Use it only if your arithmetic expression will include data found during query execution in a column. With it, you tell CA Dataquery which columns you will use in an arithmetic expression to create the results you want. You can use up to ten columns. You will also be allowed to include literal values in your calculation.

Action

Select a column from the list of numeric columns for the tables you selected that you want to be used in defining an arithmetic expression by placing a character in the select column next to the column names. Press Enter or <PF4> CONTINUE to go to the next panel.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

QUERY NAME:

Name assigned this query with the first Guided panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SELECT	N	Any character	None

Columns

Refer to the following for a discussion of each column on this panel

TABLE NAME/ COLUMN NAME

Numeric fields referenced in the query that can be used in the calculation that defines the temporary result column contents.

DESCRIPTION

The Datadictionary description of each column listed.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Define Temporary Result Columns (DQH90) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected.

<PF6> DISPLAY QUERY

Displays the query created so far.

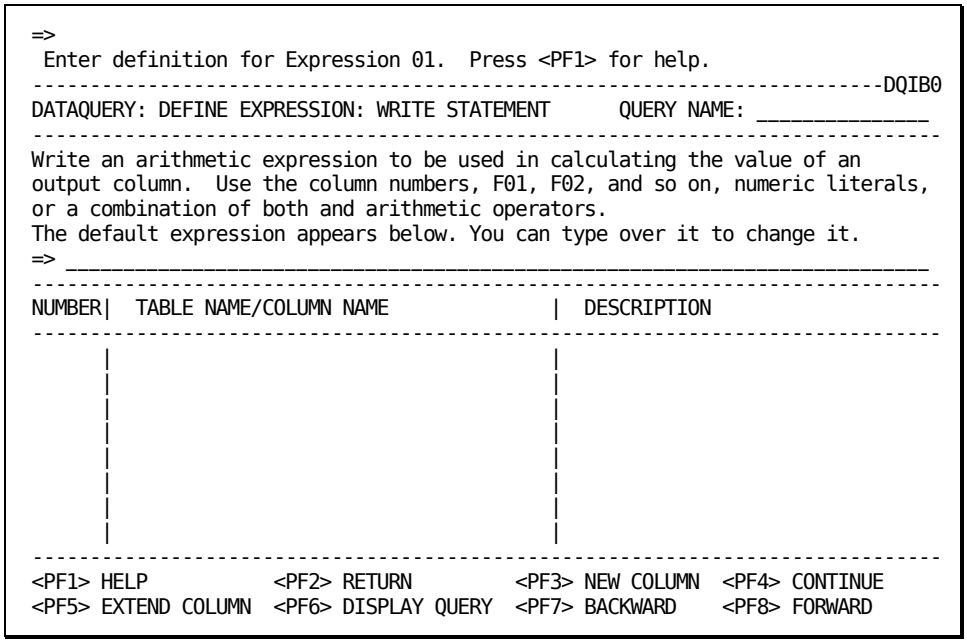
<PF7> BACKWARD

Displays a previous screen, if any.

<PF8> FORWARD

Displays next screen, if any.

(DQIB0) Define Expression: Write Statement



Description

Use this SQL Mode Guided Query panel to define an expression for use in an output row.

Action

Type an algebraic equation in the => field that can be included in the SELECT statement. Use numbers shown on the panel to represent column names. The equation can be simple or complex and can also define a character literal.

You can also type the name of any column that fits on the line following the =>. You are not limited to only previously selected columns. However, if you misspell the name, CA Dataquery does not check it here. You can also write any valid portion of a SELECT clause in the field.

You can enter a comma-separated statement in the => entry field and you can also enter a column function. Column functions are discussed in the *CA Dataquery User Guide*. An example of a valid entry of this type is:

```
'Average Salary is: $', AVG(SALARY)
```

Press <PF3> NEW COLUMN to define another expression, if needed. Press <PF4> CONTINUE to go to the next step.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

QUERY NAME:

Name assigned this query with the first Guided panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
Default expression	Y	Algebraic equation with column numbers, numeric literals, character literals, arithmetic operators, or parentheses	Addition of all columns

Columns

Refer to the following for a discussion of each column on this panel.

NUMBER

Number CA Dataquery has assigned each numeric column you selected for use in structuring your calculation.

TABLE NAME/ COLUMN NAME

Names of the columns you selected for use in structuring your calculation, along with their table names.

DESCRIPTION

Datadictionary description of each column you selected for use in structuring your calculation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Select Columns for Calculation (DQHA0) panel.

<PF3> NEW COLUMN

Displays a new DQIB0 panel to write another statement.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous screen, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQIE0) Ordering Rows: PRIORITY

```
=>

--------------------------------------------------------------------------------DQIE0
DATAQUERY: ORDERING ROWS: PRIORITY           QUERY NAME:
--------------------------------------------------------------------------------
Determine the order in which your output rows will be returned to you by
entering 1, 2, and so on, as a priority number next to any column you wish
to use in determining output sequence.  Sequence ascending or sequence
descending can be selected by placing any character under the sequence
abbreviation.
--------------------------------------------------------------------------------
PRIORITY|TABLE NAME/                        |SEQUENCE|
NUMBER  | COLUMN NAME                       |ASC/DESC| DESCRIPTION
--------------------------------------------------------------------------------
        |                                   |        |
        |                                   |        |
        |                                   |        |
        |                                   |        |
        |                                   |        |
        |                                   |        |
        |                                   |        |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode Guided Query panel allows you to select and order rows in ascending and descending sequence for presentation.

Action

Prioritize columns or expressions by entering a number from 1 to 10 in the Priority Number field preceding each column or expression name.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
PRIORITY NUMBER	N	Numeric value	None
SEQUENCE ASC/DESC	N	Any character in the ASC field or DESC field	Ascending

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME/ COLUMN NAME

Names of the columns you selected for use, including table names.

DESCRIPTION

Datadictionary description of each column you selected for use in structuring your calculation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to Sort Criteria Definition (DQHD0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

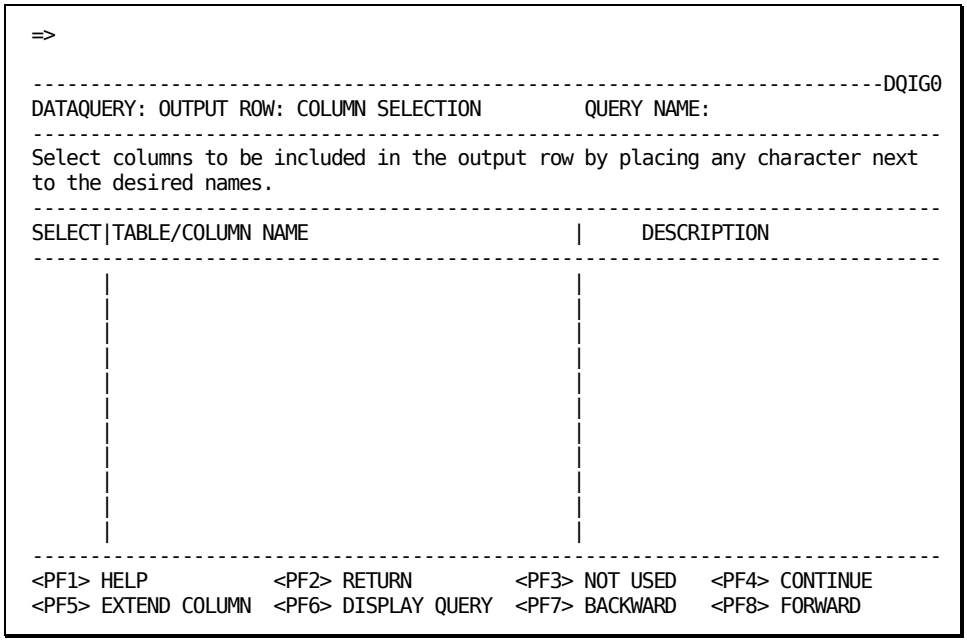
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQIG0) Output Row: Column Selection



Description

This SQL Mode Guided Query panel allows you to select the columns to include in your output. The display contains the names of all the columns from the primary and secondary tables you selected.

Action

Select one or more columns from the list displayed. Type a character over the underscore beside each column name and press Enter.

Display Fields

Refer to the following for explanations of the display field intended to contain variable data.

QUERY NAME:

Name assigned this query with the first Guided panel.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
SELECT	N	Any character	None

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME/ COLUMN NAME

Names of the columns you selected for use, including table names.

DESCRIPTION

Datadictionary description of each column you selected for use in structuring your calculation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQII0) Output Row: Definition

```
=>
 Press <PF1> for help with defining the output row
--------------------------------------------------------------------------------DQII0
DATAQUERY: OUTPUT ROW: DEFINITION             QUERY NAME:
--------------------------------------------------------------------------------
Specify the order in which the columns are to appear in the output row by
inserting an order number below.  Column functions can be selected by placing
any character in the corresponding fields.   Count output rows? _ yes _ no
--------------------------------------------------------------------------------
ORDER | TABLE NAME/                          |     |     |     |     | COUNT
NUMBER|  COLUMN NAME                         | AVG | MAX | MIN | SUM | DISTINCT
--------------------------------------------------------------------------------
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
      |                                      |     |     |     |     |
--------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode Guided Query panel can be accepted with the defaults CA Dataquery assigns, resulting in report data printed in columns in the same order in which it was selected. However, you may want to change the order in which columns are printed. You may also want to request that mathematical functions be performed on some or all numeric columns. If so, you will select math functions for columns. At the end of the report, you can obtain the results of the functions and specify a special legend to identify them on the report. See the *CA Dataquery User Guide* for a complete explanation of each available column function and an introduction to using this panel.

Action

Number the columns in the order wanted in the Order Number column. Enter any character in a column function column you want to perform on a particular column. Press Enter or <PF4> CONTINUE to go on to the next Guided Query panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name assigned the query you are defining.

Columns

Refer to the following for a discussion of each column on this panel.

ORDER NUMBER

Numbers appearing in this column are assigned automatically by CA Dataquery. Change the numbers as needed by typing over the current number to indicate the order of columns on the report.

TABLE NAME/ COLUMN NAME

Datadictionary names of the columns you selected for use, including table names.

AVG

Enter a character in this column next to any column or key for which you want CA Dataquery to produce an average at the end of the report.

MAX

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the maximum amount found at the end of the report.

MIN

Enter a character in this column next to any column or key for which you want CA Dataquery to produce the minimum amount found at the end of the report.

SUM

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a total at the end of the report.

COUNT DISTINCT

Enter a character in this column next to any column or key for which you want CA Dataquery to produce a count of the data items in this column at the end of the report.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits (DQHG0) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor.

<PF6> DISPLAY QUERY

Displays the query created so far.

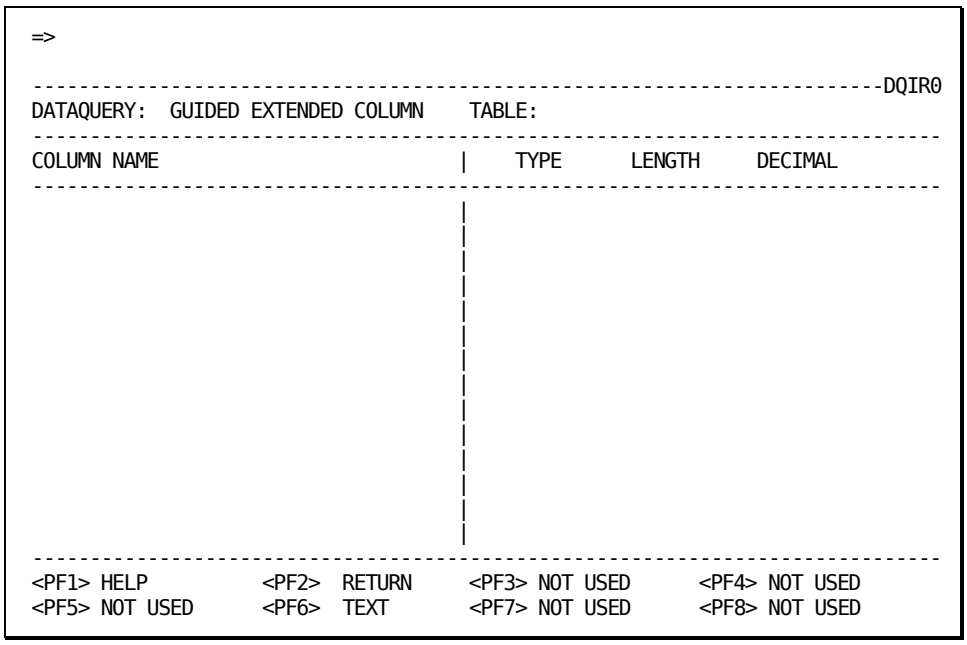
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQIR0) Guided Extended Column



Description

This SQL Mode panel displays additional Datadictionary attributes not shown on the Guided Query list of column names. Display this panel by selecting the EXTENDED DEF PF key where provided in Guided Query.

Action

When this panel is displayed, you can view the Datadictionary attributes of the column.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE:

Datadictionary name of the table whose columns are listed on the panel.

Columns

Refer to the following for explanations of the columns on this panel.

COLUMN NAME

Datadictionary name of the column for the table you selected.

TYPE

See Data Types (see page 609) for explanations of these data types.

N

Numeric (including binary)

F

Float

C

Character

G

Graphics

A

SQL DATE

T

SQL TIME

P

SQL TIMESTAMP

LENGTH

Shows the total length of the column.

DECIMAL

Number of decimal positions.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF6> TEXT

Displays text from Datadictionary on TEXT DISPLAY (COLUMN) panel (DQB40).

(DQIT0) Join Condition: Define Relation

```
=>

--------------------------------------------------------------------------DQIT0
DATAQUERY: JOIN CONDITION: DEFINE RELATION        QUERY NAME: _______________
-------------------------------------------------------------------------------
Define the relation by selecting a comparison operator to compare
     ___________________________________  in  ___________________________________
with
     ___________________________________  in  ___________________________________

Select one comparison operator:
     _ Equal ( = )
     _ Not equal ( ,= )
     _ Greater than ( > )
     _ Greater than or equal ( >= )
     _ Less than ( < )
     _ Less than or equal ( <= )

When you have completed this join condition you can press <PF3> to define
another join condition.  Press <PF1> for more information.

-------------------------------------------------------------------------------
<PF1> HELP      <PF2> RETURN          <PF3> REPEAT        <PF4> CONTINUE
<PF5> NOT USED  <PF6> DISPLAY QUERY   <PF7> NOT USED      <PF8> NOT USED
```

Description

This SQL Mode Guided Query panel is used to select a comparison operator that will be used to define the relationship between the two columns you selected to join the two tables you are using in your query. The conditions you specify will be used to determine which rows in the tables will be joined.

Action

Select the type of comparison to be performed by entering a character over the underscore beside the comparison operator. Press <PF4> CONTINUE or Enter to go to the next step.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name of the query you are creating.

Select

Refer to the following for explanations of the selections on this panel.

Equal

Data in first column matches data in the second column.

Not Equal

Data in first column does not match data in second column.

Greater than

Data in first column is greater than data in second column.

Greater than or equal to

Data in first column is equal to or greater than data in second column.

Less than

Data in first column is less than data in second column.

Less than or equal to

Data in first column matches or is less than in second column.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> REPEAT

Creates another comparison for selection.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF6> DISPLAY QUERY

View the query created so far.

(DQIU0) Grouping Rows: Priority

```
=>

-------------------------------------------------------------------------------DQIU0
DATAQUERY: GROUPING ROWS: PRIORITY           QUERY NAME:
-------------------------------------------------------------------------------
If you have more than 1 column on which grouping will occur, number them
1, 2, and so on, to show their grouping priority from major to minor.
Defaults are provided and can be modified.
-------------------------------------------------------------------------------
PRIORITY| TABLE NAME/                       |
NUMBER  |  COLUMN NAME                      | DESCRIPTION
-------------------------------------------------------------------------------
        |                                   |
        |                                   |
        |                                   |
        |                                   |
        |                                   |
        |                                   |
        |                                   |
        |                                   |
        |                                   |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

This SQL Mode Guided Query panel displays the columns on which grouping will occur. You selected both columns and functions for your output, so the columns will be used to group the retrieved rows for the column function calculations you requested.

Action

On this panel, prioritize the columns that will be used to group the rows. You can type over the default priorities with numeric values. Press <PF4> CONTINUE or Enter to go to the next panel.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name of the query you are creating.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
PRIORITY NUMBER	Y	Numeric Value	Assigned and displayed

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME/ COLUMN NAME

Identifies a grouping column.

DESCRIPTION

Datadictionary description assigned to column name.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Displays the next Guided Query panel.

<PF5> EXTEND COLUMN

Shows definition of column selected by cursor.

<PF6> DISPLAY QUERY

Displays query created so far.

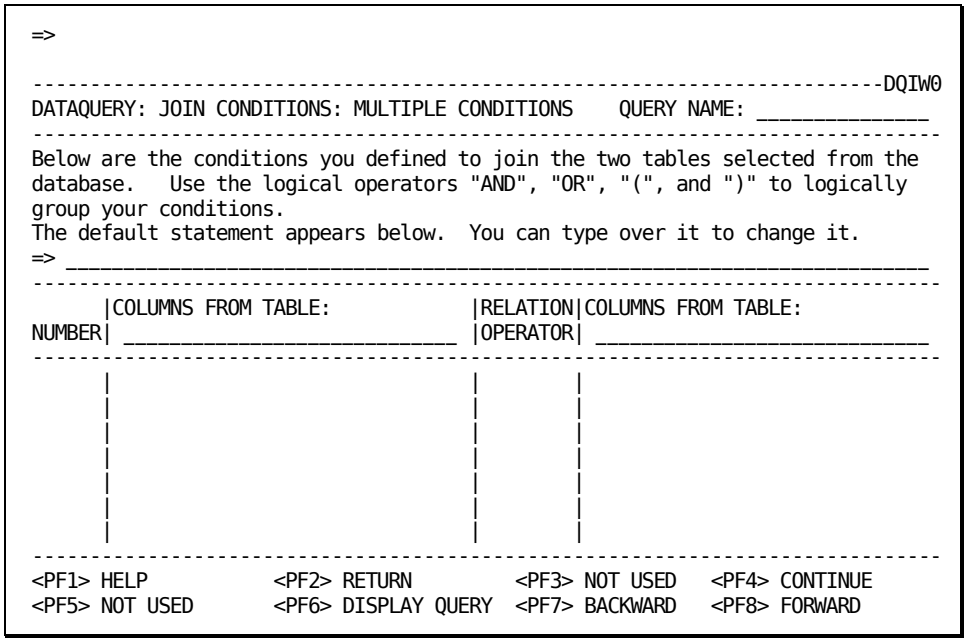
<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQIW0) Join Conditions: Multiple Conditions



Description

This SQL Mode Guided Query panel displays the join conditions you defined to describe the relationship between the two tables selected for this query. The default statement uses numeric identifiers CA Dataquery has assigned to each condition.

Action

Write over the default condition statement using *and*, *or*, or parentheses to arrange the conditions as needed. Name at least one join condition as a logical expression. Press <PF4> CONTINUE or Enter when finished.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name of the query you are creating.

default statement

Default logical expression created by CA Dataquery to group condition statements.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
=> (default statement)	Y	Condition numbers, and, or, (,)	Condition numbers added together

Columns

Refer to the following for a discussion of each column on this panel.

NUMBER

Number assigned a condition.

COLUMNS FROM TABLE (1)

Column selected from first table for this condition.

RELATION OPERATOR

Operator used to compare column from first table to column from second table.

COLUMNS FROM TABLE (2)

Column selected from second table for this condition.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Displays next Guided Query panel.

<PF6> DISPLAY QUERY

Displays query created so far.

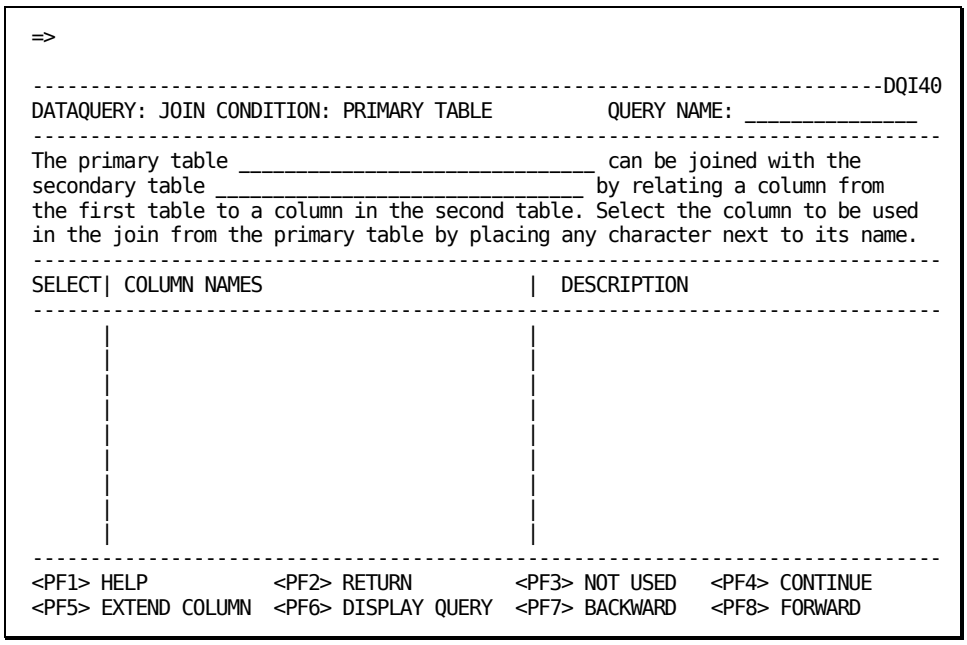
<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQI40) Join Condition: Primary Table



Description

You must complete this SQL Mode Guided Query panel if you selected a primary table and a secondary table for your query. The panel displays a list of columns from which you can choose one that will contain a value that will match a value in a key or column of the secondary table. These matching values will join rows from each table for the query's search for data.

Action

Type a character in the Select column next to the name of the key or column to be used for joining with the secondary table, key or column.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

primary table_

The table you selected first.

secondary table_

The table you selected for CA Dataquery to read second.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the name of a column in the primary table whose values will match values in a column you will choose for the secondary table.

COLUMN NAMES

Columns in the primary table.

DESCRIPTION

Description, if any, for each column.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

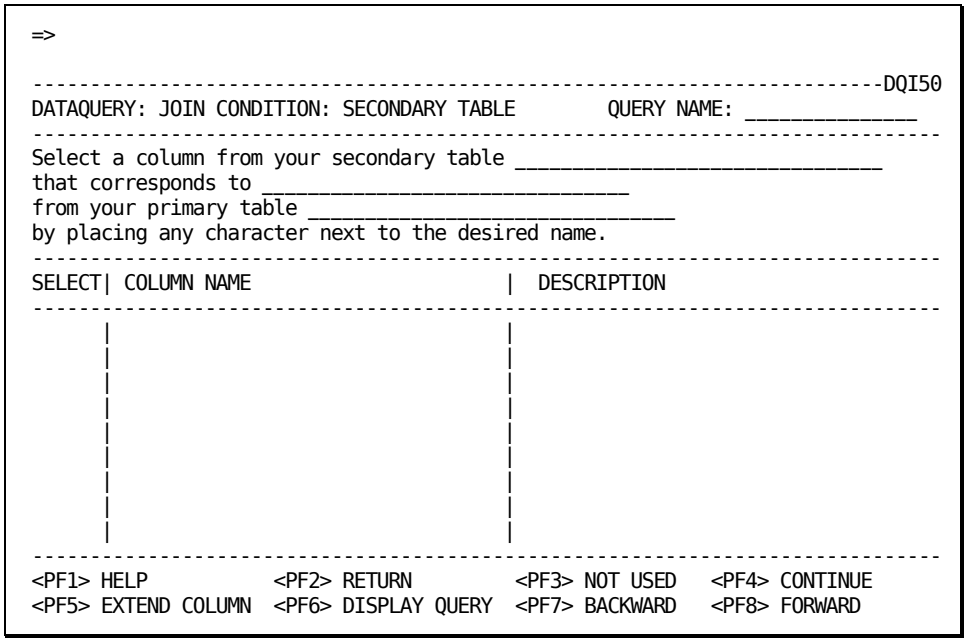
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQI50) Join Condition: Secondary Table



Description

You must complete this SQL Mode Guided Query panel if you selected a primary table and a secondary table for your query. The panel displays a list of columns from which you can choose one that will contain a value that will match a value in a column of the primary table. These matching values will join rows from each table for the query's search for data.

Action

Type a character in the SELECT column next to the name of the column to be used for joining with the primary table column.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

secondary table ____

The table you selected as the second one to be read in joining tables.

corresponds to ____

Column you selected from the primary table whose value will match values in the key you will select from the secondary table names listed on this panel.

primary table__

The table you selected as the first one to be read in joining tables.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the name of a key in the secondary table whose values will match values in the key or column selected for joining in the primary table.

COLUMN NAME

Lists all columns in the secondary table. Scroll forward to see any additional names.

DESCRIPTION

Description, if any, of each column.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column definition for column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQI60) Search Conditions: Designate Columns

```
=>
 Press <PF4> if you do not wish to specify further selection conditions
-------------------------------------------------------------------------------DQI60
DATAQUERY: SEARCH CONDITIONS: DESIGNATE COLUMNS    QUERY NAME: _______________
-------------------------------------------------------------------------------
Specify columns that will be evaluated for row selection by placing
any character next to the desired names.
-------------------------------------------------------------------------------
SELECT| TABLE/COLUMN NAME                 |  DESCRIPTION
-------------------------------------------------------------------------------
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
      |                                  |
-------------------------------------------------------------------------------
<PF1> HELP           <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> EXTEND COLUMN  <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

Description

The SQL Mode Guided Query optional Search Conditions: Designate Columns panel allows you to begin narrowing the query's search for data. You can select up to 15 columns that contain data for which you can search, by stating that you want to see a range of data, an exact match, and so forth. You use this panel to begin specifying search criteria that tell CA Dataquery to read only rows that contain data you specify. You begin defining the selection criteria by naming the columns with this panel.

For instance, you can indicate that you only want rows that contain specific values in a particular column, like CITY = DALLAS. With this panel you would select CITY and on a following panel you would specify that CITY should contain DALLAS for a row to be selected.

Action

Enter a character in the Select column next to any columns you want to use in specifying selection criteria for this query. Then press Enter or <PF4> CONTINUE to go to the next panel.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to the names of any columns for which you want to specify qualifying values.

TABLE/ COLUMN NAME

Lists the names of all columns in the tables you selected for this query. If two tables were selected, the primary table columns are listed first, followed by the secondary columns. Scroll forward to see additional panels if the list extends beyond the current panel.

DESCRIPTION

Description, if any, of each column listed.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Primary Table Selection (DQH20) panel.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> EXTEND COLUMN

Displays the extended column/definition for key/column selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query created so far.

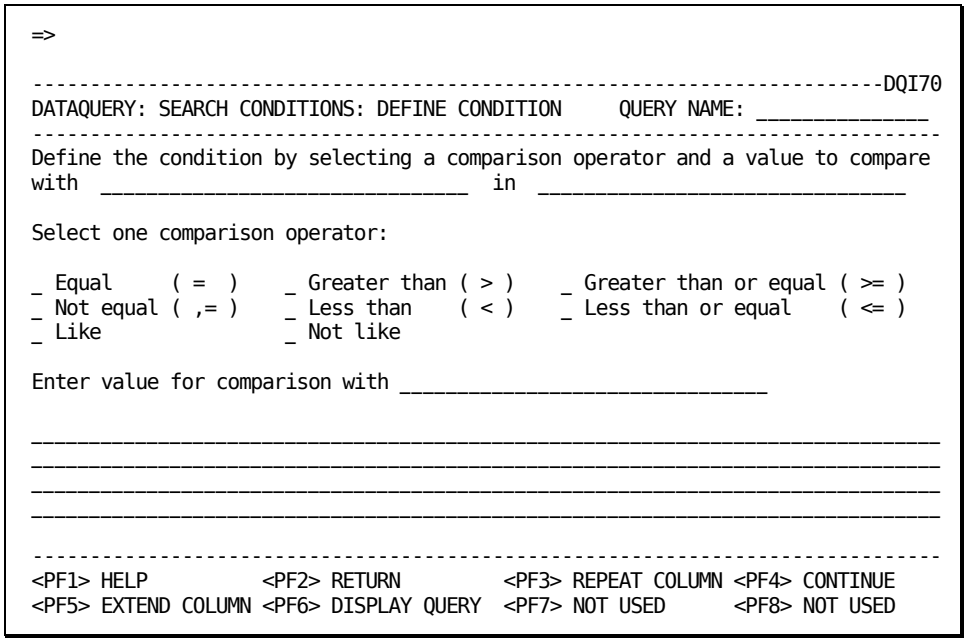
<PF7> BACKWARD

Displays a previous panel, if any.

<PF8> FORWARD

Displays next panel, if any.

(DQI70) Search Conditions: Define Condition



Description

The SQL Mode Guided Query optional Search Conditions: Define Condition panel appears for every column selected as a Selection Criteria column in the previous step. With it, you describe the data you want to be searched for in a column to qualify a row for inclusion in your report.

Action

Choose a comparison operator from the panel by entering a character next to the one that indicates how you want to compare the found column values to the values you enter at the bottom of the panel. You can choose only one operator.

Your entries on the blank lines provided must follow the rules for specifying comparisons in SQL as documented in SQL Statements (see page 515) or the *CA Datacom/DB SQL User Guide*. When the panel is complete and correct, press Enter or <PF4> CONTINUE to go to the next step.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

The name you assigned this query.

compare with ___

The column selected for comparison of the values you will enter on this panel.

in ___

The name of the table containing the selected column.

Select

Make selections on this panel according to the following.

Equal

Data in the column must be equal to the value you enter below for the row to be selected.

Greater than

Data in the column must be greater than the value you enter below for the row to be selected.

Less than

Data in the column named must be less than the value you enter below for the row to be selected.

Not equal

Data in the column named must not match the value you enter below for the row to be selected.

Greater than or equal

Data in the column must be greater than or equal to the value you enter below for the row to be selected.

Less than or equal

Data in the column named must be less than or equal to the value you enter below for the row to be selected.

Entering Comparison Values

On the lines following "Enter value for comparison with _," enter any valid comparison value, following the rules for specifying comparisons in SQL Statements (see page 515) or the *CA Datacom/DB SQL User Guide*.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Returns to the Search Conditions: Designate Columns (DQH60) panel.

<PF3> REPEAT COLUMN

Lets you use the same column again in another comparison.

<PF4> CONTINUE

Displays the next Guided Query panel when this one is complete.

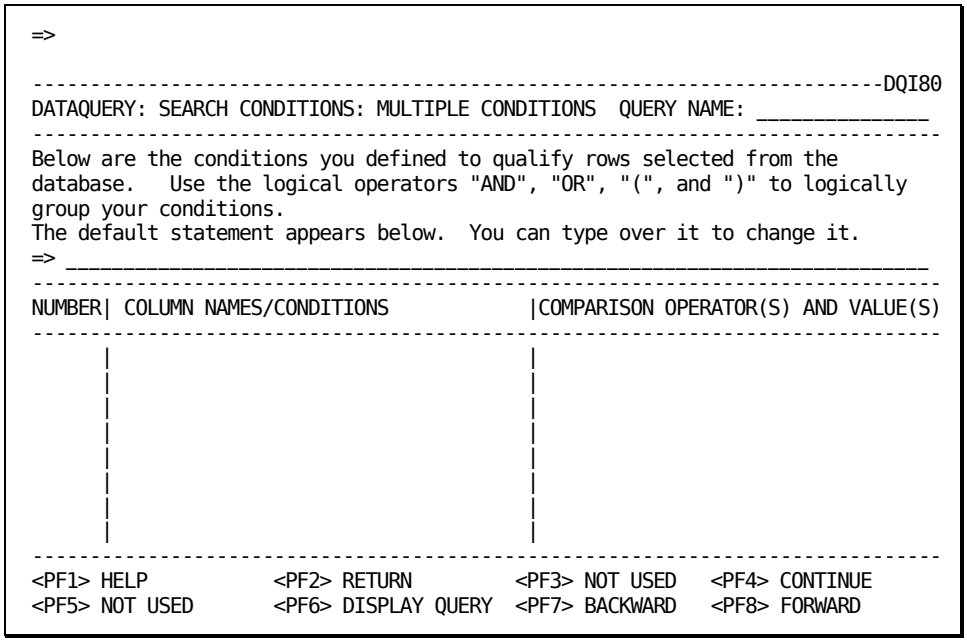
<PF5> EXTEND COLUMN

Displays the extended column definition for key or column.

<PF6> DISPLAY QUERY

Displays the query created so far.

(DQI80) Search Conditions: Multiple Conditions



Description

This SQL Mode Guided Query panel appears when you select more than one column and condition per table for narrowing your query's search for data. You select columns that contain data you want to use in qualifying the search.

When those conditions are defined, this Multiple Conditions panel appears to allow you to group the conditions into a logical expression using AND, OR, and parentheses ( ) and the number assigned to each condition. CA Dataquery always *ands* each condition sequentially as the default expression.

Action

Type over the default logical expression with your own expression, using the numbers provided for each condition. Or, accept the default statement by pressing <PF4> CONTINUE.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned this query.

default statement

Default logical expression created by CA Dataquery to group condition statements.

Entry Field

The following describes how you can use the entry field that contains the default statement.

default statement

A default logical expression combining all conditions by adding them together. You can write over this expression to tell CA Dataquery how to apply the qualifying conditions in the search for data.

Columns

Refer to the following chart for a discussion of each column on this panel.

NUMBER

Number CA Dataquery assigned a column and its condition.

COLUMN NAMES/ CONDITIONS

Names of columns/conditions selected for qualifying the search for data.

COMPARISON OPERATOR(S) AND VALUE(S)

Conditions constructed for each column.

Sample

```
=>

--------------------------------------------------------------------------------DQH80
DATAQUERY: SELECTION CRITERIA:MULTIPLE CONDITIONS QUERY NAME: MULTI-2
--------------------------------------------------------------------------------
Below are the conditions you defined to qualify rows selected from table
  CAI-RCPTS-REC                    .
Use the logical operators "AND", "OR", "(", and ")" to logically group your
conditions.
The default statement appears below.  You can type over it to change it.
=> (2 and 3) or (1 and 4)
--------------------------------------------------------------------------------
NUMBER|      KEY (*) AND COLUMN NAMES      |COMPARISON OPERATOR(S) AND VALUE(S)
--------------------------------------------------------------------------------
  01  |   RECEIPT-ID                       | GT     1
  02  |   REC-QTY                          | GTE    1
  03  |   ITM-ID                           | EQ     999
  04  |   REC-YR                           | EQ     86
      |                                    |
      |                                    |
      |                                    |
--------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> NOT USED   <PF4> CONTINUE
<PF5> NOT USED      <PF6> DISPLAY QUERY  <PF7> BACKWARD   <PF8> FORWARD
```

The preceding example tells CA Dataquery to retrieve only data from the CAI-RCPTS-REC table rows that contain a REC-QTY greater than or equal to 1 **AND** an ITM-ID equal to 999 **OR** rows that contain a RECEIPT-ID greater than 1 **AND** a REC-YR equal to 86.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> CONTINUE

Goes to the next Guided Query Creation step.

<PF6> DISPLAY QUERY

Displays the query created so far.

<PF7> BACKWARD

Scrolls to the previous panel, if any.

<PF8> FORWARD

Scrolls to the next panel, if any.

(DQKL0) User Profile

Profile Items

A discussion of each profile item follows.

User Profile (DQKL0)

```
=>
Overtype the values to be modified and press PF4 to complete the update
--------------------------------------------------------------------------------DQKL0
DATAQUERY:  USER PROFILE               FOR => ____________________________________
--------------------------------------------------------------------------------
    PROFILE ITEM                                  EXPLANATION
--------------------------------------------------------------------------------
DATADICTIONARY DATABASE ID  =>  _____  Five digit DB ID number for DD access
LIST AND DISPLAY ALIASES    =>  ___    YES show aliases, NO suppress aliases
GROUP DISPLAY               =>  ___    YES break out simple columns, NO do not
SUPPRESS DUPLICATE COLUMNS  =>  ___    YES turn on suppression, NO turn off
SUPPRESS PFKEYS ON PRINT    =>  ___    YES turn on suppression, NO turn off
SUPPRESS EXECUTE PANEL      =>  ___    YES turn on suppression, NO turn off
PRIMARY LANGUAGE            =>  __     TWO character primary language ID
SECONDARY LANGUAGE          =>  __     TWO character secondary language ID
DECIMAL POINT CHARACTER     =>  _      Character to be used for decimal point
QUERY LANGUAGE              =>  ___    Language for queries - SQL or DQL
SQL AUTHORIZATION ID        =>  ________________
ASSOCIATE USER ERRORS       =>  ___    Display validation errors for assoc


--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN        <PF3> DISP GROUPS   <PF4> UPDATE
<PF5> PRINT OPT   <PF6> NOT USED      <PF7> NOT USED      <PF8> NOT USED
```

Description

CA Dataquery displays the values for each profile item that were assigned as defaults, or those that were later specified by the CA Dataquery Administrator or by you.

Action

You can change any profile item, but you should ensure that the values meet site standards. For example, if you specify a network printer ID that is not a valid ID, you cannot receive printed output.

When you change your own profile, the changes take effect immediately. The Datadictionary database ID always reverts back to the site default when you sign off. The SQL Authorization ID returns to the default when you sign off.

Profile Items

A discussion of each profile item follows:

DATADICTIONARY DATABASE ID

(Required) Enter the CA Datacom/DB database ID that specifies the Datadictionary database to be accessed. The database specified must be a Datadictionary database accessible to CA Dataquery. CA Dataquery supplies the CA Datacom/DB database ID from the System Option Table.

LIST AND DISPLAY ALIASES

(Required) Enter YES or NO. Specify YES to include Datadictionary aliases in the Directory of DQL Tables, and Keys and Columns Display panels. NO excludes Datadictionary aliases from these display panels. (See the CA Dataquery end user documentation for details on these panels.)

GROUP DISPLAY

(Required) Enter YES or NO. Determines the way a compound field is displayed on a report. If you specify YES, fields making up the compound field are shown as individual fields. When you specify NO, a compound field is shown as a single alphanumeric field, even though one or more of the simple fields contained in the compound field is a numeric field which cannot be printed. If an invalid value is entered, the parameter defaults to NO.

SUPPRESS DUPLICATE COLUMNS

(Required) Enter YES or NO. Determines if duplicate values for columns specified as control break columns are suppressed in the generated report. If you specify YES, the value contained in a control break column is displayed only once. Each time the value in the control break column changes, the new value is displayed. If the output continues to the top of a new page, the current value in the control break column is displayed at the top of the new page.

SUPPRESS PFKEYS ON PRINT

(Required) Enter YES or NO. Specify YES to suppress the PF key descriptions on the print panel that displays the report. NO causes the PF keys descriptions to be displayed.

SUPPRESS EXECUTE PANEL

(Required) Enter YES or NO. Specify YES to suppress the display of the Online Execution Query panel. NO causes the Online Execute Query panel to be displayed. A user would want to suppress the display of the Online Execution Query panel if their queries always read and collect data and display it on their terminals. Suppressing the display saves a step during the execution process by accepting the execution defaults.

PRIMARY LANGUAGE

(Required) Enter 2 characters. Specifies the primary language to be used during the current CA Dataquery session. A valid 2-character entry overrides the language specified in the System Option Table at installation. (AE American English is the default if no other language is specified on the System Option Table.)

SECONDARY LANGUAGE

(Required) Enter 2 characters. Specifies the secondary language to be used during the current CA Dataquery session. A valid 2-character entry overrides the language specified in the System Option Table at installation. (AE American English is the default if no other language is specified on the System Option Table.)

DECIMAL POINT CHARACTER

(Optional) Enter 1 character. Specifies the decimal point character for this ID. The default is the value of the DECPT= parameter in the System Option Table.

QUERY LANGUAGE

(Required) Enter either DQL or SQL. Specifies the query language to be used for this ID. Y (yes) must have been specified on the User Table Maintenance in the field SQL AND DQL ALLOWED for the user to be able to change from DQL to SQL in this field.

SQL AUTHORIZATION ID

(Required) Enter a 1- to 18-character authorization ID. Specifies the SQL authorization ID for execution of SQL statements for this ID.

When the SQL authorization ID is changed by the PROFILE or AUTHID command, it changes only on the user profile, **and not** on the User Table. The private SQL authorization ID attaches to the table name regardless of the authid in use during creation of the table. When the DISPLAY, LIST, EXECUTE, or CREATE functions are used, the profile authid is used.

ASSOCIATE USER ERRORS

(Optional) Enter YES to display normal end-user validation errors for the associate user. Enter NO to display only DQ244E when errors occur. DQ244E states only that the user is attempting to execute a query that has an error.

PF Keys

The following lists each PF key for the User Profile panel.

CLEAR

Returns to Main Menu without updating.

<PF1> HELP

Displays Help panel PF key options.

<PF2> RETURN

Exits panel or Main Menu.

<PF3> DISP GROUPS

Displays group level names assigned to this user system.

<PF4> UPDATE

Saves the values displayed on this panel.

<PF5> PRINT OPT

Displays print options.

(DQKM0) Hardcopy Print Options

```
=>
Overtype the values to be modified and press PF4 to complete the update
-------------------------------------------------------------------------------DQKM0
DATAQUERY:  USER PROFILE              FOR => ___________________________________
-------------------------------------------------------------------------------
   HARDCOPY PRINT OPTIONS                        EXPLANATION
-------------------------------------------------------------------------------
 NETWORK PRINTER-ID         =>  ____   Four Character ID of Network printer
 PRINT QUERY TEXT           =>  ___    YES print the query text, NO do not
 PRINT STATISTICS           =>  ___    YES print the query stats, NO do not
 PRINT BANNER PAGE          =>  ___    YES print the banner page, NO do not
 PRINT USING WINDOWS        =>  ___    YES window format, NO wrap format
 PRINT PAGES TOGETHER       =>  ___    YES pages together, NO windows together
 PRINT NUMBER OF COLUMNS    =>  ___    Three digit column width of hardcopy
 PRINT NUMBER OF ROWS       =>  ___    Three digit page length in rows









-------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN        <PF3> DISP GROUPS   <PF4> UPDATE
<PF5> PROFILE     <PF6> NOT USED      <PF7> NOT USED      <PF8> NOT USED
```

Description

To view a user's print options, select <PF5> PRINT OPT on the User Profile panel. CA Dataquery displays this Hardcopy Print Options panel.

Action

When you have completed your input to this panel, press <PF4> UPDATE to save this user's profile. The next time the user signs on to CA Dataquery, the new print options are in effect.

Options

NETWORK PRINTER ID

Required. Enter the 1- to 4-character ID of the desired printer. CA Dataquery displays the value entered in the System Option Table.

PRINT QUERY TEXT

Required. Input Y (yes), the default, if the text of the query that produced the report is to be printed when the report is printed on a network printer. N (no) does not print the query text.

PRINT STATISTICS

Required. Specify Y (yes), the default, if the statistics of the query that produced the report are to be printed when the report is printed on a network printer. N (no) does not print the query statistics.

PRINT BANNER PAGE

Required. Indicate Y (yes), the default, if this user's print jobs are to be preceded with a banner page containing user name, date, and time, to aid in distributing the reports. N (no) suppresses the printing of the banner page.

PRINT USING WINDOWS

Required. Specify Y (yes) if the report extends beyond 80 columns and you do not want the report lines to wrap. N (no) is the default, if you want the print lines to wrap or continue on the next line.

PRINT PAGES TOGETHER

Required. Use this field when printing a report composed of two or more adjacent (side-by-side) pages. If the first page (left-hand page) is labeled A and the second page (right-hand page) is labeled B and the report is three pages in length, specifying Y (yes), the default, would result in these pages being printed in the order of 1A, 2A, 3A, 1B, 2B, 3B. Using the same example, specifying N (no) results in a printing order of 1A, 1B, 2A, 2B, 3A, 3B.

PRINT NUMBER OF COLUMNS

Required. Specify the width of the hardcopy on the network printer by stating the number of columns to be printed. Indicate a 1- to 3-character numeric value.

PRINT NUMBER OF ROWS

Required. Specify the number of rows to be printed on one page of hardcopy on the network printer. This is a 3-character numeric field.

PF Keys

The following lists each PF key for the User Profile panel.

CLEAR

Returns to Main Menu without updating.

<PF1> HELP

Displays Help panel PF key options.

<PF2> RETURN

Exits panel or Main Menu.

<PF3> DISP GROUPS

Displays group level names assigned to this user system.

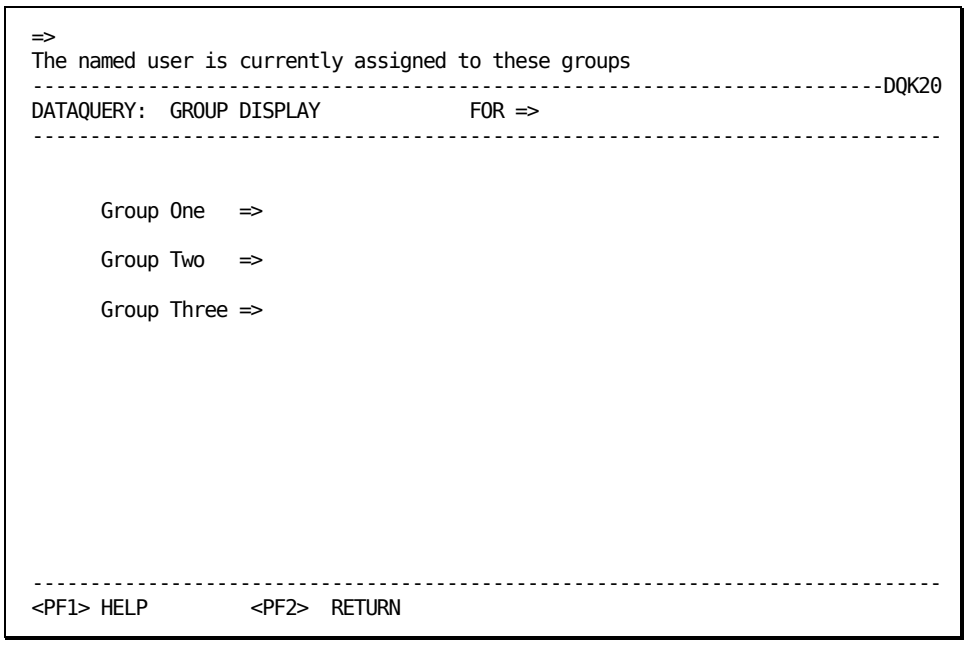
<PF4> UPDATE

Saves the values displayed on this panel.

<PF5> PRINT OPT

Displays print options.

(DQK20) Group Display



Description

The Group Display panel shows security groups to which you have been assigned, if any. If no groups are assigned, you can access database tables and queries assigned to any group.

Action

No action is required.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

(DQMI0) Estimated Statistics

```
=>

Estimated statistics have exceeded maximum limits.  To proceed with query:
press the PF4 key.  To cancel execution of query: press the PF2 key.
-------------------------------------------------------------------------------DQMI0
DATAQUERY:  ESTIMATED STATISTICS                 QUERY NAME:
-------------------------------------------------------------------------------

             NUMBER REQUESTED:

     FIND TERMINATED BECAUSE:

                                ESTIMATED
                                ---------
     TOTAL ESTIMATED ROWS:
     TOTAL JOINS EXAMINED:
     VALID JOINS EXAMINED:
     TOTAL ESTIMATED I/O




-------------------------------------------------------------------------------
<PF1> HELP     <PF2> RETURN      <PF3> NOT USED    <PF4> CONTINUE
```

Description

This DQL Mode panel displays estimated information about the execution of the active query's FIND statement. If estimated I/O exceeds the system option or user option maximum, the query will not be executed unless you press <PF4> CONTINUE.

Action

Decide whether to continue or cancel execution of the query.

Display Fields

Refer to the following for explanations of the display fields intended to contain variable data.

QUERY NAME:

Query being executed.

NUMBER REQUESTED:

All or the number of rows requested in the query.

FIND TERMINATED BECAUSE:

Total estimated I/O exceeds estimated MAX I/O.

TOTAL ESTIMATED ROWS:

Estimated number of rows expected in the output.

TOTAL JOINS EXAMINED:

Number of joins created by the query and examined by this program.

VALID JOINS EXAMINED:

Number of valid joins examined.

TOTAL ESTIMATED I/O

Estimated number of CA Datacom/DB physical I/O events for the processing of the FIND statement.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel and cancels execution.

<PF4> CONTINUE

Continues query execution.

(DQM10) List of Help Topics

```
=>
Place the cursor on the desired TOPIC and press the appropriate PFkey
-------------------------------------------------------------------------------DQM10
DATAQUERY:  LIST OF HELP TOPICS
-------------------------------------------------------------------------------
     DATAQUERY TOPIC NAME        |     DESCRIPTION
-------------------------------------------------------------------------------
  Accessing Repeating Fields     |   Using repeating fields in queries.
  Batch DATAQUERY Execution      |   Submitting a query for batch execution.
  Command Descriptions           |   What commands do.
  Command Syntax                 |   How commands are built.
  Database Terms                 |   Database terminology used in DATAQUERY
  DATACOM/DB and DATAQUERY       |   How DATAQUERY uses the database.
  DATADICTIONARY                 |   How DATAQUERY uses the dictionary.
  DATAREPORTER                   |   Description and sample program.
  DIALOG Creation                |   How to create a dialog.
  DIALOG Execution               |   Using queries with variable values.
  Easy Way to Use DATAQUERY      |   Suggestions for the new user.
  Efficient Table Searches       |   Determining the most efficient retrieval
  Error Messages                 |   Types and content of error messages.
-------------------------------------------------------------------------------
<PF1> HELP        <PF2>  RETURN         <PF3> NOT USED    <PF4> NOT USED
<PF5> NOT USED    <PF6>  DISPLAY TEXT   <PF7> BACKWARD    <PF8> FORWARD
```

Description

This panel presents a list of topics which can be selected to display Help information.

Action

Position the cursor on a topic and press <PF6> DISPLAY TEXT to see Help information.

PF Keys

The following lists each PF key for the User Profile panel.

CLEAR

Returns to Main Menu without updating.

<PF1> HELP

Displays Help panel PF key options.

<PF2> RETURN

Exits panel or Main Menu.

<PF6> DISPLAY TEXT

Displays Help panel for topic selected by cursor.

<PF7> BACKWARD

Scrolls to the previous panel, if any.

<PF8> FORWARD

Scrolls to the next panel, if any.

(DQM20) Primary Table Selection

```
=>

--------------------------------------------------------------------------------DQM20
DATAQUERY:  PRIMARY TABLE SELECTION              QUERY NAME:  _______________
--------------------------------------------------------------------------------
Select one table from which you want information by placing any character
next to the desired table name.  START WITH  ___________________________________
--------------------------------------------------------------------------------
SELECT|   TABLE NAME                      | DESCRIPTION
--------------------------------------------------------------------------------
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
   _  |                                  |
--------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> DISPLAY TEXT <PF4> CONTINUE
<PF5> DISPLAY TABLE <PF6> DISPLAY QUERY  <PF7> BACKWARD     <PF8> FORWARD
```

Description

The SQL Mode Guided Query Creation function required Primary Table Selection panel tells CA Dataquery which table to search (first if you want to search two tables) for the data you want. The primary table is the one CA Dataquery will read in its search. It should contain all of the data you want if it is to be the only table read.

Action

Place a character next to the name of the table you want the query to search and then press Enter or <PF4> CONTINUE.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
QUERY NAME:	N	Any unique name	None
START WITH	N	Any unique characters	None

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to one database table name to select the primary table you want your query to search.

TABLE NAME

Lists all database tables you are authorized to access. Lists can be continued on additional panels that you can display by scrolling forward with <PF8>.

DESCRIPTION

Description of each table, if any.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY TEXT

Displays the text description of a table selected by cursor position.

<PF4> CONTINUE

Displays the next panel and saves this selection. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> DISPLAY TABLE

Displays the keys and columns for the table selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query so far.

<PF7> BACKWARD

Scrolls back to see previous panels of multi-screen panel.

<PF8> FORWARD

Scrolls forward to see next panel, if a multi-screen panel.

(DQM30) Secondary Table Selection

```
=>
 Press <PF4> if information from a secondary table is not desired
--------------------------------------------------------------------------------DQM30
DATAQUERY:  SECONDARY TABLE SELECTION           QUERY NAME:  _______________
--------------------------------------------------------------------------------
You can use information from the primary table ____________________________
to join with related information from another table. To join two tables,
select the second table by placing any character next to the desired name.
--------------------------------------------------------------------------------
SELECT|    TABLE NAME                   | DESCRIPTION
--------------------------------------------------------------------------------
      |                                 |
      |                                 |
      |                                 |
      |                                 |
      |                                 |
      |                                 |
      |                                 |
      |                                 |
      |                                 |
--------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN         <PF3> DISPLAY TEXT <PF4> CONTINUE
<PF5> DISPLAY TABLE <PF6> DISPLAY QUERY  <PF7> BACKWARD     <PF8> FORWARD
```

Description

In the SQL Mode Guided Query Creation function, you can join your primary table (Primary Table Selection panel, H20) to a second table if they share a column. Then you can report on data that is contained in both tables. Use this panel to select an optional secondary table for joining.

Action

Place a character next to the name of the table you want the query to search and then press Enter or <PF4> CONTINUE.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

QUERY NAME:

Name you assigned the query.

primary table

Table selected first with panel DQI20 as the primary search table.

Columns

Refer to the following for a discussion of each column on this panel.

SELECT

Place a character next to one database table name to select the secondary table you want your query to search.

TABLE NAME

Lists all database tables you are authorized to access. List can be continued on additional panels that you can display by scrolling forward with <PF8>.

DESCRIPTION

Description of each table, if any.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits panel.

<PF3> DISPLAY TEXT

Displays text description, if it exists, of a table selected by cursor position.

<PF4> CONTINUE

Displays the next panel. (Returning to this panel from a later panel will require that you redisplay any entry to save it again.)

<PF5> DISPLAY TABLE

Displays the keys and columns for the table selected by cursor position.

<PF6> DISPLAY QUERY

Displays the query so far.

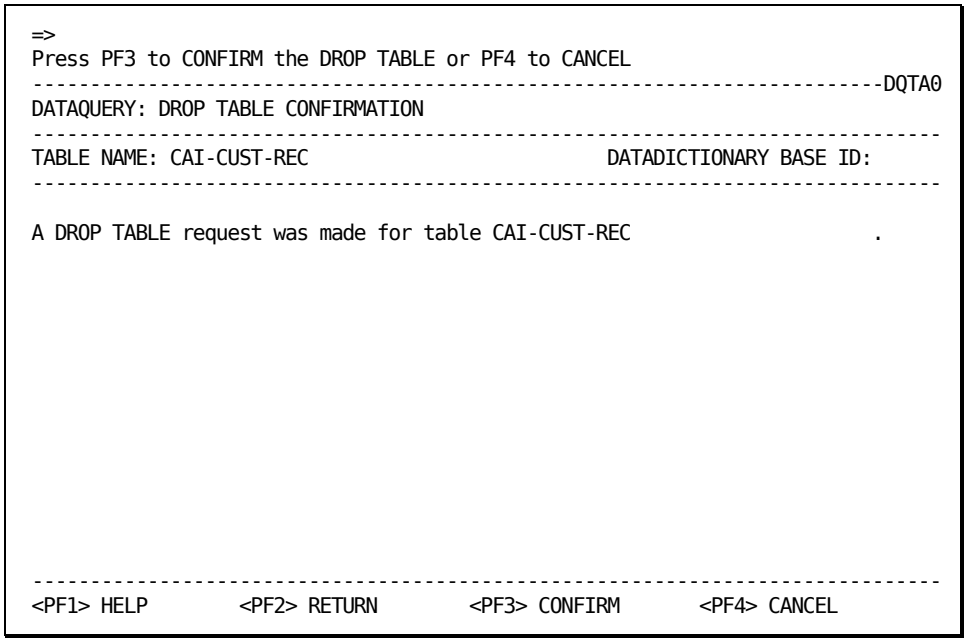
<PF7> BACKWARD

Scrolls back to see previous panels of multi-screen panel.

<PF8> FORWARD

Scrolls forward to see next panel, if a multi-screen panel.

(DQTA0) Drop Table Confirmation



Description

Deleting a table in your Personal Data Base requires you to press a PF key or issue the DROP command and confirm the deletion with this panel.

Action

To confirm the deletion, press <PF3> CONFIRM. To cancel the deletion, press <PF4> CANCEL.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> CONFIRM

Confirms that you want to delete the table.

<PF4> CANCEL

Cancels the DROP command.

(DQTB0) STORE Command Prompt

```
=>

------------------------------------------------------------------------------DQTB0
DATAQUERY:  STORE COMMAND

STORE  __________________
           (1)
--------------------------------------------------------------------------------

 (1) Enter a name for the table to be created which is
     unique for your private SQL Authorization ID.
     Press Enter to create the table.









--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

Description

The Store Command prompter appears when you issue the STORE command during query output without providing a name for the personal database table you want to create. The CA Datacom/DB SQL Option and personal database authorization are required. See STORE - Create a Personal Table from Output (see page 400) for details.

Action

Enter a valid table name and press Enter.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

(DQT10) Directory of Tables

```
=>
PLACE THE CURSOR BY THE DESIRED NAME AND PRESS THE APPROPRIATE PFKEY
--------------------------------------------------------------------------------DQT10
DATAQUERY:  DIRECTORY OF TABLES             DATADICTIONARY BASE ID: 00016
                                            START WITH:   ___________________
--------------------------------------------------------------------------------
      TABLE NAME                   | DESCRIPTION
--------------------------------------------------------------------------------
  ORG                              | ORGANIZATION INFORMATION
  STAFF                            | STAFF DATA
  PERSONNEL                        | EMPLOYEE INFORMATION
  RETIREMENT                       | RETIREMENT PLANNING
  BENEFITS                         | BENEFIT RECORDS
  QUOTAS                           | QUOTA PER SALES REP
  CLIENTS                          | CLIENT NAMES AND ADDRESSES
  BUDGET                           | DEPARTMENTAL BUDGET
  COMPETITORS                      | COMPETITOR PRODUCT INFORMATION
                                   |
                                   |
- LAST PAGE ------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN       <PF3> CREATE         <PF4> DROP
<PF5> DISPLAY       <PF6> INSERT ROWS  <PF7> BACKWARD       <PF8> FORWARD
<PF9> MAINTAIN ROWS
```

Description

This panel shows the Datadictionary database ID of your personal tables, with the names and descriptions for your authorization ID.

Action

The panel allows you to press a PF key to define a new table, delete a table, maintain a table, or copy definitions to create a new table.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

DATADICTIONARY BASE ID:

Shows the ID of the database where your personal tables are stored.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
START WITH:	N	A unique partial name	First table name

Columns

Refer to the following for a discussion of each column on this panel.

TABLE NAME

Shows the names of all tables in your personal database.

DESCRIPTION

The description assigned the table at creation.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> CREATE

Creates a new personal table.

<PF4> DROP

Drops the table indicated by the cursor.

<PF5> DISPLAY

Shows the definitions for the table indicated by the cursor.

<PF6> INSERT ROWS

Adds data to the table indicated by the cursor.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

<PF9> MAINTAIN ROWS

Finds data in the table indicated by the cursor. Changes data if needed.

(DQT20) Display Table

```
=>

-------------------------------------------------------------------------------DQT20
DATAQUERY:  DISPLAY TABLE         TABLE NAME:  STAFF
-------------------------------------------------------------------------------
COLUMN NAME                     |   TYPE    | LENGTH
-------------------------------------------------------------------------------
ID                              | SMALLINT  |      7
NAME                            | CHARACTER |      9
DEPT                            | SMALLINT  |      7
JOB                             | CHARACTER |      5
YEARS                           | SMALLINT  |      3
SALARY                          | DECIMAL   |  (7,2)
COMM                            | DECIMAL   |  (7,2)
                                |           |
                                |           |
                                |           |
                                |           |
                                |           |
                                |           |
- LAST PAGE ----------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN       <PF3> NOT USED     <PF4> DROP
<PF5> COPY          <PF6> INSERT ROWS  <PF7> BACKWARD     <PF8> FORWARD
<PF9> MAINTAIN ROWS
```

Description

This panel shows the definitions of all the columns making up a personal table.

Action

Display this panel when you want a reference to understanding table structure during query or dialog creation. Or, use the PF keys to copy the definitions to a new table or to insert rows of data in the existing table. Be aware that the maximum number of columns in a personal database table is 99.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE NAME:

Name of the personal table currently displayed.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Lists the names of all columns that make up the table.

TYPE

See Data Types (see page 609) for explanations of these data types:

CHARACTER

All displayable characters

NUMERIC

Signed or unsigned numeric data with zoned decimals

DECIMAL

Packed decimal numeric data

FLOAT

Floating-point

SMALLINT

Small integer numbers

INTEGER

Large integer numbers

LENGTH

Shows the number of positions in the column. If two numbers are separated by a comma, read the first number as the total length and the second number as the number of decimal places. This is different from the way precision is shown in DQL Mode.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> DROP

Deletes the currently displayed table.

<PF5> COPY

Copies the definitions to create a new table.

<PF6> INSERT ROWS

Adds data to the table.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

<PF9> MAINTAIN ROWS

Finds or changes data in the table.

(DQT30) Create Table

```
=>
Enter information for table columns and press appropriate PF key
-------------------------------------------------------------------------------DQT30
DATAQUERY:  CREATE TABLE          TABLE NAME:  _______________________________
-------------------------------------------------------------------------------
COLUMN NAME                     |   TYPE    | LENGTH
-------------------------------------------------------------------------------
CUST-ID                         | CHARACTER |  5
IND-CD                          | CHARACTER |  1
CUST-NO                         | CHARACTER |  4
NAME                            | CHARACTER |  30
ADDR-1                          | CHARACTER |  30
ADDR-2                          | CHARACTER |  30
CITY                            | CHARACTER |  15
STATE                           | CHARACTER |  2
ZIP                             | CHARACTER |  9
CRED-IND                        | CHARACTER |  1
PHONE                           | CHARACTER |  10
AREA-CD                         | CHARACTER |  3
PH-EXCH                         | CHARACTER |  3
PH-NO                           | CHARACTER |  4
-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> ADD TABLE     <PF4> NOT USED
<PF5> INSERT COLUMN <PF6> DELETE COLUMN <PF7> BACKWARD <PF8> FORWARD
```

Description

This panel appears when you copy an existing table definition or when you press <PF3> CREATE during display of the Display Tables (DQT10) panel. Use it to record (or change, if copying) the definitions for a new personal table. Be aware that the maximum number of columns in a personal table is 99.

Action

Type in column names and define their data types and lengths. Then press <PF5> ADD TABLE to finish creating the table.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
TABLE NAME:	Y	A one-word name (See following)	None
COLUMN NAME	Y	A one-word name	None

When no key is specified, the first column becomes the key by default. The Datadictionary limit for a key is 180 characters. This limit becomes important if the table is to be accessed with a DQL Mode query.

Table and column names must begin with an alphabetic character and use up to 32 characters, including letters, numbers, and underscores (_).

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Each display field in this column of the panel contains the name of a column defined in the table.

TYPE

Each display field in this column describes the data type defined for the column to the left. See Data Types (see page 609) for more information about data types.

CHARACTER

All displayable characters

NUMERIC

Signed, zoned, decimal numeric data

DECIMAL

Packed decimal numeric data

FLOAT

Floating-point

SMALLINT

Small integer numbers

INTEGER

Large integer numbers

LENGTH

Defines the number of positions in the column named to the left. Not used for float, smallint, or integer data types. If there are two numbers separated by a comma, the first number shows total length and the second number shows decimal positions.

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> ADD TABLE

Creates the newly defined table.

<PF5> INSERT COLUMN

Inserts a blank line for defining a new column.

<PF6> DELETE COLUMN

Erases a column definition.

<PF7> BACKWARD

Scrolls list backward one page.

<PF8> FORWARD

Scrolls list forward one page.

(DQT40) Maintain Rows

```
=>

Enter COMPARISON OPERATOR(S) AND VALUE(S) AND PRESS PF3 TO SELECT ROWS
--------------------------------------------------------------------------------DQT40
DATAQUERY:  MAINTAIN ROWS         TABLE NAME:  STAFF
--------------------------------------------------------------------------------
 COLUMN NAME                    |  TYPE   | LENGTH|COMP|     VALUE
--------------------------------------------------------------------------------
ID                              |SMALLINT|      5|    |
NAME                            |CHAR    |      9|    |
DEPT                            |SMALLINT|      5|    |
JOB                             |CHAR    |      5|    |
JOB_DESCRIPTION                 |CHAR    |     50|    |
YEARS                           |SMALLINT|      5|    |
SALARY                          |DECIMAL |  (7,2)|    |
COMM                            |DECIMAL |  (7,2)|    |
                                |        |       |    |
                                |        |       |    |
                                |        |       |    |
                                |        |       |    |
                                |        |       |    |
- LAST PAGE ----------------------------------------------------------------------
<PF1> HELP        <PF2>  RETURN       <PF3>  SELECT       <PF4>  NOT USED
<PF5> NOT USED    <PF6>  NOT USED     <PF7>  BACKWARD     <PF8>  FORWARD
```

Description

This panel defines one row of a personal table. The fields in the first three columns are display only. The fields in the last two columns allow you to enter comparison operators and search criteria to specify a subset of rows that you want to see.

Action

Press <PF3> SELECT without making entries to display all rows of the table. Or, to see a subset of data, enter operators and search criteria and then press <PF3> SELECT. To exit without finding rows, press <PF2> RETURN.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE NAME:

Names the table whose definitions are currently displayed.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Each display field in this column names a column defined in the table

TYPE

Each display field in this column describes the data type defined for the column name to the left. See Data Types (see page 609) for explanations of these data types:

CHARACTER

All displayable characters

NUMERIC

Signed or unsigned numeric data with zoned decimals

DECIMAL

Packed decimal numeric data

FLOAT

Floating-point

SMALLINT

Small integer numbers

INTEGER

Large integer numbers

LENGTH

Defines the number of positions in the column named to the left. Not used for float, smallint, or integer data types. If there are two numbers separated by a comma, the first number shows total length and the second number shows decimal positions.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
COMP	Yes, if Value contains an entry	A relational operator: ■ = or EQ *Equal* ■ < or LT *Less than* ■ > or GT *Greater than* ■ <> or LG *Less than or greater than* ■ <= or LTE *Less than or equal* ■ >= or GTE *Greater than or equal* ■ ,= or NE *Not equal* ■ ,< or NLT *Not less than* ■ ,> or NGT *Not greater than*	None
VALUE	Yes, if entry in COMP	Valid value for column named at left	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> SELECT

Displays qualified rows.

<PF7> BACKWARD

Scrolls display backward one page.

<PF8> FORWARD

Scrolls display forward one page.

(DQT50) Insert Rows

```
=>
Enter VALUE(S) IN THE VALUE COLUMN AND PRESS PF3 TO INSERT THE ROW
-------------------------------------------------------------------------------DQT50
DATAQUERY:  INSERT ROWS           TABLE NAME:  STAFF
-------------------------------------------------------------------------------
 COLUMN NAME                    |   TYPE    | LENGTH  | VALUE
-------------------------------------------------------------------------------
ID                              | SMALLINT  |       5 |
NAME                            | CHARACTER |       9 |
DEPT                            | SMALLINT  |       5 |
JOB                             | CHARACTER |       5 |
JOB_DESCRIPTION                 | CHARACTER |      50 |
YEARS                           | SMALLINT  |       5 |
SALARY                          | DECIMAL   |   (7,2) |
COMM                            | DECIMAL   |   (7,2) |
                                |           |         |
                                |           |         |
                                |           |         |
                                |           |         |
- LAST PAGE ----------------------------------------------------------------------
<PF1> HELP        <PF2>  RETURN       <PF3>  INSERT       <PF4>  NOT USED
<PF5> NOT USED    <PF6>  NOT USED     <PF7>  BACKWARD     <PF8>  FORWARD
```

Description

This panel appears as a result of pressing <PF6> INSERT ROWS during display of the Directory of Tables (DQT10) or during display of the Display Table (DQT20) panels. The above panel shows the columns defined for the personal table, the valid data types for each column, and the maximum length of each column.

Action

Enter valid values in the fields provided to create a new row and then press <PF3> INSERT to insert the row in the table.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE NAME:

Shows the name of the personal table to which you will add this row.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Each display field in this column names a column defined in the table.

TYPE

Each display field in this column describes the data type defined for the column name to the left. See Data Types (see page 609) for explanations of these data types:

CHARACTER

All displayable characters

NUMERIC

Signed or unsigned numeric data with zoned decimals

DECIMAL

Packed decimal numeric data

FLOAT

Floating-point

SMALLINT

Small integer numbers

INTEGER

Large integer numbers

LENGTH

Defines the number of positions in the column named to the left. Not used for float, smallint, or integer data types. If there are two numbers separated by a comma, the first number shows total length and the second number shows decimal positions.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
VALUE	Y	Defined at left	None

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF3> INSERT

Inserts this row in the table.

<PF7> BACKWARD

Scrolls display backward one page.

<PF8> FORWARD

Scrolls display forward one page.

(DQT60) Data Display

```
=>
DQ471I - RECORDS FOUND: 000038
-------------------------------------------------------------------------------DQT60
DATAQUERY:  DATA DISPLAY          TABLE NAME:  STAFF
                                  ROWS SELECTED: 00038 ROW DISPLAYED: 00001
-------------------------------------------------------------------------------
 COLUMN NAME                    |   TYPE    | LENGTH  | VALUE
-------------------------------------------------------------------------------
ID                              | SMALLINT  |       5 | 0
NAME                            | CHARACTER |       9 |
DEPT                            | SMALLINT  |       5 | 0
JOB                             | CHARACTER |       5 |
YEARS                           | SMALLINT  |       5 | 0
SALARY                          | DECIMAL   |   (7,2) | 0.00
COMM                            | DECIMAL   |   (7,2) | 0.00
                                |           |         |
                                |           |         |
                                |           |         |
                                |           |         |
                                |           |         |
- LAST PAGE ----------------------------------------------------------------------
<PF1> HELP        <PF2>  RETURN       <PF3>  NOT USED     <PF4>  UPDATE ROW
<PF5> DELETE ROW  <PF6>  NOT USED     <PF7>  BACKWARD     <PF8>  FORWARD
<PF9> NOT USED    <PF10> NOT USED     <PF11> PREV ROW     <PF12> NEXT ROW
```

Description

This panel shows the total number of rows found after pressing <PF3> SELECT during display of the Maintain Rows (DQT40) panel. It tells which sequential row is currently displayed and shows the definition and value of each column in the current row. If *More* appears, the display continues beyond the current panel.

Action

Page through the panels representing each row to display current data. To change a row, enter valid data in the Value fields, typing over displayed data where necessary. Press <PF4> UPDATE ROW to change the data. Other PF keys allow you to delete the entire row, page forward and backward through the current row, or see the previous or next row.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

TABLE NAME:

Names the personal table whose rows are currently displayed.

ROWS SELECTED:

Shows the number of rows included in this display.

ROW DISPLAYED:

Shows the sequential number of this row.

Columns

Refer to the following for a discussion of each column on this panel.

COLUMN NAME

Each display field in this column names a column defined in the table.

TYPE

Each display field in this column describes the data type defined for the column name to the left. See Data Types (see page 609) for explanations of these data types:

CHARACTER

All displayable characters

NUMERIC

Signed or unsigned numeric data with zoned decimals

DECIMAL

Packed decimal numeric data

FLOAT

Floating-point

SMALLINT

Small integer numbers

INTEGER

Large integer numbers

LENGTH

Defines the number of positions in the column named to the left. Not used for float, smallint, or integer data types. If there are two numbers separated by a comma, the first number shows total length and the second number shows decimal positions.

Entry Fields

Refer to the following chart for valid entries in the fields on this panel. Use the Tab and arrow keys to move the cursor. To change an entry, write or space over it. When entries are complete, press Enter or a valid PF key.

Field	Required?	Valid Entries	Default
VALUE	N	Shown at left	Displayed

PF Keys

The following lists the PF keys for this panel, with a brief description of each function.

<PF1> HELP

Displays Help information about the current panel.

<PF2> RETURN

Exits the current panel.

<PF4> UPDATE ROW

After entering new data in Value fields, press this key to save the entries.

<PF5> DELETE ROW

Removes this row and its data from the table.

<PF7> BACKWARD

Scrolls display backward one page.

<PF8> FORWARD

Scrolls display forward one page.

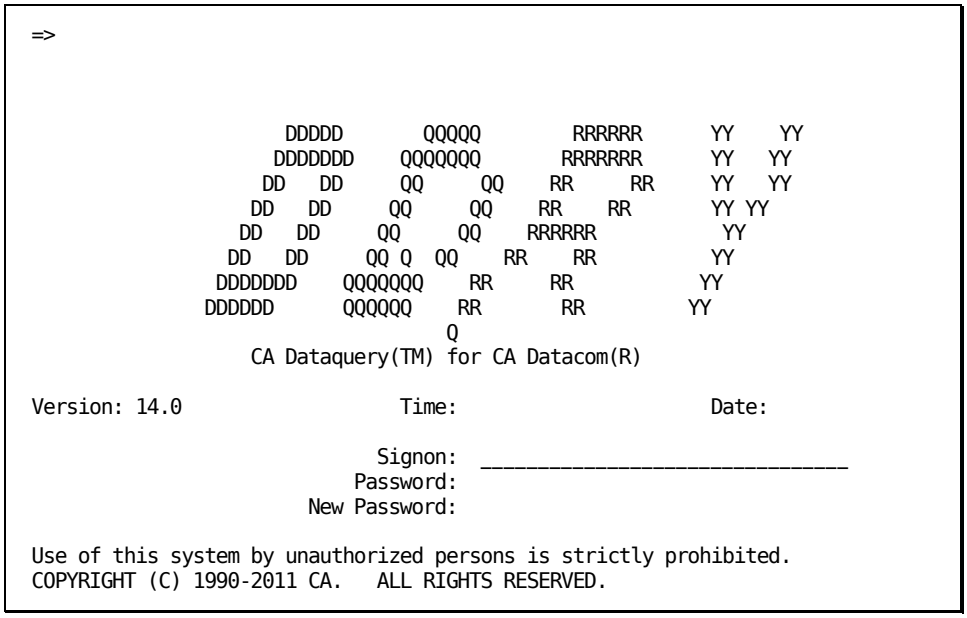
<PF11> PREV ROW

Displays first page of a previous row, if any.

<PF12> NEXT ROW

Displays first page of next row, if any.

(DQZ10) Signon



Description

After you complete the standard signon procedure for your site, CA Dataquery displays the Signon panel. If your site does not permit users to change their own passwords, the New Password field does not appear on your Signon panel.

Action

Enter your operator ID and password, if any, and press Enter to sign on. To change your password, if permitted at your site, also enter a new password in the New Password field.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

Version:

CA Dataquery version number currently running

Time:

System time

Date:

System date

Entry Fields

Signon:

Enter a valid 1- to 32-character user ID. Valid IDs are assigned by the CA Dataquery Administrator.

Password:

Enter a 1- to 9-character password, if one has been assigned. The password will not be displayed. If you have forgotten your password, your CA Dataquery Administrator can assign you a new password.

New Password:

Enter a 1- to 9-character new password to change your password assignment and press Enter, if you want to change your password. This field does not appear on the Signon panel if your site does not permit users to change passwords.

(DQZ20) SignOff

```
DQ194I - DATAQUERY SIGN/OFF COMPLETED


                    DDDDD       QQQQQ       RRRRRR       YY     YY
                   DDDDDDD    QQQQQQQ      RRRRRRR      YY     YY
                  DD    DD    QQ     QQ    RR    RR     YY     YY
                 DD    DD     QQ     QQ    RR    RR     YY  YY
                DD    DD      QQ     QQ    RRRRRR        YY
               DD    DD       QQ Q  QQ     RR    RR      YY
              DDDDDDD     QQQQQQQ     RR     RR        YY
             DDDDDD       QQQQQQ     RR      RR       YY
                                 Q


Version: 14.0                Time:                      Date:


COPYRIGHT (C) 1990-2011 CA.   ALL RIGHTS RESERVED.
```

Description

When you exit CA Dataquery, the Signoff panel appears. Use the procedure used at your site for exiting the process by which you accessed CA Dataquery. See your CA Dataquery Administrator for details of your monitor logoff procedures.

Action

Site-dependent. See your CA Dataquery Administrator if you need assistance.

Display Fields

Refer to the following chart for explanations of the display fields intended to contain variable data.

Version:

CA Dataquery version number currently running.

Time:

System time

Date:

System date

(DQZ50) Main Menu (SQL Mode)

```
=>

--------------------------------------------------------------------------------DQZ50
DATAQUERY:  MAIN MENU - SQL MODE

Enter THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries and Tables
 2.  CREATE              - Query, Dialog creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination
```

Description

If you are a fully authorized SQL Mode user, you see this Main Menu when you sign on to CA Dataquery for CA Datacom and when you exit from one of its options. The options provide access to all the functions you need for creating and running queries and controlling your environment.

If you are an associate user who executes existing queries, CA Dataquery for CA Datacom bypasses the Main Menu and displays the Directory of Queries.

Note: PDB only appears for those users authorized to use the feature. *PDB does not appear if the CA Datacom/DB SQL option is not installed.*

Action

Select an item from the menu according to the task you want to perform by entering its number in the field provided and pressing Enter. You can also enter any valid command on the command line.

Select

DIRECTORIES

Display Directory Selection panel showing types of directories you can display.

Select a directory type (queries, dialogs, and database tables) to display.

Use the directories to maintain libraries, execute queries, copy or edit queries, display database information, or see a definition of an item on the directory.

CREATE

Access the CA Dataquery Editor to write a query or dialog.

GUIDE

Display the GUIDE Template.

PDB

List, create, or maintain your personal tables.

ADMINISTRATION

Display menu listing your User Profile of terminal session options and any other administrative tasks you are authorized to perform.

HELP

Display panel listing topics you can select for additional information.

OFF

Exit CA Dataquery for CA Datacom and return to monitor.

(DQZ60) Main Menu (DQL Mode)

```
=>

--------------------------------------------------------------------------------DQZ60
DATAQUERY:  MAIN MENU - DQL MODE

Enter THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
 2.  CREATE              - Query, Dialog or Term creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination
```

Description

If you are a fully authorized user, you see the Main Menu when you sign on to CA Dataquery for CA Datacom and when you exit from one of its options. The options provide access to all the functions you need for creating and running queries and controlling your environment. If you are an associate user who executes existing queries, CA Dataquery for CA Datacom bypasses the Main Menu and displays the Directory of Queries and Terms.

Note: PDB only appears for those users authorized to use the feature. *PDB does not appear if the CA Datacom/DB SQL option is not installed*.

Action

Select an item from the menu according to the task you want to perform by entering its number in the field provided and pressing Enter. You can also enter any valid command on the command line.

Select

DIRECTORIES

Display Directory Selection panel showing types of directories you can display.

Select a directory type (queries, dialogs, and database tables) to display.

Use the directories to maintain libraries, execute queries, copy or edit queries, display database information, or see a definition of an item on the directory.

CREATE

Access the CA Dataquery Editor to write a query or dialog.

GUIDE

Display the GUIDE Template.

PDB

List, create, or maintain your personal tables.

ADMINISTRATION

Display menu listing your User Profile of terminal session options and any other administrative tasks you are authorized to perform.

HELP

Display panel listing topics you can select for additional information.

OFF

Exit CA Dataquery for CA Datacom and return to monitor.

Chapter 4: Editor Commands

Use the CA Dataquery Editor and your knowledge of DQL or SQL to build queries, dialogs and terms ranging from the simple to the complex. To aid you in your task, the Editor has line commands you use to insert, copy, move, delete, repeat and split lines, and commands which allow you to shift the position of data within a line, locate or change character strings, and scroll text forward and backward.

Special Aids

PF keys are available when you are building your query to display database information, and a template from which you can select statements to insert into query text on the Editor panel. Other PF keys allow you to scroll forward and backward if your query is more than one panel long or left and right if your terminal is an 80-character width display. Another special PF key, available in both modes, lets you validate your query, that is, check for errors in syntax and names of database tables, columns and keys.

Editor Modes

The Editor panel provides you with a Create mode where you build your query and a Process mode where you can perform such functions as executing, saving and deleting queries. A special PF key allows you to switch back and forth between these two modes.

Note: For information about using the CA Dataquery Editor to create queries, dialogs, report definitions and terms, see the *CA Dataquery User Guide*.

Create Mode

Selecting CREATE from the Main Menu or typing *CREATE* in the command line on any panel displays the Editor panel in Create mode. Using this panel, you can:

- Insert keyword formats selected from the template or type in statements in the space provided on the panel.
- Insert any comments that you want to exclude from the query by preceding them with the characters used at your site. If your site does not use the default */ to begin a comment and /* to end a comment, see your CA Dataquery Administrator for the characters used.
- Display database information using PF keys, and insert table, column and key names at the appropriate location in query statements.
- Use the CA Dataquery Editor commands to insert, copy, move, repeat, delete and split lines; scroll the display to a specific line; shift data left or right within a line; locate and change strings within the query.

- Validate the query, that is, check for errors in table, column and key names and syntax.
- Access the Editor panel in Process mode.

Process Mode

Selecting the EDIT option (<PF4> EDIT) on any Directory panel where this option is available or typing *EDIT query-name* in the command line on any panel displays the Editor panel in Process mode. The *query-name* entry must be the name of an existing query or dialog. Using this panel, you can:

- Save the query you have just created.
- Execute the query you have just created or updated.
- Delete the query displayed on this panel.
- Update a saved query of your own which is displayed on this panel.
- Validate the query you have just created, saved or updated.
- Select the dialog definition option to create a dialog. (See the *CA Dataquery User Guide* for an overview of dialog creation.)
- Access the Editor panel in Create mode.
- Use the CA Dataquery Editor commands to insert, copy, move, repeat, delete and split lines; scroll the display to a specific line; shift data left or right within a line; locate and change strings within the query.

Summary of All Editor Commands

The following summarizes the commands that are discussed in detail in this chapter.

A

After -- Destination for a move or copy after line with A.

B

Before -- Destination for a move or copy before line with B.

CH /string1/string2/

Changes the first string to the second string.

C

Copies a single line to the specified destination.

Cn

Copies the indicated number of lines starting with the line on which you entered the command to the specified destination.

CB

Copies all lines following the command through the end of the panel to the specified destination.

CC

Copies the block you define to the specified destination.

CT

Copies from the first line through the line with the command.

D

Deletes a single line.

DB

Deletes from line with the command through the last line.

DD

Deletes block of lines specified by a pair of line commands *DD*.

Dn

Deletes a specified number of lines including the one on which you enter the command.

DT

Deletes from first line through line with the command.

GC /string1/string2/

Changes every occurrence of first string to second string, and leaves display positioned at last string changed.

I

Inserts a blank line after the line with the I command.

I*n*

Inserts the specified number of lines after the line with the I*n*.

NE /string/

Searches forward for the specified text string.

PR /string/

Searches backward for the specified text string.

M

Moves a single line to the specified destination.

MB

Moves a block of text starting with the line on which you enter the command through the end of the panel.

MM

Defines and moves a block of text to the specified destination.

Mn

Moves the specified number of lines to the specified destination.

MT

Moves a block of text starting with line 01 and continuing through the line on which you enter the command.

R

Repeats the line on which you enter the R command.

Rn

Repeats the line with the R command n times.

*****

Scrolls the line on which you enter the * to the top of the display.

T

Scrolls to the top of the panel.

<n

Shifts the display the specified number of columns to the left.

>n

Shifts the display the specified number of columns to the right.

SP

Splits the line into two lines at the point you place your cursor.

X

On any line, cancels pending COPY, MOVE, BEFORE, or AFTER.

/ and / - Begin and End Comments

Description

You can enter special characters before and after any text you want to exclude from validation and processing. These characters may be changed by individual sites at installation. The CA Dataquery system option default for BEGIN COMMENT is */ and the default for ending a comment is /*. The System Profile topic selected from the List of Help Topics panel (DQN10) that appears with the HELP command tells you the characters in use at your site.

The BEGIN COMMENT command precedes any comment you want to add to a query without having it read by CA Dataquery during query execution. The END COMMENT command follows the text to be excluded.

Action

Enter the BEGIN COMMENT command anywhere in the text area of the Editor panel to tell CA Dataquery not to read the following text. Type an END COMMENT command after the text to be excluded.

Example

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                    TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
  ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
..============================== T O P ===================================
01 */THIS QUERY JOINS SALES HISTORY ROWS TO RECEIPTS ROWS FOR MONTHLY
02       REPORT DISTRIBUTED TO BRANCH MANAGERS/*
03 FIND 10 CAI-SLSHST-REC
04 WITH SHIP-QTY > 100
05 RELATED BY ITM-ID TO CAI-RCPTS-REC
06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
09 SORT CAI-RCPTS-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

CH (Change)

Summary

The following summarizes the CHANGE command.

CH /string1/string2/

Enter CH in the line number column. Enter the string you want to change as string1 in the text area, followed by the replacement string as string2.

For mass changes, see the GC command (see page 318).

Description

The CHANGE command locates a string and changes it to another string. CA Dataquery locates and changes the first occurrence of the string found.

Action

Enter the CHANGE command in the first position of the line number column and tab over to the query entry area. Type your search string and replacement string over the existing text. The text you type over is not deleted. The string size limit is 32 characters. You can use any special character to delimit the strings. Any special character in position 1 of a query text line where CH appears in the line number column is interpreted as the string delimiter. Examples of special characters are:

? @ # % /

CA Dataquery changes the first occurrence of the located string. This includes the current line. To change the next occurrence place CH in the line number column and press Enter. You do not need to reenter the search and replacement strings.

Example—Changing First Occurrence

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                     TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
ch /itm-id/ord-id/ > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                     TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
03 RELATED BY ORD-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
12 UNIT-PRICE
13 (NET-COST)
14 DISCOUNT
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Changing Next Occurrence

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
CH RELATED BY ORD-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
12 UNIT-PRICE
13 (NET-COST)
14 DISCOUNT
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
07 SORT CAI-RCPTS-REC BY (ORD-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
12 UNIT-PRICE
13 (NET-COST)
14 DISCOUNT
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

C (Copy)

Summary

The following table summarizes the COPY command.

Command	Action	Result
C	Enter a C in the line number of the line you want to copy and specify the new destination using an A or B destination command.	Copies the designated line to the destination you indicate.
Cn	Enter a C and the number of lines you want to copy and specify a number for the total number of lines to copy.	Copies the indicated number of lines including the line on which the destination command was entered.
CB	Enter the CB command on the line where you want the copy to begin. Specify a destination. The destination cannot be in the block.	Copies all lines following the command through the end of the text. Blank lines found at the end of the text are not copied to the destination.
CC	Enter CC in the line number column of the first line of the block you want to copy. Then enter CC in the line number column of the last line of the block you want to copy.	Copies the defined block to the destination you indicate.
CT	Enter the CT command in the line number column of the line through which you want to copy. Specify a destination. The destination of the copy cannot be within the block.	Copies the first line of the text through the line on which the CT command is entered.

Note: CA Dataquery does not allow you to complete a copy if the copy operation would force lines which contain text to shift beyond the maximum number of lines specified in your Option Table.

Description

You can use the COPY command to duplicate lines in the query text.

Action

Enter the command in the first position of the line number column in the source and specify the destination by entering an A (AFTER) or B (BEFORE) in the line number column of the destination. Specifying an A places the copied lines after the line on which the A is placed. B specifies that the copied lines are placed before the line on which you placed the B.

Note: The A or B line commands must be placed in the *first* position of the two-position line command area.

Example—Copy a Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY A LINE TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
A  SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
08 FROM CAI-SLSHST-REC ITM-ID
C9 SHIP-QTY
10 UNIT-PRICE
11 (NET-COST)
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY A LINE TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SHIP-QTY
08 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
09 FROM CAI-SLSHST-REC ITM-ID
10 SHIP-QTY
11 UNIT-PRICE
12 (NET-COST)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Copy Multiple Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
c2 SHIP-QTY
08 UNIT-PRICE
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
a0 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
08 FROM CAI-SLSHST-REC ITM-ID
09 SHIP-QTY
10 UNIT-PRICE
11 SHIP-QTY
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Copy Remaining Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY ALL FOLLOWING LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
A2 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 PRINT FROM CAI-SLSHST-REC
05 ORD-QTY
06 ORD-DAY
07 ORD-MO
CB ORD-YR
09 ORD-DT
.. ========================  B O T T O M  ================================

-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY ALL FOLLOWING LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 ORD-YR
04 ORD-DT
05 RELATED BY ITM-ID TO CAI-RCPTS-REC
06 PRINT FROM CAI-SLSHST-REC
07 ORD-QTY
08 ORD-DAY
09 ORD-MO
10 ORD-YR
11 ORD-DT
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Copy a Block

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY A BLOCK TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
A2 WITH SHIP-QTY > 100
03 PRINT FROM CAI-SLSHST-REC
CC ORD-QTY
05 ORD-DAY
CC ORD-MO
07 ORD-YR
08 ORD-DT
.. ========================  B O T T O M  ================================


-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   COPY A BLOCK TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 ORD-QTY
04 ORD-DAY
05 ORD-MO
06 PRINT FROM CAI-SLSHST-REC
07 ORD-QTY
08 ORD-DAY
09 ORD-MO
10 ORD-YR
11 ORD-DT
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Copy Previous Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          A-CUST-CITY-DIA                   TYPE: DIALOG  STATUS: PUBLIC
DESCRIPTION:   COPY ALL PREVIOUS LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01          */THIS PART OF THE DIALOG ALLOWS YOU TO CHANGE
CT            ONLY THE/*
03 FIND 1?10 CAI-CUST-REC
04 WITH CITY = 2?'DALLAS'
05 PRINT       TITLE1 'SAMPLE ORDER ENTRY DATABASE'
A6             TITLE2 'CUSTOMER FILE'
07             NAME
08             ADDR-1
09             ADDR-2
10             CITY
11             STATE
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          A-CUST-CITY-DIA                  TYPE: DIALOG  STATUS: PUBLIC
DESCRIPTION:   COPY ALL PREVIOUS LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01          */THIS PART OF THE DIALOG ALLOWS YOU TO CHANGE
02            ONLY THE/*
03 FIND 1?10 CAI-CUST-REC
04 WITH CITY = 2?'DALLAS'
05 PRINT       TITLE1 'SAMPLE ORDER ENTRY DATABASE'
06              TITLE2 'CUSTOMER FILE'
07          */THIS PART OF THE DIALOG ALLOWS YOU TO CHANGE
08            ONLY THE
09              NAME
10              ADDR-1
11              ADDR-2
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

D (Delete)

Summary

The following table summarizes the DELETE command.

Command	Action	Result
D	Enter a D in the line number column of the line you want to delete.	Deletes the line you specified.
DB	Enter DB in the line number column of the first line you want to delete.	Deletes from the line you specified through the last line.
DD	Enter DD in the line number column of the first and last line you want to delete.	Deletes the block of lines marked off by the DD pair.
Dn	Enter a D with the number of lines you want to delete.	Deletes the specified number of lines starting with the line on which you entered the DELETE command.

Command	Action	Result
DT	Enter DT on the line number of the last line you want to delete.	Deletes from line 1 through the line you specified.

Description

You can use the DELETE command to delete single lines or a specified number of lines.

Action

To delete one line, enter D in the first position of the line number column of the line. To use the other DELETE commands, enter them beginning in the first position of the line number column of the line you select.

Example—Delete a Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY2                    TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   DELETE ONE LINE ____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
D2 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY2                     TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   DELETE ONE LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 RELATED BY ITM-ID TO CAI-RCPTS-REC
03 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
04 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
05 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
06 SORT CAI-RCPTS-REC BY (ITM-ID)
07 CAI-SLSHST-REC BY (ITM-ID)
08 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
09 FROM CAI-SLSHST-REC ITM-ID
10 SHIP-QTY
11 UNIT-PRICE
--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Delete Multiple Lines

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY2                     TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   DELETE MORE THAN ONE LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
11 UNIT-PRICE
D3 (NET-COST)
13 DISCOUNT
14 (DISCOUNT-COST)
.. ========================  B O T T O M  ================================




--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY2                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   DELETE MORE THAN ONE LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
11 UNIT-PRICE
.. ========================  B O T T O M  ================================




-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

GC (Mass Change)

Summary

The following table summarizes the GC command.

Command	Parameters	Instructions
GC	/string1/string2/	Enter GC in the line number column. Enter the string you want to change as string1 in the text area, followed by the replacement string as string2.

Description

The GC command locates all occurrences of a string and changes them to another string.

Action

Enter the GC command in the first position of the line number column and tab over to the query entry area. Type your search string and replacement string over the existing text. The text you type over is not deleted. The string size limit is 32 characters. You can use any special character to delimit the strings. Any special character in position 1 of a query text line where GC appears in the line number column is interpreted as the string delimiter. Examples of special characters are:

? @ # % /

CA Dataquery changes all occurrences of the string. This includes the current line.

Example—Changing All Occurrences

```
=>
--------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
GC /ORD-ID/EMP-ID/ CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
12 UNIT-PRICE
13 (NET-COST)
14 DISCOUNT
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                      TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
07 SORT CAI-RCPTS-REC BY (EMP-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'EMP-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
12 UNIT-PRICE
13 (NET-COST)
14 DISCOUNT
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

I (Insert)

Summary

The following table summarizes the INSERT command.

Command	Action	Result
I	Place an I in the line number column and press Enter.	Inserts a blank line immediately after the line on which you specified the I command.
In	Enter an I along with the number of lines you want to insert and press Enter.	Inserts the specified lines immediately after the line on which you specified the I command.

Description

You can use the INSERT command to add blank lines in the query text.

Action

Enter an I in the first position of the line number column and the number of lines you want to add. If you do not specify a number CA Dataquery defaults to one. You cannot specify more than nine lines at one time.

The lines are added immediately below the line on which you enter the command.

Example—Insert a Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   INSERT ONE LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND ALL CAI-SLSHST-REC ROWS
I  PRINT SLMN-ID
.. ========================  B O T T O M  ================================




------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   INSERT ONE LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND ALL CAI-SLSHST-REC ROWS
02 PRINT SLMN-ID
03
.. ========================  B O T T O M  ================================




------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Insert Multiple Lines

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   INSERT MULTIPLE LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND ALL CAI-SLSHST-REC ROWS
I9 PRINT SLMN-ID
.. ========================  B O T T O M  ================================




------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   INSERT MULTIPLE LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND ALL CAI-SLSHST-REC ROWS
02 PRINT SLMN-ID
03
04
05
06
07
08
09
10
11
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

NE (Locate Text)

Summary

The following table summarizes the LOCATE command.

Format	Action	Result
NE /string/	Enter NE in the line number column. Enter the search string in the text entry area of the panel. Use a special character to delimit the search string.	Locates the first occurrence of the string after the line on which you entered the command. Does not find a match on the current line.

Description

Use the NE command to locate character strings by searching forward. Or, you can search backward using the PR (locate previous) command described on a following page. The string size limit is 32 characters.

Action

To search for character strings, issue the command and define the string. You issue the command in the line number column and specify the string in the actual text entry area of the editor panel. Type the search string over existing text. CA Dataquery does not delete the text you type over. Use special characters to delimit the search string. Any special character placed in position 1 (column 1) or the query text line where NE appears in the line number column is interpreted as the string delimiter. For example some special characters are:

? @ # % /

To repeat the search, reenter the command in the line number column. You do not need to reenter the search text.

Example—Locate Next

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   CHANGE QUERY TEXT_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
NE /ITM-ID/CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ITM-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>------------------------------------------------------------------------DQD1
0
 DATAQUERY:  EDITOR
 --------------------------------------------------------------------------------
 NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
 DESCRIPTION:   LOCATE QUERY TEXT
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
 03 RELATED BY ITM-ID TO CAI-RCPTS-REC
 04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
 05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
 06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
 07 SORT CAI-RCPTS-REC BY (ITM-ID)
 08 CAI-SLSHST-REC BY (ITM-ID)
 09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
 10 FROM CAI-SLSHST-REC ITM-ID
 11 SHIP-QTY
 12 UNIT-PRICE
 13 (NET-COST)
 14 DISCOUNT
 --------------------------------------------------------------------------------
 <PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
 <PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
 <PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

PR (Locate Previous)

Summary

The following table summarizes the LOCATE PREVIOUS command.

Format	Action	Result
PR /string/	Enter the string of characters you want to locate. Use any special characters (/) to delimit the search string.	Searches backward, displaying the first occurrence of the string it finds.

Description

Use the PR command to locate character strings backward. You can search forward using the NE (next) command described on a previous page.

Action

To search for character strings, issue the command and define the string. You issue the command in the line number column and specify the string in the actual text entry area of the editor panel. Type the search string over existing text. CA Dataquery does not delete the text you type over. The string size limit is 32 characters. Use special characters to delimit the search string. Any special character placed in position 1 (column 1) or the query text line where PR appears in the line number column is interpreted as the string delimiter. For example some special characters are:

? @ # % /

To repeat the search, reenter the command in the line number column. You do not need to reenter the search text.

Example—Locate Text Backward

```
=>-------------------------------------------------------------------------DQD1
0
 DATAQUERY:  EDITOR
 ------------------------------------------------------------------------------
 NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
 DESCRIPTION:   LOCATE QUERY TEXT
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
 PR /ITM-ID/
 15 (DISCOUNT-COST)
 .. ========================  B O T T O M  ================================




 ------------------------------------------------------------------------------
 <PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
 <PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
 <PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
 =>
 -------------------------------------------------------------------------DQD10
 DATAQUERY:  EDITOR
 ------------------------------------------------------------------------------
 NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
 DESCRIPTION:   LOCATE QUERY TEXT
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
 10 FROM CAI-SLSHST-REC ITM-ID
 11 SHIP-QTY
 12 UNIT-PRICE
 13 (NET-COST)
 14 DISCOUNT
 15 (DISCOUNT-COST)
 .. ========================  B O T T O M  ================================




 ------------------------------------------------------------------------------
 <PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
 <PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
 <PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

M (Move)

Summary

The following table summarizes the MOVE command.

Command	Action	Result
M	Enter an M in the line number column of the line you want to move and specify a destination.	Moves a single line to the specified destination.
MB	Enter MB in line number column of the line on which you want the copy to begin and specify a destination.	Moves the block of text starting with the line on which you entered the MB command through the end of the text. Blank lines found at the end of the text are not moved.
MM	Enter MM in the first line and the last line of the block you want to define. Specify a destination.	Moves the block of text to a specified destination.
Mn	Enter an M on the first line of the block along with the number of lines you want to include in this block and specify a destination.	Moves the specified number of lines starting with the line on which you entered the MOVE command.
MT	Enter the MT command in the line number column of the line on which you want the block to end. CA Dataquery assumes the block begins with line 01.	Moves the block of text starting with line 01 and continuing through the text to the line which contains the MT command. The line which contains the MT is included.

Description

Use the MOVE command to move lines of text from one location to another.

Action

Enter the MOVE commands in the line number column. Use an A or B to specify the destination of the move. A places the moved lines after the line on which you specify the A. B places the moved lines before the line on which you specify the B. The A or B must be placed in the first position of the line number column. The destination cannot be inside the defined block.

To cancel a block command, type an **X** in the first position of any line number column.

Example—Move One Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE ONE LINE_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
M2 RELATED BY ITM-ID TO CAI-RCPTS-REC
A3 WITH SHIP-QTY > 100
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE ONE LINE_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Move a Block

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE A BLOCK______________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
A1 FIND 10 CAI-SLSHST-REC
02 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
03 FROM CAI-SLSHST-REC ITM-ID
04 SHIP-QTY
05 UNIT-PRICE
06 (NET-COST)
07 DISCOUNT
MM WITH SHIP-QTY > 100
09 RELATED BY ITM-ID TO CAI-RCPTS-REC
10 SORT CAI-RCPTS-REC BY (ITM-ID)
MM CAI-SLSHST-REC BY (ITM-ID)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE A BLOCK   _____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SORT CAI-RCPTS-REC BY (ITM-ID)
05 CAI-SLSHST-REC BY (ITM-ID)
06 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
07 FROM CAI-SLSHST-REC ITM-ID
08 SHIP-QTY
09 UNIT-PRICE
10 (NET-COST)
11 DISCOUNT
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Move All Following Lines

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE NEXT LINES_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
A1 FIND 10 CAI-SLSHST-REC
02 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
03 FROM CAI-SLSHST-REC ITM-ID
04 SHIP-QTY
05 UNIT-PRICE
06 (NET-COST)
07 DISCOUNT
MB WITH SHIP-QTY > 100
09 RELATED BY ITM-ID TO CAI-RCPTS-REC
10 SORT CAI-RCPTS-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE NEXT LINES_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SORT CAI-RCPTS-REC BY (ITM-ID)
05 CAI-SLSHST-REC BY (ITM-ID)
06 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
07 FROM CAI-SLSHST-REC ITM-ID
08 SHIP-QTY
09 UNIT-PRICE
10 (NET-COST)
11 DISCOUNT
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Move Multiple Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE MULTIPLE LINES_____________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
A3 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SORT CAI-RCPTS-REC BY (ITM-ID)
05 CAI-SLSHST-REC BY (ITM-ID)
M3 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 (DISCOUNT-COST)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE MULTIPLE LINES_____________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 NET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 (DISCOUNT-COST)
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Move Previous Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE: CAI-SLSHST-REC
-------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE FIRST LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 IND-CD
02 CUST-ID
03 DISC-PCT
04 UNIT-PRICE
05 SHIP-QTY
06 ORD-QTY
07 ORD-DAY
MT ORD-MO
09 FIND ALL CAI-SLSHST-REC
A0 PRINT
.. ========================  B O T T O M  ================================
-------------------------------------------------------------------------------
<PF1> HELP          <PF2>  RETURN     <PF3>  DISPLAY COLUMNS <PF4> DISPLAY KEYS
<PF5> DISPLAY ALL   <PF6>  LIST TABLES <PF7> BACKWARD       <PF8>  FORWARD
<PF9> TEMPLATE      <PF10> VALIDATE   <PF11> RIGHT/LEFT     <PF12> PROCESS MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  CAI-SLSHST-REC
-------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   MOVE FIRST LINES TO ANOTHER LOCATION
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ================================
01 FIND ALL CAI-SLSHST-REC
02 PRINT
03 IND-CD
04 CUST-ID
05 DISC-PCT
06 UNIT-PRICE
07 SHIP-QTY
08 ORD-QTY
09 ORD-DAY
10 ORD-MO
.. ========================  B O T T O M  ================================
-------------------------------------------------------------------------------
<PF1> HELP          <PF2>  RETURN     <PF3>  DISPLAY COLUMNS <PF4> DISPLAY KEYS
<PF5> DISPLAY ALL   <PF6>  LIST TABLES <PF7> BACKWARD       <PF8>  FORWARD
<PF9> TEMPLATE      <PF10> VALIDATE   <PF11> RIGHT/LEFT     <PF12> PROCESS MODE
```

R (Repeat)

Summary

The following table summarizes the REPEAT command.

Command	Action	Result
R	Enter an R in the line number column of the line you want repeated.	Repeats the line on which you entered the R command.
Rn	Enter an R in the line number column of the line you want repeated and specify the number of times you want the line repeated.	Repeats the line the specified number of times.

Description

You can use the REPEAT command to repeat lines of existing text. The repeated lines are inserted below the line on which you entered the REPEAT command.

Action

Enter an R in the first position of the line number column to repeat a single line. To repeat a single line multiple times, enter an R in the first column of the line number column and the number of times you want the line repeated in the second column.

Example—Repeat One Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   REPEAT A LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
R8 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   REPEAT LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Repeat Multiple Lines

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   REPEAT LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
R2 CAI-SLSHST-REC BY (ITM-ID)
09 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
10 FROM CAI-SLSHST-REC ITM-ID
11 SHIP-QTY
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   REPEAT LINES
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Scroll

Summary

The following table summarizes the SCROLL command.

Command	Action	Result
*	Enter an * in the line number column of the line you want to scroll to the top of the display.	Scrolls the specified line to the top of the display.
n or nn	Enter the line number of the line you want scrolled to the top of the display in any line number column.	Displays the specified line at the top of the display.
T	Enter a T in the line number column of any line.	Scrolls to line 01.

Description

Use the SCROLL command to scroll a specified line to the top of the Editor panel.

Action

You enter the SCROLL commands in the first position of the line number column. You can specify a specific line number, T, or * (asterisk). If you enter a specific line number, CA Dataquery scrolls that line to the top of the Editor panel. Placing a T in the line number column scrolls to line 01. If you place an * (asterisk) in the line number column of a line, that line is scrolled to the top of the display.

Example—Scroll Current Line

```
=>
------------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL CURRENT LINE TO TOP
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
*1 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL CURRENT LINE TO TOP
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
12 FROM CAI-SLSHST-REC ITM-ID
13 SHIP-QTY
14 UNIT-PRICE
15 (NET-COST)
16 DISCOUNT
17 (DISCOUNT-COST)
.. ==========================  B O T T O M  ================================




------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Scroll Named Line to Top

```
=>
------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL NAMED LINE TO TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL NAMED LINE TO TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
12 FROM CAI-SLSHST-REC ITM-ID
13 SHIP-QTY
14 UNIT-PRICE
15 (NET-COST)
16 DISCOUNT
17 (DISCOUNT-COST)
.. ========================  B O T T O M  ================================
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Scroll to First Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL TO THE TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
12 FROM CAI-SLSHST-REC ITM-ID
13 SHIP-QTY
14 UNIT-PRICE
15 (NET-COST)
16 DISCOUNT
T7 (DISCOUNT-COST)
.. ========================  B O T T O M  ================================
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL TO THE TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
05 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
06 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
07 SORT CAI-RCPTS-REC BY (ITM-ID)
08 CAI-SLSHST-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Shift

Summary

The following table summarizes the SHIFT command.

Command	Action	Result
<n	Enter < and the number of columns you want the display to shift to the left.	Shifts the display the specified number of columns to the left.
>n	Enter > and the number of columns you want the display to shift to the right.	Shifts the display the specified number of columns to the right.

Description

The SHIFT command allows you to shift a line of text to the right or left.

Action

You can shift the display area right or left by using the < and > keys. <n shifts the display the specified number of columns left. >n shifts the display the specified number of columns right. You cannot shift to the left or the right if it would cause non-blank characters to be shifted past column 1 or column 80.

Example—Shift Right

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHIFT A LINE TO THE RIGHT n COLUMNS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
>9 */THIS PART OF THE QUERY CREATES NEW TEMPORARY RESULTS*/
05 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
06 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
07 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
08 SORT CAI-RCPTS-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHIFT A LINE TO THE RIGHT n COLUMNS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04          */THIS PART OF THE QUERY CREATES NEW TEMPORARY RES         ULTS*/
05 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
06 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
07 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
08 SORT CAI-RCPTS-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Shift a Line to the Left

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
------------------------------------------------------------------------------
NAME:          SAMPLE                          TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHIFT A LINE TO THE LEFT 3 POSITIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
<3          */THIS PART OF THE QUERY CREATES NEW TEMPORARY RES         ULTS*/
05 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
06 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
07 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
08 SORT CAI-RCPTS-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE                              TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHIFT A LINE TO THE LEFT 3 POSITIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04       */THIS PART OF THE QUERY CREATES NEW TEMPORARY RESULT         S*/
05 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
06 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
07 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
08 SORT CAI-RCPTS-REC BY (ITM-ID)
09 CAI-SLSHST-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

SP (Split)

Summary

The following table summarizes the SPLIT command.

Command	Action	Result
SP	Enter SP in the line number column of the line you want to split.	Splits the line into two lines as indicated by your cursor position.

Description

Use the SPLIT line command to split text from one line into two lines.

Action

Enter the SPLIT command in the line number column, move the cursor to the point you want the line split, and press Enter. CA Dataquery splits the text into two lines.

Example—Split a Line

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION: SPLIT A LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
SP       */THIS PART OF THE QUERY  CREATES NEW TEMPORARY RESULTS*/
```

Result

```
=>
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
-------------------------------------------------------------------------------
NAME:          SAMPLE                           TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SPLIT A LINE
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND 10 CAI-SLSHST-REC
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04       */THIS PART OF THE QUERY
05                                CREATES NEW TEMPORARY RESULTS*/
06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
09 SORT CAI-RCPTS-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
-------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

X (Cancel)

Summary

The following table summarizes the CANCEL command.

Command	Action	Result
X	Enter the X command in the line number column of any line.	Cancels any COPY, MOVE, BEFORE, or AFTER command currently pending.

Description

You can use the X command to cancel any COPY, MOVE, BEFORE, or AFTER commands which might be pending. The X command has no effect on the INSERT, DELETE, REPEAT, CHANGE, NEXT, PREVIOUS, SPLIT, and SHIFT commands (either previous or subsequent).

Action

Enter the X command in the line number column of any line to cancel any COPY, MOVE, BEFORE, or AFTER commands that are pending.

Example—Invalid Command

The following sample shows the result of entering an invalid block command. The destination was next to the block and would have moved the block to its own location. Therefore, CA Dataquery scrolled the line containing the invalid destination command (B) to the top of the text area. The block command (MOVE or COPY) must be canceled before CA Dataquery permits any further operations.

```
 =>
DQ206E - THE DESTINATION YOU SPECIFIED FOR THE BLOCK COMMAND IS INVALID
 ------------------------------------------------------------------------DQD10
 DATAQUERY:  EDITOR
 ------------------------------------------------------------------------------
 NAME:
 SAMPLE                            TYPE: QUERY   STATUS:  PRIVATE DESCRIPTION:
 ____________________________________________________
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
 B  WITH SHIP-QTY > 100
 03 RELATED BY ITM-ID TO CAI-RCPTS-REC
 04       */THIS PART OF THE QUERY
 05                                CREATES NEW TEMPORARY RESULTS*/
 06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
 07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
 08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
 09 SORT CAI-RCPTS-REC BY (ITM-ID)
 10 CAI-SLSHST-REC BY (ITM-ID)
 11 CAI-SLSHST-REC BY (ITM-ID)
 12 CAI-SLSHST-REC BY (ITM-ID)
 13 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
 ------------------------------------------------------------------------------
 <PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
 <PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
 <PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Example—Cancel a Command

```
=>
DQ206E - THE DESTINATION YOU SPECIFIED FOR THE BLOCK COMMAND IS INVALID
------------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE
TYPE: QUERY   STATUS:  PRIVATE DESCRIPTION:   SCROLL A LINE TO THE TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
X  WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04       */THIS PART OF THE QUERY
05                                 CREATES NEW TEMPORARY RESULTS*/
06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
09 SORT CAI-RCPTS-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
12 CAI-SLSHST-REC BY (ITM-ID)
13 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Result

```
=>

------------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE                            TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SCROLL A LINE TO THE TOP OF DISPLAY
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
02 WITH SHIP-QTY > 100
03 RELATED BY ITM-ID TO CAI-RCPTS-REC
04       */THIS PART OF THE QUERY
05                                 CREATES NEW TEMPORARY RESULTS*/
06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
07 SET DISCOUNT (5.2) = NET-COST * DISC-PCT
08 SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
09 SORT CAI-RCPTS-REC BY (ITM-ID)
10 CAI-SLSHST-REC BY (ITM-ID)
11 CAI-SLSHST-REC BY (ITM-ID)
12 CAI-SLSHST-REC BY (ITM-ID)
13 PRINT FROM CAI-RCPTS-REC ITM-ID 'ORD-ID'
--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

Chapter 5: Operation Commands

Using Operands

End users can enter a command on the command line at the top of any panel and press Enter to proceed directly to any authorized function or panel. Some CA Dataquery commands require you to enter additional information called *operands* after the command. If the needed operands are not entered, or if invalid operands are entered, CA Dataquery displays a helpful panel that prompts you for valid operands. These panels are documented in Panels (see page 27).

Effect on <PF2> RETURN

Entering an operation command terminates all outstanding CA Dataquery processes, such as directory displays. If <PF2> RETURN is used on a function initiated by a command, CA Dataquery returns to the Main Menu.

Administrative Commands

If you are authorized to use Administrative commands, see the *CA Dataquery User Guide* for administrative command usage and syntax.

Chapter Organization

This chapter provides detailed instructions on CA Dataquery conventional user commands. The commands appear in alphabetic order with the command name listed in the top outside corner of the page. You can abbreviate any command by typing the fewest characters that uniquely identify it to CA Dataquery, as in EXE for EXECUTE.

Summary

The following summarizes valid user commands.

+nnnn

(Where nnnn represents a number) Scrolls forward nnnn pages on a multi-screen panel.

-nnnn

(Where nnnn represents a number) Scrolls backward nnnn pages on a multi-screen panel.

ADMIN

Displays Administrative Menu.

AUTHID

With an SQL authorization ID as an operand, changes the authid for the current session.

BOTTOM

Scrolls to bottom of multi-screen panel.

CREATE

Displays CA Dataquery Editor panel.

DETAIL

Entered on the report output panel, formats report to display the report as it is defined in the query, with detail and totals.

DIRECTORY

Displays Directory Selection panel.

DISPLAY

Displays Key and Columns Display panel for named table.

DRAW

With a table-name operand, creates a simple query that finds all simple columns.

DQL

Changes mode of operation from SQL Mode to DQL Mode.

EDIT

With operand, displays Editor panel containing named text.

EXECUTE

Display online execution panel for query named in command.

EXTRACT

Displays Extract Active Found Set panel to save a set of data found by a query.

FORMAT

Followed by the name of an existing query or dialog, displays first panel for formatting the output created by executing an SQL query.

GUIDE

Displays the first panel for constructing a query with the Guided Query Creation function.

HELP

Displays Help panel topics you can select for additional Help.

KEEP

Displays panel for saving query output.

LIST

With operands, displays directory of queries, terms, dialogs, files, or saved found data sets.

MENU

Displays Main Menu.

MSG

With operands, sends a message to designated user.

NO-DETAIL

Entered on the DQL Mode report output panel, formats report to display query output without detail lines.

NO-TOTALS

Entered on the DQL Mode report output panel, formats report to display query output without totals.

NOWRAP

Issue after WRAP command to return wrapped report lines to multi-screen display.

OFF

Signs CA Dataquery off and returns to monitor.

PDB

Initiates Personal Database Facility.

PFn

Where n is a number from 1 to 12, acts as a numbered PF key.

PROFILE

Displays User Profile panel.

STALL

Displays Execution Statistics panel with expanded statistics and a PF key for viewing tables statistics.

STATS

Displays Execution Statistics panel for current query.

STORE

With unique, new table name operand, creates a personal table from active found set of data in DQL Mode or SQL Mode.

SUBMIT

With query name, displays Batch Execution panel for query named.

SQL

Changes mode of operation from DQL Mode to SQL Mode. To use personal tables, the SQL option of CA Datacom/DB must be installed at your site and you must be authorized by your CA Dataquery Administrator to access the Personal Database Facility.

TIME

Displays current time and date.

TOP

Scrolls to the top of a multi-screen panel.

TOTALS

Entered on the report output panel, formats report to display only total lines and suppresses detail lines.

WHEN-ONLY

Entered on the report output panel, formats report to display only the results of DQL Mode WHEN/DO statements.

WRAP

Moves off-screen lines to current screen of multi-screen display by wrapping each report line before starting the next report line.

+nnnn - Scroll Forward to a Page

Purpose

Use this command to page forward during display of a panel which can be scrolled forward. The number you enter following the plus sign (+) is the number of screens you want to skip.

When to Use It

To scroll forward on a multi-screen panel.

Syntax

Enter	With	When You Want to::
+2	No operand	Skip the next screen of a scrolling display.
+9999	No operand	Skip the next 9,998 screens of a scrolling display.
+nnnn (any number from 1 to 9999)	No operand	Skip the next n screens.

Example

The following example shows a typical use for the +nnnn command. We want to skip three screens and display page 4.

```
 => +3

01/04/2010          DATAQUERY SAMPLE                      PAGE 1
12:51:43                                                          DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01008  C10000      0000002    00029.50          $59.00
          C10001      0000004    00021.00          $84.00
          C10002      0000006    00014.00          $84.00
          C10005      0000002    00066.75         $133.50

  TOTAL ORD-ID 01008
                                                  $360.50

  AVERAGE ITEM PRICE 00032.81

   01009  A60005      0000001    00219.99         $219.99
          A60008      0000002    00179.99         $359.98
          A70000      0000004    00099.99         $399.96
------------------------------- MORE .... -----------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12>RIGHT
```

Result

You now see page 4 of a multi-screen report.

```
 =>

01/04/2010          DATAQUERY SAMPLE                          PAGE 4
12:51:43                                                        DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01011  C50005      0000006    00116.00         $696.00
          C50004      0000004    00048.00         $192.00
          C50003      0000003    00036.00         $108.00
          C50002      0000002    00042.00          $84.00
          C50001      0000002    00049.50          $99.00
          C50000      0000007    00029.00         $203.00
          C40000      0000004    00044.00         $176.00
          C30001      0000006    00009.75          $58.50
          C50006      0000005    00145.00         $725.00
          C60002      0000006    00009.00          $54.00
          C60001      0000002    00013.50          $27.00
          C60000      0000004    00019.00          $76.00
          C50008      0000008    00119.00         $952.00
 ------------------------------ MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

-nnnn - Scroll Backward to a Page

Purpose

Use this command to page backward during display of a multi-page panel. The number you enter following the minus sign (-) is the number of screens you want to skip backwards.

When to Use It

To scroll backward on a multi-screen panel.

Syntax

Enter	With	When You Want to:
-2	No operand	Go back two previous screens.
-9999	No operand	Go back 9,998 previous screens of a scrolling display.
-nnnn	No operand	Skip backward over n previous screens. (Use any number from 1 to 9999.)

Example

The following example shows a typical use for the -nnnn command. We want to skip back two screens and display page 2.

```
 => -2

01/04/2010          DATAQUERY SAMPLE                        PAGE 4
12:51:43                                                          DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01011  C50005      0000006    00116.00         $696.00
          C50004      0000004    00048.00         $192.00
          C50003      0000003    00036.00         $108.00
          C50002      0000002    00042.00          $84.00
          C50001      0000002    00049.50          $99.00
          C50000      0000007    00029.00         $203.00
          C40000      0000004    00044.00         $176.00
          C30001      0000006    00009.75          $58.50
          C50006      0000005    00145.00         $725.00
          C60002      0000006    00009.00          $54.00
          C60001      0000002    00013.50          $27.00
          C60000      0000004    00019.00          $76.00
          C50008      0000008    00119.00         $952.00
 ------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

You now see page 2 of a multi-screen report if you enter -2 on page 4.

```
 =>

01/04/2010          DATAQUERY SAMPLE                          PAGE 2
12:51:43                                                        DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------

  TOTAL ORD-ID 01009
                                                  $979.95

  AVERAGE ITEM PRICE 00166.65

   01010  H20006      0000005    00012.99          $64.95
          H30000      0000004    00124.99         $499.96
          H40000      0000015    00015.99         $239.85
          H20001      0000003    00021.99          $65.97
          H30002      0000006    00069.99         $419.94
          H10002      0000010    00004.99          $49.90
          H20004      0000002    00021.99          $43.98
------------------------------- MORE .... -----------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

ADMIN - Display Administrative Menu

Purpose

The ADMIN command displays the Administrative Menu listing administrative tasks you are authorized to perform. Other administrative commands are available for authorized users. You can get more information about them in the *CA Dataquery Administrator Guide*.

When to Use It

When you want to change your User Profile or perform any other authorized administrative function. See Panels (see page 27) for more information.

Syntax

Enter	With	When You Want to:
ADMIN	No operand	Display the Administrative Menu

Example

```
 => ADMIN

01/04/2010          DATAQUERY SAMPLE                    PAGE 1A
14:15:01                                                        DETAIL

  CUST-ID-KEY  NAME                          PHONE       CRED-IND  ORD-ID
  -----------  ----------------------------  ----------  --------  ------
  I4790I4790   CARPET CLEANERS               2143947734  I          01002
  H4130H4130   WINDOWS AND FLOORING          2145335534  H          01003
  A0030A0030   CANNON TOOLS CO               4045553322  A          01011
  A0030A0030   CANNON TOOLS CO               4045553322  A          01223
```

Result

```
 =>

 ------------------------------------------------------------------------DQKE0
 DATAQUERY:  ADMINISTRATIVE MENU
 ------------------------------------------------------------------------

 Enter DESIRED OPTION NUMBER ===>  __

  1.  PROFILE              - Display and update user profile
```

AUTHID - Change SQL Authorization ID

Purpose

The AUTHID command, followed by an SQL authorization ID, changes your current authorization ID for the current session.

When to Use It

When you want to change your User Profile SQL authorization ID for any reason, such as displaying lists of tables for another ID. See Panels (see page 27) for more information.

If you change your SQL authorization ID with the PROFILE or AUTHID command, it changes only on your User Profile, and it does **not** change your PRIVATE SQL AUTHID which is maintained by the CA Dataquery for CA Datacom Administrator. Therefore, if you create a table in PDB, your permanent SQL authorization ID is attached to the table. However, if you DISPLAY tables or create or execute queries, the SQL authid currently specified on your user profile is used by CA Dataquery for CA Datacom.

Syntax

Enter	With	When You Want to:
AUTHID	SQL-authid	Change authorization IDs.

Example

```
=> AUTHID SYSUSR
CREATION PANEL
--------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  _____________________________
------------------------------------------------------------------------------
NAME:          _______________                  TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   ____________________________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ================================ T O P ===================================
..
..
```

Output

```
=>
DQ353I - THE USER PROFILE WAS UPDATED SUCCESSFULLY.
--------------------------------------------------------------------------DQZ50
DATAQUERY:  MAIN MENU - SQL MODE

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

1.  DIRECTORIES         - Lists of Queries and Tables
2.  CREATE              - Query, Dialog creation
3.  GUIDE               - Structured query creation
4.  PDB                 - List, create and maintain personal tables
5.  ADMINISTRATION      - DATAQUERY system management
6.  HELP                - DISPLAY Help Information
7.  OFF                 - DATAQUERY session termination
```

BOTTOM - Display Last Page

Purpose

The BOTTOM command displays the last screen of a multi-screen display.

When to Use It

To see the last lines of a multi-screen display.

Syntax

Enter	With	When You Want to:
BOTTOM	No operand	Go to the last screen of a multi-screen display.

Example

```
 => BOTTOM
01/04/2010          DATAQUERY SAMPLE                              PAGE 1
12:54:30                                                            DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01008  C10000      0000002    00029.50          $59.00
          C10001      0000004    00021.00          $84.00
          C10002      0000006    00014.00          $84.00
          C10005      0000002    00066.75         $133.50

  TOTAL ORD-ID 01008
                                                  $360.50

  AVERAGE ITEM PRICE 00032.81

   01009  A60005      0000001    00219.99         $219.99
          A60008      0000002    00179.99         $359.98
          A70000      0000004    00099.99         $399.96
 -------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>

01/04/2010          DATAQUERY SAMPLE                          PAGE 14
12:54:30                                                           DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------

* GRAND TOTAL
                                              $29,256.78









 -------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

CREATE - Display Editor Creation Panel

Purpose

The CREATE command displays the CA Dataquery Editor Creation panel for building a query. For information about the Editor panel operation, see Panels (see page 27).

Note: For more information about using the Editor, see the *CA Dataquery User Guide*.

When to Use It

Any time you want to build a query from scratch using the CA Dataquery Editor.

Syntax

Enter	With	When You Want to:
CREATE	No operand	Create a new query.

Example

```
 => CRE

01/04/2010          DATAQUERY SAMPLE                          PAGE 14
12:54:30                                                           DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
```

Result

```
=>
CREATION PANEL
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  ____________________________
-------------------------------------------------------------------------------
NAME:          _______________                  TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   ____________________________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ================================ T O P ====================================
..
..
..
..
..
..
..
..
..
..
   ========================  B O T T O M  ================================
-------------------------------------------------------------------------------
<PF1> HELP          <PF2>  RETURN     <PF3>  DISPLAY COLUMNS <PF4> DISPLAY KEYS
<PF5> DISPLAY ALL   <PF6>  LIST TABLES <PF7> BACKWARD       <PF8>  FORWARD
<PF9> TEMPLATE      <PF10> VALIDATE   <PF11> RIGHT/LEFT     <PF12> PROCESS MODE
```

DETAIL - Display Detail Report

Purpose

Use this command to show the online report output as defined, with detail lines and totals, if any.

When to Use It

During display of any report.

Syntax

Enter	With	When You Want to:
DETAIL	No operand	Change the output display to show complete report.

Example

```
 => DETAIL

01/04/2010         DATAQUERY SAMPLE           PAGE 1
16:53:20                                                          TOTALS-ONLY

  ACTIVITY  YEAR TO DATE
   MONTH       SALES
  --------  ------------
  02             452.00
  03            2000.00
  04          341924.50
  06            5000.00
  08               0.00
  09               0.00
  11         1030318.34

* GRAND TOTAL
             1379694.84

 ------------------------------- LAST PAGE ----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>

01/04/2010         DATAQUERY SAMPLE          PAGE 1A
16:53:20                                                                DETAIL

  CUSTOMER                                                 PHONE       SALESMAN
    NAME                           CITY            STATE  NUMBER        ID
  ------------------------------  --------------  -----  ----------  --------
  UNITED ATLANTIC SHARES          CHARLOTTE       NC     7045554852  52733

  TOTAL ACTIVITY/MONTH 02


  AVERAGE SALE   0000452.00

  INTERNATIONAL BANK CORP.        NEW YORK        NY     2125559889  52733

  TOTAL ACTIVITY/MONTH 03

 -------------------------------- MORE .... --------------------------------- =>
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

DIRECTORY - Display Directory Selection Panel

Purpose

Entering DIRECTORY or DIR on the command line and pressing Enter displays the Directory Selection panel from which you can select the type of directory you want to display. An alternative command is LIST followed by a valid operand, which you can use to immediately display any directory. LIST command usage is discussed later in this chapter.

When to Use It

If you want to display a list of queries, dialogs, terms, and so forth.

Syntax

Enter	With	When You Want to:
DIRECTORY	No operand	Display the Directory Selection panel.

Example

```
=> DIRECTORY
CREATION PANEL
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  _____________________________
-------------------------------------------------------------------------------
NAME:          _______________                  TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   ____________________________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ================================ T O P ====================================
..
```

Result

```
=>
Mark the desired Directory and press Enter.
-------------------------------------------------------------------------DQA00
DATAQUERY:  DIRECTORY SELECTION
-------------------------------------------------------------------------------

  _ Queries and Terms      - List all queries and terms accessible to you.
  _ Queries Only           - List queries accessible to you.
  _ Terms Only             - List terms accessible to you.
  _ Dialogs                - List Dialogs accessible to you.
  _ Public Queries         - List public queries.
  _ Queries and Terms      - List queries and terms created by user:
                               _______________________________

  _ Tables                 - List the tables accessible to you.
                             Start Table Directory with Letter:  _

  _ Saved Sets             - List the saved sets.



-------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

DISPLAY - Display Keys and Columns

Purpose

Entering the DISPLAY command followed by a database table name you are authorized to access displays the Key and Columns Display panel for that table. See Panels (see page 27) for details about using the Key and Columns Display panel.

When to Use It

If you need to know about the keys and columns in any production status database table, or see the definition of any key or column.

Syntax

Enter	With	When You Want to:
DISPLAY	table-name	See a list and description of each key and column for the production status table you name.
DISPLAY	No operand	Display the Display Command prompt panel to see command syntax or see a list of tables you can access.

Example

```
=> DISPLAY
CREATION PANEL
-------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  ____________________________
-------------------------------------------------------------------------
NAME:          _______________                  TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   ____________________________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ================================
..
```

Result

```
=>

-------------------------------------------------------------------------------DQB00
DATAQUERY:  DISPLAY COMMAND

DISPLAY _____________________________
                  (1)
--------------------------------------------------------------------------------

 (1) Enter the name of a table and press <Enter>
     to display its keys and columns.

     Press <PF3> for a list of tables.









--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  LIST TABLES
```

DRAW - Create a Simple Query

Purpose

Entering the DRAW command creates a simple query that finds all the simple columns for the table named in the operand. Additional operands differ according to whether you are in DQL Mode or SQL Mode.

When to Use It

Use DRAW to automatically create a query or, on the Editor panel, to quickly copy all column names at the end of any query text. If on the Editor panel and part of a statement already exists, simply delete the extra statement.

DQL Mode Syntax

Enter	With	When You Want to Create a Query That:
DRAW	table-name	Finds data in all simple columns.
DRAW	table-name ALL	Find data in all simple and compound columns.

SQL Mode Syntax

Enter	With	When You Want to Create a Query That:
DRAW	table-name	Finds data in all simple columns.
DRAW	table-name INSERT	Contains an INSERT clause instead of SELECT.
DRAW	table-name UPDATE	Contains an UPDATE clause instead of SELECT.
DRAW	table-name DELETE	Contains a DELETE clause instead of SELECT.

See the chapter on SQL query syntax for information about using SQL statements (see page 515).

Example

```
=> draw cai-accts-rec
Place the cursor on the desired name and press the appropriate PFkey
------------------------------------------------------------------------------DQA20
DATAQUERY:  DIRECTORY OF TABLES             DATADICTIONARY BASE ID: 189
------------------------------------------------------------------------------
      TABLE  NAME                |STATUS|     DESCRIPTION
------------------------------------------------------------------------------
  ADDRESS                        |  0   |
  CAI-ACCTS-REC                  |  0   | ORDER ENTRY DEMO ACCTS RECORD
  CAI-CUST-REC                   |  0   | CUSTOMERS RECORD
  CAI-DETAIL-REC                 |  0   | DETAILS RECORD
  CAI-ITEMS-REC                  |  0   | ITEMS RECORD
```

Result

```
=>

------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:           _______________                 TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
01 FIND      CAI-ACCTS-REC  ROWS
02 PRINT      FROM  CAI-ACCTS-REC  ORD-ID  BILL-DT  BILL-YR  BILL-MO
03 BILL-DAY  ORD-AMT  FRT-AMT  DISC-AMT  CUST-ID  IND-CD  CUST-NO
.. ========================  B O T T O M  ================================




--------------------------------------------------------------------------------
<PF1>  HELP       <PF2>  RETURN       <PF3>  EXECUTE      <PF4>  SAVE
<PF5>  DIALOG DEF <PF6>  DELETE       <PF7>  BACKWARD     <PF8>  FORWARD
<PF9>  UPDATE     <PF10> VALIDATE     <PF11> RIGHT/LEFT   <PF12> CREATE MODE
```

DQL - Change to DQL Mode

Purpose

Enter the DQL command to activate DQL Mode.

When to Use It

When you are operating in SQL Mode and want to change to DQL Mode.

Syntax

Enter	With	When You Want to:
DQL	No operand	Change from SQL Mode to DQL Mode.

Example

```
=> DQL
PLACE THE CURSOR BY THE DESIRED NAME AND PRESS THE APPROPRIATE PFKEY
-------------------------------------------------------------------------DQT10
DATAQUERY:  DIRECTORY OF TABLES             DATADICTIONARY BASE ID: 002
                                            START WITH:   _________________
-------------------------------------------------------------------------------
      TABLE NAME                    | DESCRIPTION
-------------------------------------------------------------------------------
  ORG                               | ORGANIZATION INFORMATION
  STAFF                             | STAFF DATA
  PERSONNEL                         | EMPLOYEE INFORMATION
  RETIREMENT                        | RETIREMENT PLANNING
```

Result

```
=>
-------------------------------------------------------------------------DQZ60
DATAQUERY:  MAIN MENU - DQL MODE

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
 2.  CREATE              - Query, Dialog or Term creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination
```

EDIT - Display a Member

Purpose

The EDIT command allows you to edit any private member you own or copy and edit any member you can access, if you give the copied version a new name. You become the new member's author. You can enter the command with a valid existing member name and display the CA Dataquery Editor with the current version of the member. Or, you can enter EDIT * to edit the active query. If you enter EDIT without an operand and press Enter, CA Dataquery presents the Edit Command prompt panel. From that panel, you can see a list of public members that you can access. See Panels (see page 27) for details on using the Edit Command prompt panel or the Directory of Queries and Terms.

When to Use It

When you want to copy and change an existing member or make changes to one of your private members.

Syntax

Enter	With	When You Want to:
EDIT	*	Display the active query or dialog on the Editor panel. If you do not own it, you can change it and give it a new name to make your own version.
EDIT	member-name	View an existing member and copy it by giving it a new name or, with authorization, update an existing query with changed text.
EDIT	No operand	Display the Edit command prompt panel to determine command syntax or display a directory.

Example

```
=> EDIT

--------------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
--------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE               - Display and update user profile
```

DQL Result

```
 =>
DQ022E - THERE IS NO ACTIVE QUERY
 --------------------------------------------------------------------------------DQD00
 DATAQUERY:  EDIT COMMAND

 EDIT ACTIVE-QUERY
           (1)
 --------------------------------------------------------------------------------

  (1) Enter 'ACTIVE-QUERY' to display query currently in use
      or enter the name of any query, term, or dialog.
      Press <Enter> to display it for editing.

      Press <PF3> for a list of queries.









 --------------------------------------------------------------------------------
 <PF1> HELP        <PF2> RETURN       <PF3> LIST QUERIES
```

SQL Result

```
=>
DQ022E - THERE IS NO ACTIVE QUERY
------------------------------------------------------------------------------DQD00
DATAQUERY:  EDIT COMMAND

EDIT ACTIVE-QUERY
           (1)
--------------------------------------------------------------------------------

 (1) Enter 'ACTIVE-QUERY' to display query currently in use
     or enter the name of any query, term, or dialog.
     Press <Enter> to display it for editing.

     Press <PF3> for a list of queries.









--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> LIST QUERIES
```

EXECUTE - Begin Online Execution

Purpose

Entering the EXECUTE command and a query name on the command line and pressing Enter displays the online execution panel containing the query name. You can use that panel to execute the query and view the results. If you enter the EXECUTE command without an operand and no query is currently active, CA Dataquery displays the online execution panel with no entry in the Execute Query Named field. If you have accessed a query during this CA Dataquery session, and you enter EXECUTE *, CA Dataquery displays the online execution panel with ACTIVE-QUERY in the Execute Query Named field, meaning the last query you accessed can be executed.

You can type over any entry in the Execute Query Named field. See Panels (see page 27) for detailed information about the online execution panel.

When to Use It

To tell CA Dataquery to execute a query immediately online. With an active found set and query, use EXECUTE if you want to re-execute with different execution specifications, or from a different execution step, as you may want to do after editing a private query to save processing time.

Syntax

Enter	With	When You Want to:
EXECUTE	*	Display online execution panel for the active query, if any, so you can re-execute the query from a different execution step.
EXECUTE	query name	Display online execution panel containing operand query name.
EXECUTE	No operand	If no active query, display online execution panel containing no query name. If a query is currently active, display online execution panel for the active query.

Example

```
=> EXECUTE

------------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
-------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE              - Display and update user profile
```

DQL Result

```
=>
 Enter the desired options and press PF3 to execute
------------------------------------------------------------------------------DQE10
DATAQUERY:  ONLINE EXECUTION
-------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  EXECUTE STEP                 The first query step to execute
    _  SELECTION               - Read and collect the data
    _  COMPUTATION             - Perform the user defined calculations
    _  SORTING                 - Order the collected data
    _  REPORTING               - Produce the report

  REPORT FORMAT                The report format
    _  COLUMNAR                - Show the data arranged one row per line
    _  LIST                    - Show the data arranged one row per page

  REPORT DESTINATION           The destination for the report
    _  VIDEO TERMINAL          - Produce the report on the terminal
    _  NETWORK PRINTER ____    - Produce the report on a network printer
    _  SYSTEM PRINTER          - Produce the report on the system printer
-------------------------------------------------------------------------------
<PF1> HELP       <PF2>  RETURN       <PF3> EXECUTE     <PF4> TOTALING OPTIONS
```

SQL Result

```
=>
 Enter the desired options and press PF3 to execute
------------------------------------------------------------------------DQE60
DATAQUERY:  ONLINE EXECUTION SQL QUERY
--------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  EXECUTE STEP                 The first query step to execute
   _  SELECTION                - Read and collect the data
   _  REPORTING                - Produce the report

  REPORT DESTINATION           The destination for the report
   _  VIDEO TERMINAL           - Produce the report on the terminal
   _  NETWORK PRINTER ____     - Produce the report on a network printer
   _  SYSTEM PRINTER           - Produce the report on the system printer









--------------------------------------------------------------------------------
<PF1> HELP       <PF2> RETURN       <PF3> EXECUTE       <PF4> FORMAT REPORT
```

EXTRACT - Extract Found Data

Purpose

The DQL Mode EXTRACT command creates a data table of the found set produced by executing a query. The data table can be used by other CA products such as CA Ideal, or by programs written at your site. The data table created cannot be used by CA Dataquery. Since the data base table is subject to database maintenance, the contents of the extracted data table may become outdated.

The size of an EXTRACT data set can be limited by the CA Dataquery Administrator. The size of the SET statement result can be limited in the System Option Table, causing the result to be zero (0). See the CA Dataquery Administrator about EXTRACT limits and see the *CA Dataquery User Guide* for additional information about sets and EXTRACT.

When to Use It

When you want to save the results of a query or dialog so it can be accessed by other programs.

Syntax

Enter	With	When You Want to:
EXTRACT	No operand	Display the Extract Active Found Set panel to save a found set produced by executing a query.

Example

```
 => EXTRACT

01/04/2010          DATAQUERY SAMPLE                    PAGE 14
13:01:47                                                        DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01008  C10000      0000002    00029.50          $59.00
          C10001      0000004    00021.00          $84.00
          C10002      0000006    00014.00          $84.00
          C10005      0000002    00066.75         $133.50
```

Result

```
=>
Enter set attributes and press PF3 to extract
-------------------------------------------------------------------------DQA90
DATAQUERY:  EXTRACT ACTIVE FOUND SET

NAME: ________    PASSWORD: _________   DESCRIPTION: _______________

-------------------------------------------------------------------------------









-------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN         <PF3>  SAVE
```

FORMAT - Change SQL Format

Purpose

The FORMAT command, issued in SQL Mode, displays the first of a series of panels for viewing or defining a report definition for the output of a specific SQL query.

When to Use It

When you want to see the specifications for an SQL query report or when you want to change the specifications for the current execution.

Syntax

Enter	With	When You Want to:
FORMAT	query-name	Change or view report format definition.

Example

```
 => FORMAT
 DQ471I - ROWS FOUND: 000034
01/15/2010       DATAQUERY SAMPLE             PAGE 1
15:01:06                    TO PRODUCE REPORT AAA                      DETAIL


     ID  EMP_NAME               DEPT  JOB    NUMYEARS     SALARY       COMM
  ------  --------------------  ------  -----  --------  ---------  ---------
  00010   SANDERS               00020   MGR       00009   17530.00          -
  00020   PERNAL                00020   SALES     00008   18171.25   00612.45
  00030   MARENGHI              00038   MGR       00005   17506.75          -
  00040   O'BRIEN               00038   SALES     00006   18006.00   00846.55
  00050   HANES                 00015   MGR       00010   20659.80          -
  00060   QUIGLEY               00038   SALES         -   16808.30   00650.25
  00070   ROTHMAN               00015   SALES     00007   16502.83   01152.00
  00080   JAMES                 00020   CLERK         -   13504.60   00128.20
  00090   KOONITZ               00042   SALES     00006   18001.75   01386.70
  00100   PLOTZ                 00042   MGR       00007   18352.80          -
  00110   NGAN                  00015   CLERK     00005   12508.20   00206.60
  00120   NAUGHTON              00038   CLERK         -   12954.75   00180.00
 ------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Output

```
=>

-------------------------------------------------------------------------------DQF30
DATAQUERY:  REPORT FORMAT - REPORT TITLE      QUERY NAME:    SQLREP1
-------------------------------------------------------------------------------

ENTER UP TO TWO LINES FOR A
REPORT TITLE:      => TO PRODUCE REPORT AAA
                   =>

SELECT THE DESIRED REPORT FORMAT:    X  COLUMNAR FORMAT: ONE ROW PER LINE.
                                     _  LIST FORMAT: ONE ROW PER PAGE.









-------------------------------------------------------------------------------
<PF1> HELP       <PF2> RETURN       <PF3> DELETE FORMAT      <PF4> CONTINUE
```

GUIDE - Access Guided Query

Purpose

The GUIDE command invokes the Guided Query Creation function that lets you build a query by responding to panels that CA Dataquery presents. See the *CA Dataquery User Guide* for an introduction to this function and to query concepts. Instructions for using the Guided Query panels appear in Panels (see page 27). DQL Mode and SQL Mode present different panels for Guided Query.

When to Use It

When you want to build a new query and you want assistance from the CA Dataquery system.

Syntax

Enter	With	When You Want to:
GUIDE	No operand	Begin creating a query with the Guided Query Creation function.

Example

```
=> guide
CREATION PANEL
--------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR           CURRENT TABLE:  ____________________________
--------------------------------------------------------------------------------
NAME:          _______________                  TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   ____________________________________________________________
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ================================ T O P ===================================
..
```

Result

```
=>

--------------------------------------------------------------------------DQH10
DATAQUERY:   QUERY IDENTIFICATION
--------------------------------------------------------------------------------

Enter a unique name for the new query ==>   _______________

Select the accessibility level for your query by placing any character
next to your choice.

   _   PUBLIC
  X    PRIVATE

Enter a description for the new query
  ==> ____________________________________________________________




--------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN        <PF3> NOT USED      <PF4>  CONTINUE
```

HELP - View Help Topics

Purpose

You can enter the HELP command and press Enter during display of any CA Dataquery panel or message to display a list of topics for which you can get more HELP. You can position the cursor on a topic you want to know more about and press the appropriate PF key.

When to Use It

When you need an explanation of a particular topic related to CA Dataquery.

Syntax

Enter	With	When You Want to:
HELP	No operand	Know more about a concept, function, or site-specific symbol.

Example

```
=> help
-------------------------------------------------------------------------DQH10
DATAQUERY:   QUERY IDENTIFICATION
--------------------------------------------------------------------------------

ENTER A UNIQUE NAME FOR THE NEW QUERY ==>   _______________

SELECT THE ACCESSIBILITY LEVEL FOR YOUR QUERY BY PLACING ANY CHARACTER
NEXT TO YOUR CHOICE.
--------------------------------------------------------------------------------
<PF1> HELP         <PF2> RETURN        <PF3> NOT USED      <PF4>  CONTINUE
```

Result

```
=>
Place the cursor on the desired TOPIC and press the appropriate PFkey
------------------------------------------------------------------------DQM10
DATAQUERY:  LIST OF HELP TOPICS
-------------------------------------------------------------------------------
     DATAQUERY TOPIC NAME        D     DESCRIPTION
-------------------------------------------------------------------------------
  Accessing Repeating Fields     D   Using repeating fields in queries.
  Batch DATAQUERY Execution      D   Submitting a query for batch execution.
  Command Conversion Table       D   3.1 commands with 9.0 commands/procedure
  Command Descriptions           D   What commands do.
  Command Syntax                 D   How commands are built.
  Database Terms                     Database terminology used in DATAQUERY
  DATACOM/DB and DATAQUERY       D   How DATAQUERY uses the database.
  DATADICTIONARY                 D   How DATAQUERY uses the dictionary.
  DATAREPORTER                   D   Description and sample program.
  DIALOG Creation                D   How to create a dialog.
  DIALOG Execution               D   Using queries with variable values.
  Easy Way to Use DATAQUERY      D   Suggestions for the new user.
  Efficient Table Searches       D   Determining the most efficient retrieval
  Error Messages                 D   Types and content of error messages.
-------------------------------------------------------------------------------
<PF1> HELP        <PF2>  RETURN         <PF3> NOT USED    <PF4> NOT USED
<PF5> NOT USED    <PF6>  DISPLAY TEXT   <PF7> BACKWARD    <PF8> FORWARD
```

KEEP - Save Found Set Location

Purpose

The DQL Mode KEEP command stores the active found set in the Saved Set library. An active found set is a collection of internal database directions to the data found with your most recent query execution during the current session. You may want to KEEP a found set for a query that requires a lengthy search for data. Saving the set allows you to minimize processing time the next time you execute the query while still accessing current data. We advise including the query name as part of the saved set description. (See the *CA Dataquery User Guide* for more information.) Only one KEEP command per active found set is permitted.

The data itself is not saved with the KEEP command. Therefore, if the data is updated after the set is saved, and the set is then used, the results can change. Other than the STORE command, which creates a personal table of query output, the only way to store current results of query execution so that it is not subject to database maintenance is by using the EXTRACT command online or the Export function during batch execution.

You may want to read about the EXTRACT command which allows other software products to access data found by CA Dataquery. See EXTRACT - Extract Found Data (see page 371) for more information about EXTRACT.

When to Use It

After executing a query that requires a lengthy search for data, you may want to change certain report characteristics and re-execute the query at a later time. KEEP allows you to do this without doing FIND processing again.

Syntax

Enter	With	When You Want to:
KEEP	No operand	Display the Save Active Found Set panel for saving the active found set.

Example

```
 => keep

01/04/2010          DATAQUERY SAMPLE                      PAGE 1
14:17:22                                                        DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01008  C10000      0000002    00029.50          $59.00
          C10001      0000004    00021.00          $84.00
          C10002      0000006    00014.00          $84.00
```

Result

```
=>
Enter the sets attributes and press PF3 to save
-------------------------------------------------------------------------DQA80
DATAQUERY:  SAVE ACTIVE FOUND SET

NAME: ________  DESCRIPTION: ____________________

--------------------------------------------------------------------------------









--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN         <PF3>  SAVE
```

LIST - Display a Directory

Purpose

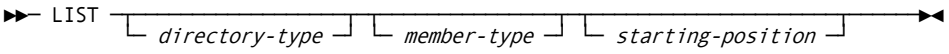
Entering LIST on the command line and pressing Enter displays the List Command prompt panel that prompts you for the directory type, library type (if a query or term), and a list starting position. You can enter LIST followed by operands and immediately display a directory of queries, terms, dialogs, tables, or saved sets. An alternative command is DIRECTORY which you can use to display a list of directories for selection. DIRECTORY ;l command usage was discussed previously in this chapter.

When to Use It

If you want to display a list of queries, dialogs, terms, tables, or saved sets.

Syntax

The full format for the LIST permits three operands. No operand is required because entering LIST alone displays a prompt panel.



General Lists

Enter	With	When You Want to:
LIST	No operand	Display the List Command prompt panel.
LIST	all	Display the main directory listing all public members for your group assignment (if any) and your own private members.

List Queries

Enter	With	When You Want to:
LIST	QUERIES PUBLIC start-with	Display the Directory of Queries listing all public queries starting with the letters designated, if any.
LIST	QUERIES PRIVATE start-with	Display the Directory of Queries listing your private queries starting with the letters designated, if any.
LIST	QUERIES BOTH start-with	Display the Directory of Queries listing your private queries and all public queries starting with the letters shown, if any.
LIST	QUERIES user-id start-with	Display the Directory of Queries listing another user's public queries starting with the letters shown, if any.

List Terms (DQL Mode only)

Enter	With	When You Want to:
LIST	TERMS PUBLIC start-with	Display the Directory of Terms listing all public terms starting with the letters shown, if any.
LIST	TERMS PRIVATE start-with	Display the Directory of Terms listing your private terms starting with the letters shown, if any.
LIST	TERMS BOTH start-with	Display the Directory of Terms listing your private terms and all public terms starting with the letters shown, if any.
LIST	TERMS user-id start-with	Display the Directory of Terms listing another user's private terms starting with the letters shown, if any.

List Dialogs

Enter	With	When You Want to:
LIST	DIALOGS PUBLIC start-with	Display the Directory of Dialogs listing all public dialogs starting with the letters shown, if any.
LIST	DIALOGS PRIVATE start-with	Display the Directory of Terms listing your private dialogs starting with the letters shown, if any.
LIST	DIALOGS BOTH start-with	Display the Directory of Dialogs listing your private dialogs and all public dialogs starting with the letters shown, if any.
LIST	DIALOGS user-id start-with	Display the Directory of Dialogs listing another user's private dialogs starting with the letters shown, if any.

List Tables

Enter	With	When You Want to:
LIST	TABLES start-with	Display the Directory of Tables panel listing all production status tables you can access, beginning with the starting characters you entered, if any.

List Sets (DQL Mode only)

Enter	With	When You Want to:
LIST	SETS	Display the Directory of Sets panel listing all saved sets you can access.

Example

```
=> list
Enter the sets attributes and press PF3 to save
------------------------------------------------------------------------DQA80
DATAQUERY:  SAVE ACTIVE FOUND SET

NAME: ________  DESCRIPTION: ____________________

--------------------------------------------------------------------------------
```

Result

Entering LIST without an operand displays the following prompt panel.

```
=>

------------------------------------------------------------------------DQA10
DATAQUERY:  LIST COMMAND

LIST ALL     BOTH                            A
     (1)       (2)                          (3)
-------------------------------------------------------------------------------
(1) Select a        (2) Select the type       (3) Enter 1 to 32 letters of the
    Directory Type:     of Query or Term          alphabet to indicate the
                        Library:                  beginning of the display.
     ALL (Queries,                                Not valid for LIST SETS.
       terms, and       PRIVATE
       dialogs)         PUBLIC
     QUERIES            BOTH (Public and private)
     TERMS              An Operator (Enter ID)
     DIALOGS
     TABLES
     SETS



-------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

MENU - Display the Main Menu

Purpose

Entering MENU and pressing Enter displays the Main Menu.

When to Use It

You want to list or select the major functions.

Syntax

Enter	With	When You Want to:
MENU	No operand	See the Main Menu.

Example

```
=> menu

-------------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
-------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE              - Display and update user profile
 2.  CONDITIONS           - List create and maintain conditions
 3.  RESTRICTIONS         - List create and maintain restrictions
```

DQL Result

```
=>

-------------------------------------------------------------------------------DQZ60
DATAQUERY:  MAIN MENU - DQL MODE

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
 2.  CREATE              - Query, Dialog or Term creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination
```

SQL Result

```
=>

-------------------------------------------------------------------------------DQZ50
DATAQUERY:  MAIN MENU

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries and Tables
 2.  CREATE              - Query, Dialog creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination
```

MSG - Send a Message

Purpose

Entering MSG, the user ID of another active user, and a short message (to the end of the command line) and then pressing Enter sends the message to the user named.

When to Use It

You want to send a message to another active user without leaving your work station or you want to respond to a message you just received.

Syntax

Enter	With	When You Want to:
MSG	user-id text-of-message	Send a message to another active user.

Example

Following is how a message looks when you send it:

```
=> MSG SMITH PLEASE TRY QUERY NAMED NEWORDERS

--------------------------------------------------------------------------DQZ60
DATAQUERY:  MAIN MENU

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
 2.  CREATE              - Query, Dialog or Term creation
```

Result

Following is how a message looks when it is received:

```
=> MSG FROM FULTON 11:53:31
   PLEASE TRY QUERY NAMED NEWORDERS
--------------------------------------------------------------------------DQZ60
DATAQUERY:  MAIN MENU

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>

 1.  DIRECTORIES         - Lists of Queries, Terms, Tables, and Saved Sets
 2.  CREATE              - Query, Dialog or Term creation
 3.  GUIDE               - Structured query creation
 4.  ADMINISTRATION      - DATAQUERY system management
 5.  HELP                - Display Help Information
 6.  OFF                 - DATAQUERY session termination
```

NO-DETAIL - Suppress Detail Lines

Purpose

Enter NO-DETAIL during online display of query or dialog output to see only total lines of the report. NO-DETAIL suppresses all detail data.

When to Use It

During display of query output, as an alternative to the PF key for this command.

Syntax

Enter	With	When You Want to:
NO-DETAIL	No operand	Suppress detail lines.

Example

```
 => NO-DETAIL
01/08/2010         DATAQUERY SAMPLE                 PAGE 1
10:45:04                                                          DETAIL

     ID  NAME                    DEPT       SALARY  DEPTNAME         MANAGER
 ------  --------------------  ------  -----------  --------------  -------
  00010  SANDERS               00020   0018357.50  MID ATLANTIC      00010
  00020  PERNAL                00020   0001234.56  MID ATLANTIC      00010
  00128  SEPULVEDA             00020   0029100.00  MID ATLANTIC      00010
  00130  WILSON                00020   0016111.00  MID ATLANTIC      00010
  00090  GRAY                  00020   0018199.00  MID ATLANTIC      00010
```

Result

```
 =>
01/19/2010         DATAQUERY SAMPLE           PAGE 1
10:45:04                                                              NO-DETAIL


  TOTAL DEPTNUMB 00020
                                        *0083002.06

  AVERAGE SALARY DEPT 00020  16600.41

  TOTAL DEPTNUMB 00038
                                        *0076262.30

  AVERAGE SALARY DEPT 00038  15252.46

  TOTAL DEPTNUMB 00040
                                        *0097328.60

  AVERAGE SALARY DEPT 00040  16221.43
------------------------------- MORE .... -----------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

NO-TOTALS - Suppress Report Totals

Purpose

Enter NO-TOTALS during online display of query or dialog output to see only detail lines of the report. NO-TOTALS suppresses all defined totals, including control break totals.

When to Use It

During display of query output, as an alternative to the PF key for this command.

Syntax

Enter	With	When You Want to:
NO-TOTALS	No operand	Suppress totals.

Example

```
 => NO-TOTALS

01/08/2010         DATAQUERY SAMPLE          PAGE 1A
16:53:20                                                          DETAIL

  CUSTOMER                                                PHONE       SALESMAN
    NAME                          CITY            STATE  NUMBER         ID
  ------------------------------  --------------  -----  ----------  --------
  UNITED ATLANTIC SHARES          CHARLOTTE       NC     7045554852  52733

  TOTAL ACTIVITY/MONTH 02


  AVERAGE SALE   0000452.00

  INTERNATIONAL BANK CORP.        NEW YORK        NY     2125559889  52733

  TOTAL ACTIVITY/MONTH 03

 ------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>
01/08/2010         DATAQUERY SAMPLE          PAGE 1A
16:53:20                                                       NO-TOTALS

  CUSTOMER                                                PHONE       SALESMAN
    NAME                          CITY            STATE  NUMBER         ID
  ------------------------------  --------------  -----  ----------  --------
  UNITED ATLANTIC SHARES          CHARLOTTE       NC     7045554852  52733
  INTERNATIONAL BANK CORP.        NEW YORK        NY     2125559889  52733
  SUN DIAL CITRUS GROWERS         LOS ANGELES     CA     2135554800  52733
  CANNON TOOLS CO                 ATLANTA         GA     4045553322  34222
  GARY                            DUNCAN          OK     4052559900  13729
  CARPET CLEANERS                 CARROLLTON      TX     2143947734  34222
  WINDOWS AND FLOORING            DALLAS          TX     2145335534  34222
  IMPERIAL BANKCORP               NEW YORK        NY     2125558878  15266
  WEST LIFE INSURANCE             BALTIMORE       MD     3015554002  13729
  CARTERET STEEL CORPORATION      SAN MATEO       CA     4155552002  13729
  SOUTHWEST STATE OIL REFINING    SHERMAN OAKS    CA     8185559003  13729
  HARVESTLAND FOODS INC           HONOLULU        HI     8085551003  13729
 ------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

NOWRAP - Full Width Report

Purpose

Use the NOWRAP command to display a wrapped columnar report at full width. Use the LEFT and RIGHT PF keys to view horizontal pages.

When to Use It

During output display, if the report output is displayed in WRAP format (not extending beyond the current window).

Syntax

Enter	With	When You Want to:
NOWRAP	No operand	Displays full width of report which has been condensed using the WRAP command.

Example

```
 => nowrap

01/08/2010         DATAQUERY SAMPLE           PAGE 1
16:53:20                                                          DETAIL

-------------------------------------------------------------------------
  CUSTOMER/ID  CUSTOMER/NUMBER  CUSTOMER/NAME                  CREDIT/INDICATOR
    PHONE/NUMBER  YEAR TO DATE/SALES  ACTIVITY/DATE  TOTAL/SALES
-------------------------------------------------------------------------

  A0030        0030             CANNON TOOLS CO                A
    4045553322            320823.00   870420          99,823.00

  A0090        0090             INTERNATIONAL BANK CORP        A
    2125559889               275.59   861113          23,275.59

  A0130        0130             SUN DIAL CITRUS GROWERS        A
    2135554800              3061.46   861113          26,061.46

 -------------------------------- MORE .... ---------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>
01/05/2010          DATAQUERY SAMPLE          PAGE 1A
16:53:20                                                                DETAIL

  CUSTOMER  CUSTOMER  CUSTOMER                          CREDIT    PHONE
     ID      NUMBER     NAME                          INDICATOR  NUMBER
  --------  --------  ------------------------------  ---------  ----------
  A0030     0030      CANNON TOOLS CO                 A          4045553322
  A0090     0090      INTERNATIONAL BANK CORP         A          2125559889
  A0130     0130      SUN DIAL CITRUS GROWERS         A          2135554800
  A0150     0150      IMPERIAL BANKCORP               A          2125558878
  A0170     0170      UNITED ATLANTIC SHARES          A          7045554852
  B0190     0190      SOUTHWEST STATE OIL REFINING    A          8185559003
  B0210     0210      WEST LIFE INSURANCE             A          3015554002
  B0230     0230      CHEMICAL MUTUAL                 A          8175551044
  B0250     0250      MICHIGAN LIGHTING INC.          A          6025558306
  B0270     0270      HARTFORD IRON WORKS INC         A          2155552208
  B0290     0290      NATIONAL HARRIS CORPORATION     A          4045553001
  B0310     0310      CARTERET STEEL CORPORATION      A          4155552002
 ------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

OFF - Sign Off

Purpose

Entering the OFF command and pressing Enter signs you off of CA Dataquery.

When to Use It

When you finish the current CA Dataquery session. Your site may have special procedures in effect that will change the results of using the OFF command. Your system may display a screen not documented here. Contact your CA Dataquery Administrator if special signoff procedures are used.

Syntax

Enter	With	When You Want to:
OFF	No operand	Exit CA Dataquery

Example

```
=> OFF

--------------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
--------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE              - Display and update user profile
```

Result

```
 DQ194I - DATAQUERY SIGN/OFF COMPLETED



                     DDDDD        QQQQQ       RRRRRR       YY     YY
                    DDDDDDD     QQQQQQQ      RRRRRRR      YY    YY
                   DD    DD    QQ     QQ     RR    RR      YY    YY
                  DD    DD    QQ      QQ    RR    RR       YY YY
                 DD    DD    QQ      QQ    RRRRRR          YY
                DD    DD    QQ Q QQ     RR    RR          YY
               DDDDDDD     QQQQQQQ     RR     RR         YY
              DDDDDD       QQQQQQ     RR       RR       YY
                                  Q


Version: 14.0            Time:                        Date:


COPYRIGHT (C) 1990-2011 CA.   ALL RIGHTS RESERVED.
```

PDB - Access Personal Database Facility

Purpose

Use the PDB command when you want to display the Directory of Tables (DQT10) panel of the Personal Database Facility to see a list of table names. You must be authorized to use the Facility in order to use the command. (The Personal Database Facility is not available to users of products installed with CA Datacom/AD.)

When to Use It

To create, view, or maintain personal tables.

Syntax

Enter	With	When You Want to:
PDB	No operand	Display the Directory of Tables panel.

Example

```
=> PDB

--------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
--------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE              - Display and update user profile
```

Result

```
=>
PLACE THE CURSOR BY THE DESIRED NAME AND PRESS THE APPROPRIATE PFKEY
-------------------------------------------------------------------------DQT10
DATAQUERY:  DIRECTORY OF TABLES             DATADICTIONARY BASE ID: 002
                                            START WITH:   _________________
-------------------------------------------------------------------------------
      TABLE NAME                    | DESCRIPTION
-------------------------------------------------------------------------------
  ORG                               | ORGANIZATION INFORMATION
  STAFF                             | STAFF DATA
  PERSONNEL                         | EMPLOYEE INFORMATION
  RETIREMENT                        | RETIREMENT PLANNING
  BENEFITS                          | BENEFIT RECORDS
  QUOTAS                            | QUOTA PER SALES REP
  CLIENTS                           | CLIENT NAMES AND ADDRESSES
  BUDGET                            | DEPARTMENTAL BUDGET
  COMPETITORS                       | COMPETITOR PRODUCT INFORMATION
                                    |
                                    |
- LAST PAGE ----------------------------------------------------------------
<PF1> HELP          <PF2> RETURN       <PF3> CREATE         <PF4> DROP
<PF5> DISPLAY       <PF6> INSERT ROWS  <PF7> BACKWARD       <PF8> FORWARD
<PF9> MAINTAIN ROWS
```

PFn - Execute a PF Key Function

Purpose

Entering PF followed by a number (as in PF1) and pressing Enter causes CA Dataquery to perform the PF key function as indicated on the PF key menu of the current panel.

When to Use It

Use a PF key command when there is no corresponding PF key on your keyboard.

Syntax

Enter	With	When You Want to:
PFn	No operand	Perform a PF key function for the current panel.

Example

```
=> pf4
 Enter the desired options and press PF3 to execute
------------------------------------------------------------------------------DQE10
DATAQUERY:  ONLINE EXECUTION
--------------------------------------------------------------------------------
  EXECUTE QUERY NAMED => ACTIVE QUERY

  EXECUTE STEP                 The first query step to execute
   _  SELECTION                - Read and collect the data
   _  COMPUTATION              - Perform the user defined calculations
   _  SORTING                  - Order the collected data
   _  REPORTING                - Produce the report

  REPORT FORMAT                The report format
   _  COLUMNAR                 - Show the data arranged one row per line
   _  LIST                     - Show the data arranged one row per page

  REPORT DESTINATION           The destination for the report
   _  VIDEO TERMINAL           - Produce the report on the terminal
   _  NETWORK PRINTER ____     - Produce the report on a network printer
   _  SYSTEM PRINTER           - Produce the report on the system printer
```

Result

```
=>
Enter the desired options and press the appropriate PFkey
------------------------------------------------------------------------------DQE30
DATAQUERY:  TOTALING OPTIONS
--------------------------------------------------------------------------------

QUERY NAME =>  USER-SAMPLE

TOTALING OPTIONS

  X  DETAIL and TOTALS           - Produce report with detail and totals
  _  DETAIL ONLY                 - Suppress any requested totaling
  _  TOTALS ONLY                 - Produce the report with totals only
  _  TOTALS BY NAME              - Only use the entered control-break name
                                   _________________________________




--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN     <PF3> EXECUTE
```

PROFILE - Display Your User Profile

Purpose

Entering PROFILE and pressing Enter displays the User Profile panel containing CA Dataquery options you can change. See Panels (see page 27) for a complete discussion of your user options.

When to Use It

When you want to view or change your session options.

Syntax

Enter	With	When You Want to:
PROFILE	No operand	Display or change your options on the User Profile panel.

Example

```
=> PROFILE
Enter the desired options and press the appropriate PFkey
-------------------------------------------------------------------------DQE30
DATAQUERY:  TOTALING OPTIONS
-------------------------------------------------------------------------------

QUERY NAME =>  USER-SAMPLE

TOTALING OPTIONS

  X  DETAIL and TOTALS             - Produce report with detail and totals
```

Result

```
=>
Overtype the values to be modified and press PF4 to complete the update
-------------------------------------------------------------------------------DQKL0
DATAQUERY:  USER PROFILE              FOR => BIRDS
-------------------------------------------------------------------------------
    PROFILE ITEM                                    EXPLANATION
-------------------------------------------------------------------------------
DATADICTIONARY DATABASE ID  =>  _____    Five digit DB ID number for D D access
LIST AND DISPLAY ALIASES    =>  ___      YES show aliases, NO suppress aliases
GROUP DISPLAY               =>  ___      YES break out simple cols, NO do not
SUPPRESS DUPLICATE COLUMNS  =>  ___      YES turn on suppression, NO turn off
SUPPRESS PFKEYS ON PRINT    =>  ___      YES turn on suppression, NO turn off
SUPPRESS EXECUTE PANEL      =>  ___      YES turn on suppression, NO turn off
PRIMARY LANGUAGE            =>  __       Two character PRIMARY Language ID
SECONDARY LANGUAGE          =>  __       Two character SECONDARY Language ID
DECIMAL POINT CHARACTER     =>  _        Character to be used for decimal point
QUERY LANGUAGE              =>  ___      Language for queries - SQL or DQL
SQL AUTHORIZATION ID        =>  ________________
ASSOCIATE USER ERRORS       =>  ___      Display validation errors for assoc

-------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN       <PF3> DISP GROUPS   <PF4> UPDATE
<PF5> PRINT OPT   <PF6> NOT USED     <PF7> NOT USED      <PF8> NOT USED
```

SQL - Change to SQL Mode

Purpose

Use the SQL command when you want to leave DQL Mode to use CA Dataquery for CA Datacom in SQL Mode. To use SQL, the SQL option of CA Datacom/DB must be installed at your site and you must be authorized by your CA Dataquery for CA Datacom Administrator to use SQL Mode.

When to Use It

To create, execute, or maintain SQL queries and dialogs.

Syntax

Enter	With	When You Want to:
SQL	No operand	Use functions in SQL Mode.

Example

```
=> sql

-------------------------------------------------------------------------------DQKE0
DATAQUERY:  ADMINISTRATIVE MENU
-------------------------------------------------------------------------------

ENTER DESIRED OPTION NUMBER ===>  __

 1.  PROFILE              - Display and update user profile
```

Result

```
=>

-------------------------------------------------------------------------------DQZ50
DATAQUERY:  MAIN MENU

ENTER THE NUMBER OF THE DESIRED FUNCTION ===>  _

 1.  DIRECTORIES         - Lists of Queries and Tables
 2.  CREATE              - Query, Dialog creation
 3.  GUIDE               - Structured query creation
 4.  PDB                 - List, create and maintain personal tables
 5.  ADMINISTRATION      - DATAQUERY system management
 6.  HELP                - Display Help Information
 7.  OFF                 - DATAQUERY session termination









-------------------------------------------------------------------------------
```

STALL - Produce Additional Statistics

Purpose

Administrators or Support personnel may need certain statistics when managing the system and they may ask you to provide them. The DQL Mode STALL command displays statistics about total joins, valid joins, estimated I/O and estimated rows during display of the FIND Statistics (DQEF0) panel. Pressing the STATS PF key during query output displays the basic FIND statistics. Entering the STALL command next displays the additional statistics and the TABLE STATS PF key.

Entering the STALL command during query output (without pressing the STATS key at all) displays the FIND statistics panel with the additional statistics and with the TABLE STATS key. See (DQEF0) Query Statistics (see page 111) for additional information about the Find Statistics (DQEC0) panel.

When to Use It

Enter the STALL command when requested by the CA Dataquery Administrator or CA Support.

Syntax

Enter	With	When You Want to:
STALL	No operand	See extended statistics for DQL FIND processing.

Example

```
=>
CURRENT DATAQUERY FIND/SELECT STATISTICS.
------------------------------------------------------------------------------DQEF0
DATAQUERY:  FIND STATISTICS                        QUERY NAME:
------------------------------------------------------------------------------

     NUMBER REQUESTED:                             COMPLETION DATE:
     NUMBER FOUND:                                 COMPLETION TIME:

     FIND TERMINATED BECAUSE:

                               OVERALL    OPTIMIZATION     SEARCH
                               -------    ------------    -------
     ELAPSED TIME (SECONDS):
     I/O EVENTS:
     SELFR TOTAL:
     SELNR TOTAL:
     TOTAL BYTES:



------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN
```

Result

```
=>
CURRENT DATAQUERY FIND/SELECT STATISTICS.
------------------------------------------------------------------------------DQEF0
DATAQUERY:  FIND STATISTICS                        QUERY NAME:
------------------------------------------------------------------------------

     NUMBER REQUESTED:                             COMPLETION DATE:
     NUMBER FOUND:                                 COMPLETION TIME:

     FIND TERMINATED BECAUSE:
                               OVERALL    OPTIMIZATION     SEARCH
                               -------    ------------    -------
     ELAPSED TIME (SECONDS):
     I/O EVENTS:
     SELFR TOTAL:
     SELNR TOTAL:
     TOTAL BYTES:
     TOTAL JOINS:
     VALID JOINS:
     ESTIMATED I/O:
     ESTIMATED ROWS:
------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN        <PF3> TABLE STATS
```

STATS - Display Execution Statistics

Purpose

This command displays the execution statistics for the active query or dialog.

When to Use It

Use the STATS command when you want to know how many rows were found, why a query terminated if not due to normal end of search, and system statistics about the search for data.

Note: If the STALL command has been issued during this session, additional statistics appear. See (DQEC0) FIND Statistics (see page 106) for more information.

Syntax (DQL Mode)

Enter	With	When You Want to:
STATS	No operand	Display Find Statistics panel for the active query.

Syntax (SQL Mode)

Enter	With	When You Want to:
STATS	No operand	Display search statistics or table statistics for the active query.

Example

```
=> STATS
------------------------------------------------------------------------------DQD10
DATAQUERY:  EDITOR
--------------------------------------------------------------------------------
NAME:          SAMPLE-QUERY                    TYPE: QUERY   STATUS: PRIVATE
DESCRIPTION:   SHOW EDITOR LINE COMMAND FUNCTIONS
   ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
.. ============================== T O P ===================================
03 FIND 10 CAI-SLSHST-REC
04 WITH SHIP-QTY > 100
05 RELATED BY ITM-ID TO CAI-RCPTS-REC
06 SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
```

Result

```
=>
Current Dataquery FIND/SELECT Statistics.
--------------------------------------------------------------------------DQEF0
DATAQUERY:  FIND STATISTICS                       QUERY NAME: USER-SAMPLE
--------------------------------------------------------------------------------

     NUMBER REQUESTED: ALL                        COMPLETION DATE: 06/04/88
     NUMBER FOUND:         97                     COMPLETION TIME: 14:25:46

     FIND TERMINATED BECAUSE:  NORMAL END OF SEARCH WAS REACHED

                               OVERALL    OPTIMIZATION     SEARCH
                               -------    ------------    -------
     ELAPSED TIME (SECONDS):         2               0          0
     I/O EVENTS:                     2               0          2
     SELFR TOTAL:                    2               1          1
     SELNR TOTAL:                   97                         97
     TOTAL BYTES:                  351



--------------------------------------------------------------------------------
<PF1> HELP        <PF2> RETURN
```

STORE - Create a Personal Table from Output

Purpose

Use the STORE command to automatically create a personal table containing the output data of a query. CA Dataquery creates a query containing a CREATE TABLE command that becomes the active query. (The Personal Database Facility is not available to users of products installed with CA Datacom/AD.)

When to Use It

During display of the output of a query in either mode.

Syntax

Enter	With	When You Want to:
STORE	table-name	Create a Personal Table containing query output.

The table name should begin with an alphabetic character. You can use letters, numbers, and underscores (_) as part of the name. The maximum name length is 32 characters.

After creation, Datadictionary assigns the table an entity-name and that name appears on directories. Where duplicate names occur, Datadictionary may modify a name to make it unique. Also, any table you create is stored under your permanent SQL authorization ID, maintained by your CA Dataquery Administrator.

Example

```
 => STORE STAFF2
 DQ471I - ROWS FOUND: 000037
01/03/2010         DATAQUERY SAMPLE  PAGE 1
14:12:25                    DOCUMENT-STORE-BIRDS                    DETAIL

        ID  NAME           DEPT  JOB       YEARS       SALARY         COMM
  --------  ---------  --------  -----  --------  -----------  -----------
    00010   SANDERS      00020   MGR      00007     18357.50     00000.00
    00020   PERNAL       00020   SALES    00008     18171.25     00612.45
    00030   MARENGHI     00038   MGR      00005     17506.75     00000.00
    00040   O'BRIEN      00038   SALES    00006     18006.00     00846.55
    00050   HANES        00015   MGR      00010     20659.80     00000.00
    00060   QUIGLEY      00038   SALES    00000     16808.30     00650.25
    00070   ROTHMAN      00015   SALES    00007     16502.83     01152.00
    00080   JAMES        00020   CLERK    00000     13504.60     00128.20
    00090   KOONITZ      00042   SALES    00006     18001.75     01386.70
    00100   PLOTZTZ      00042   MGR      00007     18352.80     00000.00
    00110   NGAN         00015   CLERK    00005     12508.20     00206.60
    00120   NAUGHTON     00038   CLERK    00000     12954.75     00180.00
 -------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
=>
DQ154I - THE TABLE WAS SAVED ALONG WITH THE ROWS SELECTED BY YOUR QUERY
------------------------------------------------------------------------------DQTB0
DATAQUERY:  STORE COMMAND

STORE  STAFF3
          (1)
--------------------------------------------------------------------------------

 (1) Enter a name for the table to be created which is
     unique for your private SQL Authorization ID.
     Press <ENTER> to create the table.









--------------------------------------------------------------------------------
<PF1> HELP         <PF2>  RETURN
```

SUBMIT - Begin Batch Execution

Purpose

Entering the SUBMIT command and a query name on the command line and pressing Enter displays the batch execution panel containing the query name. You can use that panel to submit the query for offline (batch) execution. If you enter the SUBMIT command without an operand and no query is currently active, CA Dataquery displays the batch execution panel with no entry in the Execute Query Named field. If you have accessed a query during this CA Dataquery session, and you enter SUBMIT *, CA Dataquery displays the batch execution panel with ACTIVE-QUERY in the Execute Query Named field, meaning the last query you accessed can be executed.

You can type over any entry in the Execute Query Named field. See Panels (see page 27) for detailed information about the Batch Execution panel.

When to Use It

To submit a query, whether active or not, for batch execution.

Syntax

Enter	With	When You Want to:
SUBMIT	*	Display batch execution panel for the active query, if any, so you can re-execute the query with different specifications.
SUBMIT	query name	Display batch execution panel containing operand query name.
SUBMIT	No operand	If no active query, display batch execution panel containing no query name. If a query is currently active, display batch execution panel for the active query.

Example

```
=> SUBMIT
Current Dataquery FIND/SELECT Statistics.
--------------------------------------------------------------------------DQEB0
DATAQUERY:  FIND STATISTICS                      QUERY NAME: USER-SAMPLE
--------------------------------------------------------------------------------

     NUMBER REQUESTED: ALL                       COMPLETION DATE: 06/04/88
     NUMBER FOUND:         97                    COMPLETION TIME: 14:25:46
```

Result

```
=>
--------------------------------------------------------------------------DQEN0
DATAQUERY:  BATCH EXECUTION
--------------------------------------------------------------------------------
Enter name of query to submit:          ACTIVE-QUERY
Select the type of execution:           X  Immediate
                                        _  Defer execution until time __ : __
Enter the name of the JCL member to use: $$DQJCL
Select the report type:                 X  Detail and totals
                                        _  Detail only (no totals)
                                        _  Totals only (summary)
                                        _  Suppress report
Enter the name for an output set to export print data to a sequential file
or leave blank for no export:           ________________
Select the export output type:          _  Detail
                                        _  Totals
Select the output file device type:     _  Tape
                                        _  Disk


--------------------------------------------------------------------------------
<PF1> HELP      <PF2>  RETURN      <PF3>  SUBMIT     PORT
```

TIME - Display Time and Date

Purpose

Enter TIME on the command line and press Enter to display the current system date and time.

When to Use It

You want to know the exact date or time.

Syntax

Enter	With	When You Want to:
TIME	No operand	Display system time and date on the message line.

Example

```
=>
-------------------------------------------------------------------------------DQEN0
DATAQUERY:  BATCH EXECUTION
--------------------------------------------------------------------------------
Enter name of query to submit:          ACTIVE-QUERY
Select the type of execution:           X  Immediate
                                        _  Defer execution until time __ : __
Enter the name of the JCL member to use: $$DQJCL
```

Result

```
=>
CURRENT TIME 14:38:17, DATE 01/02/2010
-------------------------------------------------------------------------------DQEN0
DATAQUERY:  BATCH EXECUTION
--------------------------------------------------------------------------------
Enter name of query to submit:          ACTIVE-QUERY
Select the type of execution:           X  Immediate
                                        _  Defer execution until time __ : __
Enter the name of the JCL member to use: $$DQJCL
Select the report type:                 X  Detail and totals
```

TOP - Display First Screen

Purpose

The TOP command displays the first screen of a multi-screen display.

When to Use It

To see the first lines of a multi-screen display.

Syntax

Enter	With	When You Want to:
TOP	No operand	Go to the first screen of a multi-screen display.

Example

```
 => TOP

01/03/2010          DATAQUERY SAMPLE                        PAGE 14
14:38:51                                                          DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------

* GRAND TOTAL
                                              $29,256.78









 ------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>
01/08/2010          DATAQUERY SAMPLE                          PAGE 1
14:38:51                                                            DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE        AMT-SHIP
  ------  ----------  --------  ----------  --------------
   01008  C10000      0000002    00029.50          $59.00
          C10001      0000004    00021.00          $84.00
          C10002      0000006    00014.00          $84.00
          C10005      0000002    00066.75         $133.50

  TOTAL ORD-ID 01008
                                                   $360.50

  AVERAGE ITEM PRICE 00032.81

   01009  A60005      0000001    00219.99         $219.99
          A60008      0000002    00179.99         $359.98
          A70000      0000004    00099.99         $399.96
 ------------------------------- MORE .... ----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

TOTALS - Display DQL Mode PRINT Totals Only

Purpose

Use the TOTALS command to display only the PRINT command column totals defined in the query or dialog. TOTALS suppresses detail lines and WHEN/DO results.

When to Use It

Use this command in DQL Mode to display only PRINT statement control break and end of report totals.

Syntax

Enter	With	When You Want to:
TOTALS	No operand	Suppress detail lines and WHEN/DO results in an output display.

Example

```
 => TOTALS

01/03/2010         DATAQUERY SAMPLE                PAGE 1A
16:53:20                                                          DETAIL

  CUSTOMER                                                PHONE       SALESMAN
    NAME                           CITY             STATE  NUMBER         ID
  ------------------------------  ---------------  -----  ----------  --------
  UNITED ATLANTIC SHARES          CHARLOTTE        NC     7045554852  52733

  TOTAL ACTIVITY/MONTH 02


  AVERAGE SALE   0000452.00

  INTERNATIONAL BANK CORP.        NEW YORK         NY     2125559889  52733

  TOTAL ACTIVITY/MONTH 03

 -------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>

01/05/2010         DATAQUERY SAMPLE                PAGE 1
16:53:20                                                      TOTALS-ONLY

  ACTIVITY  YEAR TO DATE
   MONTH       SALES
  --------  ------------
  02             452.00
  03            2000.00
  04          341924.50
  06            5000.00
  08               0.00
  09               0.00
  11         1030318.34

* GRAND TOTAL
             1379694.84

 -------------------------------- LAST PAGE ---------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

WHEN-ONLY - Show Only Calculation Results

Purpose

If a query in DQL Mode contains one or more WHEN/DO statements, you can issue this command during online output display to show only the results of WHEN/DO statements.

When to Use It

Use this command during online DQL Mode report display.

Syntax

Enter	With	When You Want to:
WHEN-ONLY	No operand	See only the results of a DQL Mode WHEN/DO query statement.

Example

```
 => WHEN-ONLY
01/19/2010         DATAQUERY SAMPLE                 PAGE 1
11:10:12                                                          DETAIL

    DEPT  NAME                          SALARY           COMM          WAGES
  ------  --------------------  -------------  -------------  -------------
  00015   ROTHMAN                   1,152.00      16,502.83      17,654.83
          HANES                    22,659.80            .00      22,659.80

  TOTAL DEPT 00015
                               *   23,811.80  *   16,502.83

  SUM WAGES     40,314.63

  00020   WILSON                   16,111.00         219.00      16,330.00
          PERNAL                    1,234.56         612.45       1,847.01
          SEPULVEDA                29,100.00             -              -
          SANDERS                  18,357.50            .00      18,357.50
          GRAY                     18,199.00         750.00      18,949.00
 -------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>
01/19/2010         DATAQUERY SAMPLE                PAGE 1
11:10:12                                                          WHENS-ONLY


  SUM WAGES     40,314.63


  SUM WAGES     55,483.51


  SUM WAGES     95,362.30


  SUM WAGES     99,748.96


  SUM WAGES             -
-------------------------------- MORE .... -----------------------------------
<PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
<PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
<PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

WRAP - Wrap Report Lines

Purpose

Use the WRAP command to display a columnar report that extends beyond the terminal screen on one screen. The report lines and column headings wrap so that the report can be viewed without changing windows.

When to Use It

To display all data on one screen.

Syntax

Enter	With	When You Want to:
WRAP	No operand	See all columnar data on one screen.

Example

```
 => wrap
01/19/2010         DATAQUERY SAMPLE                PAGE 1A
16:53:20                                                          DETAIL

  CUSTOMER  CUSTOMER  CUSTOMER                        CREDIT    PHONE
     ID      NUMBER     NAME                        INDICATOR  NUMBER
  --------  --------  ------------------------------  ---------  ----------
  A0030     0030      CANNON TOOLS CO                 A          4045553322
  A0090     0090      INTERNATIONAL BANK CORP         A          2125559889
  A0130     0130      SUN DIAL CITRUS GROWERS         A          2135554800
  A0150     0150      IMPERIAL BANKCORP               A          2125558878
  A0170     0170      UNITED ATLANTIC SHARES          A          7045554852
  B0190     0190      SOUTHWEST STATE OIL REFINING    A          8185559003
  B0210     0210      WEST LIFE INSURANCE             A          3015554002
  B0230     0230      CHEMICAL MUTUAL                 A          8175551044
  B0250     0250      MICHIGAN LIGHTING INC.          A          6025558306
  B0270     0270      HARTFORD IRON WORKS INC         A          2155552208
  B0290     0290      NATIONAL HARRIS CORPORATION     A          4045553001
  B0310     0310      CARTERET STEEL CORPORATION      A          4155552002
 ------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Result

```
 =>

01/05/2010         DATAQUERY SAMPLE           PAGE 1
16:53:20                                                          DETAIL

-------------------------------------------------------------------------------
  CUSTOMER/ID  CUSTOMER/NUMBER  CUSTOMER/NAME                  CREDIT/INDICATOR
    PHONE/NUMBER  YEAR TO DATE/SALES  ACTIVITY/DATE  TOTAL/SALES
-------------------------------------------------------------------------------

  A0030        0030             CANNON TOOLS CO                A
    4045553322            320823.00   870420          99,823.00

  A0090        0090             INTERNATIONAL BANK CORP        A
    2125559889               275.59   861113          23,275.59

  A0130        0130             SUN DIAL CITRUS GROWERS        A
    2135554800              3061.46   861113          26,061.46

 ------------------------------- MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Chapter 6: DQL Statements

Whether you build your queries using Guided Query or the CREATE function, your query uses the same DQL statements. These statements instruct CA Dataquery to search a table, limit the search, relate two or more tables, create temporary results, sort the data, determine the format of the report and perform mathematical functions. DQL statements are English-like sentences and clauses whose keywords are verbs like FIND, SORT, or PRINT.

This chapter provides detailed information about DQL syntax. The CA Dataquery verbs appear in alphabetic order with the verb name listed on the top outside corner of the page. For help in understanding the syntax diagrams, turn to Reading Syntax Diagrams (see page 16).

The following chart summarizes the DQL statements. Statements appear in the order in which they should appear in a query.

Required YES/NO	DQL Keyword	Description
YES	FIND or COUNT	FIND starts a query and specifies how many rows to retrieve in the table being searched. COUNT begins a query that counts the number of rows in a table and does not produce a report. Can use either with **WITH** to limit the rows counted and a join keyword to retrieve rows from more than one table.
NO	WITH	Limits the search to rows which match specified selection criteria. Can use with **FIND** and relationship keywords to limit the rows which are retrieved. Uses comparison and logical operators to build logical expressions for the selection criteria.
NO	RELATED OUTER-JOINED LEFT-JOINED RIGHT-JOINED OUTER-DISJOINED LEFT-DISJOINED RIGHT-DISJOINED	Relationship keywords that tell CA Dataquery to search more than one table to retrieve specific rows. Retrieves those that are related by a key or column and those that are not, depending on the keyword used. See Relationship Clauses (see page 444) and Outer Joins (see page 455).
NO	SET	Creates a temporary result to hold results of calculations involving column data from the database or literal values. Can perform mathematical functions, such as finding the average or maximum of a list of specific values, within the arithmetic expression.

Required YES/NO	DQL Keyword	Description
NO	SORT	Specifies key or columns with which to sort the data in a specific order (UP or DOWN) for report output. Can also specify control break columns.
YES (If output is wanted.)	PRINT or DISPLAY	PRINT produces a report in columnar format for specified database columns or temporary results. Can also specify report title, alternate headings for columns, the accumulation of totals for numeric columns at a control break, and alternate edit patterns for numeric data in the report output. DISPLAY produces a report in one row per page format listing specified database columns or temporary results. Can also specify alternate headings for columns and alternate edit patterns for numeric data in the report output.
NO	WHEN/DO	Specifies when a mathematical function or page break is to be performed and what column is the object of the mathematical function. Can specify mathematical function or page break based on a control break, or at the end of the report.

Performance Considerations

Using a COUNT statement rather than a FIND statement can significantly improve the performance of your query in its search for the specified rows. Because no data is retrieved, the search requires fewer resources. In some cases, the difference in performance will be small.

Using keys in the selection criteria for your WITH clause can significantly improve the performance of your FIND or COUNT statement in its search for the specified rows. The purpose of keys is to expedite the search for data. In some cases, specifying a column in your selection criteria can be almost as efficient as using a key. Using an arithmetic expression in your selection criteria will nearly always require more overhead than specifying a key or column in the WITH clause.

When you specify relational criteria in a relationship clause, again the use of keys can be more efficient than specifying a column or a literal value to join the tables your query searches.

Your choice of operators in the WITH clause can also affect the efficiency of the FIND. For example, using a key and EQUAL as the operator for your selection criteria can be more efficient than choosing the combination of masking and another operator, such as GREATER THAN. See WITH Clauses (see page 504) for details about operators.

When using a left join or left disjoin, as discussed on Outer Joins (see page 455), performance should not be negatively impacted. Performing a right join or right disjoin could be costly for large tables. These joins are not reversible.

In the case of the join of three or more tables, the result of the processing would be as if the joins were done two at a time in the order written, with the result of the join of the first pair of tables being joined to the third table, and so on.

Getting Assistance

If you have concerns about the performance of your selection criteria and relationship criteria in a query, your Database Administrator or CA Dataquery Administrator can help you determine which combination of keys or columns and operators will make your query most efficient.

COUNT Statement

Purpose

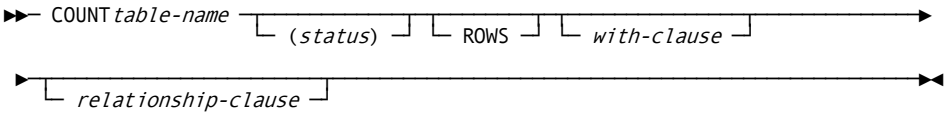
The COUNT statement specifies the information to be located in the database. The COUNT statement contains three major parts: the COUNT, the selection criteria and the relationship criteria. Criteria can be omitted.

Operation

The COUNT statement must be the first entry of a COUNT query. When you execute a COUNT statement, CA Dataquery locates the specified data and reports on the number of rows found by displaying a message on the screen.

Syntax Diagram

Following is the syntax for the COUNT statement.



Explanation

The following explains each keyword in a COUNT statement.

table-name

Represents the name of the primary table containing the rows CA Dataquery is to count. Only those tables shown in your Directory of Tables are available for query access.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following chart describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

ROWS

Used for readability only.

with-clause

Represents one or more logical expressions which specify the criteria CA Dataquery is to use to qualify rows to be counted. A WITH clause contains the criteria which qualify the rows. Only those rows which meet the specified criteria in the WITH clause are counted, as in the following example:

```
COUNT CAI-SLSHST-REC ROWS
          WITH SHIP-QTY GT 0
```

The selection criteria can consist of multiple conditions which serve to limit the number of rows to those which meet all conditions. See WITH Clauses (see page 504) for more information.

You can decide not to include any selection criteria. This causes CA Dataquery to count all the rows in the specified table.

relationship-clause

Represents a relationship expression which relates the data in two or more tables. A relationship clause establishes the relationship between tables. When you use RELATED BY to join tables, you can access all the data contained in each row of the relationship, as in the following example:

```
COUNT CAI-SLSHST-REC ROWS
     WITH SHIP-QTY GT 0
RELATED BY SLMN-ID TO CAI-ORDER-REC ROWS
     WITH ORD-YR LT '87'
```

When you use an outer join keyword, you can retrieve unrelated rows, when needed. See Outer Joins (see page 455) for details. Multiple relationship expressions can exist in a COUNT statement. In addition, you can specify selection criteria for each relationship clause. See Relationship Clauses (see page 444) for more information.

DELETE Statement

You need special authorization to use the DELETE statement in a query to delete data from the database. See your CA Dataquery Administrator for authorization and to see the documentation in the *CA Dataquery Administrator Guide*.

DISPLAY Statement

Purpose

Use the DISPLAY statement to present the information produced from your query one row at a time as a list of columns. The DISPLAY statement allows you to:

- Present column and key names of a table in a list format.
- Specify alternate heading names to represent columns and keys.
- Designate how numeric column values appear in the report.

Operation

Each report page has a standard heading that includes the row number being viewed and the page within the row. When you specify a large number of columns to be displayed, CA Dataquery provides additional pages for each row. Column headings appear on the left side of the page with the data on the right side.

Before CA Dataquery sends the DISPLAY output to the terminal, it edits the data for errors relating to the DISPLAY format.

- If the data consists of any hexadecimal values which are not acceptable to the terminal, CA Dataquery translates those values to a character. This can occur when you print a compound field composed of one or more packed or binary fields.
- To cause the compound field to be displayed as individual columns, specify YES as the value for GROUP DISPLAY in your user profile. If the compound field is composed of repeating fields, only the first occurrence is broken down and displayed unless other occurrences are explicitly requested.

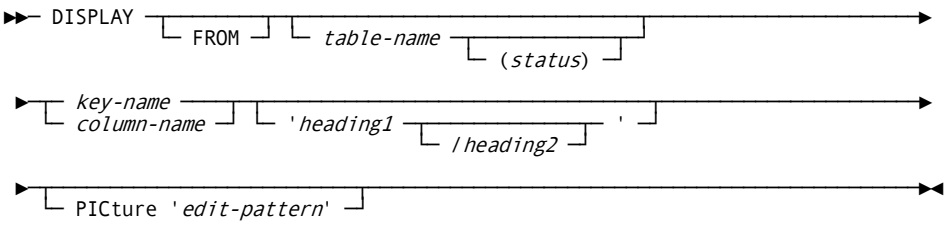
  Note: For more information about the Group Display option, see the *CA Dataquery User Guide*.

Caution Output from queries containing a DISPLAY statement cannot be printed on a system printer unless the query is executed in batch.

If CA Dataquery encounters an error in a column being edited, a special error code surrounded by asterisks is reported in place of the column. (See the *CA Dataquery Message Reference Guide* for a complete explanation of PRINT and DISPLAY error codes if you receive one of these messages.)

After executing a query containing a DISPLAY statement, you can page through the rows found.

Syntax Diagram



Repeat parameters once for each key or column.

Caution To simplify discussion, important elements of the DISPLAY statement appear in the following separate sections.

Basic statement

Section Basic Syntax (see page 417)

For alternate output headings

Section Alternate Output Headings (see page 419)

For edit patterns

Section PICTURE Clause (see page 424)

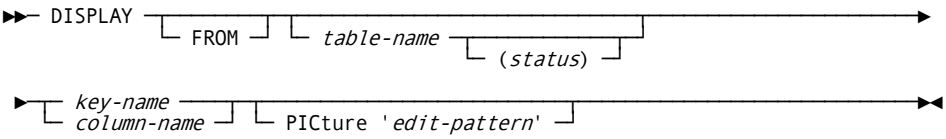
Basic Syntax

Purpose

You can create a DISPLAY statement without including headings or a PICTURE clause. The basic statement is simpler without it.

Syntax Diagram

The following is the basic syntax of DISPLAY with a PICTURE clause.



Explanation

This section explains each keyword in the DISPLAY format.

FROM

Used for readability only.

table-name

Specifies the Datadictionary name of the table where CA Dataquery obtains the values of the columns and keys indicated. You only need to include the table name if the column or key in the statement refers to a table which is different than the most recently named table in the query.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following chart describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

key-name

Specifies the name of a key contained in the table specified in the DISPLAY statement or the most recently named table in a previous statement.

column-name

Specifies the name of a column that is either in the table named in the DISPLAY statement or in the previously named table. A column also can be a result from a previous SET statement in the query. And, a column can be one occurrence in a repeating field. If that is the case, use subscripts to identify which occurrence is to be used. A typical subscript looks like this:

SALES-RESULTS (5,2)

(See the *CA Dataquery User Guide* for more information.)

PICTURE 'edit-pattern'

See PICTURE Clause (see page 424) for information about PICTURE clauses.

Example

The following illustrates a basic query which displays the data resulting from query execution.

```
FIND 10 CAI-SLSHST-REC ROWS
DISPLAY FROM CAI-SLSHST-REC SLMN-ID
                            ORD-ID
```

Output

A sample of the first screen of output generated from the preceding query follows:

```
=>

-------------------------------------------------------------------------------DQED0
DATAQUERY:  COLUMN VALUE DISPLAY                        ROW 00001 PAGE  00001
-------------------------------------------------------------------------------
SLMN-ID                          I4790
ORD-ID                           01002









-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> PREV ROW      <PF4> NEXT ROW
<PF5> NOT USED      <PF6> STATS         <PF7> BACKWARD      <PF8> FORWARD
```

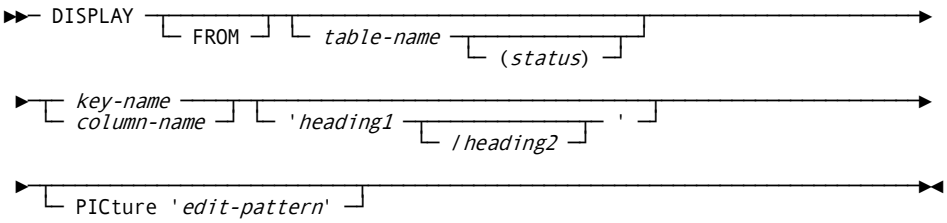
Alternate Output Headings

Purpose

The DISPLAY statement also allows you to specify alternate headings for the displayed output.

Syntax Diagram

To assign alternate headings to display for each column or key, use the following DISPLAY format.



Repeat parameters once for each key or column.

Explanation

The following explains each keyword in the DISPLAY syntax used to produce alternate output headings.

'heading1 / heading2'

Assigns an alternate column heading name for the preceding column or key. A heading name can be up to 32 characters, including blanks, and must appear within apostrophes ('). If you do not specify a heading name, CA Dataquery prints an alternate heading from Datadictionary, if your site management selected that choice as a system option; otherwise, CA Dataquery prints the Datadictionary name of the column or key. Column headings can be two lines of 22 characters each, separated by a site-specified heading separator character, such as a slash (/). See the System Profile topic on the Topic Help panel for the heading separator used at your site.

PICTURE 'edit-pattern'

See PICTURE Clause (see page 424) for information about PICTURE clauses.

Example

The following is a sample query and query output illustrating the use of designated one- and two-line headings.

```
FIND 10 CAI-SLSHST-REC ROWS
DISPLAY FROM CAI-SLSHST-REC SLMN-ID
                            'REPRESENTATIVE/ID NUMBER
                            ORD-ID
                            'ORDER NUMBER'
```

Output

```
=>

-------------------------------------------------------------------------DQED0
DATAQUERY:  COLUMN VALUE DISPLAY                     ROW 00001 PAGE 00001
-------------------------------------------------------------------------------
REPRESENTATIVE
ID NUMBER                        I4790
ORDER NUMBER                     01002









-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> PREV ROW      <PF4> NEXT ROW
<PF5> NOT USED      <PF6> STATS         <PF7> BACKWARD      <PF8> FORWARD
```

ERASE Statement

You need special authorization to use the ERASE statement in a query to remove rows from the database. See your CA Dataquery Administrator for authorization and to see the documentation in the *CA Dataquery Administrator Guide*.

FIND Statement

Purpose

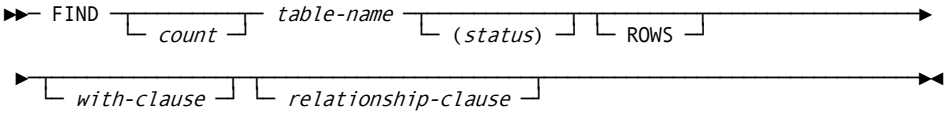
The FIND statement specifies the information to be retrieved from the database. The FIND statement contains three major parts: the FIND, the selection criteria (WITH clause) and the relationship criteria (RELATE or outer join clause).

Operation

The FIND statement must be the first entry of each new query because it requests the data which later statements manipulate and display. When you execute a FIND statement, CA Dataquery establishes an active found set of data replacing any found set created from the execution of a previous FIND statement.

Syntax Diagram

Following is the syntax of the FIND statement.



Explanation

The following explains the keywords in the FIND statement.

count

Represents the maximum number of rows you want to select. You can specify ALL or a positive number. When you specify a positive number, CA Dataquery continues selecting rows until it can find no more rows or it finds the number of rows specified. If you do not specify any value for count, CA Dataquery defaults to ALL.

table-name

Represents the name of the table from which CA Dataquery is to retrieve the specified data. Only those tables shown in your Directory of Tables are available for your query access.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following chart describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

ROWS

Used for readability only.

with-clause

Represents one or more logical expressions which specify the criteria on which CA Dataquery is to select rows from the database. A WITH clause contains the criteria which qualify the rows. Only those rows which meet the specified criteria in the WITH clause are selected, as in the following example:

```
FIND ALL CAI-SLSHST-REC ROWS
     WITH SHIP-QTY GT 0
```

The selection criteria can consist of multiple conditions which serve to limit the number of rows to those which meet all conditions. See WITH Clauses (see page 504) for more information.

You can decide not to include any selection criteria. This causes CA Dataquery to select all the rows in the specified table. However, this can cause the query selection process to run for a long period of time due to CA Dataquery selecting a potentially large amount of rows, depending on the size of the table involved.

relationship-clause

Represents a relationship expression which relates the data in two or more tables. A relationship clause establishes the relationship between tables. When you use RELATED BY to join tables, you can access all the data contained in each row of the relationship, as in the following example.

```
FIND ALL CAI-SLSHST-REC ROWS
     WITH SHIP-QTY GT 0
RELATED BY SLMN-ID TO CAI-ORDER-REC ROWS
     WITH ORD-YR LT '87'
```

When you use an outer join keyword, you can retrieve unrelated rows, when needed. See Outer Joins (see page 455) for details.

Multiple relationship expressions can exist in a FIND statement. In addition, you can specify selection criteria for each RELATED clause. See Relationship Clauses (see page 444) for more information.

INSERT Statement

You need special authorization to use the INSERT statement in a query to add rows to the database. See your CA Dataquery Administrator for authorization and to see the documentation in the *CA Dataquery Administrator Guide*.

PICTURE Clause

Purpose

You can specify the appearance of numeric data on a report by adding a PICTURE clause to the PRINT or DISPLAY statement. An edit pattern establishes a format for the display of non-floating point numeric data. The purpose of an edit pattern is to convert numeric data into a more readable format. This format can indicate a dollar value, a minus sign, a credit or debit, and so forth.

Explanation

You can specify an edit pattern in the PRINT or DISPLAY statement in your query. When you include a PICTURE clause in one of these statements, this overrides the default edit pattern defined by Datadictionary. Overriding a default edit pattern allows you to manipulate the final appearance of numeric data without affecting the decimal precision used in computation.

CA Dataquery selects an edit pattern according to the following order of precedence.

1. A PICTURE clause specified in a query.
2. An edit pattern as specified in Datadictionary (when such an option is specified at your site).
3. A default pattern based on the precision specified by the column definitions in Datadictionary.

A numeric value is shifted to fit a PICTURE clause by aligning the decimal points. Truncation (dropping all digits right of the decimal) of a numeric value is permitted when the value can be rounded and truncated to fit the PICTURE clause. However, an error condition results if truncation is attempted on the integers to the left of the decimal.

When CA Dataquery processes a query, it presents the values of numeric columns or keys as specified in the PICTURE clause of the PRINT or DISPLAY statement of the query. If you do not include a PICTURE clause in the PRINT or DISPLAY statement, CA Dataquery presents the data as established in the edit pattern for that column in Datadictionary, or it uses the information pertaining to the column from the Datadictionary definition for the column, as shown in the extended column display.

If, however, you do not define an edit pattern for numeric columns to Datadictionary, CA Dataquery creates its own edit pattern by displaying all digit positions and inserting a decimal point if needed. This generally results in leading zeros.

Explanation

To override the Datadictionary edit pattern and the standard data format that CA Dataquery uses to present numeric columns, CA Dataquery provides the PICTURE clause. The PICTURE clause consists of the word PICTURE followed by a sequence of characters, called an edit pattern, which is enclosed in apostrophes (').

An edit pattern can contain several types of characters that are used to specify the way in which each digit of the numeric column appears when displayed. The types of characters which you can include in an edit pattern are replacement characters, insertion characters, sign and currency symbols.

Euro Currency Symbol

You can use the euro currency symbol in EDIT PATTERNS and PICTURE clauses in CA Dataquery currency fields, that is, the following symbol:

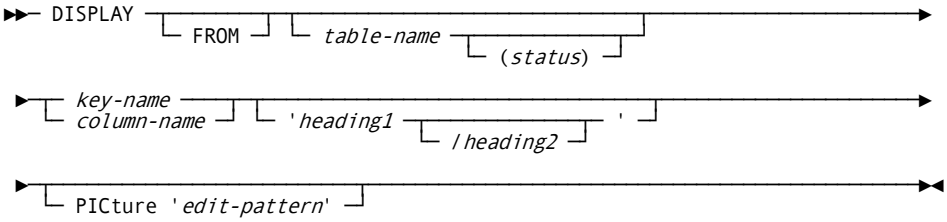
€

To use the euro symbol:

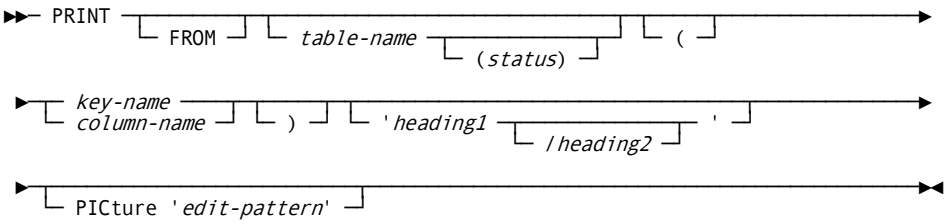
- In DQL mode, the euro currency symbol can be specified in the PICTURE clauses on the PRINT statement in a query.
- In SQL mode, CA Dataquery handles the euro currency symbol in the edit patterns of the SQL Report Formatting for SQL queries on the DQF70 panel.

CA Datacom/DB Datadictionary Edit Patterns

When using CA Datacom/DB Datadictionary edit patterns to display CA Dataquery currency fields, be aware that Datadictionary does not check the edit pattern, that is, if a terminal is capable of entering and displaying an edit pattern, Datadictionary accepts it. CA Dataquery handles the euro currency symbol in Datadictionary edit patterns but does not handle other nonstandard currency symbols.



Repeat parameters once for each key or column.



Repeat parameters once for each key or column.

Explanation

The following explains 'edit pattern' in the preceding diagrams.

PICTURE 'edit pattern'

Specifies how data appears when displayed. Entering an edit pattern overrides the edit pattern specified in Datadictionary. (The short form for PICTURE is PIC.)

The edit pattern must appear within apostrophes ('). If you are using the PICTURE clause with a signed result of a SET statement or with a signed numeric column, the PICTURE clause should include a sign (although one will be added automatically, if needed). For example, PICTURE 'ZZZZZZ99.99-'.

The new edit pattern must conform to CA Dataquery rules for editing numeric columns. See the following sections.

Replacement Characters

The following is a list of the valid replacement characters you can include in an edit pattern. View the extended column display to determine the number of digits in the specified column.

9

Specifies that the corresponding digit position in the value of the column is to contain a decimal digit.

Z

Indicates that if a leading zero appears in the corresponding position in the value of the column, a blank character replaces the zero when the value is displayed. If a nonzero digit appears in the corresponding position in the value, the digit is displayed.

*****

Indicates that if a leading zero appears in the corresponding position in the value of the column, an asterisk replaces it when the value is displayed. If a nonzero digit appears in the corresponding position, the digit is displayed.

Insertion Characters

Insertion characters cause the specified character to be inserted into the corresponding position of the value of the specified column. An insertion character indicates that a character is to be inserted between digit positions; it is not a digit position itself.

Decimal Point

A *period* is the default. See the System Profile for other valid values. The character is inserted in the corresponding position of the data column. CA Dataquery shifts, rounds, and, if necessary, truncates decimal places of the value being edited to match the specified edit pattern.

Number Separator

A *comma* is the default but any character is acceptable except &., *, or the decimal point. It causes the character to be inserted into the corresponding position of the value. If the edit pattern requests zero suppression, the number grouping character is also suppressed. This character is inserted only when an unsuppressed digit appears to its left.

Other insertion characters

A *slash* causes the slash character to be inserted into the corresponding position of the value when the value is presented on a report. A *hyphen* causes the hyphen character to be inserted into the corresponding position of the value when the value is presented on a report.

Signs and Symbols

You can specify only one type of sign character or currency symbol in an edit pattern. You can specify a sign only in the rightmost position of the edit pattern. A currency symbol can be only in the leftmost position of the edit pattern.

$

Specifies that a dollar sign currency symbol is to appear with the value when displayed. If you use this symbol more than once in succession, it becomes a floating character. A floating character specifies that the dollar sign is to appear left of the leftmost digit in the value. To specify a floating character, use multiple dollar signs in the edit pattern in each digit position through which it can float.

€

Specifies that a euro currency symbol is to appear with the value when displayed. If you use this symbol more than once in succession, it becomes a floating character. A floating character specifies that the euro symbol is to appear left of the leftmost digit in the value. To specify a floating character, use multiple euro symbols in the edit pattern in each digit position through which it can float.

-

Specifies that the minus sign is to appear only if the value of the column is less than zero. If the value is greater than zero, no sign appears.

CR

Specifies that the credit symbol is to appear only if the value is less than zero.

DB

Specifies that the debit symbol is to appear only if the value is less than zero.

Examples

The following examples illustrate the use of a PICTURE clause. These edit patterns refer to numeric columns with a length of five digits. In the first column are sample PICTURE clauses. The other three columns show what the output looks like using the PICTURE clause in the first column and the data appearing at the top of each column.

PICTURE Clause	9876.5	0012.3-	0000.9
PIC '9999.9-'	9876.5	0012.3-	0000.9
PIC 'ZZZZ.Z-'	9876.5	12.3-	0.9
PIC 'ZZZ9.9-'	9876.5	12.3-	0.9
PIC '****.*CR'	9876.5	**12.3 CR	****.9
PIC '$$$$$.$CR'	$9876.5	$12.3 CR	$.9
PIC '$$$$9.9CR'	$9876.5	$12.3 CR	$0.9

Sample DISPLAY Query

The following sample query illustrates the use of a PICTURE clause in the DISPLAY statement.

```
FIND 10 CAI-SLSHST-REC ROWS
     WITH SHIP-QTY > 0
DISPLAY FROM CAI-SLSHST-REC SLMN-ID      'REPRESENTATIVE'
                            ORD-ID       'ORDER NUMBER'
                            ITM-ID       'ITEM NUMBER'
                            SHIP-QTY     'QUANTITY SHIPPED'
                              PICTURE 'ZZZZZZ9-'
                            UNIT-PRICE   'PRICE'
                              PICTURE '$$$$$9.99-'
                           DISC-PCT     'PERCENT DISCOUNT'
                              PICTURE '99.9-'
```

Zs

Replace leading zeros with blanks.

9s

Indicate that the corresponding digit position in the column is to be printed, even if it is a zero.

$

Instructs CA Dataquery to place a dollar sign to the left of the first digit which is not 0. For UNIT-PRICE, an extra dollar sign is required to allow for a floating dollar sign even when the column is a full seven digits long.

-

Indicates a signed column (determined by displaying the Extended Column Definition).

The number of positions identified in each PICTURE clause represents the maximum width of the column (plus any floating dollar sign).

Output

When you execute this DISPLAY query, the following appears as output. Notice the result of the PICTURE clause.

DISPLAY Output with PICTURE Clause

```
=>

-------------------------------------------------------------------------DQED0
DATAQUERY:  COLUMN VALUE DISPLAY                      ROW 00001 PAGE 00001
-------------------------------------------------------------------------------
REPRESENTATIVE                  I4790
ORDER NUMBER                    01002
ITEM NUMBER                     A10002
QUANTITY SHIPPED                4
PRICE                           $19.99
PERCENT DISCOUNT                .0









-------------------------------------------------------------------------------
<PF1> HELP          <PF2> RETURN        <PF3> PREV ROW      <PF4> NEXT ROW
<PF5> NOT USED      <PF6> STATS         <PF7> BACKWARD      <PF8> FORWARD
```

Sample PRINT Query

The following query demonstrates the use of specifying an edit pattern in the PRINT statement.

```
FIND 10 CAI-SLSHST-REC ROWS
     WITH SHIP-QTY > 0
PRINT FROM CAI-SLSHST-REC SLMN-ID      'REPRESENTATIVE'
                ORD-ID       'ORDER NUMBER'
                ITM-ID       'ITEM NUMBER'
                SHIP-QTY     'QUANTITY SHIPPED'
                  PICTURE 'ZZZZZZ9-'
                UNIT-PRICE   'PRICE'
                  PICTURE '$$$$$9.99-'
                DISC-PCT     'PERCENT DISCOUNT'
                  PICTURE '99.9-'
```

Zs

Replace leading zeros with blanks.

9s

Indicate that the corresponding digit position in the column is to be printed, even if it is a zero.

$

Instructs CA Dataquery to place a dollar sign to the left of the first digit which is not 0. For UNIT-PRICE, an extra dollar sign is required to allow for a floating dollar sign even when the column is a full seven digits long.

-

Indicates a signed column (determined by displaying the Extended Column Definition).

The number of positions identified in each PICTURE clause represents the maximum width of the column (plus any floating dollar sign).

Output

When you execute the preceding query, the following appears as output. Notice the result of the PICTURE clause for SHIP-QTY and UNIT-PRICE. Also notice that the DISC-PCT column output does not appear on this page of the report.

PRINT Output with PICTURE Clause

```
 =>

01/02/2010                  CA Dataquery                            PAGE 1A
15:30:52                                                                 DETAIL

  REPRESENTATIVE  ORDER NUMBER  ITEM NUMBER  QUANTITY SHIPPED     PRICE
  --------------  ------------  -----------  ----------------  --------
  I4790                  01002  A10002                      4    $19.99
  I4790                  01002  A20000                      1   $149.99
  I4790                  01002  A30000                      2    $39.99
  15266                  01004  A40000                      5     $5.99
  I4790                  01002  A60000                      4    $49.99
  52733                  02121  C10001                     23    $21.00
  H4130                  01003  H10003                    100    $12.99
  H4130                  01003  H20005                    200    $17.99
  H4130                  01003  H30001                    102    $14.99
  H4130                  01003  H40000                    900    $15.99



 ------------------------------- LAST PAGE -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

PRINT Statement

Purpose

Use the PRINT statement to show the information produced from your query in a report-like format. The values of the columns and keys you specify in the PRINT statement appear in columns. Each column has a standard heading which can be specified by you as the name of the column. Names used in a PRINT statement are first evaluated as columns, then as keys if no column is found with the specified name. The PRINT statement allows you to:

- Present column and key names of a table in a columnar format.
- Specify alternate heading names to use as column headings.
- Designate a title to appear at the top of each page of the report.
- Designate how numeric column values appear in the report.
- Name a column or key more than once to assure that it appears on each page of a wide report that requires multiple pages to print all the columns.

CA Dataquery also allows you to specify in the PRINT statement the accumulation of numeric columns. For those columns which you want to accumulate, enclose the column name in parentheses. The column to be accumulated can be a specific column within a table or a result from a SET statement in the query.

Operation

If you specify control breaks in a SORT statement within your query, the columns you want totals on are subtotaled according to the control breaks designated there. You can specify up to ten levels of control break totals in the report. A grand total of the columns to be totaled is provided automatically at PRINT time.

Before CA Dataquery sends the PRINT output to your terminal, it edits the data for errors relating to how it is presented.

- If the data consists of any hexadecimal values which are not acceptable to the terminal, those values are translated to # characters. This can occur when you print a compound field composed of one or more packed or binary fields.

  Note: The default literal masking character is the pound sign (#), but another symbol can be used at your site. Display Topic Help and choose System Profile to display the symbol used at your site.

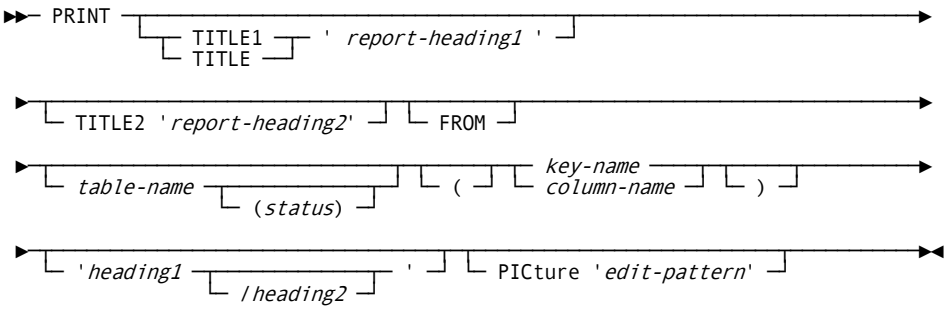
- To cause a compound field to be presented as individual columns, specify YES as the value for GROUP DISPLAY in your user profile. See the *CA Dataquery User Guide* for more information about this option.

If CA Dataquery encounters an error in a column being edited before it presents the output, a special error code surrounded by asterisks is reported in place of the column in the report. See the *Message Guide* for a list of PRINT and DISPLAY error codes if you receive one of these errors.

If you have specified control breaks in the SORT statement and accumulation columns in the PRINT statement, totals can be viewed several ways. See the *CA Dataquery User Guide* for more information.

Syntax Diagram

The following syntax shows the most complex version of the PRINT statement.



Caution! To simplify discussion, this document divides the PRINT statement into its important elements and discusses them in the following sections.

Basic statement

Next section.

For output headings

Headings Syntax Diagram (see page 436)

For edit patterns

PICTURE Clause (see page 424)

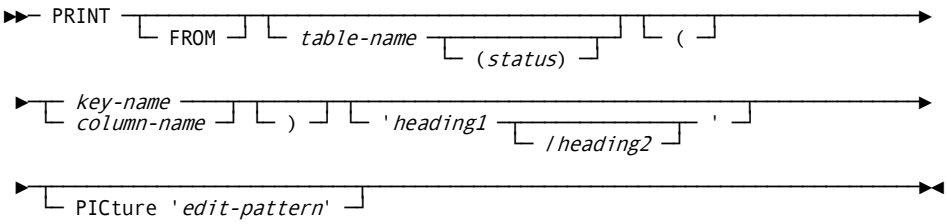
Basic Syntax Diagram

Purpose

You can create a PRINT statement without including a title for the report. The basic statement is simpler without it.

Syntax Diagram

The following is the syntax for the basic PRINT statement.



Repeat parameters once for each key or column.

Explanation

This section explains each keyword in the basic PRINT format.

FROM

Used for readability only.

table-name

Specifies the Datadictionary name of the table where CA Dataquery obtains the values of the columns and keys indicated. You only need to include the table name if the column or key in the statement refers to a table which is different than the most recently named table in the query.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following chart describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

key-name

Specifies a key that belongs to the table specified in the PRINT statement or the most recently named table in a previous statement.

column-name

Specifies the name of a column that is either in the table named in the PRINT statement or in the previously named table. A column also can be a result from a previous SET statement in the query. And, a column can be one occurrence within a repeating field. If that is the case, use subscripts to identify which occurrence is to be used. A typical subscript looks like this:

```
SALES-RESULTS (5,2)
```

(See the *CA Dataquery User Guide* for more information.) The named column cannot exceed 1700 characters in length.

If a specified column is numeric, you can designate the accumulation of the values of the column by enclosing the column name within parentheses.

PICTURE 'edit-pattern'

See PICTURE Clause (see page 424) for information about PICTURE clauses.

Example

The following sample query contains a basic PRINT statement.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
```

When you execute the preceding query, CA Dataquery displays in a columnar format the order number identifying the customer and the corresponding items ordered (which are greater than 100) and the price per item.

Output

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1
15:29:32                                                                   DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE
  ------  ----------  --------  ----------
   08811  O50006      0000203    00014.75
   09811  G10013      0000164    00110.25
   04222  O50006      0000934    00014.75









 -------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

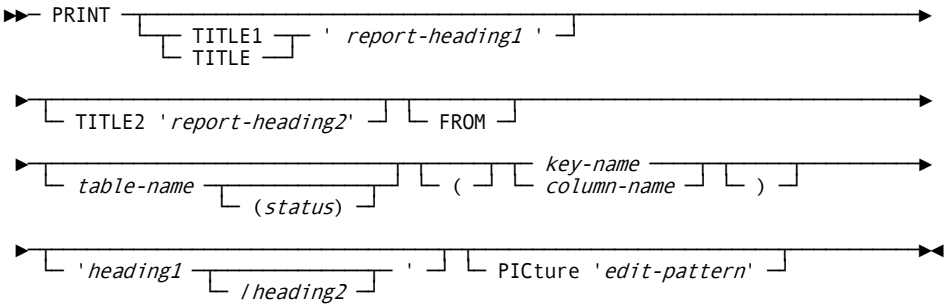
Headings Syntax Diagram

Purpose

You can use the PRINT statement to override standard column headings and to specify a one- or two-line report heading to appear at the top of the report.

Syntax Diagram

The following syntax shows the PRINT statement with keywords that add a report title and column headings.



Explanation

The following explains the keywords for headings in the preceding PRINT format.

TITLE 'report-heading'

Specifies the only line of the report heading. The maximum length of the report heading is 55 characters. When using this option, enclose the specified heading in apostrophes (').

TITLE1 'report-heading1'

Specifies the first line of the report heading. The maximum length of the report heading is 55 characters. When using this option, enclose the specified heading in apostrophes (').

TITLE2 'report heading2'

Specifies the second line of the report heading. The maximum length of the report heading is 55 characters. When using this option, enclose the specified heading within apostrophes (').

'heading1/ heading2'

Assigns an alternate one- or two-line column heading name for the preceding column or key. A heading name can be up to 32 characters, including blanks, and must appear within apostrophes ('). If you do not specify a heading name, CA Dataquery prints an alternate heading which it finds in Datadictionary, if your site management selected that choice as a system option; otherwise, CA Dataquery prints the Datadictionary name of the column or key. A column heading can be two lines long, with each line separated by a heading separator character. See the System Profile Topic Help to determine which character is used at your site as a heading separator.

PICTURE 'edit-pattern'

See PICTURE Clause (see page 424) for information about PICTURE clauses.

Example

An example illustrating these options follows:

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
PRINT FROM CAI-ORDERS-REC ORD-ID 'ORDER\NUMBER'
      FROM CAI-SLSHST-REC ITM-ID 'ITEM\NUMBER'
                          SHIP-QTY 'QUANTITY\ORDERED'
                          UNIT-PRICE 'PRICE PER ITEM'
           TITLE1 'LARGE ORDERS'
```

The title for your report is LARGE ORDERS. The report contains the order number identifying the customer, and the items ordered, how many were ordered and the cost per item.

Output

The output for the preceding query appears below.

```
 =>

01/01/2010                  CA Dataquery                              PAGE 1
15:30:19                        LARGE ORDERS                             DETAIL


  ORDER NUMBER  ITEM NUMBER  QUANTITY ORDERED  PRICE PER ITEM
  ------------  -----------  ----------------  --------------
         08811  O50006                0000203        00014.75









 -------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Totals Syntax Diagram

Purpose

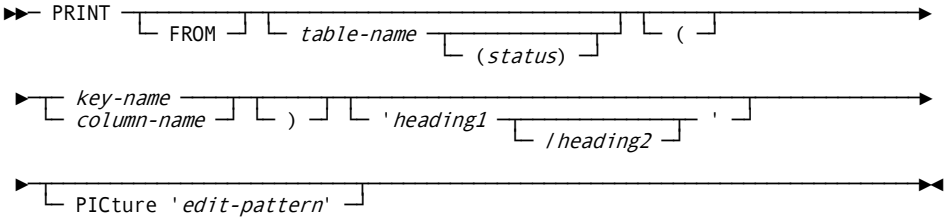
The PRINT statement can be used to accumulate subtotals and totals for specified numeric columns, including floating-point columns.

Operation

You can use the basic PRINT syntax to accumulate totals. The column or columns to be accumulated must be enclosed in parentheses.

Syntax Diagram

The following syntax diagram shows how to add parentheses to key or column names in the PRINT statement to cause totaling at control breaks (if any) and at the end of the report.



Accumulating Subtotals

Purpose

You can accumulate subtotals of a temporary result in the same way you accumulate subtotals of a database column.

Operation

Select one or more control break columns by enclosing their names in parentheses in a SORT statement and then enclose column names in the PRINT statement in parentheses where you want to show totals of those columns at control breaks or at the end of the report.

Sample

The following sample query illustrates the use of the PRINT statement to accumulate subtotals for results from a SET statement in the query.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY (ORD-ID) AND
     CAI-SLSHST-REC BY ITM-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
```

Explanation

The preceding query specifies that NET-COST is to be accumulated by enclosing that computed column in parentheses in the PRINT statement. In the SORT statement, ORD-ID is enclosed in parentheses to indicate that it is a control break column. A subtotal for NET-COST will be shown each time the value in the ORD-ID column changes.

Output

Following is the output after executing the preceding query.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1
15:31:50                                                                   DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  ----------  --------  ----------  -------------  ---------
   08811  O50006      0000203    00014.75    00002994.25   00010.00
          G00222      0000100    00100.00    00010000.00   00000.00
          910011      0001000    00100.00    00100000.00   00020.00

  TOTAL ORD-ID 08811
                                          * 00112994.25

* GRAND TOTAL
                                          * 00112994.25


 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Explanation

In the preceding output screen the values contained in ORD-ID appear only once in the ORD-ID column. This is a result of setting the SUPPRESS DUPLICATE COLUMNS option to YES in your User Profile. (See the *CA Dataquery User Guide* for more information.)

Subtotals are designated by TOTAL, followed by the specific control break column (ORD-ID) and the current value contained in the column. The SUBTOTAL value is preceded by an asterisk. The total is designated by * GRAND TOTAL. An asterisk also precedes the accumulated value.

Omitting Control Break Column Total

Purpose

The control break columns only control what groups of data get totaled on the report. They do not control which columns get totaled. That specification occurs in the PRINT statement when you enclose a column name in parentheses.

Sample

You do not need to include the control break column in the PRINT statement to display the totals for accumulated columns. In the following query, the control break column ORD-ID is not included in the PRINT statement.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY (ORD-ID) AND
     CAI-SLSHST-REC BY SHIP-QTY
PRINT FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
```

Output

Following is the output when the control break column is not included in the PRINT statement.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1
15:32:34                                                                   DETAIL

  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ----------  --------  ----------  -------------  ---------
  O50006      0000203    00014.75    00002994.25   00010.00

  TOTAL ORD-ID 08811
                                   * 00002994.25

* GRAND TOTAL
                                   * 00002994.25




 -------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

In the preceding sample screens the subtotal lines provide the name of the control break column and the information in the column. This allows you to identify the order number and the accumulated subtotal for that order.

Specifying Multiple Control Breaks

If you specify more than one control break column in the SORT statement, your output will include a PRINT column subtotal for each specified control break column. The following query illustrates the use of multiple control breaks to obtain multiple subtotals.

```
FIND ALL CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY (ORD-ID) AND
     CAI-SLSHST-REC BY (ITM-ID)
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
```

Output

The results of the preceding sample query appear on the following screen.

```
 =>
01/02/2010                  CA Dataquery                              PAGE 1
15:34:44                                                                   DETAIL

  ORD-ID  ITM-ID     SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  ---------  --------  ----------  -------------  ---------
   08001  O50006      0000203    00014.75    00002994.25   00010.00

    TOTAL ITM-ID 05006
                                  * 00002994.25
   08001  O60011      0000193    00010.00    00001930.00   00010.00
    TOTAL ITM-ID 060011
                                  * 00001930.OO
   08001  O60012      0000100    00020.00    00002000.00   00010.00
    TOTAL ITM-ID 060012
                                  * 00002000.00
TOTAL ORD-ID 08001
                                  * 00006924.25
* GRAND TOTAL
                                  * 00006924.25
 ------------------------------- LAST PAGE -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Explanation

Notice in the preceding sample screen that the line identifying the ITM-ID subtotal is indented. The indented line indicates that this is the secondary control break column and the subtotal for this control break is accumulated first. The line designating the ORD-ID subtotal is not indented. The subtotal line for ORD-ID is not indented since this column is the primary control break column. The first column enclosed in parentheses in a SORT statement is always the primary control break column. Both subtotal lines identify the column used for the control break and the value contained in that column at the time the subtotal is accumulated.

Relationship Clauses

Purpose

The CA Dataquery relationship clauses allow you to combine or join information from a maximum of 17 tables to temporarily create a single output table. You can search the temporary table with your query or dialog and produce output containing data from each related table.

The RELATED keyword allows you to retrieve only rows that are related by a common key, column, or value. The outer join keywords allow you to retrieve rows that cannot be joined using a common key or value. The rules for using both types of clauses in a query are the same, except that FIRST cannot be used with an outer join keyword.

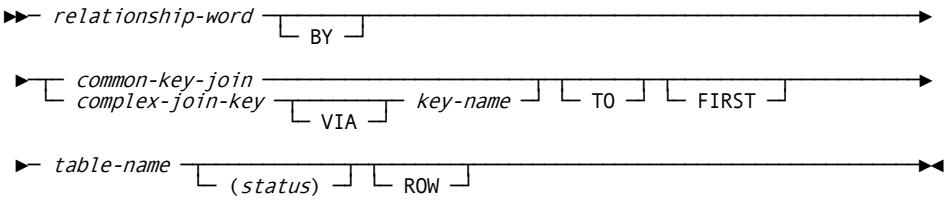
Background

When you use a RELATED BY clause to specify a relationship, the FIND statement gathers information from all related tables. The key or common column specified in the RELATED BY clause is the link between the database tables that allows the join to occur. CA Dataquery supports the traditional *equijoin* concept, meaning that tables will be joined when the common key or column contains equal values in each related table, and only joined rows will appear in the output. To learn about producing output containing unjoined rows, see Outer Joins (see page 455).

CA Dataquery retrieves all established relationships within the scope of the query, whether the relationships are one-to-one, one-to-many, many-to-one, or many-to-many. A *logical row* (the combined temporary row) exists when all of the established relationships exist and all selection criteria are met. You can eliminate duplicates in your results by using the FIRST option to select only the first row that meets all FIND statement criteria. See Complex Relationships - No Common Key (see page 449) for more information about FIRST.

Syntax Diagram

The following syntax diagram shows a complete, complex relationship statement.



Variations

The CA Dataquery RELATED and outer join keywords support two ways of retrieving rows from tables:

- Searching for common keys in each table
- Comparing the value of a column, key or literal to a key or column value of a secondary table

Combinations of these relationships can be made. A discussion of those combinations appears in Combining Relationship Types (see page 465) and Outer Joins (see page 455).

To simplify discussion, we present the options of the relationship clauses in the following sections.

Common key relationships

Section Simple Relationship with a Common Key (see page 446)

Using selection criteria

Section Adding Selection Criteria (see page 462)

Complex relationships with different types of links

Section Complex Relationships - No Common Key (see page 449)

Chained relationships

Section Chained Relationships (see page 466)

Repeating relationships

Section Repeating Relationships (see page 467)

Combining chained and repeating relationships

Section Combining Chained and Repeating Relationships (see page 469)

Deciding what type of relationship combination to use

Section Deciding Which Type of Relationship to Use (see page 465)

Outer joins

Page Outer Joins (see page 455)

JOIN-AS-IS

CA Dataquery DQL Mode provides a method for manually specifying the order in which tables are joined. In place of the RELATED BY syntax, specify JOIN-AS-IS. The tables will be joined in the same order they appear in the query syntax, and no join optimization will be performed.

Technical Support may suggest the temporary use of JOIN-AS-IS as a tool to help analyze poorly performing queries. In the case of certain queries, Technical Support may even suggest the use of JOIN-AS-IS on an ongoing basis to achieve the best join order. JOIN-AS-IS should not be used routinely for all queries. The use of JOIN-AS-IS presumes that the user is knowledgeable about the data and keys involved in the query and can thus predict the best order to use to join the tables. If the table's characteristics change significantly over time, the use of JOIN-AS-IS should be re-evaluated.

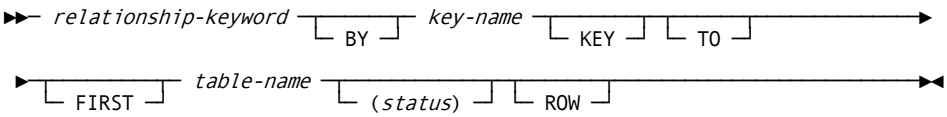
Simple Relationship with a Common Key

Purpose

A basic expression relates different types of tables by a common key. For two tables to share a common key, the key in each table must have the same structure, including name, length, type of column (numeric or character), and so forth. If the keys are multi-column keys, the key with the fewest columns must be structured the same as the corresponding columns of the other table's key.

Syntax Diagram

The following syntax diagram shows the simplest form of a relationship statement.



Explanation

The following explains the basic relationship clause.

relationship-keyword

Use RELATED to specify an inner equijoin. See Relationship Clauses (see page 444) for details.

Use one of the following keywords to retrieve rows that are not related:

OUTER-JOIN(ED)	OUTER-DISJOIN(ED)
LEFT-JOIN(ED)	LEFT-DISJOIN(ED)

RIGHT-JOIN(ED)	RIGHT-DISJOIN(ED)

See Types of Outer Joins and Disjunctions (see page 456) for details.

BY

Used for readability only.

key-name

Represents the Datadictionary name of a key or a column name that is the high-order part of a key. The key relates two tables and must be identified by the same name in both tables.

Note: Using different names will cause a CA Dataquery error condition. Be certain the key for each table has the same name and structure. If keys differ only by name, but have the same structure, use the complex relationship expression syntax diagram discussed on Complex Relationships - No Common Key (see page 449). Each key must have a similar structure (length, type, and so forth) in Datadictionary for both tables in the relationship. Verify names and key information with the extended key display or a Datadictionary report, if necessary.

KEY

Used for readability only.

TO

Used for readability only.

FIRST

Specifies that the first related row satisfies the relationship search. This feature is used to limit the scope of a relationship to a one-to-one or many-to-one relationship. FIRST will not be satisfied unless the row will also satisfy any other relationships specified in the query. The use of FIRST is not permitted for outer joins and disjoins. See Outer Joins (see page 455) for more information about outer joins.

table-name

Represents the Datadictionary name of the table to be related to the primary table contained in the FIND clause by a common key.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

ROW

Used for readability only.

Example

The following query illustrates a basic relationship between CAI-DETAIL-REC and CAI-ACCTS-REC tables sharing the common key ORD-ID-KEY.

```
FIND ALL CAI-DETAIL-REC ROWS
     WITH SHIP-QTY GREATER THAN 0
  RELATED BY ORD-ID-KEY TO CAI-ACCTS-REC ROWS
     WITH ORD-AMT GREATER THAN 3000
PRINT FROM CAI-DETAIL-REC ORD-ID
                          SHIP-QTY
      FROM CAI-ACCTS-REC  ORD-AMT
                          DISC-AMT
```

By relating the CAI-DETAIL-REC table to the CAI-ACCTS-REC table, you have more flexibility in selecting only those rows which contain the information that you require. When you execute the preceding query, CA Dataquery selects all the CAI-DETAIL-REC rows for those orders which have been shipped and which total more than $3000.

You can relate two tables which have a common column name. For example, if you relate CAI-DETAIL-REC and CAI-ITEMS-REC using KEY as the common key, you can create the following query if you only want those rows which have a common item number.

```
FIND 10 CAI-DETAIL-REC ROWS
  RELATED BY KEY TO CAI-ITEMS-REC ROWS
     WITH ITM-ID = CAI-DETAIL-REC ITM-ID
```

The first column ITM-ID in the WITH clause is contained in the last table named, CAI-ITEMS-REC. However, the second column ITM-ID is contained in the CAI-DETAIL-REC table and must be prefixed with a qualifying table name. You can prefix the first column with CAI-ITEMS-REC for your readability.

Additional Information

You can use the outer join keywords to retrieve the remaining rows. See Outer Joins (see page 455) for details.

Complex Relationships - No Common Key

Purpose

When you do not have a common key in each table, you can use relationship clauses to establish a complex relationship between two types of tables by relating a column, key or value contained in the primary table to a key or column in the secondary table.

Link in Table1	Table2
Column Key Value	Column Key

Look for examples of these types of relationships later in this section.

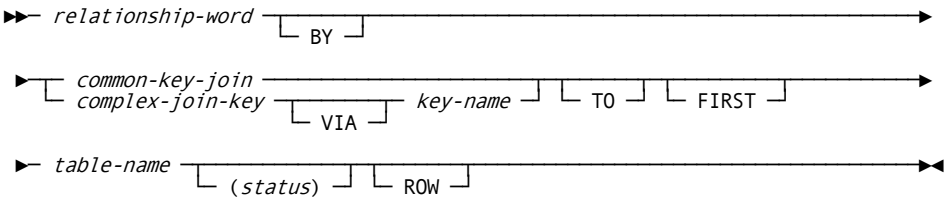
Rules

Before establishing a complex relationship between two tables, the column, key or value you use must meet the following criteria.

1. If you use a column, the value of the column must be the same as the key or column in the secondary table; if the key is a multi-column key, the value of the column must equal the value of the high-order key column.
2. If you use a key, the key from the primary table must have the same attributes as the key in the secondary table; however, the names of the keys can be different.
3. If you use a value, the value must equal the value of the key in the secondary table; if the key is a multi-column key, the value must equal the value of the high-order column.

Syntax Diagram

The following is the syntax for a complex relationship.



Explanation

The following explains the keywords used for complex relationships in the preceding diagram. See Simple Relationship with a Common Key (see page 446) for explanations of the basic keywords.

relationship-word

Use RELATED to specify an inner equijoin. See Relationship Clauses (see page 444) for details.

Use one of the following keywords to retrieve rows that are not related.

OUTER-JOIN(ED)	OUTER-DISJOIN(ED)
LEFT-JOIN(ED)	LEFT-DISJOIN(ED)
RIGHT-JOIN(ED)	RIGHT-DISJOIN(ED)

Note: See Types of Outer Joins and Disjunctions (see page 456) for details.

link

Tables can be related by matching a specified value in any column of table1 with values in a specified key or column in another table.

When the link is a column or key, a relationship forms when CA Dataquery finds a row in table2 containing the same value in key2 as the value in the link in table1. CA Dataquery relates the row found in table1 to the row in table2 to create a composite row in the active found set. Link can also be a literal value which will match a key or column in table2. Also, an output table can be produced that contains rows whose values did not match, if the outer join keywords are used in place of RELATED.

VIA

Used for readability only.

key-name

Represents the Datadictionary name of a key that is contained within the secondary table of the relationship. The key and link must be structured the same as defined to Datadictionary to have a relationship between two tables. (See the extended key display or a Datadictionary report for information on key names and structures.)

column-name

Represents the Datadictionary name of a column that is contained in the secondary table of the relationship. This column must have the same structure as the key or column named in the primary table or the link value. Using a column in the secondary table may have negative performance implications if there are not selection criteria associated with the table that reference keys.

Consult with your CA Dataquery Administrator to determine if using columns for secondary tables is appropriate for your query. Also, a column can be one occurrence within a repeating field.

If that is the case, use subscripts to identify which occurrence is to be used. A typical subscript looks like this:

```
SALES-RESULTS (5,2)
```

See the *CA Dataquery User Guide* for more information.

TO

Used for readability only.

FIRST

Specifies that the first related row satisfies the relationship. This feature is used to limit the scope of a relationship to a one-to-one or many-to-one relationship. The use of FIRST is not permitted for outer joins and disjoins. See Outer Joins (see page 455) for more information about outer joins.

table-name

Represents the Datadictionary name of the secondary table in the relationship. CA Dataquery uses the key or column from the secondary table to establish a relationship with the primary table by matching the value specified by the link in this expression. It is possible to retrieve rows whose values do not match by using an outer join keyword rather than RELATE.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

Key or Column Link

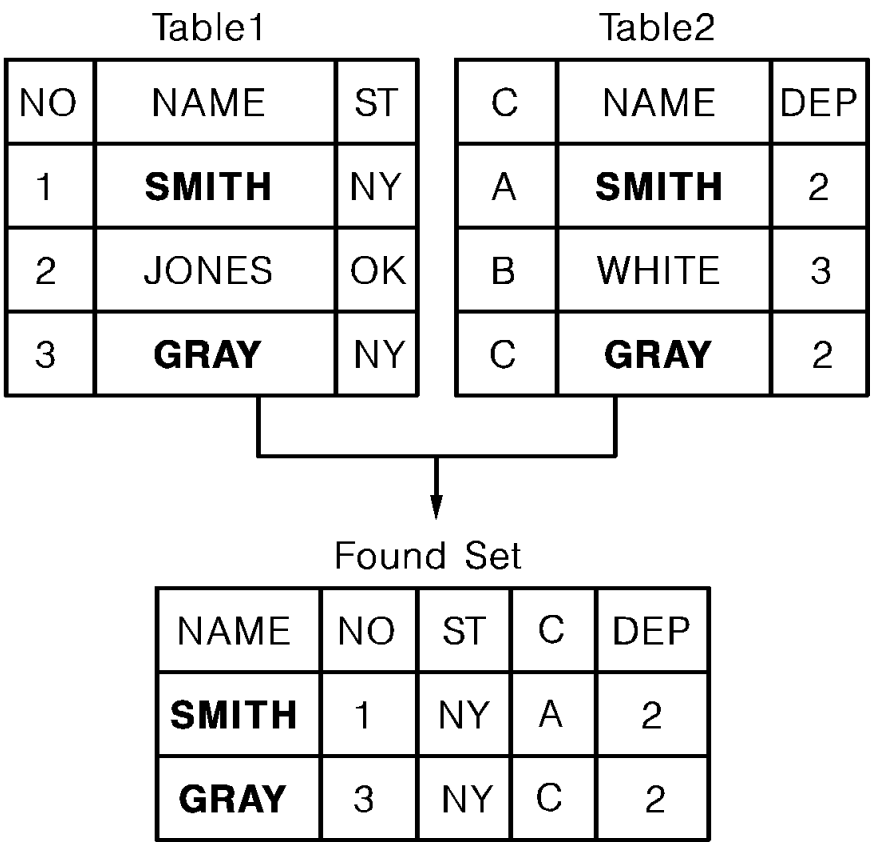
Tables can be related by matching a specified link value in one table with values in a specified key or column in another table. When the link is a column or key, a relationship forms when CA Dataquery finds a row in table2 containing the same value in the named key or column. CA Dataquery relates the row found in table1 to the row in table2 to create a composite row in the active found set. It is possible to retrieve rows whose values do not match by using an outer join keyword rather than RELATE. See Outer Joins (see page 455).

Sample Query

The following sample RELATED BY clause has a key as a link.

```
        FIND ALL TABLE1 ROWS
         WITH STATE = 'NY'
RELATED BY KEY1 VIA KEY2 TO TABLE2
```

Key or Column Link Example



Additional Information

You can use the outer join keywords to retrieve the remaining rows. See Outer Joins (see page 455) for details.

Literal Value Link

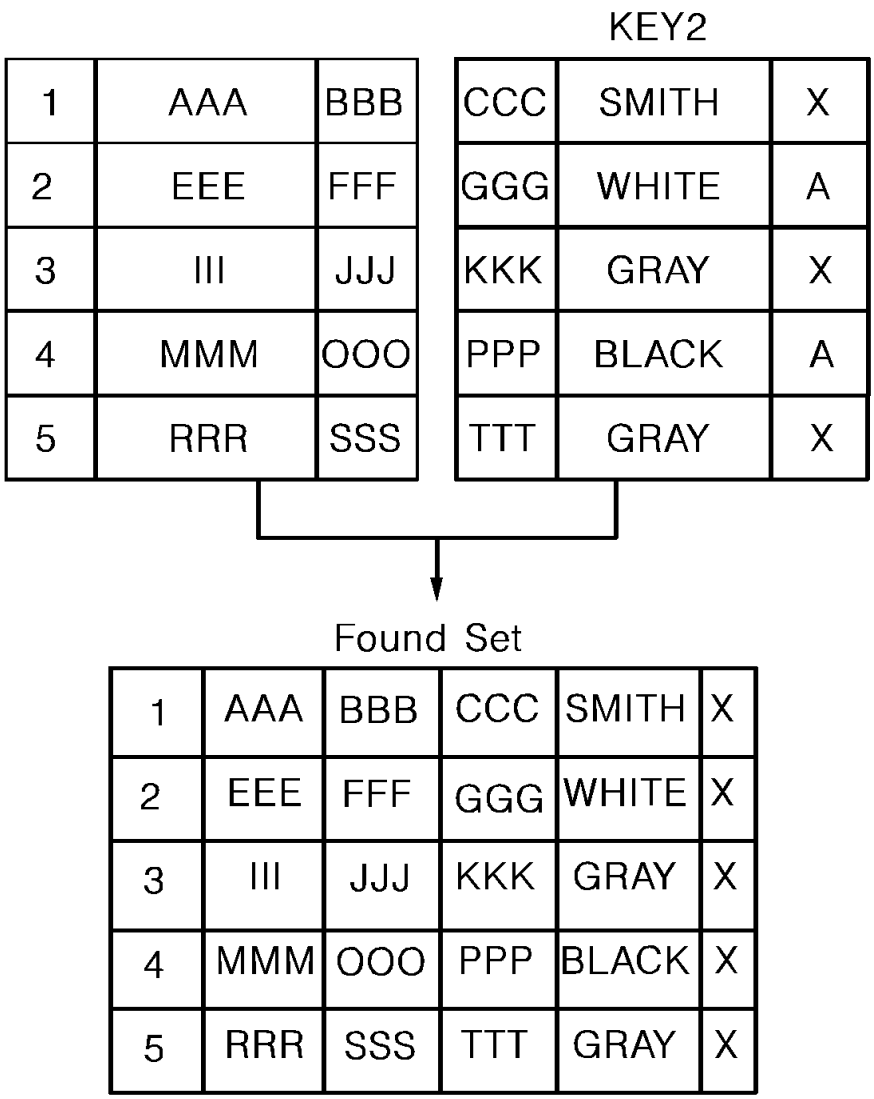
When the link is a literal value, a relationship forms when CA Dataquery finds the named literal value in the named key or column of table2. CA Dataquery creates one composite record in the active found set for each row in table1 plus the rows in table2 containing the literal.

Query

The following sample RELATED BY clause has a literal value as the link.

```
FIND ALL TABLE1 RECORDS
  RELATED BY 'SMITH' VIA KEY2 TO TABLE2
```

Literal Value Link Example



Note: If more than one occurrence of the literal appears in table2, every row in table1 will be repeated once in the active found set for each occurrence found in table2. For example, if ten rows are found in table1 and the literal occurs twice in table2, the composite rows in the active found set will number twenty.

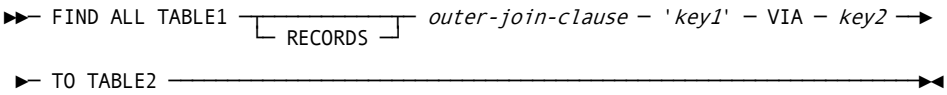
Outer Joins

When rows from one table are combined with rows from another table based on an equal comparison of the participating rows' values, the join bypasses rows in all searched tables which have no matching value. The rows not retrieved are not included in the active found set when the RELATED BY clause is used.

When the RELATED BY clause is specified in the query, CA Dataquery performs an inner equijoin, producing output rows by joining rows. To specify that you want the output rows to include rows **not** retrieved, use an outer join clause in place of the RELATED BY clause.

Syntax Diagram

Following is the syntax diagram for the outer join clause that replaces the RELATED BY clause when rows not joined are wanted in the output.



FIND

See FIND Statement (see page 421).

ALL

(Optional) Use ALL or a number. Defaults to ALL.

TABLE1

Represents the first table to be searched to retrieve data.

RECORDS

Optional.

outer-join-clause

A clause specifying what kinds of unmatched rows are wanted in the output. See Relationship Clauses (see page 444).

'key1'

Represents the name of a key in the first table that is also found in the second table.

VIA

Optional.

'key2'

Represents the name of a key in the second table that is also found in the first table.

TABLE2

Name of the second table listed in the query.

Tables can be related by comparing a specified value in any column of table1 with values in a specified key or column in another table.

Types of Outer Joins and Disjunctions

The following table describes the keywords to be used in a query to describe the types of outer joins available with CA Dataquery. These keywords should be used in place of the RELATED keyword.

OUTER-JOIN(ED)

Rows that do not form an inner equijoin are included in the active found set along with the rows joined with an inner equijoin (as in RELATED BY). They are extended to the length of other rows by adding null values. If an unmatched row is in the first table named, the null values are added to the right side and the row is considered left-joined.

If an unmatched row is in the second table named, the null values are added to the left side and the row is considered to be right-joined.

LEFT-JOIN(ED)

The output rows consist of all joined rows and the rows not retrieved from TABLE1, extended with null values. No unmatched rows from TABLE2 are included.

RIGHT-JOIN(ED)

The output rows consist of all inner equijoin rows and rows not retrieved from TABLE2, extended with null values. No unmatched rows from TABLE1 are included.

OUTER-DISJOIN(ED)

The output rows consist of only rows that would be left after an inner equijoin, extended with null values. The rows from TABLE1 are extended with nulls on the right and the rows from TABLE2 are extended with nulls on the left.

LEFT-DISJOIN(ED)

The active found set consists of only unmatched rows from TABLE1.

RIGHT-DISJOIN(ED)

The active found set consists of only unmatched rows from TABLE2.

Note: The use of OUTER JOIN, RIGHT-JOIN, or RIGHT-DISJOIN adds additional processing time to FIND statement processing. Outer joins and disjunctions cause the CA Dataquery optimization process to be bypassed. Any query containing an outer join or disjoin keyword is processed by accessing the tables in the order they appear in the query. Each table named on the "right side" must be read twice to accomplish the join. The use of LEFT-JOIN or LEFT-DISJOIN does not cause this additional processing.

Sample Table

The next sections describe and illustrate the outer join keywords using specific examples. All illustrations are based on the following three tables:

TABLEA		TABLEB		TABLEC	
A#	ACITY	B#	BCITY	C#	CCITY
A1	London	B1	London	C1	London
A2	Paris	B2	Oslo	C2	Rome
A3	?	B3	?	C3	?
A4	NY	B4	NY	C4	Madrid
A5	Madrid	B5	LA	C5	LA

The tables are related by the CITY fields. Null valued (empty) fields are indicated by a question mark (**?**).

When the RELATED BY clause is used to join output rows, only joined rows are produced. This is called an *inner equijoin*. For more information, see Relationship Clauses (see page 444).

The following query produces an inner equijoin of the tables ACITY and BCITY:

```
FIND ALL TABLEA RELATED BY ACITY VIA BCITY TO TABLEB
```

This is what the inner equijoin looks like:

A#	ACITY	B#	BCITY
A1	London	B1	London

A#	ACITY	B#	BCITY
A4	NY	B4	NY

Full (Symmetric) Outer Join

To preserve the rows in both tables that cannot be joined with RELATED BY requires an outer join. Each unjoined row is extended with null values where data would have appeared had a match been found. The following query produces a full outer join:

```
FIND ALL TABLEA OUTER-JOINED BY ACITY VIA BCITY TO TABLEB
```

The outer joined output shows an output row that has been extended with nulls (shown as **?**) because no match was found.

A#	ACITY	B#	BCITY
A1	London	B1	London
A2	Paris	?	?
A3	?	?	?
A4	NY	B4	NY
A5	Madrid	?	?
?	?	B2	Oslo
?	?	B3	?
?	?	B5	LA

To extend the full outer equijoin to three tables, first join tables A, B, and C using ACITY = BCITY and ACITY = CCITY as in the following query.

When joining three or more tables, it may be best to use a key from a table other than the last named table. In this case, the table name can be repeated before the keyword OUTER-JOINED. For example:

```
     FIND ALL   TABLEA
      OUTER-JOINED BY  ACITY VIA  BCITY TO  B
TABLEA OUTER-JOINED BY  TABLEA.ACITY  VIA  CCITY  TO  C
```

This is illustrated by the following table:

A	ACITY	B	BCITY	C	CCITY
A1	London	B1	London	C1	London
A2	Paris	?	?	?	?

A	ACITY	B	BCITY	C	CCITY
A3	?	?	?	?	?
A4	NY	B4	NY	?	?
A5	Madrid	?	?	C4	Madrid
?	?	B2	Oslo	?	?
?	?	B3	?	?	?
?	?	B5	LA	?	?
?	?	?	?	C2	Rome
?	?	?	?	C3	?
?	?	?	?	C5	LA

If the outer join of the tables A, B, and C had been done with ACITY = BCITY and BCITY = CCITY, the resultant table would have been:

A	ACITY	B	BCITY	C	CCITY
A1	London	B1	London	C1	London
A2	Paris	?	?	?	?
A3	?	?	?	?	?
A4	NY	B4	NY	?	?
A5	Madrid	?	?	?	?
?	?	B2	Oslo	?	?
?	?	B3	?	?	?
?	?	B5	LA	C5	LA
?	?	?	?	C2	Rome
?:	?:	?:	?:	C3:	?
?	?	?	?	C4	Madrid

Left Outer Join

The left outer equijoin of tables A and B using ACITY = BCITY would be the union of the inner join of A and B and the unused rows of A extended with null values. Following is a query that would produce a left outer join:

```
FIND ALL  tableA  LEFT-JOIN(ED) BY keyA VIA keyB TO tableB
```

Following is the active found set:

A	ACITY	B	BCITY
A1	London	B1	London
A2	Paris	?	?
A3	?	?	?
A4	NY	B4	NY
A5	Madrid	?	?

Right Outer Join

The following query produces a right outer join of tables A and B:

```
FIND ALL  tableA  RIGHT-JOINED BY keyA VIA keyB TO tableB
```

The right outer join of tables A and B with ACITY = BCITY produces the active found set below:

A#	ACITY	B#	BCITY
A1	London	B1	London
?	?	B2	Oslo
?	?	B3	?
A4	NY	B4	NY
?	?	B5	LA

Disjunction of Tables

The words *disjoin* or *disjunction* describe the result of an outer join with the rows of the inner join removed. The following query produces a left disjoin:

```
FIND ALL tableA LEFT-DISJOINED BY keyA VIA keyB TO tableB
```

For example, the left disjoin of tables A and B using ACITY = BCITY produces this active found set:

A	ACITY	B	BCITY
A2	Paris	?	?
A3	?	?	?
A5	Madrid	?	?

The preceding sample represents an active found set resulting from the left outer join of A and B with rows of the inner join removed. A right disjoin of tables A and B could be similarly defined.

The following query produces a full outer disjoin:

```
FIND ALL tableA OUTER-DISJOIN(ED) BY keyA VIA keyB TO tableB
```

The full outer disjoin of tables A and B using ACITY = BCITY results in the following active found set:

A	ACITY	B	BCITY
A2	Paris	?	?
A3	?	?	?
A5	Madrid	?	?
?	?	B2	Oslo
?	?	B3	?
?	?	B5	LA

Adding Selection Criteria

Purpose

When you want to relate tables to one another but only select certain rows, add a WITH clause to the relationship statement. The WITH clause specifies which rows should be selected to make up the active found set.

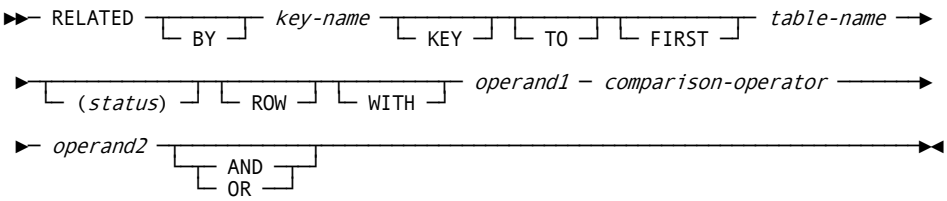
Operation

The selection criteria can consist of multiple conditions which serve to limit the number of rows to those which meet all conditions. See section WITH Clauses (see page 504) for more information.

You can decide not to include any selection criteria. This causes CA Dataquery to select all the rows in the specified table. However, this can cause the query selection process to run for a long period of time due to CA Dataquery selecting a potentially large amount of rows, depending on the size of the table involved.

Syntax Diagram

The following shows the syntax of RELATED BY when specifying selection criteria.



WITH-clause

Represents one or more logical expressions which specify the criteria on which CA Dataquery is to select rows from the database. A WITH clause contains the criteria which qualify the rows. Only those rows which meet the specified criteria in the WITH clause are selected.

See Simple Relationship with a Common Key (see page 446) for an explanation of the other keywords in the RELATED BY syntax.

Equijoin Examples

The following example shows how selection criteria works with a RELATED BY statement that uses a common key.

Query

The following sample query relates two tables, CAI-CUST-REC and CAI-ACCTS-REC using the common key CUST-ID-KEY and selecting only rows where ORD-AMT is greater than or equal to 2000.

```
FIND ALL CAI-CUST-REC ROWS
 RELATED BY CUST-ID-KEY TO CAI-ACCTS-REC ROWS
  WITH ORD-AMT GTE 2000.00
PRINT FROM CAI-CUST-REC CUST-ID-KEY
                         NAME
                         PHONE
                         CRED-IND
      FROM CAI-ACCTS-REC ORD-ID
                         ORD-AMT
                         FRT-AMT
                         PAYMT-AMT
```

Explanation

CA Dataquery joins rows from the two tables whenever the value in the common key, CUST-ID-KEY, matched. Now that the two tables are related, we can access information for customers from both tables with one query.

Joining the two tables creates a new *temporary table* during query execution. Only rows containing the specified selection criteria were selected. This temporary table is called the *active found set.*

Output

Following is the first page of output from the preceding illustrated query and active found set.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1A
14:15:01                                                                   DETAIL

  CUST-ID-KEY  NAME                            PHONE       CRED-IND  ORD-ID
  -----------  ------------------------------  ----------  --------  ------
  A0030        CANNON TOOLS CO                 4045553322  A          01011
  A0030        CANNON TOOLS CO                 4045553322  A          01223
  G0090        INTERNATIONAL BANK CORP.        2125559889  G          05999




 ------------------------------- MORE .... -------------------------------- =>
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Scrolling right with PF12 displays page 1B containing the remaining columns requested by the sample query.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1B
14:15:01                                                                   DETAIL

      ORD-AMT      FRT-AMT  PAYMT-AMT
  -----------  -----------  ---------
  0004250.00   0000425.00   04547.50
  0032950.50   0003295.00   34597.97
  0002000.00   0000200.00   00000.00




 <= ------------------------------ MORE .... -----------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

In the previous figure, the Cannon Tool Company has two entries in the CAI-ACCTS-REC table. In the active found set, there are also two separate *rows* for Cannon Tool Company.

Combining Relationship Types

You can relate tables by combining simple and complex joins in one of three ways. Choose the RELATED BY clause when combining relationship types. The following describes the methods.

Chained relationship

Establishes a series of relationships through common keys or columns. Can combine simple and complex joins.

Repeating relationship

Establishes relationships by repeating the name of a table as the subject of a clause. Can combine simple and complex joins.

Both chained and repeating

Establishes relationships by combining both of the preceding types of joins.

Deciding Which Type of Relationship to Use

Sometimes the same relationship can be stated in either a chained or repeating fashion. Both types of relationships join unrelated tables. The following example demonstrates the use of both types of relationships to accomplish the same objectives.

```
Chained:
TABLE-A RELATED KEY1 TO TABLE-B RELATED KEY2 TO TABLE-C
Repeating:
TABLE-B RELATED KEY1 TABLE-A AND TABLE-B RELATED KEY2 TO TABLE-C
```

Some relationships can only be stated in a repeating fashion. For example:

```
TABLE-A RELATED KEY1 TABLE-B RELATED KEY2 TABLE-C AND TABLE-B
RELATED KEY3 TABLE-D
```

Whenever you have a table that joins two or more other tables, you must repeat the table name, hence the term *repeating relationship.*

Chained Relationships

Purpose

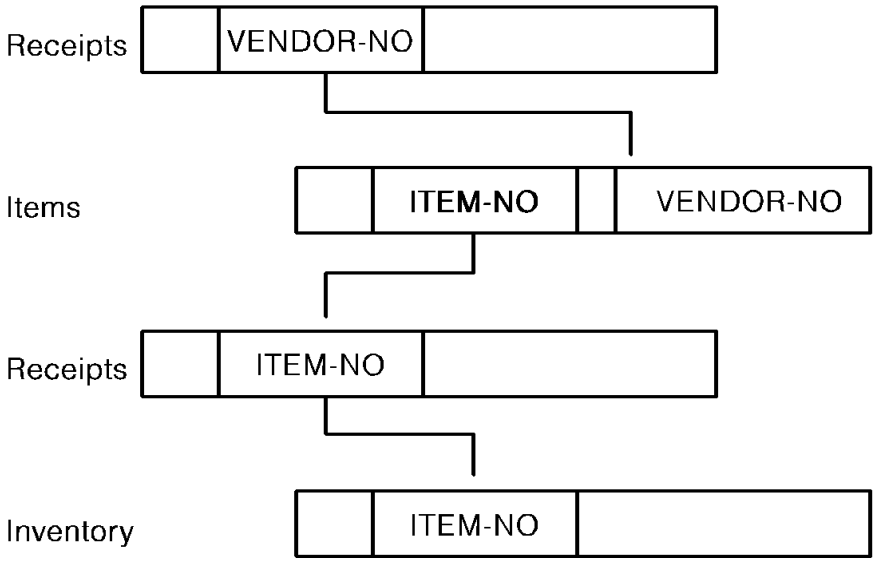
You can join tables to one another in a series by establishing a series of relationships through common keys or columns. The common key for one relationship can be different from the common key for another relationship in the series. The relationships you establish can be a combination of simple and complex joins.

Example

The following query shows a chained relationship among four tables. The diagram illustrates the relationship.

```
FIND ALL VENDOR ROWS
  RELATED BY VENDOR-NO TO ITEMS
  RELATED BY ITEM-NO TO RECEIPTS
  RELATED BY ITEM-NO TO INVENTORY
```

Chained Relationship Example



Repeating Relationships

Purpose

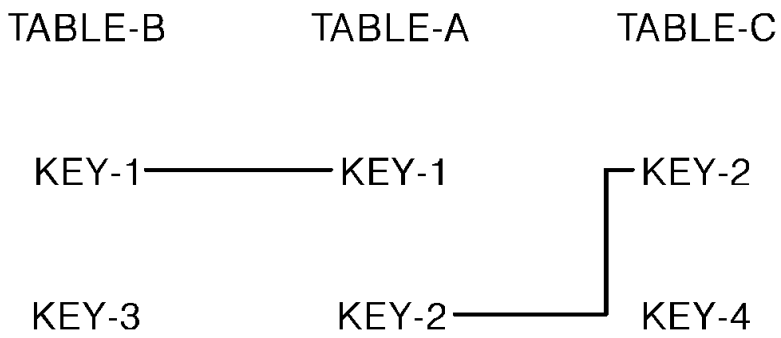
A repeating relationship occurs when the name of a table is repeated as the subject of a relationship. In general, a repeating relationship should be used when the subject of the relationship wanted is not the object of the last relationship specified. The relationships established can be a combination of complex and simple joins. For example, A is related to B and A is related to C; therefore, A contains two different keys or columns which are common to either B or C.

Example

The following example shows how TABLE-B can be related to TABLE-C through their relationship to TABLE-A.

Repeating Relationship Illustration

Figure 1: Figure shows repeating relationships



Example

The following FIND statement uses the common key CUST-ID-KEY to connect the CAI-CUST-REC table with the CAI-DETAIL-REC table and the CAI-ORDERS-REC table in a repeating relationship.

```
FIND 10 CAI-CUST-REC ROWS
  RELATED BY CUST-ID-KEY TO
     CAI-DETAIL-REC ROWS
  AND CAI-CUST-REC RELATED BY CUST-ID-KEY
     TO CAI-ORDERS-REC ROWS
        WITH ORD-YR = '87'
```

Explanation

The preceding example contains two RELATED BY statements. This repetition indicates that a relationship must exist between more than two tables before a logical row is selected. In the example, the relationship of three tables is established.

In the preceding sample FIND statement, a relationship must exist between CAI-CUST-REC and CAI-DETAIL-REC tables before a row is chosen. This relationship is established through the common key, CUST-ID-KEY.

For example, if a CAI-CUST-REC row, a CAI-DETAIL-REC row and a CAI-ORDERS-REC row all exist in their respective table types with a CUST-ID-KEY of 01009, all the relationships qualify. When edited and executed, the preceding query statement can yield information from each of the named tables. However, the relationship between CAI-DETAIL-REC rows and CAI-ORDERS-REC rows in the example preceding does not need to specify the same key name as specified in the first relationship. The second relationship can specify an entirely different key.

Example

The following sample illustrates a repeating relationship.

```
FIND 10 CAI-DETAIL-REC ROWS
RELATED BY ITEM-ID-KEY TO CAI-ITEMS-REC ROWS AND
CAI-DETAIL-REC ROWS RELATED BY ORD-ID-KEY TO CAI-ACCTS-REC ROWS
    WITH ORD-AMT = 0
```

Explanation

In the preceding example, CAI-DETAIL-REC and CAI-ITEMS-REC tables are related by their common key ITEM-ID-KEY while CAI-DETAIL-REC and CAI-ACCTS-REC tables are related by their common key ORD-ID-KEY.

In a repeating relationship, the first table is repeatedly joined, or related, to each succeeding table. Each table is linked with the first table based on the value in a common column or key. However, the same column or key need not be used to link the first table with all the others.

Example

The following DQL statements show a repeating relationship of four tables:

```
FIND ALL CUSTOMER ROWS
  RELATED BY CUST-ID TO ACCOUNTS ROWS
    AND CUSTOMER ROWS RELATED BY CUST-ID TO ORDERS ROWS
    AND CUSTOMER ROWS RELATED BY VENDOR-NO TO VENDORS ROWS
```

Combining Chained and Repeating Relationships

You can combine chained and repeating relationships, as in the following example.

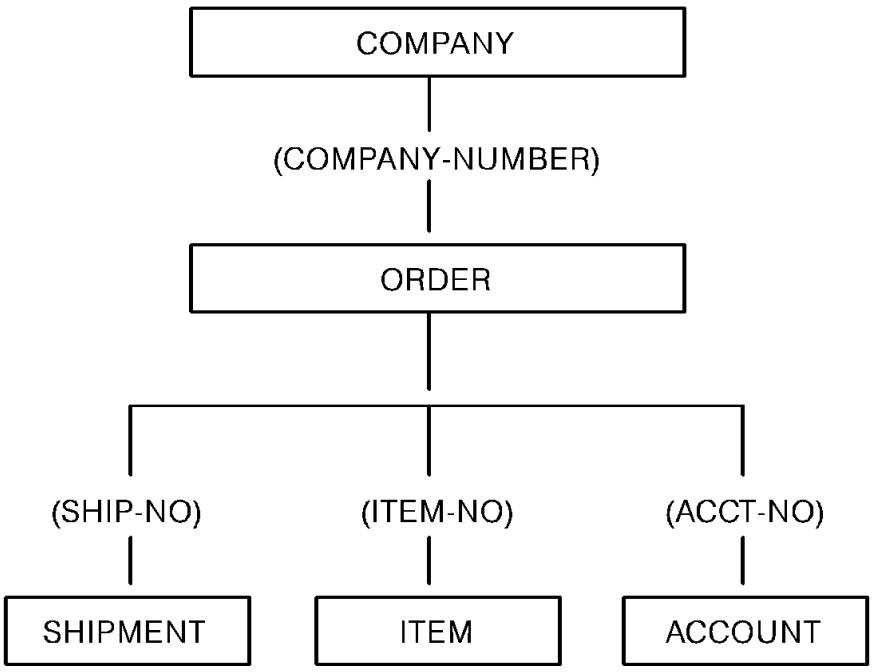
Example

The FIND statement below uses both relationships to join five different tables. The portion of the FIND statement in the first box is the selection criteria. The second box contains the chained relationship and the third box contains the repeating relationship.

```
 FIND ALL COMPANY ROWS
  WITH COMPANY-NAME = 'CA'
RELATED BY COMPANY-NUMBER TO ORDER ROWS
     AND
    RELATED BY SHIP-NO TO SHIPMENT ROWS
     WITH SHIP-CODE = 'INTERSTATE'
AND
    ORDER ROWS RELATED BY ITEM-NO TO ITEM ROWS
     WITH ITEM-CLASS = 'FLAMMABLE'
     AND
     ORDER ROWS RELATED BY ACCT-NO TO ACCOUNT ROWS
```

This diagram shows how chained and repeating relationships in the preceding query relate five tables using four keys.

Chained Relationship Example



Example

The following example uses both chained and repeating relationships. In the sample query, the CAI-DETAIL-REC, CAI-ACCTS-REC and CAI-ITEMS-REC tables are chained together while the CAI-ACCTS-REC table is repeated to connect with CAI-SLSHST-REC tables.

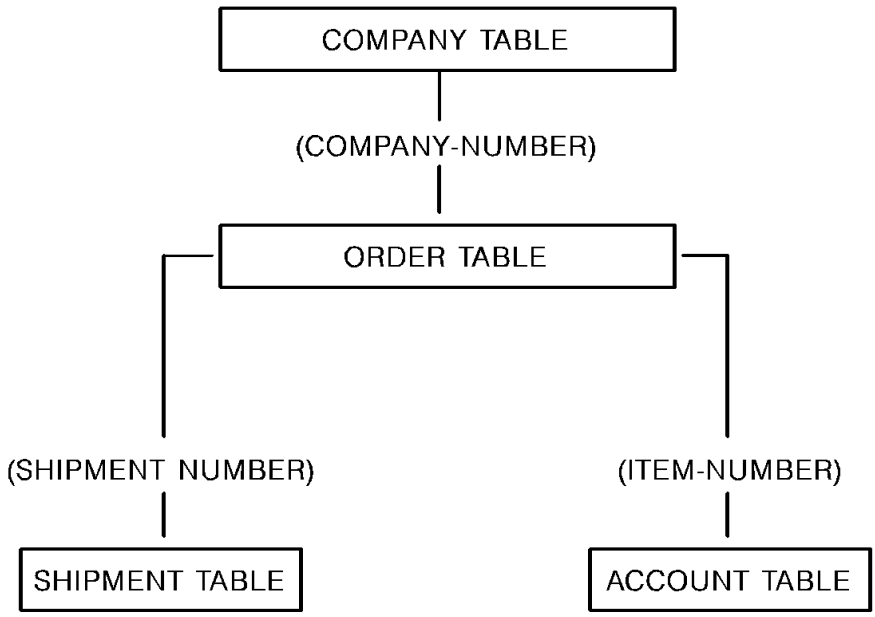
```
FIND ALL CAI-DETAIL-REC ROWS
     WITH ACT-YR = '87'
  RELATED BY ORD-ID-KEY TO CAI-ACCTS-REC ROWS AND
  RELATED BY ITM-ID-KEY TO CAI-ITEMS-REC ROWS
     WITH B-O-QTY ,= 0  AND
  CAI-ACCTS-REC ROWS RELATED BY CUST-ID-KEY TO
     CAI-SLSHST-REC ROWS
     WITH SLMN-ID = '14830'
```

In this example, the CAI-SLSHST-REC table does not connect directly to the CAI-DETAIL-REC table, but their rows can be related by the use of the intermediate table CAI-ACCTS-REC.

Example

The following diagram shows how three different keys can relate four tables in spite of the fact that the four tables do not share a single common key.

Chained Relationship Example with Three Keys



SET Statements

Purpose

You can use the SET statement in a query to do the following:

- Perform arithmetic calculations on the data found by a query.
- Access user-defined functions.
- Perform calculations which make use of values from database tables and from the result created by other SET statements.

Operation

A SET statement must immediately follow the FIND statement or another SET statement. To determine how many SETs can be in a query, use the following formula:

```
(18 x Number of Tables Joined) + (Length of SORT key)
+ (17 x Number of SET Statements) + 6
= (A Number That is Less Than 760)
```

There is an absolute maximum of 43 SET statements for a query with a FIND on one table and no SORT statement.

When you use a SET statement in a query, you can create a temporary result and assign it a name and a value. You then can use the result during sorting or reporting in the same way you use a database column.

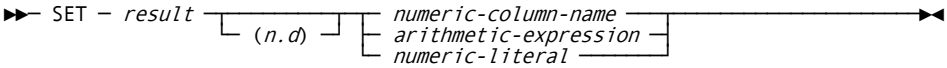
When a result is first named in a SET statement, CA Dataquery assigns it certain characteristics or attributes. Initially, it is numeric and allows 13 digits, a decimal point, two decimal places, and a plus or minus sign. The default precision can be changed by means of the precision (n.d) entry. The initial value of a result is zero.

After establishing the final value of the SET, CA Dataquery moves the value to the result based on the result's precision (the number of digits before and after the decimal point). CA Dataquery then rounds and truncates the decimals to comply with the precision. If the final value is too large for the result, an overflow can result in an error when CA Dataquery processes the PRINT or DISPLAY statement.

To create a temporary result with SET, specify the desired name and assign a value to that name. The value to which you set the result can be a value of another numeric column, the result of one or more arithmetic operations, a specific numeric value, or any combination of these.

Syntax Diagram

The following shows the syntax of the SET statement.



Repeat statement once for each result wanted.

The following sections discuss each option of the SET statement.

Numeric columns

Next section.

Arithmetic expressions

Page Arithmetic Result Syntax Diagram (see page 473).

Numeric literals

Section Numeric Literal Syntax Diagram (see page 479).

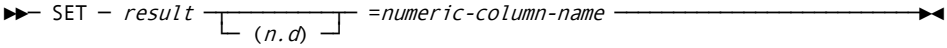
Numeric Column Syntax Diagram

Purpose

You can set a temporary result to the value of a numeric column.

Syntax Diagram

The following is the syntax for the SET statement when you want to set the result to a numeric column.



Explanation

The following explains the keywords in the preceding format.

result

Assigns a name to the temporary result. A result name can be up to 32 alphanumeric characters. The first character of the name must be alphabetic.

(n.d)

Specifies the precision (the specific number of digits before and after the decimal) of the result, overriding the site default precision. (n.d) indicates that the result contains n integers and d decimals. The total of n + d cannot exceed 15 digits.

For example, (9.2) specifies a result with nine integer positions and two decimal positions. The total number of digits is 11.

numeric-column name

Indicates the name of a result in another SET statement or a non-floating-point numeric column defined to Datadictionary as part of a database table referenced in your query.

The letters N, B and D in the Datadictionary report and the letter N in the TYPE column of the extended column display indicate a numeric column.

If the column is from a table, as opposed to a temporary (SET-created) result, CA Dataquery assumes it is contained within the last table named in the query. If you want to specify a column within a table other than the last table named, you must qualify the column name by preceding it with the name of the table, for example, PAYROLL ACTIVITY-CODE. CA Dataquery assumes that each column specified after this qualifying table name is contained in that table.

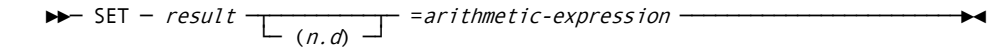
Arithmetic Result Syntax Diagram

Purpose

You can set the value of a temporary result to be the value that results from an arithmetic expression.

Syntax Diagram

The following is the syntax for the SET statement when you want to set the result to equal the value resulting from an arithmetic expression.



Explanation

The following explains the keywords in the preceding format.

result

Assigns a name to the new result. A result name can be up to 32 alphanumeric characters, with an alphabetic first character.

(n.d)

Specifies the precision (the specific number of digits before and after the decimal) of the result, overriding the default precision. (n.d) indicates that the result contains n integers and d decimals. The total of n + d cannot exceed 18 digits.

For example, (9.2) specifies a result with nine integer positions and two decimal positions. The total number of digits is 11.

arithmetic-expression

Specifies one or more arithmetic operations to be performed on selected data, including null-valued columns. If a computation is done involving null values, the result is null. CA Dataquery evaluates the expression according to the hierarchy of operations. (For more information, see Evaluation of Arithmetic Expressions (see page 477).)

Sample

This is a sample query which illustrates SET statements using arithmetic expressions.

```
FIND CAI-SLSHST-REC ROWS
    WITH SHIP-QTY GTE 200
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          NET-COST
                          DISCOUNT
                          DISCOUNT-COST
```

Sample

The CAI-SLSHST-REC table looks like this:

SHIP -QTY	ORD -DT	CUST -ID	SLMN -ID	DISC -PCT	IND -CD	UNIT -PRICE	ITM -ID
103	880117	QW10000	SMITH	0.10	XZ-234	1.25	CAPS
400	880319	RM10000	JONES	0.10	AM-789	7.00	MATS
200	880215	UV9999	WATT	0.10	XZ-744	3.00	LOCKS
167	890117	RV8009	ROBB	0.05	AA-909	2.37	DIAL
300	890215	XY3375	ADAMS	0.10	ZM-889	5.00	VISOR

Sample

The CAI-ORDERS-REC table looks like this:

ORD -ID	ORD -TOT	CUST -PO	SLMN -ID	SHIP -ID	ITM -ID	FRT -TOT	GROUP
BR950	2595.00	RT900	JONES	LMN-12-1	MATS	75.00	AMA
CQ322	573.99	RT907	WATT	LMN-33-7	LOCK	33.99	AMA
RR995	17.50	RT899	SMITH	ALV-01-1	CAPS	1.25	AMA
AW009	1433.00	RT999	ADAMS	ARR-33-9	VISOR	83.00	CRA

Sample

The active found set for the preceding query looks like this:

ORD -ID	ITM -ID	SHIP -QTY	UNIT -PRICE	NET -COST	DISCOUNT	DISCOUNT -COST
BR950	MATS	400	7.00	2800.00	280.00	2520.00
CQ322	LOCK	200	3.00	600.00	60.00	540.00
AW009	VISOR	300	5.00	1500.00	150.00	1350.00

Arithmetic Expressions in SET

An arithmetic expression can include the following data items:

- Numeric columns from the database
- Results from previous SET statements
- Numeric literals

The following are valid arithmetic operators.

Function	Entry	Description
Add	+	Adds the specified numeric operands.
Subtract	-	Subtracts the specified numeric operands.
Divide	/	Divides the specified numeric operands (divisor cannot be 0).
Multiply	*	Multiplies the specified numeric operands.

Function	Entry	Description
Parenthetical expression	( )	Clarifies which operations are to be performed first in a complex calculation (can specify up to 5 levels of nested parentheses).
Unary +	+	Indicates that the specified numeric operand is a positive value (must precede the numeric data without a space between the symbol and the data).
Unary -	-	Indicates that the specified numeric operand is a negative value (must precede the numeric data without a space between the symbol and the data).

A space must precede and follow each operator. However, no space follows a unary arithmetic operator (an operator accompanying a single value). For example, if your data includes negative numbers, the minus sign (-) in front of a number is the unary operator, such as in -2.

An arithmetic expression can begin only with one of the following:

- Left parenthesis
- Unary +
- Unary -
- Numeric constant
- Numeric column name

An arithmetic expression can end only with one of the following:

- A right parenthesis
- A numeric constant
- A numeric column-name

Evaluation of Arithmetic Expressions

CA Dataquery evaluates arithmetic expressions according to the arithmetic operator it finds. This standard processing sequence is called the hierarchy of operations and is performed in the following order.

1. Expressions within parentheses (innermost parentheses first)
2. Expressions within parentheses (same level of parentheses, performing from left to right)
3. Mathematical functions
4. Unary + and unary -
5. * and /
6. + and -

CA Dataquery evaluates expressions within parentheses first. When expressions are contained within nested parentheses, evaluation proceeds from the least inclusive (innermost) to the most inclusive (outermost) set.

When the order of consecutive operations on the same hierarchic level is not completely specified, the order of operation is left-to-right; for example, the following two expressions are evaluated the same.

```
A + B / C - D / C * F + G * C / H + I

A + (B/C) - ((D / C) * F) + ((G * C) / H) + I
```

Note: There must be a one-to-one correspondence between left and right parentheses in an arithmetic expression.

Sample

The result of an arithmetic expression is dependent on the precision of the result and the arithmetic operation which is performed. The following example illustrates how CA Dataquery calculates the value for FLDX, using the following expression.

```
FLDX (3.3) = ((2. + 3) * (4 + (5 / 6 + (7 - 8)))) + 9
```

Explanation

The following chart shows the hierarchy of operations, the part of the expression which is evaluated in each step, and the result of that evaluation. After each step, we show the expression with the value from the just completed step replacing the part that was evaluated.

Step	Part	Result	Comments
1	(7 - 8)	-1	CA Dataquery evaluates the innermost parenthetical expression first.
		FLDX (3.3) = ((2. + 3) * (4 + (5 / 6 + (-1)))) + 9	
2	5 / 6	0.833	CA Dataquery evaluates the part of the inner parenthetical expression involving division, since division has a higher precedence than addition.
		FLDX (3.3) = ((2. + 3) * (4 + (0.833 + (-1)))) + 9	
3	(0.833 + (-1))	-.167	CA Dataquery uses the results from the evaluation in Steps 1 and 2 to evaluate the second innermost parenthetical expression.
		FLDX (3.3) = ((2. + 3) * (4 + (-.167))) + 9	
4	(2 + 3)	5	The two remaining inner parenthetical expressions both involve addition and are at the same level, so CA Dataquery evaluates the left expression first.
		FLDX (3.3) = ((5) * (4 + (-.167))) + 9	
5	(4 + (-.167))	3.833	CA Dataquery evaluates the inner parenthetical expression on the right.
		FLDX (3.3) = (5 * 3.833) + 9	
6	(5 * 3.833)	19.165	CA Dataquery completes the evaluation of the multilevel parenthetical expression by multiplying the values within parentheses of FLDX, (3.3), designates the number of digits CA Dataquery calculates on either side of the decimal point. The precision (3.3) indicates that the value has three digits to the left and right of the decimal point.
		FLDX (3.3) = 28.165	

You can also define FLDX with a precision different from (3.3), as in the following alternative formats:

```
FLDX (7.5) = +0000028.16665
FLDX (2.4) = +28.1665
FLDX (2.0) = +28
FLDX (1.0) = ***E3***  (Display indicates error because of overflow.)
```

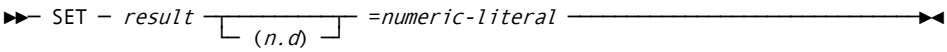
Numeric Literal Syntax Diagram

Purpose

You can set a temporary result to be a specific numeric value. This is often useful in performing calculations.

Syntax Diagram

The following is the syntax for the SET statement when you want to set the result to a numeric literal value.



Explanation

The following explains the preceding keywords.

result

Assigns a name to the new result. A result name can be up to 32 alphanumeric characters. The first character of the name must be alphabetic.

(n.d)

Specifies the precision (the specific number of digits before and after the decimal) of the result, overriding the default precision. (n.d) indicates that the result contains n integers and d decimals. The total of n + d cannot exceed 18 digits.

For example, (9.2) specifies a result with nine integer positions and two decimal positions. The total number of digits is 11.

numeric-literal

Assigns a numeric constant value to the result. A unary plus (+) or minus (-) sign can precede the value. A numeric-literal can be up to 18 numbers (integers and decimals).

Note: If DECPT=COMMA in the System Option Table, the numeric literal must be enclosed in quotes. See your CA Dataquery Administrator with any questions about system options.

Sample

The name assigned to a result can be used in subsequent statements as shown in the following example.

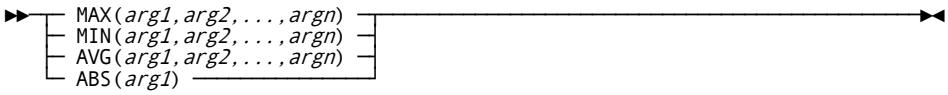
```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-ORDERS-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT-PCT = .06
SET DISCOUNT (5.2) = NET-COST * DISCOUNT-PCT
SET DISCOUNT-COST (8.2) = NET-COST - DISCOUNT
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       NET-COST
       DISCOUNT
       DISCOUNT-COST
```

Mathematical Functions in SET

CA Dataquery provides mathematical functions which help to simplify certain arithmetic operations. Two basic types of functions exist, standard and user-defined. Both types work with column values at the row level and not with a mixture of column values and column totals.

Standard Functions

CA Dataquery provides the following standard functions for use in a SET statement of a query.



Description

MAX*(arg1,arg2,...,argn)*

Returns the highest value of a list of arguments.

MIN*(arg1,arg2,...,argn)*

Returns the lowest value of a list of arguments.

AVG*(arg1,arg2,...,argn)*

Returns the average value of a list of arguments.

ABS(arg1)

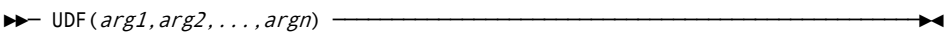
Returns the absolute value of the argument.

The argument list in parentheses must immediately follow the function name with no intervening blanks. An argument can be a literal value, numeric database column or the result from a previous SET statement.

User-Defined Functions

If you need a specific mathematical function which CA Dataquery does not provide as a standard function, site management can create a user-defined function (UDF) to meet your request.

The syntax diagram for the user-defined function follows:



The first argument (*arg1*) is a numeric value which represents the number assigned to a particular mathematical function by site management. For example, if 14 functions have been defined, *arg1* could be 10 and the arguments *arg2* through *argn* would specify result names, column names or values for numeric literals to be used in the specified function.

Sample

This sample SET statement shows the use of a user-defined function as part of a calculation.

```
SET A = (FLDB * FLDC) / AVG(X,Y,Z) + UDF(7,X,Y,Z)
```

In the example preceding, 7 represents the number of the mathematical function established by site management that you want to use. X, Y and Z specify the result name, column name or a numeric literal to be used in the mathematical function assigned to 7. The result A equals the result of the calculations performed in the UDF expression and the other expressions in the SET statement.

Using Multiple SET Statements

The SET statement performs a variety of arithmetic operations, allowing you to arrange calculations in several steps. In the following example, a series of SET commands makes use of columns and numeric constants to define a group of simple equations.

```
SET RATE (5.2) = 10.75
SET VALUE (2.2) = 14.22
SET INTRT (2.5) = 14.34721
SET A (2.5) = AVG(14,9,6)
SET B (6.0) = (-VALUE * 100)
SET SUBTOT (4.5) = (INTRT + 7) / (-16)
SET AMT (4.4) = (RATE / INTRT) + (.6392 * VALUE)
SET SUM (4.4) = (RATE + A - B) + (SUBTOT + AMT / 5)
```

After each SET statement is calculated, its value can then be used in the next calculation until the last result has been evaluated. The following represent the results obtained from the previous calculations.

```
A = 09.66667
B = -001422
SUBTOT = -0001.33420
AMT = 0009.8387
SUM = 1443.0502
```

The final result SUM contains the value calculated from the last equation involving values from the other SET statements. Such use of nested or connected SET statements permits you considerable flexibility in performing calculations, especially when the standard operators and functions are employed in conjunction with user-defined functions.

SORT Statements

Purpose

The SORT statement allows you to organize the rows selected with the FIND statement. You can include a SORT statement in your query to list the selected rows in a specific order based on the values contained in selected columns.

Operation

With the SORT statement, you can order found set rows in ascending or descending order according to the contents of one or more columns. In addition, you can specify multiple SORT control-columns to allow sequencing of a group within a group; for example, you can sequence a group of customers by locale and then alphabetically by name within that locale.

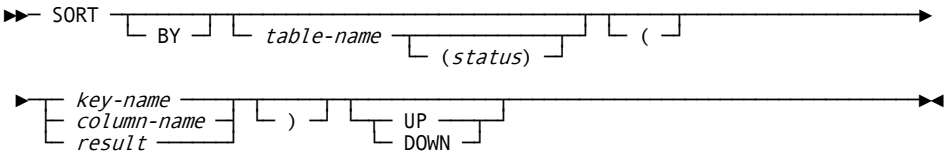
You can designate control break columns in the SORT statement to cause columns designated for accumulation in the PRINT statement to be subtotaled whenever a value in a control break column changes. You can only specify accumulation of numeric columns in a PRINT statement. (See Designating Control Breaks (see page 489) for more information on accumulation of columns. See Totals Syntax Diagram (see page 438) for more information on printing totals.)

The total length of the sort key cannot exceed 100 characters. The sort key length is equal to the sum of:

- Seventeen (17) for each temporary result named in the SORT statement
- The length of each database table column
- One (1) for each signed database table column

Syntax Diagram

The following syntax diagram shows the keywords in a SORT statement.



Repeat the parameters once for each SORT key or column name.

Explanation

The following explains the keywords in the SORT statement.

BY

Used for readability only.

table-name

Specifies the Datadictionary defined table name with which the specified control-column is associated. When the column to be sorted is a temporary result from a SET statement, a table name should not be used since the result is not associated with any table.

If the control-column to be sorted belongs to the table most recently named in the query, repeating the table name is not necessary. However, if the control-column belongs to another table specified in the query, you must qualify the control-column with the corresponding table name.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following chart describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

key-name

Represents the name of the key which CA Dataquery uses to sort the found set. When this is a key, it must be contained within one of the tables specified in the FIND statement of the query. If a sort column is named UP, ASC, ASCENDING, DOWN, DESC or DESCENDING, use an alias for the column in the SORT statement. (An alias, or nickname, is assigned in Datadictionary to give an additional name to a column, key, or table.) Parentheses enclosing a SORT key or column name designate a control break. See Designating Control Breaks (see page 489) for more information.

column-name

Represents the name of the column which CA Dataquery uses to sort the found set. If named UP, ASC, ASCENDING, DOWN, DESC or DESCENDING, use an alias for the column in the SORT statement. (An alias, or nickname, is assigned in Datadictionary to give an additional name to a column, key, or table.) Parentheses enclosing a SORT key or column name designate a control break. See Designating Control Breaks (see page 489) for more information.

result

Represents the name of the result of a previous SET statement which CA Dataquery uses to sort the found set.

UP

Indicates to arrange the row collection in an ascending direction based on the value of the control-column. If you do not specify the direction, CA Dataquery defaults to UP. You also can use the word ASCENDING or ASC to represent this direction.

DOWN

Indicates to arrange the row collection in a descending direction based on the value of the control-column. You also can use the word DESCENDING or DESC to indicate this direction.

Sample

The following query illustrates sorting in an ascending direction.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
    RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY ORD-ID UP
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       NET-COST
       DISCOUNT
```

Output

The following screen is sample output generated from executing this query.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1
16:52:32                                                                   DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  ----------  --------  ----------  -------------  ---------
   08811  O50006      0000203    00014.75    00002994.25   00010.00
   08811  O60005      0000003    00004.33    00000012.99   00002.12
   08811  C70006      0001293    00010.00    00001293.00   00015.34
   09031  C10006      0000011    00006.00    00000066.00   00003.99
   09722  A90011      0000302    00003.00    00000906.00   00012.00
   12810  O50006      0000100    00014.75    00001400.00   00005.23









 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Sample

The following query and output illustrates using SORT to order data in a descending direction.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY ORD-ID DOWN
PRINT FROM CAI-ORDERS-REC ORD-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       NET-COST
       DISCOUNT
```

Output

The following screen is sample output from executing the preceding query.

```
 =>

01/02/2010                  CA Dataquery                              PAGE 1
16:52:32                                                                   DETAIL

  ORD-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  ----------  --------  ----------  -------------  ---------
   12810  O50006      0000100    00014.75    00001400.00   00005.23
   09722  A90011      0000302    00003.00    00000906.00   00012.00
   09031  C10006      0000011    00006.00    00000066.00   00003.99
   08811  C70006      0001293    00010.00    00001293.00   00015.34
   08811  O60005      0000003    00004.33    00000012.99   00002.12
   08811  O50006      0000203    00014.75    00002994.25   00010.00









 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Sorting Groups Within Groups

To sort a group within a group, include additional sort control-columns. The data is sorted according to the order in which the sort columns are specified.

Sample

For example, the SORT statement in the following sample specifies ORD-ID and SHIP-ID as the two sort columns. CA Dataquery first sorts the ORD-ID data and then sorts by SHIP-ID within each order.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY ORD-ID AND SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
       SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       NET-COST
       DISCOUNT
```

Output

The following sample output is generated from executing the preceding query.

```
 =>

01/02/2010                          CA Dataquery                    PAGE 1
18:11:54                                                                DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   08811  00001    O50006      0000203    00014.75    00002994.25   00010.00
   08811  00002    O70001      0000002    00010.00    00000020.00   00010.00
   09012  00007    C00006      0000100    00004.00    00000400.00   00002.00
   09013  00003    O50006      0000112    00100.00    00001112.00   00002.01









 -------------------------------- LAST PAGE ---------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT           <PF12> RIGHT
```

Designating Control Breaks

When a column in a SORT statement is enclosed in parentheses, it becomes a control break. Whenever the value contained in a control break column changes, a control break occurs.

Purpose

The primary use for control breaks is to specify when subtotals will be displayed for columns that have been designated for accumulation in the PRINT statement and to trigger processing of special functions defined in the WHEN/DO clause. (See Specifying Functions at Control Breaks (see page 493) for more information.) When the PRINT statement is executed, a subtotal for each accumulation column is presented whenever the value changes in the control break. After all data has been presented, a grand total appears as the last line of output.

You can also use control breaks to make reports easier reading. If the SUPPRESS DUPLICATE COLUMNS options is set to YES in your User Profile, the value contained in the control break is displayed only once on the screen. The value appears on the first line in the column designated for the control break. The column remains blank until the value in the control break changes.

You can designate up to ten control breaks in the SORT statement. Control breaks and ordinary columns can be freely intermixed.

Sample

The following query illustrates the use of control breaks. The control break column is ORD-ID, which is enclosed in parentheses in the SORT statement. Accumulation columns are not specified in this example.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY (ORD-ID) AND SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
       SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       NET-COST
       DISCOUNT
```

Sample

The following screen illustrates the effect when the SUPPRESS DUPLICATE COLUMNS option is set to YES. The value in the control break is presented only one time and the report column remains blank until the value in the control break changes.

Results with Suppress Duplicate Columns Set to YES

```
 =>

01/02/2010                          CA Dataquery                    PAGE 1
16:59:12                                                                DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   08811  00001    O50006      0000203    00014.75    00002994.25   00010.00
          00002    O70001      0000002    00010.00    00000020.00   00002.33
   09012  00007    C00006      0000100    00004.00    00000400.00   00012.82
          00008    900006      0000100    00006.00    00000600.00   00002.82
          00007    070006      0000100    00010.00    00001000.00   00010.00
   09013  00008    O50006      0000112    00100.00    00001112.00   00002.01
   09014  00008    O50006      0000002    00100.00    00000200.00   00002.09









 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD       <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

Sample

The following query illustrates the use of control breaks to present subtotals for columns designated as accumulation columns in the PRINT statement. The accumulation column in this example is NET-COST and it is enclosed in parentheses in the PRINT statement.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-ORDERS-REC BY (ORD-ID) AND SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
       SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
       SHIP-QTY
       UNIT-PRICE
       (NET-COST)
       DISCOUNT
```

Output

When the preceding query is executed, the screen displays a subtotal for NET-COST each time the value of ORD-ID changes. The following screen shows the results of the query.

```
 =>

01/02/2010                          CA Dataquery                    PAGE 1A
17:03:11                                                                 DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   08811  00001    O50006      0000203    00014.75    00002994.25   08982.75
          00002    O50006      0000010    00014.75    00000147.50   00002.75

  TOTAL ORD-ID 08811
                                                    * 00003141.75

   09312  00012    O50006      0000100    00014.75    00001475.00   04432.75
          00015    O90001      0000010    00006.00    00000060.00   00012.75

  TOTAL ORD-ID 09312
                                                    * 00001535.00

* GRAND TOTAL
 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP          <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS     <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED       <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

After the word TOTAL, the name of the control break (ORD-ID) appears, followed by the value in the column. An asterisk precedes the accumulated value for the subtotal.

The last line of output presented is a total for the accumulated columns. The total is designated by * GRAND TOTAL and an asterisk preceding the accumulated value.

UPDATE Statement

You need special authorization to use the UPDATE statement in a query to add data to the database. See your CA Dataquery Administrator for authorization and to see the documentation in the *CA Dataquery Administrator* Guide.

WHEN/DO Statements

Purpose

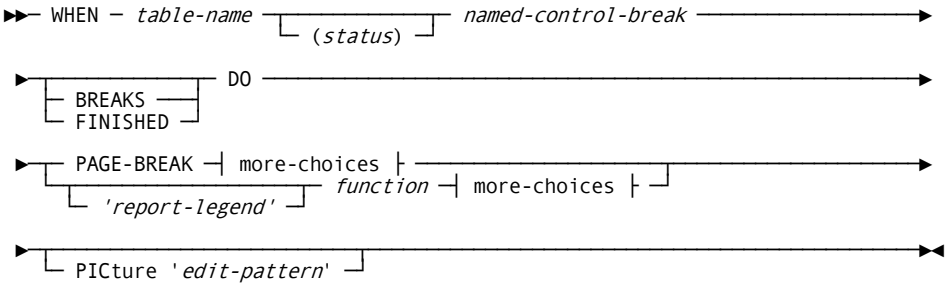
The WHEN/DO statement specifies when a mathematical function or page break is to be performed and what column will be the object of the mathematical function.

Since only one PRINT statement can be present in the query, the PRINT statement must be completed before the WHEN/DO statement is given. There can be multiple WHEN statements in the query, and there may be multiple DO statements for any WHEN statement.

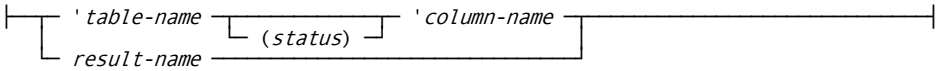
You can perform mathematical functions when a control break occurs, or at the end of the report. Mathematical functions which you can specify in the DO statement are AVG, CNT, MAX, MIN, and SUM. See Designating Control Breaks (see page 489) for more information.

Syntax Diagram

The following diagram shows the syntax of a WHEN/DO statement.



Expansion of more-choices



Repeat WHEN statement once for each function or page break wanted. Multiple DO statements may be used with one WHEN statement.

To simplify discussion, we present the options of the WHEN/DO statement in the following sections.

Mathematical functions at control breaks

Section Specifying Functions at Control Breaks (see page 493)

Mathematical functions at the end of report

Section End of Report (see page 497)

Specifying page breaks in the output

Section Page Break (see page 501)

Multiple DO clauses

Section Specifying Multiple DO Statements (see page 502)

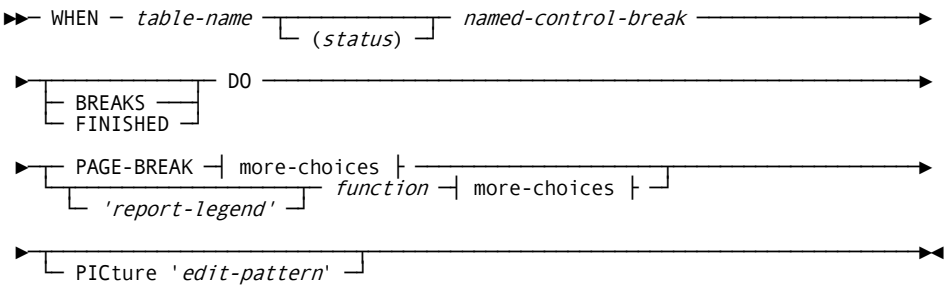
Specifying Functions at Control Breaks

Purpose

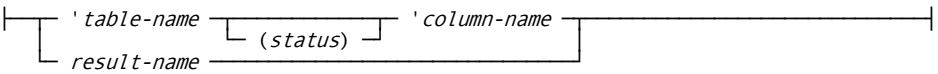
You can have CA Dataquery perform a calculation at control breaks and print the result on the report, along with a legend.

Syntax Diagram

The syntax for specifying a mathematical function at a control break is:



Expansion of more-choices



Repeat WHEN statement once for each function or page break wanted. Multiple DO statements may be used with one WHEN statement.

Explanation

The following explains the previous syntax diagram for WHEN/DO statements.

table-name

Refers to the specific table which contains the designated column or key which follows. It is not needed if it is the same as the most recently named table in the query.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

named-control-break

Specifies the name of a column or key which was designated as a control-break in a SORT statement.

BREAKS

Used only for readability.

'report-legend'

Specifies a legend to be printed beside the results of the function calculation. Can be up to 64 characters in length. It can contain a heading substitution string which will result in the value of the column being substituted in the legend at the indicated place when it is printed. It must be enclosed in single apostrophes. If no report legend is given, the default will be the function followed by the name of the function column.

Note: Select the System Profile Help Topic from the List of Help Topics panel (HELP command) to determine what heading substitution string is used at your site. && is the default provided by CA.

function

Specifies the function to be performed. The following chart describes each function you can perform and the entry for your query:

Function	Entry	At Designated Control Break:
Average	AVG	Averages the values for the specified numeric column.
Count	CNT	Counts the number of occurrences of the specified column.
Maximum	MAX	Finds the maximum value for the specified numeric column.
Minimum	MIN	Finds the minimum value for the specified numeric column.
Sum	SUM	Sums the values for the specified numeric column.

PAGE-BREAK

Specifies that CA Dataquery should start a new page or a new screen when the value of the control break changes. (WHEN/DO PAGE-BREAK does not operate with totals-only.)

column-name

Specifies the name of the column on which to perform the specified function. The named column must be defined in Datadictionary as a valid numeric type, which includes floating-point types.

Also, a column can be one occurrence within a repeating field. If that is the case, use subscripts to identify which occurrence is to be used. A typical subscript looks like this:

```
SALES-RESULTS (5,2)
```

(See the *CA Dataquery User Guide* for more information.)

result-name

Specifies the name of the temporary result on which to perform the specified function.

PICTURE

(Optional) Specifies that the data should be formatted according to the edit pattern specified next.

'edit-pattern'

A series of characters that specify how you want the data to be formatted. See PICTURE Clause (see page 424).

Sample

The following query illustrates a mathematical function performed at a control break.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-SLSHST-REC BY (ORD-ID) AND CAI-ORDERS-REC SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
                          SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
WHEN CAI-SLSHST-REC ORD-ID BREAKS
DO MAX CAI-SLSHST-REC UNIT-PRICE
```

Output

The following sample output is generated by the preceding query.

```
 =>
01/02/2010                          CA Dataquery                      PAGE 1
17:21:16                                                                   DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   08811  00001    O50006      0000203    00014.75    00002994.25   00010.00
          00002    O50006      0000010    00014.75    00000147.50   00002.75

  TOTAL ORD-ID 08811
                                                    * 00003141.75
  MAXIMUM UNIT-PRICE    14.75

   09312  00012    O50006      0000100    00014.75    00001475.00   00010.00
          00015    O90001      0000010    00090.00    00000900.00   00002.75
  TOTAL ORD-ID 08811
                                                    * 00001375.00
  MAXIMUM UNIT-PRICE    90.00
* GRAND TOTAL
                                                    * 00079923.76
 ------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP          <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS     <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED       <PF10> NOT USED     <PF11> LEFT            <PF12> RIGHT
```

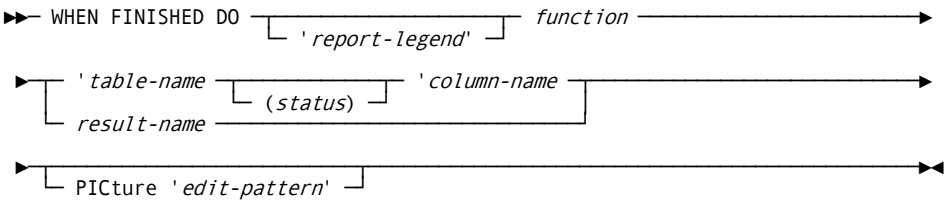
End of Report

Purpose

You can specify that a mathematical function be performed at the end of a report.

Syntax Diagram

The syntax for specifying a mathematical function when the report is finished is:



Repeat WHEN statement once for each function or page break wanted. Multiple DO statements may be used with one WHEN statement.

Explanation

The following explains the preceding WHEN/DO statement format.

FINISHED

Refers to end of report and implies grand totals on sums and overall averages, maximums, and minimums.

'report-legend'

Specifies a legend to be printed beside the results of the function calculation. Can be up to 64 characters in length. It must be enclosed in single apostrophes. If no report legend is given, the default will be the function followed by the name of the function column.

Note: Do not use the heading substitution string in a legend for an end-of-report function.

function

Specifies the function to be performed. The following describes each function you can perform and the entry in your query.

AVG

Averages the values for the specified numeric column.

CNT

Counts the number of occurrences of the specified column.

MAX

Finds the maximum value for the specified numeric column.

MIN

Finds the minimum value for the specified numeric column.

SUM

Sums the values for the specified numeric column.

table-name

Refers to the specific table which contains the designated column or key which follows. It is not needed if it is the same as the most recently named table in the query.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

column-name

Specifies the name of the column on which the function is to be performed. Also, a column can be one occurrence within a repeating field. If that is the case, use subscripts to identify which occurrence is to be used. (See the *CA Dataquery User Guide* for more information.)

result-name

Specifies the name of the temporary result on which the function is to be performed.

PICTURE

(Optional) Specifies that the data should be formatted according to the edit pattern specified next.

'edit-pattern'

A series of characters that specify how you want the data to be formatted. See PICTURE Clause (see page 424).

Example

The following query illustrates a mathematical function performed at the end of the report.

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 100
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-SLSHST-REC BY (ORD-ID) AND CAI-ORDERS-REC SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
                          SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
WHEN FINISHED     DO SUM SHIP-QTY
```

Output

The following screen is sample output generated from executing the preceding query.

```
 =>
01/02/2010                          CA Dataquery                    PAGE 1
17:36:57                                                                 DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   08811  00001    O50006      0000203    00014.75    00002994.25   00010.00
          00002    O50006      0000010    00014.75    00000147.50   00002.75

  TOTAL ORD-ID 08811
                                                   * 00003141.75

   09312  00012    O50006      0000100    00014.75    00001475.00   00002.75
          00015    O90001      0000010    00090.00    00000900.00   00002.75

  TOTAL ORD-ID 08811
                                                   * 00001375.00
* GRAND TOTAL
                                                   * 00004516.75
  SUM SHIP-QTY  0000323
 -------------------------------- LAST PAGE --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED      <PF11> LEFT            <PF12> RIGHT
```

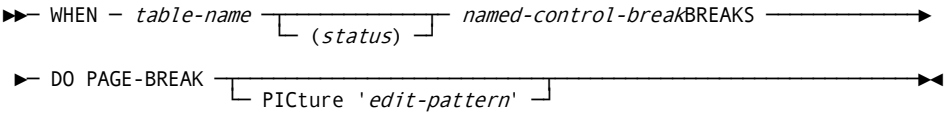
Page Break

Purpose

In addition to performing mathematical functions on a control break, you can also specify a page break. This option allows you to start report output related to a specified control break on a new page.

Syntax Diagram

The syntax for specifying a page break at a control break is:



Repeat WHEN statement once for each function or page break wanted. Multiple DO statements may be used with one WHEN statement.

Explanation

The following explains the preceding format.

table-name

Refers to the specific table which contains the designated column or key which follows. It is not needed if it is the same as the most recently named table in the query.

(status)

(Optional) If authorized, use to specify a Datadictionary definition of a table that is being tested by programmers responsible for its maintenance. If the FIND or COUNT statement specifies a TEST status for a table, all subsequent statements in the same query reference the same status and version of the table, unless a different status is specified. The following describes each status you can specify.

Note: The status must be enclosed in parentheses, and there must be no spaces between the table name and the left parenthesis.

TEST

Finds the named table in the first test version that meets the specifications.

Tnnn

(Where nnn refers to the version number of the test status you want.) Finds the named table in the specified test version.

PROD

PRODuction is the default status if not specified.

named-control-break

Specifies the name of a column or key which was designated as a control-break in a SORT statement.

BREAKS

Used only for readability.

PAGE-BREAK

You can have CA Dataquery start a new page or new screen every time the value in a specific control break changes. (WHEN/DO PAGE-BREAK does not operate with totals-only.)

PICTURE

(Optional) Specifies that the data should be formatted according to the edit pattern specified next.

'edit-pattern'

A series of characters that specify how you want the data to be formatted. See PICTURE Clause (see page 424).

Specifying Multiple DO Statements

The following query illustrates multiple DO statements for a single WHEN statement. The multiple DO statements perform mathematical functions and a page break when a control break is encountered during execution.

Sample

```
FIND 10 CAI-SLSHST-REC ROWS
    WITH SHIP-QTY > 0
  RELATED BY SLMN-ID TO CAI-ORDERS-REC
SET NET-COST (8.2) = CAI-SLSHST-REC SHIP-QTY * UNIT-PRICE
SET DISCOUNT (5.2) = NET-COST * DISC-PCT
SORT CAI-SLSHST-REC BY (ORD-ID) AND CAI-ORDERS-REC SHIP-ID
PRINT FROM CAI-ORDERS-REC ORD-ID
                          SHIP-ID
      FROM CAI-SLSHST-REC ITM-ID
                          SHIP-QTY
                          UNIT-PRICE
                          (NET-COST)
                          DISCOUNT
WHEN CAI-SLSHST-REC ORD-ID BREAKS
DO MAX UNIT-PRICE
DO SUM CAI-SLSHST-REC SHIP-QTY
DO PAGE-BREAK
```

Output

This screen is the first page of sample output generated from executing this query.

Sample (Continued)

```
 =>

01/02/2010                          CA Dataquery                    PAGE 1
18:21:32                                                                DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   07333  00001    O10001      0000006    00998.50    00005991.00   00010.00
          00002    O30012      0000001    01000.00    00001000.00   00003.00

  TOTAL ORD-ID 07333
                                                    * 00006991.00

  MAXIMUM UNIT-PRICE  01000.00

  SUM SHIP-QTY  0000007




 ------------------------------- MORE .... --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED      <PF11> LEFT            <PF12> RIGHT
```

Sample (Continued)

This screen is a later page of sample output for the preceding example.

```
 =>

01/02/2010                          CA Dataquery                    PAGE 3A
18:21:32                                                                DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST   DISCOUNT
  ------  -------  ----------  --------  ----------  -------------  ---------
   09311  00001    C10001      0000023    00021.00    00000483.00   00032.00

  TOTAL ORD-ID 09311
                                                    * 00000483.00

  MAXIMUM UNIT-PRICE  00021.00

  SUM SHIP-QTY  0000023


 ------------------------------- MORE .... --------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED      <PF11> LEFT            <PF12> RIGHT
```

Sample (Continued)

This screen is the last page of output for the preceding example.

```
 =>

01/02/2010                          CA Dataquery                    PAGE 5A
18:21:32                                                                DETAIL

  ORD-ID  SHIP-ID  ITM-ID      SHIP-QTY  UNIT-PRICE       NET-COST DISCOUNT
  ------  -------  ----------  --------  ----------  ------------- --------

* GRAND TOTAL
                                                    * 00056053.40









 -------------------------------- LAST PAGE ---------------------------------
 <PF1>  HELP         <PF2>  RETURN       <PF3>  TOTALS ONLY     <PF4>  DETAIL
 <PF5>  NO TOTALS    <PF6>  STATS        <PF7>  BACKWARD        <PF8>  FORWARD
 <PF9> NOT USED      <PF10> NOT USED      <PF11> LEFT            <PF12> RIGHT
```

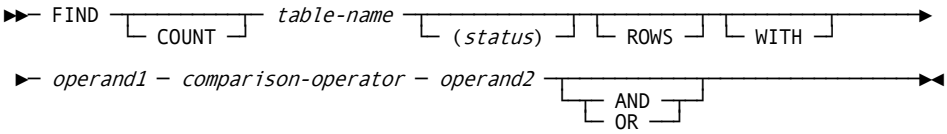
WITH Clauses

Purpose

Use WITH clauses to apply selection criteria to the selection of rows for the active found set. WITH clauses can be added to FIND statements and/or relationship clauses to qualify rows from any table read by the query for selection. Their use can improve performance. A WITH clause contains selection criteria stating the characteristics a row must have to qualify for selection. The selection criteria are expressed as one or more logical expressions that CA Dataquery evaluates during the selection process. See Multiple WITH Clauses (see page 511) for more information about multiple WITH clauses.

Syntax Diagrams

The following diagrams show the use of a WITH clause in a FIND statement and in a RELATED BY clause.



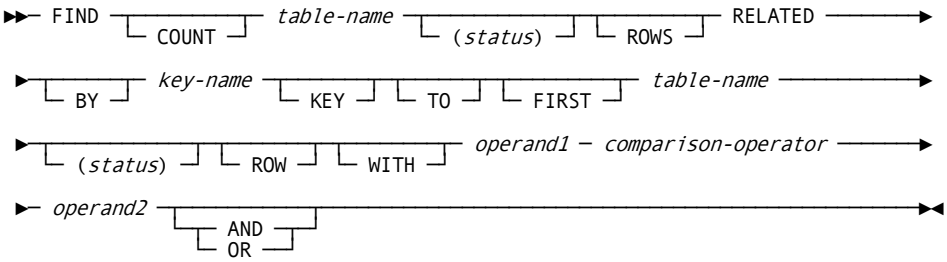
Additional comparison expressions may be added using AND and OR. For example:

FIND all personnel rows

WITH name-column = smith

AND city-column = dallas

OR city-column = houston



Additional comparison expressions may be added using AND and OR.

For example:

FIND all personnel ROWS

WITH dept-column = sales

RELATED BY social-security-column TO payroll

WITH payroll street-column NOT CONTAINING 'p.o.'

OR 'box'

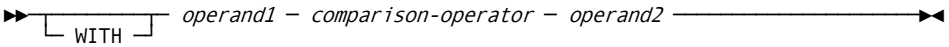
Note: A series of values following an EQ (=) or NE (,=) cannot be separated with AND or OR. For example:

```
Right:
WITH FIELDA = 'TX' 'CA' 'NY' or
WITH FIELDA NE 'TX' 'CA' 'NY'
Wrong:
WITH FIELDA NE 'TX' and 'CA' and 'NY'
```

Only the keywords for the WITH clause are described in the following sections. See FIND Statement (see page 421) for a description of the FIND statement and Relationship Clauses (see page 444) for a description of relationship clauses.

Further information appears in Multiple WITH Clauses (see page 511) and Evaluation of Logical Expressions (see page 512).

WITH Clause Operand1



Explanation

Operand1 in the WITH clause diagram represents any valid column or key contained in the specified table or an arithmetic expression enclosed in parentheses.

CA Dataquery evaluates the specified name as a column first, then as a key if no column with that name is found. CA Dataquery compares the column, key or arithmetic expression to the value of operand2. See Evaluation of Arithmetic Expressions (see page 477) for more information about how CA Dataquery processes these expressions. In comparing columns, CA Dataquery compares data types. In comparing keys, CA Dataquery only performs logical compares.

Columns and Keys

Operand1 can be any valid column or key, including a character column, a numeric column, a key, or a repeating field.

Key Name:
If either operand1 or operand2 is a key and the other is a column, the key can contain only one column, but it must be a character column.

Repeating Fields:
A column can be one occurrence within a repeating field. If that is the case, use subscripts to identify which occurrence is to be used. A typical subscript looks like this:

```
SALES-RESULTS (5,2)
```

See the *CA Dataquery User Guide* for detailed instructions.

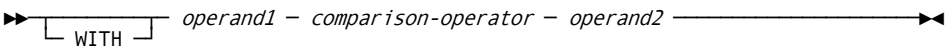
Table Names:
When operand1 and operand2 represent values in two different tables, prefix one or both with table names and use a period between the table name and column name. Do not use a table name if operand1 is an arithmetic expression.

Numeric Column:
If operand1 of a WITH clause is a single numeric column, its comparison with operand2 is based on the decimal precision of operand1.

Arithmetic Expressions

If you are using an arithmetic expression to represent operand1, you must enclose the expression within parentheses. In addition, you can have multiple expressions within the parentheses. However, an arithmetic expression cannot contain any mathematical functions (MAX, AVG, and so forth).

The following is a sample WITH clause with operand1 as an arithmetic expression.



If operand1 is an arithmetic expression, such as (FLD/A + FLD/B) the comparison is based on the precision of the object after the number is rounded. For example, assume that the value of FLD/A is .41 and the value of FLD/B is 0. If the WITH clause is

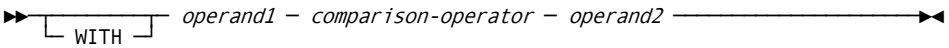
```
WITH FLD/A = 0.00
```

the WITH clause evaluates false. If the WITH clause is

```
WITH (FLD/A + FLD/B) = 0
```

the WITH clause evaluates true, because the .41 result of operand1 expression rounds to 0. If the object were specified as 0.00, no row would be found. Rounding is not required and 0.41 does not equal 0.00.

WITH Clause Comparison Operators



Explanation

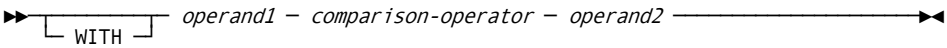
Comparison-operator in the preceding diagram indicates the type of comparison to be performed between operand1 and operand2. The following chart lists the operators you can use in a WITH clause, shows alternate entries (if they exist), and gives examples on how to use the operators.

Note: See WITH Clauses (see page 504) for restrictions on using AND with EQ or NE.

Operator	Alternate Entry	Explanation/Example
EQUAL	EQ, =	CA Dataquery selects only those rows for which the value of operand1 equals the value of operand2. **WITH STATE EQUAL 'TX'** In addition, you can use THRU with the EQUAL operation to specify a range of values for a numeric column. **WITH YTD-SALES = 1000 THRU 2000** You can include a series of values for operand2 in a list fashion using the operator EQUAL. **WITH STATE = 'TX','NY', 'CA'**
NOT EQUAL	NE, ,=	CA Dataquery selects only those rows for which operand1 is not equal to the value of operand2. **WITH STATE NE 'TX','NY', 'CA'**
GREATER THAN	GT, >	CA Dataquery selects only those rows for which the value of operand1 is greater than the value of operand2 **WITH YTD-SALES GT 1000**
GREATER THAN OR EQUAL TO	GTE	CA Dataquery selects only those rows for which the value of operand1 is greater than or equal to the value of operand2. **WITH YTD-SALES GTE 1000**
LESS THAN	LT, <	CA Dataquery selects only those rows for which the value of operand1 is less than the value of operand2. **WITH YTD-SALES LT 1000**
LESS THAN OR EQUAL TO	LTE	CA Dataquery selects only those rows for which the value of operand1 is less than or equal to the value of operand2. **WITH YTD-SALES LTE 1000**

Operator	Alternate Entry	Explanation/Example
CONTAINING	(none)	CA Dataquery selects only those rows for which the value of operand1 contains the value of operand2 in any position of operand1. The operand2 value must be a literal. This instructs CA Dataquery to select only those rows which contain the literal. **WITH CITY CONTAINING 'DA'**
NOT	,	CA Dataquery selects all rows for which the value of operand1 does not compare with value for operand2. You can use NOT with any of the operators listed preceding except NOT EQUAL. **WITH YTD-SALES NOT LT 1000**

WITH Clause Operand2



Explanation

Use one of the following as *Operand2* of a WITH clause, depending on its compatibility with operand1 or with a particular operator.

- A Datadictionary name of a column in the last table named
- A Datadictionary name of a key in the last table named
- A literal value
- The word **null** (meaning *empty*)

See the following pages for explanations of these operand2 options.

CA Dataquery evaluates the name specified for operand2 as a key first, then as a column if no key is found with that name. If a key or column is named, CA Dataquery compares the contents of the column to operand1. If a literal value or **null** is specified, that value is compared to operand1. *The value of the column, key, or literal must be structured the same as the value for operand1.*

Note: Do not use the operator CONTAINING if operand2 is a column or key.

Columns and Keys

See Columns and Keys (see page 506) for detailed information about columns and keys in a WITH clause.

Literal Value

If operand1 is a character column, the literal value for operand2 can contain any alphabetic or numeric character. If the literal value is not a number, it must be enclosed within apostrophes ('). The length of the literal cannot exceed the length of operand1. If the length of the literal is smaller, CA Dataquery adjusts the value to make the length of the literal the same as operand1 by padding with blanks to the right, the same as operand1.

If operand1 is a numeric column, the literal must contain numeric characters, which do not exceed the length of operand1, and a decimal point. The literal should not be enclosed within apostrophes ('). You can indicate a negative value by prefixing the literal with a minus sign (-).

If operand1 is a single-column key, the literal must conform to the structure of the column in the key and must be enclosed within apostrophes (').

Also, you can use *literal masking* to compare only portions of operand1 and operand2 to each other.

If operand1 is a multi-column or compound key, you can specify a literal value for each key column; however, if you specify only one value, CA Dataquery compares the literal value to the high-order column in the key. If you do specify a literal value for each key column, delimiters must separate each value. The entire literal must be enclosed within apostrophes ('). The following is a sample diagram for a literal for a multi-column key.

Note: A delimiter can be any character except a blank, the literal masking character, a dollar sign ($), a hyphen (-), the decimal point character, an apostrophe ('), a letter (a-z), a number from 0 to 9, the Kanji shift in character, or the Kanji shift out character.

```
'/lit1/lit2/lit3/'
```

Literal Masking

You can use a literal value which specifies that only certain positions within a column are to be compared. This is called literal masking. A literal mask designates a set of characters which CA Dataquery compares position by position to the value of operand1. In those positions which you do not want to specify a character, use the literal masking character.

If the column is numeric, only numeric characters other than the literal masking character can be specified in the literal. Unsigned, zoned, decimal columns and signed or unsigned, packed decimal columns can be masked. Do not include decimal points, dollar signs or minus signs in the literal. You can specify up to 15 literal values per WITH clause.

Note: Select the System Profile topic from the List of Help Topics panel (HELP) command to see which character has been designated as the Literal Masking character at your site. The default is the pound sign (#).

For example, if operand1 is a five-character column, and you want to select all the rows which have a C in the second position of its value, you can specify the following literal mask:

'#C#'

In this example, the pound sign is a literal masking character.

Notice that the literal does not contain five characters. Whenever the (#) appears as the last position of the mask, CA Dataquery assumes the special character fills the other positions to the proper length of the column.

Multiple WITH Clauses

A WITH clause can contain multiple logical expressions joined by AND or OR. CA Dataquery selects the data meeting the conditions stated in each expression of the WITH clause.

If expressions within the WITH clause are joined by AND, CA Dataquery retrieves data which matches all of the stated expressions. If expressions within the WITH clause are joined by OR, CA Dataquery retrieves data which matches at least one of the stated expressions.

Example

The following example illustrates the use of AND in a WITH clause.

```
FIND 10 CAI-CUST-REC ROWS
     WITH STATE = 'TX' AND
          CITY = 'DALLAS' AND
          PH-EXCH = '248'
```

The result of this FIND statement is a set of 10 customer rows for customers located in Dallas, Texas, and within the 248 phone exchange.

Example

In the example below, a WITH clause contains expressions joined by OR.

```
FIND ALL CAI-CUST-REC ROWS
     WITH CITY = 'DALLAS' OR
          AREA-CD = '214'
```

When you execute this FIND statement, CA Dataquery selects all the customer rows for Dallas or customers who are located within the area code 214.

You can group logical expressions together within parentheses to change the standard order of evaluation. CA Dataquery evaluates expressions within parentheses before any other expressions. You can specify up to five levels of parentheses in a FIND statement.

If NOT precedes an entire WITH clause contained within parentheses, it reverses the meaning of the operators within the parentheses. However, if more WITH clauses follow the parentheses, the NOT operator does not negate them.

Example

The following two examples illustrate the use of NOT. In the first example, CA Dataquery selects those customer rows which do not have a state address of TX, NY or CA.

```
FIND 10 CAI-CUST-REC ROWS
     WITH STATE NOT ='TX' 'NY' 'CA'
```

In the example below, CA Dataquery selects those personnel rows which do not have a state address of TX, NY or CA, but do have a city address which contains DA.

```
FIND 10 CAI-CUST-REC ROWS
      WITH STATE NOT = 'TX ' 'NY' 'CA'
      AND CITY CONTAINING 'DA'
```

Evaluation of Logical Expressions

CA Dataquery evaluates logical expressions in the following order:

1. Expressions contained within parentheses
2. Expressions within parentheses preceded by NOT
3. Expressions joined with AND
4. Expressions joined with OR

The following chart illustrates how CA Dataquery evaluates logical expressions which contain parentheses or NOT.

Example and Explanation
EXPR1 **AND** EXPR2 **AND** EXPR3 *CA Dataquery selects only those rows which meet the criteria specified in all three expressions.*

Example and Explanation
EXPR1 **AND** (EXPR2 OR EXPR3) *CA Dataquery selects only those rows which meet the criteria in EXPR1 and the criteria in either EXPR2 or EXPR3.*
(EXPR1 AND EXPR2) **OR** EXPR3 *CA Dataquery selects only those rows which meet the criteria stated in both EXPR1 and EXPR2 or just the criteria stated in EXPR3.*
NOT (EXPR1 AND EXPR2) **AND** EXPR3 *CA Dataquery selects only those rows which do not meet the criteria specified in either EXPR1 or EXPR2, but do meet the criteria in EXPR3.*

If you do not use parentheses to group your logical expressions, CA Dataquery will evaluate expressions joined by AND first. For example,

```
EXPR1 AND EXPR2 OR EXPR3

is equivalent to

(EXPR1 AND EXPR2) OR EXPR3
```

Chapter 7: SQL Statements

This chapter provides an overview of the SQL keywords used by CA Dataquery to create queries and dialogs that produce reports.

SQL Overview

Whether you build your queries using Guided Query or the CREATE function, your query uses the same SQL clauses. These clauses instruct CA Dataquery to search a table, limit the search, join one or more tables to the first table, group data, sort the data for presentation, and perform mathematical functions.

CA Dataquery SQL queries are English-like clauses whose keywords are SELECT, FROM, WHERE, HAVING, GROUP BY, and ORDER BY. These clauses together form an SQL SELECT statement. The UNION keyword can be used to combine SELECT statements. Also, SQL clauses can use the same basic elements, like expressions and search conditions.

For complete details on creating SQL statements for CA Dataquery, see the following pages.

For a tutorial on the basics of SQL query creation, see the *CA Dataquery User Guide*.

Note: Only one SQL command is allowed per query.

SQL Summary

The following chart summarizes the SQL keywords used most often in queries. The keywords appear on the chart in the order in which they are used in an SQL query or dialog.

Keyword:	Operand:	Explanation:
SELECT	selection-list	Starts an SQL query or dialog. Specifies the data to be retrieved. The items in a selection list can be: column names, mathematical functions, arithmetic expressions, and literal constants.
FROM	table-names	Identifies the table or tables where data can be found.

Keyword:	Operand:	Explanation:
WHERE	predicate	Tests each row in the named tables. If the result of the test is true, the row is selected. A predicate can be a comparison or a special search condition.
GROUP BY	column-names	Arranges data in groups. Usually used when needed for applying functions (like SUM) to groups of data.
HAVING	predicate	Used with GROUP BY to retrieve rows whose groups meet the search condition.
ORDER BY	select-list-items	Lists column names to specify the order in which data is to be presented. Columns listed must be in the SELECT clause. Ascending order is the default but DESC (descending) can be specified.

Other Acceptable SQL Keywords

Additional SQL keywords can also be submitted with CA Dataquery. You must have special authorization from your CA Dataquery Administrator to use these keywords in CA Dataquery.

Data definition authorization allows you to use:

- ALTER
- CREATE INDEX
- CREATE SYNONYM
- CREATE TABLE
- CREATE VIEW
- COMMENT ON
- DROP
- GRANT
- REVOKE

Data maintenance authorization allows you to use:

- DELETE
- INSERT
- UPDATE

Some information about using these keywords in CA Dataquery can be found in the *CA Dataquery Administrator Guide*. Also, see the *CA Dataquery User Guide* to compare DQL and SQL Query usage.

ALTER TABLE

This SQL statement can be executed in the following ways:	**Through the Datadictionary Interactive SQL Service Facility *(interactive)***	**In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)***	**By using CA Dataquery *(SQL & Batch Modes)***
ALTER TABLE	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

Note: If you are using CA Datacom/DB as part of the CA Datacom/AD environment, you cannot use the ALTER TABLE statement. With regard to table partitioning, ALTER statements may not be issued against a table which is partitioned nor against a partition. An SQL integrity constraint cannot reference a partitioned table, nor a partition of a partitioned table. That is to say, constraints and partitioned tables are mutually exclusive. For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

The ALTER TABLE statement is a CA Datacom/DB extension. ALTER TABLE allows you to alter a table definition without having to reload manually. You may change the table's column definition and/or the table's constraints. You may make only one change to a column in an ALTER TABLE statement.

You can query the SYSCONSTRSRC and SYSCONSTRDEP tables (see the *CA Datacom/DB SQL User Guide*) to view information about constraints. You can then use ALTER TABLE to drop the constraint or modify the constraint's selection criteria.

When a table is altered, the prepared statements in plans that reference the table are marked non-executable until rebound. An attempt to execute these statements invokes automatic rebind. Depending on the change, the rebind may be successful and require no manual intervention. But if the plans containing referencing statements are currently executing or binding, the ALTER request is aborted. This is always the case when the plan containing the ALTER statement references the table, because this is an attempt to invalidate its own plan and is not allowed.

A Datadictionary Relationship Report can tell you what plans are dependent on a table. See the *CA Datacom Datadictionary Batch Reference Guide* for information on Relationship Reports.

See the *CA Datacom/DB SQL User Guide* for information about the results in Datadictionary of using the ALTER TABLE statement.

The ALTER TABLE statement causes the CONSTRAINT attribute-value to be marked A (for ALTERED) for all TEST and HISTory versions of the Datadictionary definition. The next time you try to copy that definition from TEST to PRODuction status, Datadictionary issues an ALT DSF return code that indicates it cannot be copied because the PROD version has been altered.

Note: For details, see the appendix on constrained tables in the *CA Datacom Datadictionary Online Reference Guide* or *CA Datacom Datadictionary Batch Reference Guide*.

The table must be at least half empty before alteration, if using the ALTER TABLE statement causes a table's rows to be reformatted (adding, dropping, or modifying columns). But the table must be *more than* half empty before alteration (that is, less than half full) if the adding, dropping, or modifying of columns causes the size of the table's rows to *grow.*

Issuing an ALTER TABLE statement involves a process that:

1. Copies the contents of the table to the Temporary Table Manager (TTM),
2. Changes the definition of the table, and then
3. Copies the reformatted rows back to the table.

However, a check is made before this operation begins to see if enough space is available in the TTM, and if the answer is no, SQL code -561 (SQLSTATE 57S05) is generated: **TTM TOO SMALL - SEE ERROR ACTION FOR SUGGESTION.** In response to this message you could make the TTM larger, but it can take a long time to perform that operation if there are many rows in the table. Therefore, instead of making the TTM larger consider doing the following:

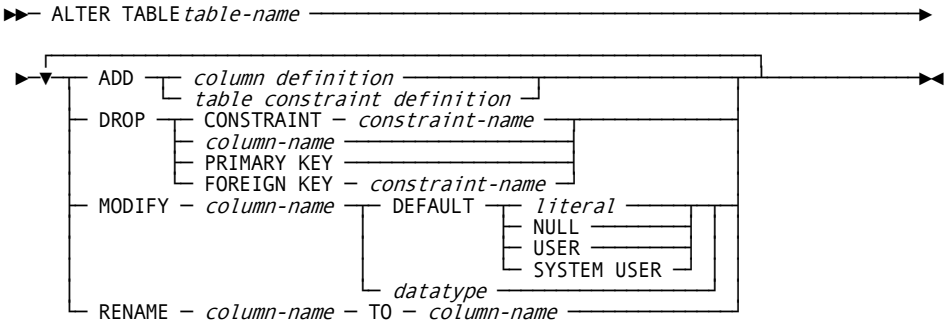
1. Copy the rows to an external file using the EXTRACT function of DBUTLTY.
2. Convert the rows yourself.

3. For single table areas, do a null LOAD. For multi-table areas, do a REPLACE on the specific table.
4. Perform the ALTER of the empty table.
5. For single table areas, LOAD the converted table. For multi-table areas, do a REPLACE using the converted data.

Note: Be aware that because the TTM space check is made at the beginning of the process, there is the possibility that other tasks could use some space and the TTM still become full even if the -561 SQL code was not initially received.

When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, an ALTER TABLE statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force.

Following is the syntax diagram for the ALTER TABLE statement:



Description

table-name

Specify the name of the table you want to alter. The table-name must identify a table described in the Datadictionary, but not a Datadictionary table.

ADD

Allows you to add column definitions or table constraint definitions.

Important! Constraints should not be added to an open table. If a constraint (CHECK, FOREIGN KEY, or PRIMARY KEY) is added to a table which previously had no constraints, the constraint is not be enforced in programs which opened the table before the execution of the ALTER TABLE statement. Should this occur, the program should close and reopen the table to activate constraint enforcement for that table. (This caution only applies to record-at-a-time programs referencing tables to which the constraint was added.)

When you use the ALTER TABLE statement to add domain or foreign key constraint(s) to a table, they are added to the Schema Information Tables (SIT) (see the *CA Datacom/DB SQL User Guide*) in check pending status. The CONFIRM function of DBUTLTY is then called to validate existing data against the added constraint(s). Previous existing foreign key(s) are not revalidated. If a violation is found, CA Datacom/DB issues an SQL warning message and changes the table's Directory (CXX) status to prevent access to the table's rows.

Note: You must CONFIRM tables in the following order, from referenced-to-referencing, or parent-to-child, that is to say, the referenced table cannot be in check pending status. For example, if there is a foreign key from a REMARKS table to a LINE_ITEM table on PO_Nbr and Line_Item_Nbr, and the LINE_ITEM table has a foreign key that references the PO table on PO_Nbr, the PO table must be loaded first, then the LINE_ITEM table can be confirmed, then the REMARKS table can be confirmed. Therefore, the confirm order would be: from PO table to LINE_ITEM table to REMARKS table.

If violations are found, with the table in check pending status you can do one of two things:

1. You can execute the CONFIRM function of DBUTLTY to delete the row(s) in error and optionally write them to an exception table for further action. To identify the violated constraint(s), after writing the row(s) in error to an exception table you can write a program to read those row(s) and attempt to insert them back into the original table so that the violated constraint(s) are reported.
2. Or you can use the ALTER TABLE statement to drop the violated constraint(s), causing the table to be reconfirmed and taken out of check pending status. If all the added constraints are dropped, the CONFIRM function of DBUTLTY notes that there are no constraints in check pending status in the SIT and resets the table's Directory (CXX) status to allow access to the table's rows.

If you try to add a new constraint other than domain or foreign key constraint(s) previously described and there is data already in the table that violates the constraint, CA Datacom/DB issues an error message. You can correct the error by either changing the constraint definition or removing the data from the table.

column definition

When a new column is added (through a column definition), rows that already exist in the table receive NULL values in the added column unless NOT NULL WITH DEFAULT is specified in the column constraint definition. If NOT NULL WITH DEFAULT is specified, they receive zero (0) for NUMERIC data types, blanks for fixed-length strings, 1/1/1 for dates, 0:00:00 for times, and 1/1/1-0:00:00.000000 for timestamps.

The new column is added at the end of the row.

When a table is created through SQL, an element with the name SQLEL is added spanning the full row. If the SQLEL element exists, the length of this new column is added. If the SQLEL element does not exist, it is added.

See Column Definition (see page 597) for information about the column definition and Column Constraint Definition (see page 599) for information about the column constraint definition.

table constraint definition

See Table Constraint Definition (see page 602).

DROP

Allows you to drop CONSTRAINTs, columns, PRIMARY KEYs, or FOREIGN KEYs. If you try to drop a constraint by name during an ALTER TABLE statement and the specified constraint name is not found, CA Datacom/DB issues an error message. You can query the SYSCONSTRSRC and SYSCONSTRDEP tables (see the *CA Datacom/DB SQL User Guide*) to determine the correct name of the constraint.

You cannot drop a primary or unique constraint if there is one or more foreign references to it. In such cases, CA Datacom/DB issues an error message. You can query the SYSCONSTRSRC and SYSCONSTRDEP tables (see the *CA Datacom/DB SQL User Guide*) to locate the foreign reference. You can then remove all foreign references before dropping the primary or unique constraint.

Note: When a PRIMARY KEY or UNIQUE constraint is dropped, the corresponding index (key) remains but is made non-unique, even if the key was originally created using Datadictionary.

You cannot drop a WITH CHECK OPTION constraint from a view definition by using the ALTER TABLE statement. To drop a WITH CHECK OPTION constraint from a view definition, use the DROP statement to drop the view definition, then use the CREATE VIEW statement to re-create the view definition without the WITH CHECK OPTION constraint.

Note: Dropping the view and re-creating it requires the rebinding of all dependent statements and views. You also have to re-create any views which were dependent on the dropped view.

A column may not be dropped if it is:

- The only column in any element,
- The only column of a key, or
- Involved in any constraint, that is to say, a FOREIGN KEY, PRIMARY KEY, UNIQUE, or CHECK constraint. If you attempt to drop a column involved in any constraint, you receive an SQL -242 return code (CONFLICT ALTERING COLUMN).

DROP does not process and you receive a -118 return code when the Datadictionary entity-occurrence definition of the table or view you specify is protected with a password or a Lock Level 1 or 2. The error message also includes a Datadictionary Service Facility (DSF) return code. The DSF return codes are:

- IPW (for password protected)
- IOR (for Lock Level 1 protected)
- NTF (for Lock Level 2 protected)

See the *CA Datacom/DB SQL User Guide* for more information, and see the Datadictionary documentation for information on passwords and lock levels.

CONSTRAINT *constraint-name*

Specifies the CONSTRAINT to be dropped. See Column Constraint Definition (see page 599) for information about the CONSTRAINT *constraint-name.*

column-name

Specifies the name of a column. The column must belong to the table you have specified with the table-name.

PRIMARY KEY

Specifies that the PRIMARY KEY constraint for a table name is to be dropped. See Column Constraint Definition (see page 599) for information about the PRIMARY KEY.

FOREIGN KEY *constraint-name*

Specifies the FOREIGN KEY to be dropped. See Referential Constraint Definition (see page 604) for information about the FOREIGN KEY.

MODIFY

Allows you to modify a column's DEFAULT or data type.

You cannot modify a column that is involved in any constraint, that is to say, a FOREIGN KEY, PRIMARY KEY, UNIQUE, or CHECK constraint. If you attempt to modify a column that is involved in any constraint, you receive an SQL -242 return code (CONFLICT ALTERING COLUMN).

DEFAULT

Allows you to specify a default.

literal

Specifies a literal as the default. The literal you specify must be consistent with the data type of the specified column.

A user-supplied DEFAULT literal can be up to 20 bytes long, or the length of the column involved, whichever is shorter. A default value may be specified for a character column where the column is greater than 20 bytes long, but the default literal itself is limited to 20 bytes, with the remaining bytes padded with blanks by the system.

NULL

Specifies a NULL value as the default.

USER

Specifies the current authorization ID as the default.

SYSTEM USER

This CA Datacom/DB extension specifies the accessor ID of the currently signed-on user as the default.

datatype

A column's data type may be changed with the following rules:

- DATE, TIME, TIMESTAMP: these data types may not be modified.
- Changing the length of a character field results in padding with blanks or truncation. You receive an error message if truncation would delete non-blank data.

  Important! All data types that are known to Datadictionary as special data types but are unknown to SQL are treated by SQL as character. Therefore, if you alter one of these columns, the ALTER uses the character rules.

- Any of the numeric data types may be converted to any other numeric data type (FLOAT, REAL, DOUBLE PRECISION, DECIMAL, SMALLINT, INTEGER, NUMERIC). Conversion of values from the old to the new data type is treated with the same truncation rules as assignment.

See Data Types (see page 609) for more information about data types.

RENAME *column-name* TO *column-name*

Allows you to rename a column. This changes the SQL name of the column throughout the Datadictionary and SIT tables. It does not correct any external source statements.

You cannot rename a column that is involved in any constraint, that is to say, a FOREIGN KEY, PRIMARY KEY, UNIQUE, or CHECK constraint. If you attempt to rename a column that is involved in any constraint, you receive an SQL -242 return code (CONFLICT ALTERING COLUMN).

Processing

In one ALTER TABLE statement, all ADD column-name, DROP column-name, and MODIFY column-name with data type functions are done first and as one group. Processing this group is done in the following sequence:

1. Lock the table.
2. Read the first/next row in the table.
3. Add the row to the SQL Temporary Table (TTM).
4. Delete the row.
5. Loop back through all the rows.
6. Update the definitions in the Datadictionary and Directory (CXX).
7. Read the records from the SQL Temporary Table (TTM).
8. Reformat the row by adding new columns to the end, deleting any dropped columns, and changing the format and/or length of any modified column.
9. Add the row to the original table.
10. Loop back to process all rows.

Primary and unique constraints are validated as rows are added back in step 9. If a duplicate is found, the entire ALTER statement is backed out.

Foreign keys and domain constraints are added after all rows have been added back. After adding the constraint definition(s), the rows of the table are read in Native Key sequence to validate the constraints. If a row violates one or more domain constraints, the table is placed in check pending status and an SQL return code 170 is returned. You must use the CONFIRM function of DBUTLTY to remove the check pending status before the table can be used. If a row violates foreign keys, the ALTER TABLE statement fails with an SQL return code -176.

If the ALTER TABLE statement cannot complete, it is rolled back to its beginning. If the ALTER TABLE process is force checkpointed (because of the size of the Log Area (LXX) and the concurrent activity), the rollback is incomplete. Therefore, before altering large tables, ensure that the Log Area (LXX) is large enough and take a backup of the data area.

Example 1

Modify an existing table column decimal number to precision 6 with a scale of 2 decimals. This assumes the previous column was such that no data is lost.

1. Table name: DEPTTBL
2. Column name: PROJSTAFF
3. Data type: DECIMAL

```
ALTER TABLE DEPTTBL
     MODIFY PROJSTAFF DECIMAL(6,2)
```

Example 2

Modify two columns to allow for more character data than previously defined.

1. Table name: DEPTTBL
2. Column names: ADMDEPT (to be 3 characters) and DEPTNAME (to be 30 characters)

```
ALTER TABLE DEPTTBL
  MODIFY ADMDEPT CHAR(3)
  MODIFY DEPTNAME CHAR(30)
```

Example 3

Alter table ORDERS to prevent the acceptance of any order that would require the financing of a purchase of less than $1000.00.

1. Require a down payment of 100 percent on orders of less than $1000.00.
2. Give a meaningful constraint name: ORDERS_MIN_AMT_FINANCED.

```
ALTER TABLE ORDERS
   ADD CHECK (GROSS_AMOUNT >= 1000.0 OR PERCENT_DOWN_PMT = 100.0)
   CONSTRAINT ORDERS_MIN_AMT_FINANCED
```

Example 4

Respond to customers' requests to allow financing on small purchases.

1. Table name: ORDERS
2. Remove constraint: ORDERS_MIN_AMT_FINANCED

```
ALTER TABLE ORDERS
    DROP CONSTRAINT ORDERS_MIN_AMT_FINANCED
```

Example 5

Prevent the acceptance of orders from customers who do not pay their bills by ensuring that any order accepted comes from a customer whose name is on a "customers in good standing" list.

1. Table containing customers in good standing: PAYING_CUSTOMERS
2. Table to be protected: ORDERS
3. Columns on which a match is required:

   ```
   ORDERS.CUSTOMER_NAME to PAYING_CUSTOMERS.COMPANY_NAME
   ```

4. Constraint name: ORDERS_TO_PAYING_CUST_NAME

```
ALTER TABLE ORDERS ADD FOREIGN KEY
    (CUSTOMER_NAME) REFERENCES PAYING_CUSTOMERS(COMPANY_NAME)
     CONSTRAINT ORDERS_TO_PAYING_CUST_NAME
```

Example 6

Prevent the entrance of a shipment into the system unless there is an outstanding order for the goods to be shipped.

1. Table containing outstanding orders: ORDERS
2. Table to be protected: CURRENT_SHIPMENTS
3. Columns requiring a match:

   ```
   CURRENT_SHIPMENT columns ORDER_ID and SHIPMENT_NUM to ORDERS columns ORDER_ID and
   SHIPMENT_ID
   ```

4. Constraint name: SHIPMENTS_TO_ORDERS_IDS_MATCH

```
ALTER TABLE CURRENT_SHIPMENTS ADD FOREIGN KEY
    (ORDER_ID, SHIPMENT_NUM) REFERENCES
    ORDERS(ORDER_ID, SHIPMENT_ID)
```

Note: The columns referenced by a foreign key must correspond to the primary key of the referenced table.

Example 7

Remove the foreign key restriction from table CURRENT_SHIPMENTS in order to ship goods that were mistakenly left out of the original shipments, when the order is no longer current.

- Foreign key name: SHIPMENTS_TO_ORDERS_IDS_MATCH

```
ALTER TABLE CURRENT_SHIPMENTS
   DROP FOREIGN KEY SHIPMENTS_TO_ORDERS_IDS_MATCH
```

Note: If a domain constraint is added to a table containing rows that violate the constraint, you receive an SQL return code to warn you of this fact. The constraint is added, but the table is placed in "check" status to make it unusable until the problem has been corrected by either running the CONFIRM function of DBUTLTY with the DELETE=YES option specified to delete violating rows, or by dropping the constraint and then running CONFIRM.

A similar situation occurs when adding a foreign key for which the referenced table lacks rows to match all of the values in the referencing table. The same options (previously described) are available to correct the problem. However, in the case of dropping a foreign key to correct the situation, an automatic CONFIRM is initiated for you so that you do not need to perform this step yourself.

Assignment Statement

For details about this statement, see Assignment Statement (see page 553).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

CALL/EXECUTE PROCEDURE

For an overview and examples of procedures and triggers, see the *CA Datacom/DB SQL User Guide*.

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CALL EXECUTE PROCEDURE	YES if no parms passed	YES	YES if no parms passed

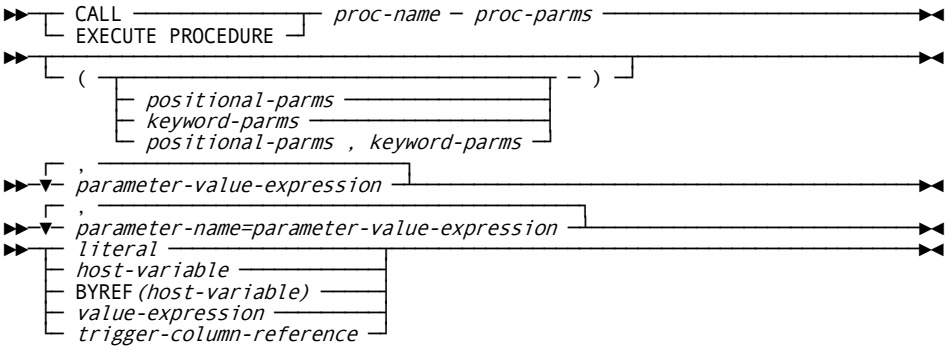
Note: YES indicates a valid execution method for this statement.

A CALL or EXECUTE PROCEDURE statement is used to call user-written procedure logic. This statement can be coded directly by the end-user or can be embedded in a trigger definition. Please note that procedures themselves can contain CALL/EXECUTE PROCEDURE statements, subject to the following discussion of nesting limitation.

A procedure can also cause procedures to execute (*trigger* them) by executing INSERT, UPDATE, or DELETE statements. The depth of nesting of these recursive procedure calls is limited by the PROCEDURE Multi-User startup option that can be used to limit or eliminate nesting.

If null indication variables are passed to the procedure, a list of pointers to the indicators follows the list of pointers to the variables themselves. Although the pointers are contiguous, the indicators themselves are not.

Note: Because SQL has no way of verifying the compatibility of the user-written code with the procedure defined by the CALL/EXECUTE PROCEDURE statement, it is the sole responsibility of the creator of the procedure to ensure that the CALL/EXECUTE PROCEDURE statement precisely reflects the parameter list expected by the user-written program. Failure to properly coordinate parameter lists can cause the procedure subtask to abnormally terminate.



CALL

(Required)

EXECUTE PROCEDURE

(Required)

proc-name

Specify a procedure name. The procedure name cannot be specified as a host variable reference.

ANSI supports *overloading* of the procedure name, with duplicate procedure names resolved to procedure definitions using the layout of the parameter list. For syntax compatibility purposes, a second, *specific* name may be given to a procedure to uniquely identify it, but it must match the nonspecific name, and the nonspecific name must be unique.

parameter-name

Enter the name of the parameter as specified in the CREATE PROCEDURE statement.

literal

Specify a *literal* (value) for the parameter.

host-variable

Specify a *host-variable* by naming a variable that is described in the program in accordance with the rules for declaring host variables.

BYREF(*host-variable*)

This clause, supported for compatibility purposes, is treated as a host variable reference.

value-expression

Specify a value expression (value expressions are used in procedure parameter lists).

trigger-column-reference

(Valid only in a CREATE TRIGGER or CREATE RULE statement.) A reference to a base table column value through a correlation name defined in the old-row/new-row syntax of the CREATE TRIGGER/CREATE RULE statements.

CASE Statement

For details about this statement, see CASE Statement (see page 555).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

COMMENT ON

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
COMMENT ON	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

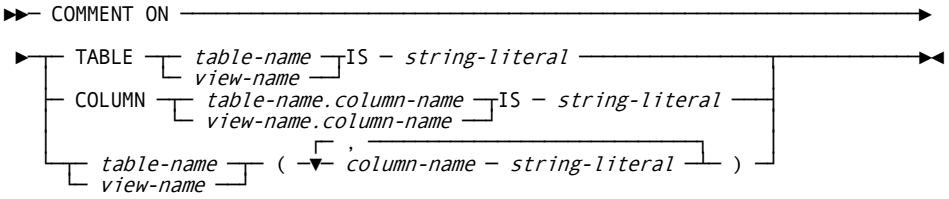
The COMMENT ON statement is a CA Datacom/DB extension. COMMENT ON adds or replaces comments in Datadictionary. The comments apply to tables, views or columns.

If a comment does not exist for the table, view or column, the comment you specify is added to text in Datadictionary. If a comment already exists for the table, view or column, and you specify a new comment, the existing comment is replaced by the new comment.

You cannot retrieve a comment through SQL. You can display the text created by the COMMENT ON statement through the Datadictionary batch and online functions. See the *CA Datacom Datadictionary User Guide* for information on displaying text.

Datadictionary provides many capabilities not available through SQL. For example, text classifications allow you to store text about your SQL tables, views and columns in addition to that specified in the COMMENT ON statement. See the *CA Datacom Datadictionary User Guide* for more information about specifying additional text.

Following is the syntax diagram for the COMMENT ON statement:



Description

TABLE

Indicates you want to comment on a table or view.

table-name* or *view-name

Must identify a table or view described in the Datadictionary. The comment is placed in text in Datadictionary.

COLUMN

Indicates you want to comment on a column.

table-name.column-name* or *view-name.column-name

The name of the column, qualified by the name of the table or view in which it appears. The column must be described in the Datadictionary. The comment is placed in text in Datadictionary.

IS *string-literal*

The string-literal can be any character string literal enclosed between apostrophes.

column-name* IS *string-literal

To comment on more than one column in a table or view, do not use TABLE or COLUMN. Give the table or view name and a list of the form:

(column-name IS 'string-literal', column-name IS 'string-literal', column-name IS 'string-literal')

The column comments must be separated by commas and the list must be enclosed with parentheses. All columns named must appear in the same table or view and the table or view must be described in the Datadictionary.

Example 1

Enter a comment on table EMP.

```
COMMENT ON TABLE EMP
IS 'REFLECTS 1ST QTR 87 REORGANIZATION'
```

Example 2

Enter a comment on view MGR.

```
COMMENT ON TABLE MGR
IS 'VIEW OF TABLE EMP THAT ARE MANAGERS'
```

Example 3

Enter a comment on the EMPNBR column of table EMP.

```
COMMENT ON COLUMN EMP.EMPNBR
IS 'EMPNBR IS UNIQUE'
```

Example 4

Enter comments on two columns in table DEPTTBL.

```
COMMENT ON DEPTTBL
(MGRNBR IS 'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
 ADMDEPT IS 'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT')
```

CREATE INDEX

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CREATE INDEX	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The CREATE INDEX statement is a CA Datacom/DB extension. CREATE INDEX defines a new index on one or more columns of a base table. Adding an index to a table may improve the performance of some queries which reference that table.

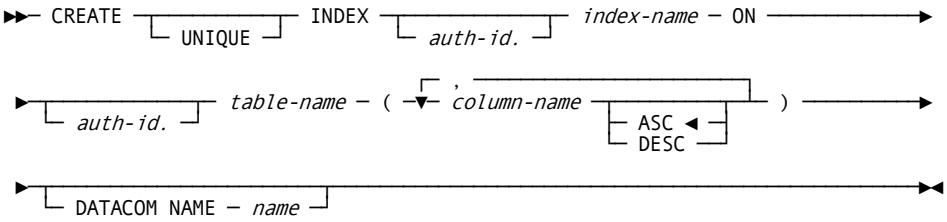
When a CREATE INDEX statement successfully executes, a KEY entity-occurrence is defined in Datadictionary in PRODuction status. The five-character key name is generated for you by Datadictionary. The key's attributes include MASTER-KEY=N, NATIVE-KEY=N, INCLUDE-NIL-KEY=Y, and UNIQUE=N.

Note: For information about specifying SQL key selection override keys in either the correlation name or synonym name of a query, see the section on overriding SQL key selection in the *CA Datacom/DB SQL User Guide*.

CREATE INDEX causes all plans dependent on the indexed table to be marked invalid. You can run a Datadictionary Relationship Report to find out what plans are dependent on a table. See the *CA Datacom Datadictionary Batch Reference Guide* for information on Relationship Reports.

Note: When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, a CREATE INDEX statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force. With regard to table partitioning, CREATE statements may not be issued against a table which is partitioned nor against a partition. For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

Following is the syntax diagram for the CREATE INDEX statement:



Description

UNIQUE

If UNIQUE is specified, the created index enforces value uniqueness and provides an efficient means of both retrieving rows containing specific data values and of ordering the data. The index is rejected if the table contains non-unique values.

auth-id.

An optional identifier of the schema for the index and table. If specified for the index-name, it must be the same as the authorization ID of the table you have specified with the table-name. Use a period (.) to concatenate the authorization ID to the index name or table name (for example, *auth-id.table-name*).

***index-name* ON**

The name for the index you are creating. The index-name you supply, including the implicit or explicit qualifier (authid), must not identify an index already described in Datadictionary.

table-name

Specify the name of the table to be indexed. Must be a table described in Datadictionary.

(column-name)

Specifies the name of a column or a list of columns. The column(s) must belong to the table you have specified with the table-name. If more than one column name is listed, each name must be different and separated by commas. The name of a column or a list of columns must be enclosed in parentheses.

ASC

Places the values of the column in ascending order. ASC is the default order.

DESC

Places the values of the column in descending order.

DATACOM NAME *name*

Specifies a five-character DATACOM NAME name (a KEY entity-occurrence in Datadictionary) for the index. If you do not specify a DATACOM NAME name, Datadictionary generates a name for you. (This is not the Datadictionary entity-occurrence name.)

Processing

CREATE INDEX processing is done by:

1. Locking the table to be indexed,
2. Copying the rows to a temporary table while deleting them from the original table,
3. Adding the definition of the new index, and then
4. Adding the rows from the temporary table back to the original table.

Because of the amount of processing involved, adding an index through the CREATE INDEX statement may not be appropriate for tables with a large number of rows. You may wish to use DBUTLTY's BACKUP function to unload the table's area and then use DBUTLTY's LOAD function to null load the area before executing the CREATE INDEX statement. After adding the new key definition by executing the CREATE INDEX statement, DBUTLTY's LOAD function can be used to reload the area.

Note: For information on using DBUTLTY, see the *CA Datacom/DB DBUTLTY Reference Guide*.

The sum of the lengths of all columns to be included in the index must be no more than 180.

Example

The following example creates an EMPLOYEE_INDEX index on table-name EMPLOYEES, column-name EMPNO.

```
CREATE INDEX EMPLOYEE_INDEX ON EMPLOYEES (EMPNO)
```

CREATE PROCEDURE

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CREATE PROCEDURE	YES/NO	YES	YES/NO

Note: YES indicates a valid execution method for this statement.

YES/NO means YES for External Procedures but NO for SQL Procedures (see following explanations of External and SQL Procedures).

External Procedure definitions and SQL Procedures are created with this statement. External Procedures (see External Procedures (see page 536)) are written in a host programming language (COBOL, PL/I, C, or ASSEMBLER) with SQL statements imbedded in their programs. For an overview and examples of External Procedures and triggers, see the *CA Datacom/DB SQL User Guide*. SQL Procedures (see SQL Procedures (see page 537)) are written in SQL and have all of their execution-time logic included in an SQL CREATE PROCEDURE statement.

There are three ways to execute procedures:

- CALL statements (see CALL/EXECUTE PROCEDURE (see page 527))
- EXECUTE PROCEDURE statements (see CALL/EXECUTE PROCEDURE (see page 527))
- Triggers (see CREATE TRIGGER/RULE (see page 616))

We recommend using PLNCLOSE=T for procedures instead of PLNCLOSE=R. For External Procedures, the PLNCLOSE= specification is determined by the Preprocessor options used for the procedure body compilation. For SQL Procedures, the PLNCLOSE= specification is determined by the options used by the plan in which the CREATE PROCEDURE statement is executed. When using DBSQLPR, specify PLNCLOSE=T or allow PLNCLOSE to default to that value.

When you code procedures that perform INSERTs or UPDATEs, keep in mind that if they are called or triggered (directly or indirectly) from an application that reads the same table on which that procedure is operating, the procedure affects the results of the calling or triggering application and, in extreme cases, can cause loops to occur. For example, if an application is performing a searched UPDATE of a table, and the UPDATE triggers a procedure that does UPDATEs itself, the procedure could cause an endless loop by moving previously updated rows back into the traversal path of the controlling searched update. Because CA Datacom/DB cannot prevent such a situation from occurring, it is important that you make every effort to ensure your coding of a procedure does not cause this to occur.

Triggers whose procedures perform INSERTs, UPDATEs, or DELETEs have the potential to trigger themselves recursively, triggering themselves immediately, or passing through other triggers first. However, be aware that the *maximum recursion depth* specified in the PROCEDURE Multi-User startup option cannot be exceeded.

External Procedures

External Procedure logic is created by executing a preprocessor against user-written host-language source code. The program source code for the procedure must be preprocessed before the CREATE PROCEDURE statement is allowed to run.

The following in the CREATE PROCEDURE statement distinguishes External Procedures from SQL Procedures:

- External Procedures allow any valid choice for the language in the *proc-attributes* syntax except SQL.
- External Procedures allow, in the *proc-body* syntax, the use of the *proc-external* and *parameter-style* clauses, but not the *proc-SQL-stmt* clause.

In a z/OS environment, COBOL, PL/I, and C procedures must be made Language Environment (LE) conforming by being written and compiled using COBOL for z/OS, PL/I for MVS, and z/OS C/C++. Assembler procedures must also be made LE-conforming by use of the CEEENTRY and associated macros.

In a z/VSE environment, COBOL, PL/I, and C procedures must be made Language Environment (LE) conforming by being written and compiled using COBOL for z/VSE, PL/I for z/VSE, IBM C for z/VSE, and High Level Assembler.

Because SQL has no way of verifying the compatibility of the user-written code with the procedure defined by the CREATE PROCEDURE statement, it is the sole responsibility of the creator of the procedure to ensure that the CREATE PROCEDURE statement precisely reflects the parameter list expected by the user-written program. Failure to properly coordinate parameter lists can cause the procedure subtask to abnormally terminate.

Note: As described in the *CA Datacom/DB SQL User Guide*, you should add to the Multi-User Facility library concatenation the libraries containing the programs to be executed as procedures, along with any associated subroutines. Also, if an existing procedure is being replaced, do a NEWCOPY console command, as described in the *CA Datacom/DB Database and System Administration Guide*, to that procedure so that the next job executes the latest copy. For additional information, see the section on Multi-User Facility considerations for procedures in the *CA Datacom/DB SQL User Guide*.

SQL Procedures

SQL procedures support a basic procedural language with support for the following:

- declaration of SQL variables
- value assignment
- decision-making logic (IF-THEN statement) and CASE statement
- looping constructs (LOOP, REPEAT-UNTIL, and WHILE statements)
- error and condition handling, including user-defined conditions (see Diagnostics and Condition Handling (see page 548))
- As many SQL statements as the size of your RWTSA area allows

For SQL Procedures, the CREATE PROCEDURE statement is specified as follows:

- SQL is specified for LANGUAGE in the *language* syntax.
- In the *proc-body* syntax, a *proc-SQL-stmt* clause is specified. The *proc-SQL-stmt* clause is composed of SQL Procedure statements that are coded and executed directly. For instructions about coding a list of statements and program-like logic, see the compound statement information in Compound Statement (see page 558).

Following is a list of the SQL statements that support SQL Procedures:

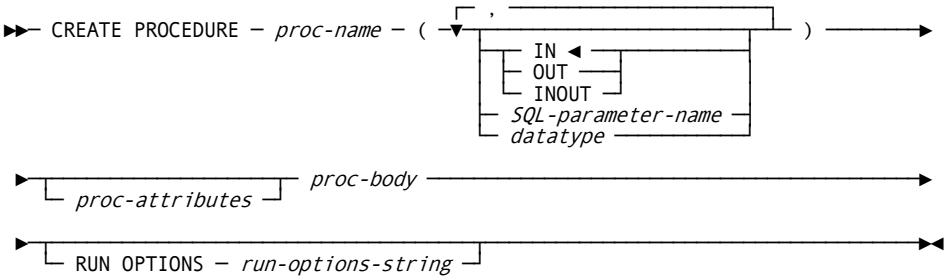
- Assignment Statement (see page 553)
- CASE Statement (see page 555) (different from a case expression)
- Compound Statement (see page 558) (the building block from which SQL Procedures are mainly composed)
- condition declaration
- DATACOM DUMP Statement (see page 563)
- GET DIAGNOSTICS Statement (see page 566)
- IF-THEN Statement (see page 569)
- ITERATE Statement (see page 570)
- LEAVE Statement (see page 570)
- SQL variable declaration
- looping constructs
  - LOOP Statement (see page 571)
  - REPEAT-UNTIL Statement (see page 575)
  - WHILE Statement (see page 583)

- RAISE ERROR Statement (see page 573)
- RESIGNAL Statement (see page 577)
- SIGNAL Statement (see page 579)
- SIMULATE DATACOM PROCEDURE Statement (see page 582) (DBSQLPR only)

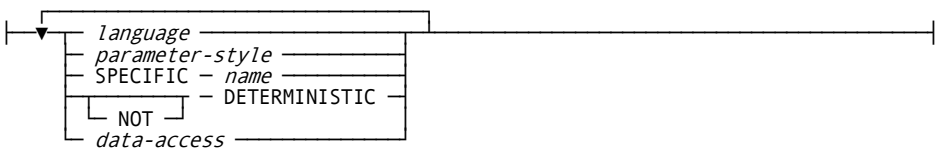
CREATE PROCEDURE Syntax and Description

Following is the syntax for the CREATE PROCEDURE statement:

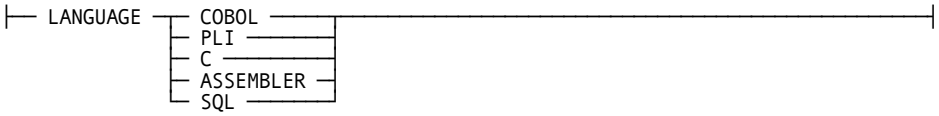
Note: SQL-parameter-name is required for SQL Procedures and, though optional for all others, is recommended for all others as well. See the separate *proc-body* syntax diagram that follows.



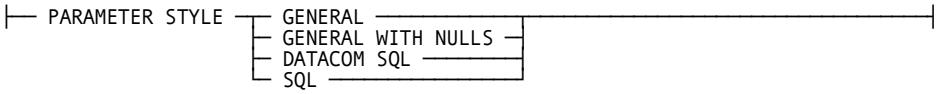
Expansion of Where proc-attributes is defined as



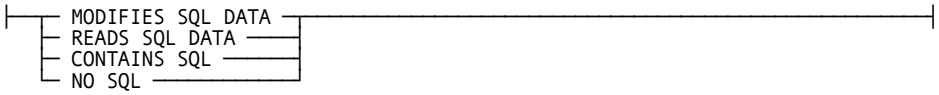
Expansion of Where language is defined as



Expansion of Where parameter-style is defined as



Expansion of Where data-access is defined as



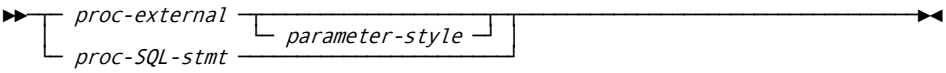
Note: All of the *proc-attributes* choices can be used together, but each can only be used once. If you are using SQL Procedures, *data-access* and *parameter-style* do not apply.

You can specify the parameter style only once, either here as part of the procedure attribute syntax or as part of the EXTERNAL syntax. Parameter style applies only to External Procedures. It does not apply to SQL Procedures.

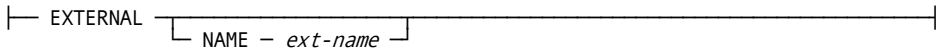
Note: The *proc-external* clause is only used for External Procedures.

The optional *parameter-style* clause is only used for External Procedures.

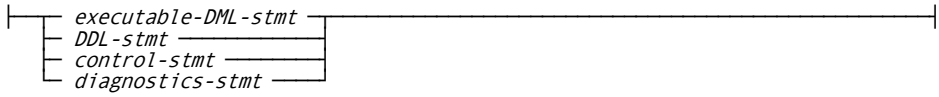
The *proc-SQL-stmt* clause is only used for SQL Procedures.



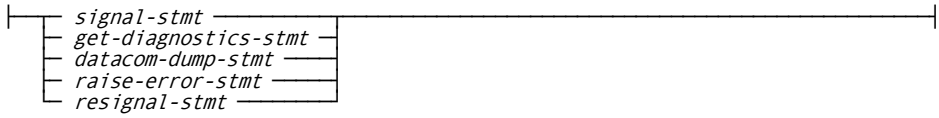
Expansion of Where proc_external is defined as



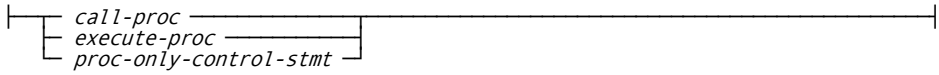
Expansion of Where proc-SQL-stmt is defined as



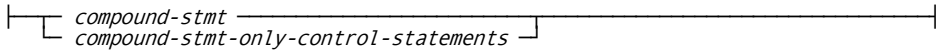
Expansion of Where diagnostics-stmt is defined as



Expansion of Where control-stmt is defined as



Expansion of Where proc-only-control-stmt is defined as

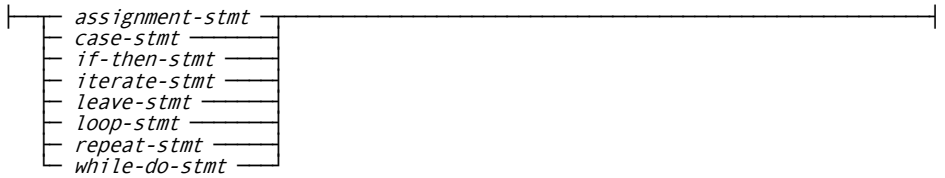


Note: Syntax for the compound-stmt-only-control-statement is given in a separate diagram on the next page.

Note: The executable-DML-stmt is any DML statement except for DECLARE CURSOR (DECLARE CURSOR is supported as part of the compound statement, see Compound Statement (see page 558)). For a list of DML statements, see the *CA Datacom/DB SQL User Guide*. For a list of these statements, see the *CA Datacom/DB SQL User Guide*. See the *CA Datacom/DB SQL User Guide*. For a list of these statements, see the *CA Datacom/DB SQL User Guide*.

See the separate *diagnostics-stmt* syntax diagram that follows and see the GET DIAGNOSTICS statement on GET DIAGNOSTICS Statement (see page 566).

Expansion of Where compound-stmt-only-control-statements are defined as



Following is the description of the CREATE PROCEDURE syntax:

proc-name

(Required) For the *proc-name* specify the SQL-name of the procedure name. The name can be qualified with an authorization ID. This name is an SQL-identifier.

We recommend that you use unique procedure names with regard to the names of any record, table, synonym, view, or constraint. If you code and reference a procedure parameter, for example, whose name is identical to a column name that is also referenced inside a particular SQL Procedure, uniqueness in naming would allow you to use the procedure name and table name to distinguish a procedure parameter reference from a reference to the database-table column.

ANSI supports *overloading* of the procedure name, with duplicate procedure names resolved to procedure definitions using the layout of the parameter list. For syntax compatibility purposes, a second, *specific* name may be given to a procedure to uniquely identify it, but it must match the nonspecific name, and the nonspecific name must be unique.

IN/OUT/INOUT

(Optional) Part of the SQL parameter definition. Specifies to SQL whether this parameter is used for input (IN), output (OUT), or both (INOUT). If this parameter is not specified, the default (IN) is used.

Valid Entries:

IN, OUT, INOUT

Default Value:

IN

SQL-parameter-name

(Optional) The *SQL-parameter-name* is required for SQL Procedures and, though optional for all others, is recommended for all others as well. The *SQL-parameter-name* is the name of a parameter passed to a SQL Procedure (a LANGUAGE SQL procedure). When used in a SQL Procedure containing an SQL variable with a conflicting (matching) name, or a table or view reference where the table or view contains a conflicting column name, the name should be qualified by using the procedure name.

datatype

Part of the SQL parameter definition. Specify a datatype.

Valid Entries:

The same data types valid for the CREATE TABLE statement.

Default Value:

(No default)

LANGUAGE COBOL/PLI/C/ASSEMBLER/SQL

Part of the procedure attributes definition.

Specify COBOL, PLI, C, or ASSEMBLER as the programming language when writing a External Procedure.

Specify SQL when writing a SQL Procedure. When you create a LANGUAGE SQL procedure, CA Datacom/DB creates a plan to hold the SQL statements contained by the procedure. The plan name that is created consists of the first 18 bytes of the procedure name. We recommend creating a SQL Procedure using a method such as DBSQLPR. We support but do not recommend creating an SQL procedure from a preprocessed program.

See the note about language conformance on External Procedures (see page 536).

Note: For all except LANGUAGE SQL, the program source code for the procedure must be preprocessed before the CREATE PROCEDURE statement is allowed to run.

Valid Entries:

COBOL, PLI, C, ASSEMBLER, SQL

Default Value:

(No default)

SPECIFIC *name*

Part of the procedure attributes definition.

ANSI supports *overloading* of the procedure name, with duplicate procedure names resolved to procedure definitions using the layout of the parameter list. For syntax compatibility purposes, a second, *specific* name may be given to a procedure to uniquely identify it, but it must match the nonspecific name, and the nonspecific name must be unique.

DETERMINISTIC/NOT DETERMINISTIC

Part of the procedure attributes definition. This is informational only, indicating whether the procedure always returns the same output, given a certain input.

PARAMETER STYLE

Defines how parameters are passed into and out of the procedure program you write, and how errors are handled.

Note: Parameter style applies only to External Procedures. It does not apply to SQL Procedures.

See the *CA Datacom/DB SQL User Guide* for detailed information.

You can specify PARAMETER STYLE only once, either as:

- Part of the optional *proc-attributes* (procedure attributes) syntax that is specified before the EXTERNAL keyword as shown in the CREATE PROCEDURE *proc-attributes* syntax
- Part of the optional parameter style definition following the EXTERNAL keyword as shown in the CREATE PROCEDURE syntax in CREATE PROCEDURE Syntax and Description (see page 538).

Parameter style applies only to External Procedures. It does not apply to SQL Procedures.

PARAMETER STYLE GENERAL

This parameter style means that the user parameter list is passed to the procedure devoid of null indicators (nulls are not allowed). Since no formal method is provided for passing error information back to the caller, the success or failure of the CALL procedure statement is determined by the contents of SQL's internal SQLCODE variable following the last SQL request made by the procedure. This also applies to parameter style GENERAL WITH NULLS (see following).

PARAMETER STYLE GENERAL WITH NULLS

This parameter style differs from GENERAL only in that a null indicator is passed to the procedure for each user parameter.

PARAMETER STYLE DATACOM SQL

This parameter style passes nulls to the procedure as does GENERAL WITH NULLS and SQL, but it also passes some additional parameters. These parameters are modeled after those passed for the ANSI SQL3 parameter style SQL but with this difference: instead of a SQLSTATE, DATACOM SQL passes an SQLCODE in the corresponding parameter. Following are the additional parameters (the first four are modeled after SQL3):

- SQLCODE—passed to the procedure as 0 and used to set the SQLCODE of the CALL PROCEDURE statement on output.
- A variable-length character string containing the name of the procedure.
- A variable-length character string reserved for future use.

- A variable-length character string containing 80 blanks on input, and an 80-byte or shorter error message on output. This error message is placed in the SQLCA and used as the SQL error message for the CALL PROCEDURE statement.
- A two-byte fixed length character string containing the CA Datacom/DB external error code on output.
- A single character (a 4-byte integer) containing the CA Datacom/DB internal error code on output. You cannot have spaces in the field, so it must be 0000 (hex 'F0F0F0F0') if no internal error is returned. Or if, for example, a return code 94 is returned with an internal error code of 100, hex 'F0F1F0F0' would be returned. Your code should convert the binary one-byte field to a four byte unsigned display number.

PARAMETER STYLE SQL

When a procedure is created using PARAMETER STYLE SQL in the CREATE PROCEDURE statement, the SQLSTATE status indicator is returned in the SQLCA. For detailed information about the SQLSTATE status indicator, see the *CA Datacom/DB Message Reference Guide*.

Parameter style SQL passes nulls to the procedure as does GENERAL WITH NULLS and DATACOM SQL. It also passes these additional four parameters that are added to the end of the parameter/null indicator list:

- The SQLSTATE (INOUT, but always passed in as 00000, similar to the SQLCODE in style DATACOM SQL, that is, it is passed to the procedure as 00000 and used to set the SQLSTATE of the CALL PROCEDURE statement on output).
- Authid.procedure-name (IN, same as in style DATACOM SQL, that is a variable-length character string containing the name of the procedure).
- Authid.specific name (IN, same as in style DATACOM SQL, that is, a variable-length character string reserved for future use).
- Error message text (INOUT, passed in as 0-length string, same as DATACOM SQL, that is, a variable-length character string containing 80 blanks on input, and an 80-byte or shorter error message on output that is placed in the SQLCA and used as the SQL error message for the CALL PROCEDURE statement).

Unlike style DATACOM SQL, the CA Datacom/DB external and internal return codes are not a part of this parameter list but are encoded in the generated SQLSTATE value. For example, the SQLSTATE that equates to SQL return code -117 is S*eeii*, and the SQLSTATE that equates to SQL return code -118 is R*eeii*, where *ee* represents the 2-byte external CA Datacom/DB return code, and *ii* is the CA Datacom/DB internal return code in hexadecimal characters.

Valid Entries:

GENERAL, GENERAL WITH NULLS, DATACOM SQL, SQL

Default Value:

DATACOM SQL

How procedure errors are handled depends on the PARAMETER STYLE specified in the CREATE PROCEDURE statement. See the parameter style information in the error handling section in the *CA Datacom/DB SQL User Guide*.

data-access

Data access is part of the *proc-attributes* (procedure attributes) definition. If you are using SQL Procedures, *data-access* is ignored.

Valid Entries:

MODIFIES SQL DATA, READS SQL DATA, CONTAINS SQL, NO SQL

Default Value:

(No default)

proc-SQL-stmt

A *proc-SQL-stmt* is part of the *proc-body* syntax. The *proc-SQL-stmt* is composed of SQL Procedure statements that are coded and executed directly. Use the compound statement to create program-like logic. For details about compound statements, see Compound Statement (see page 558).

In statements that are within SQL Procedures, that is, not External Procedures, SQL parameters and SQL variables can be referenced anywhere expressions are allowed. Colons are not allowed in these references. When a parameter or variable is referenced in a context containing or within the scope of an identically named variable or column, you must disambiguate the name as follows:

For parameters, use the following:

```
authid.SQL-proc-name.SQL-parameter-name
```

The *authid* is optional if the procedure name is unique in this particular context. For variables, use the *start-label* of the compound statement as follows:

```
start-label.variable-name
```

If your procedure fails to avoid ambiguity as required, CA Datacom/DB attempts to resolve the ambiguity in the following way. If the statement involves a table or view, an attempt is made to find a matching column name. All currently executing contexts are next searched for a matching SQL variable, inner-most context first (inner-most active compound-statement, including active condition handlers). The inner-most SQL parameter list is then searched.

We recommend you do not label a compound statement using the SQL Procedure name. If you do so, know that we do not guarantee any specific result or its consistency over time.

Recursive procedure calls are supported. Each occurrence has separate copies of parameters and SQL variables.

The statements in a *proc-SQL-stmt* are as follows:

executable-DML-stmt

The *executable-DML-stmt* is any DML (Data Manipulation Language) statement, except for DECLARE CURSOR. Use DML statements to access and manipulate data in SQL tables.

DDL-stmt

The *DDL-stmt* (Data Definition Language statements) are used to define SQL objects such as tables and views. DDL statements are only allowed when they do not interfere with the preparation, execution, or rebinding of your procedure.

control-stmt

Part of the *proc-SQL-stmt* clause that is used only with SQL Procedures.

Following are the control statements you can choose:

call-proc Part of the *control-stmt* in the *proc-SQL-stmt* that is used only with SQL Procedures.

execute-proc Part of the *control-stmt* in the *proc-SQL-stmt* that is used only with SQL Procedures.

proc-only-control-stmt Part of the *control-stmt* in the *proc-SQL-stmt* that is used only with SQL Procedures. The *proc-only-control-stmt* can be a compound statement (*compound-stmt*) or compound statement control statements (*compound-stmt-only-control-statements*).

compound-stmt For information about compound statements, see Compound Statement (see page 558).

compound-stmt-only-control-statements Compound statement only control statements can appear only within the context of a compound statement. They can be nested. They can appear immediately inside a compound statement, or inside other statements that are themselves contained by a compound statement. Choices include the following statements:

assignment-stmt (see Assignment Statement (see page 553))

case-stmt (not a case-expression, see CASE Statement (see page 555))

if-then-stmt (see IF-THEN Statement (see page 569))

iterate-stmt (see ITERATE Statement (see page 570))

leave-stmt (see LEAVE Statement (see page 570))

loop-stmt (see LOOP Statement (see page 571))

while-do-stmt (see WHILE Statement (see page 583))

repeat-stmt (see REPEAT-UNTIL Statement (see page 575))

Notice the recursive nature of these definitions. Each of these *proc-only-control-stmt* statements qualify indirectly as a *proc-SQL-stmt* (each *proc-only-control-stmt* is a *control-stmt*), but many of them can also have lists of *proc-SQL-stmt* statements embedded in them. This means that the various statement types may be nested within each-other to an unlimited depth. Notice however the following limitations.

DDL statements (Data Definition) are only allowed when they do not interfere with the preparation, execution, or rebinding of your procedure. For example, if your procedure creates a table and then uses it, the statements that use the table might fail with TABLE NOT FOUND or a similar error during the table resolution process of the prepare phase, unless that table is manually created before the CREATE PROCEDURE is prepared (it can be dropped afterwards). If your procedure uses and then ALTERs or DROPs a table, the procedure could be marked invalid as it executes, then attempt an auto-rebind (which could fail, depending on the change you made and how the table is used by the procedure).

diagnostics-stmt

Part of the *proc-SQL-stmt* clause that is used only with SQL Procedures. See the GET DIAGNOSTICS statement on GET DIAGNOSTICS Statement (see page 566).

EXTERNAL

Begins the external body syntax that can be used to specify a name (NAME *ext-name*) for the program you write that constitutes the body of the procedure (see following for more information on NAME *ext-name*). EXTERNAL is only used for External Procedures, not for SQL Procedures.

A parameter style can also be specified as part of the EXTERNAL syntax but only if you have not specified a parameter style as part of the *proc-attributes* (procedure attribute) syntax. Parameter style is only used for External Procedures, not for SQL Procedures.

NAME *ext-name*

(Optional) If you specify EXTERNAL NAME *ext-name*, the external name you specify as the *ext-name* must match both the *load-module* name and PROGRAM-ID (Datadictionary PROGRAM entity-occurrence name) of the program. We recommend you specify a NAME *ext-name* to avoid confusion. NAME **ext-name** is only used for External Procedures, not for SQL Procedures.

Valid Entries:

A valid external name that meets the listed requirements

Default Value:

If you do not specify a NAME, the SQL name of the procedure is used as the default which in this case must be unique even without an AUTHID, and must be a valid load-module name.

RUN OPTIONS *run-options-string*

(Optional) Specifying RUN OPTIONS *run-options-string* provides control over the environment in which the procedure executes. The *run-options-string* is a quoted string constant of up to 128 characters in length containing run options for use by the IBM OS/390 Language Environment (LE), which controls procedure execution. Following is an example:

```
RUN OPTIONS 'MSGFILE(SYSOUT,FBA,121,0,ENQ)'
```

Using the MSGFILE option prevents unmanaged contention for the SYSOUT data set (SYSLST in z/VSE) from procedures running concurrently. The ENQ parameter allows competing procedure executions (running in distinct LE environments) to share the data set by single-threading access.

The destination of print statements (for example, printf in C or DISPLAY in COBOL), executed in procedures across the Multi-User Facility, tends to be a single data set, usually DDNAME SYSOUT (SYSLST in z/VSE), identified by the Multi-User Facility startup JCL. For that reason, *we recommend using these statements with caution.* Assuming no print statements have been coded, use of MSGFILE is still useful, however, in enqueuing the messages that LE itself occasionally produces.

Valid Entries:

A quoted string constant of up to 128 characters in length containing run options for use by the IBM OS/390 Language Environment

Note: Details about the contents of the string are documented in the IBM *OS/390 Language Environment for OS/390 & VM Programming Reference.*

Default Value:

(No default)

Diagnostics and Condition Handling

Understanding the concepts of a Diagnostics Area (see following section) and condition handler (see Condition Handler (see page 550)) provides you with the ability to use the full capabilities of SQL Procedure condition handling.

Diagnostics Area

A Diagnostics Area is a data structure that represents a list of errors, warnings, and user-defined conditions that have occurred during the execution of a single SQL statement.

SQL maintains an array of Diagnostics Areas, known as a Diagnostics Area Stack, for each logical unit of work (LUW). At any given time, the stack represents the error status of both the currently-active user request and any related condition handler executions (see following) or of the previous request if a request is not currently active.

Each Diagnostics Area contains a header containing information such as a description of the SQL statement that was executed, and an array of condition information areas (CIAs), each area representing a single error, warning, or user-defined condition that occurred during execution of the statement. The number of CIAs that are maintained in each Diagnostics Area is determined by the SQL_COND_INFO_AREAS Multi-User startup option. The amount of memory used by a Diagnostics Area can be computed as follows, where active LUWs is the number of concurrently active LUWs and CIAs is the number of CIAs as specified by SQL_COND_INFO_AREAS:

```
9k * (active LUWs) * (CIAs)
```

You can obtain an estimate of the number of concurrent SQL LUWs by querying the LUWS column in the SQL_STATUS (SQS) Dynamic System Table. Assuming you have 50 SQL applications running concurrently, and SQL_COND_INFO_AREAS is 2, then additional memory usage would then become:

```
9k * 50 * 2 = 900k
```

There are four ways to populate a Diagnostics Area. The first way is to simply experience an error or warning in an SQL statement. The second, third, and fourth ways are to execute the statements SIGNAL (see SIGNAL Statement (see page 579)), RESIGNAL (see RESIGNAL Statement (see page 577)), and RAISE ERROR (see RAISE ERROR Statement (see page 573)). To retrieve diagnostics information in your application, execute the GET DIAGNOSTICS statement (see GET DIAGNOSTICS Statement (see page 566)).

Condition Handler

A condition handler is a user-written routine embedded in a user-Witten procedure that detects and acts to recover from errors, warnings, and any user-defined condition that requires action or remediation. When an end-user merges the functionalities provided by condition handlers and the Diagnostics Area Stack, basic error recovery can be provided, with complete sequences of errors available for retrieval, and business-rule related functions can be performed automatically. The condition declaration and condition handler are found in the syntax for the compound statement (see Compound Statement (see page 558)).

A procedure that, for example, takes customer orders from a phone operator, checks inventory, and reserves warehouse stock to cover them should also, if every item is in stock, ship the orders immediately. A customer, however, could in addition request that individual items be shipped as they become available. The logic of a condition handler could therefore include that situation as follows (for details about condition handling, see Condition Handler (see page 550). To implement the procedure just described, do the following:

1. Declare names for the conditions to be handled:

   ```
   DECLARE ALL_ITEMS_IN_STOCK CONDITION;
   DECLARE SHIP_AS_AVAILABLE CONDITION;
   ```

   Lists of SQLSTATEs can also be named as conditions using a FOR clause that is not included in this example, but can be seen in the example starting in Sample Procedure 2. General conditions of SQLEXCEPTION, SQLWARNING, and NOT FOUND are automatically defined (see the description of *cond-declarations*).

2. Define the condition handler (see the two examples that follow).

   This example sends a message to initiate action by warehouse personnel. In this example, a compound statement is supplied that includes SQL variable references and some logic. A condition handler, however, could also consist of only one SQL statement, as shown in the example that follows this one.

```
DECLARE CONTINUE HANDLER FOR ALL_ITEMS_IN_STOCK, SHIP_AS_AVAILABLE
ship_it: BEGIN ATOMIC
           DECLARE messageLocal VARCHAR(80);
           IF numberItemsAvailable = numberItemsOrdered THEN
              SET messageLocal = 'ORDER IS COMPLETE.  SHIP NOW.';
           ELSE
               SET messageLocal = 'SHIP ALL AVAILABLE ITEMS NOW.';
           END IF;
           INSERT INTO warehouseToDo (orderId, instructions)
                       VALUES  (orderIdParameter, messageLocal);
         END end_ship_it;
```

   The following example performs essentially the same function as the example just shown. It presents, however, an alternative to the more complicated logic of the previous example by using an INSERT statement coded into the warehouseToDo table.

```
DECLARE CONTINUE HANDLER FOR ALL_ITEMS_IN_STOCK, SHIP_AS_AVAILABLE
  INSERT INTO warehouseToDo (orderId, instructions)
              VALUES (orderIdParameter, 'SHIP ALL AVAILABLE ITEMS NOW');
```

3. When you detect the defined condition, signal this fact to the condition handler. If, for example, the procedure determines that some items are not in-stock, but the client has requested that items ship as they become available, the following statement could be executed to trigger the handler to execute.

```
SIGNAL SHIP_AS_AVAILABLE;
```

   Note: Condition handlers triggered by particular SQLSTATEs execute automatically and therefore have no need for explicit SIGNAL statements.

For examples of error handlers embedded in procedures, see the example procedure starting in Sample Procedure 2

Condition Handlers Optional

Because condition Handlers are optional, you can have the caller of a procedure handle any error condition by not coding any handlers. When an error is then encountered, your procedure backs out any maintenance it performed and aborts. You need handlers for NOT FOUND and SQLWARNING, however, if you do not want completion codes to cause a procedure to abort, for example if you have an SQLSTATE '02000' or SQLCODE 100 at the end of each cursor and (or) warning conditions such as when an SQLSTATE starts with '01' or there is a positive SQLCODE. For an example showing a procedure without handlers, see Sample Procedure 1.

SQLSTATE and SQLCODE Special Variables

As an alternative to using the GET DIAGNOSTICS statement in a handler to retrieve the SQLSTATE into an SQL variable, and for compatibility with other implementations, supply an SQLSTATE or SQLCODE local variable and code the following as the first statement in your handler:

```
SET SQL-variable-name = sqlstate-or-sqlcode;
```

SQL then assigns a value to your SQLSTATE or SQLCODE variable after every statement execution. Instructions for using this feature follows:

1. Declare SQLSTATE (or SQLCODE) as an SQL variable in your condition handler or a compound statement that contains the handler.
2. As the first (or only) statement in a handler, assign SQLSTATE (or SQLCODE) to your SQL variable. Again, note that if you supply an SQLSTATE (or SQLCODE) variable, Datacom will reset its' value each time a statement is executed, hence the need for a separate SQL variable copy of any value you need to save.

For example, from a functional point of view code example 1 (following on this page) and code example 2 (see Code Example 2 (see page 553)) are equivalent. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

Code Example 1

```
.
.
.
DECLARE sqlState, sqlStateLocal CHAR(5) DEFAULT '00000';
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND
   SET sqlStateLocal = sqlState;
.
.
.
INSERT INTO orderTable VALUES (customerId, orderId, orderStatus, orderComments);
IF sqlStateLocal <-> '00000' THEN
   LEAVE thisCompoundStatement;
END IF;
.
.
.
```

In the example just shown, you could have substituted for every occurrence of sqlCode and sqlCodeLocal the variables sqlState and sqlStateLocal. We do not recommend using SQLSTATE and SQLCODE at the same time, because each statement, including assignment statements, reset both the SQLSTATE and SQLCODE, meaning that the following situation could result. Consider the following two statements:

```
SET sqlCodeLocal = sqlCode;
SET sqlStateLocal = sqlState;
```

If the two statements just shown should be the first and second statements in a handler, SQL variable sqlStateLocal would receive '00000' as the (reset) value. Because the condition information in the Diagnostics Area is preserved until the handler succeeds, the following would work as a coding alternative:

```
SET sqlCodeLocal = sqlCode;
GET STACKED DIAGNOSTICS sqlStateLocal = RETURNED_SQLSTATE;
```

Following is code example 2, which is equivalent from a functional point of view, to the previously shown code example 1. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

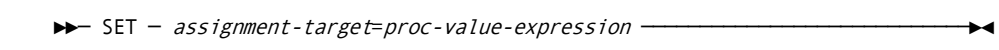
Code Example 2

```
.
.
.
DECLARE sqlState, sqlStateLocal CHAR(5) DEFAULT '00000';
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND
   GET STACKED DIAGNOSTICS sqlCodeLocal =
RETURNED_SQLSTATE;
.
.
.
INSERT INTO orderTable VALUES (customerId, orderId, orderStatus, orderComments);
IF sqlStateLocal <-> '00000' THEN
   LEAVE thisCompoundStatement;
END IF;
.
.
.
```

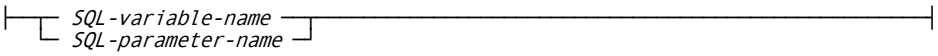
Assignment Statement

The assignment statement stores a value into an SQL variable or SQL parameter.

Following is the syntax for the assignment statement:



Expansion of Where assignment-target is defined as



assignment-target

For the *assignment-target* specify an *SQL-variable-name* or an *SQL-parameter-name.*

proc-value-expression

For a description of a *proc-value-expression,* see CASE Statement (see page 555).

SQL-variable-name

The *SQL-variable-name* is the name of a variable that is declared within a compound statement inside a SQL Procedure (a LANGUAGE SQL procedure). If the name conflicts with (matches) another SQL variable name from a nested compound statement (for example, a condition handler), or an SQL parameter name, or a column contained within a referenced table or view, the SQL variable name should be qualified using the start-label of the compound statement that immediately contains it.

SQL-parameter-name

The *SQL-parameter-name* is required for SQL Procedures and, though optional for all others, is recommended for all others as well. The *SQL-parameter-name* is the name of a parameter passed to a SQL Procedure (a LANGUAGE SQL procedure). When used in a SQL Procedure containing an SQL variable with a conflicting (matching) name, or a table or view reference where the table or view contains a conflicting column name, the name should be qualified by using the procedure name.

Example

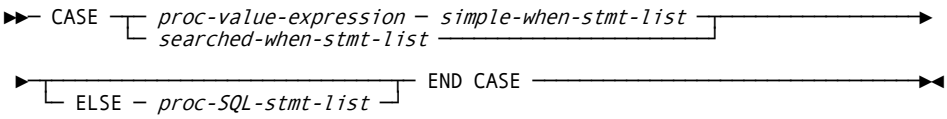
An assignment statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
.
.
.
DECLARE mySalary, yourSalary decimal(15,0) DEFAULT 1000000.;
DECLARE errorClass char(2);
DECLARE sqlStateLocal char(5);
.
.
.
SET errorClass = SUBSTR(sqlStateLocal, 1, 2);
.
.
.
SET mySalary = yourSalary * 2;
.
.
.
```

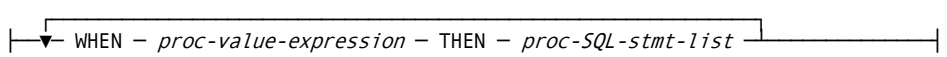
CASE Statement

The CASE works in the same way an IF statement works to control which SQL statements, based on predicates you specify, are executed.

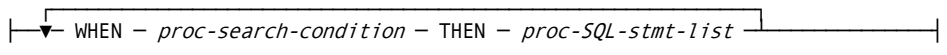
Following is the syntax for the CASE statement:



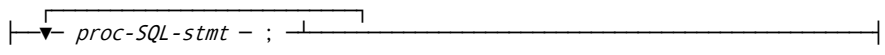
Expansion of Where simple-when-stmt-list is defined as



Expansion of Where searched-when-stmt-list is defined as



Expansion of Where proc-SQL-stmt-list is defined as



Note: For *proc-SQL-stmt* syntax, see the syntax fragment in CREATE PROCEDURE Syntax and Description (see page 538).

proc-value-expression

A *proc-value-expression* is a variant form of the expressions used to specify values (see the information about expressions in the *CA Datacom/DB SQL User Guide*) as follows:

- To the syntax for expressions, the *proc-value-expression* adds SQL variable names (see *SQL-variable-declarations* in Compound Statement (see page 558)) and SQL parameter names (see *SQL-parameter-name* in CREATE PROCEDURE Syntax and Description (see page 538)). To view SQL variable names and SQL parameter names in the syntax diagram for expressions, see the information about expressions in the *CA Datacom/DB SQL User Guide*.
- A *proc-value-expression* does not allow the use of column functions.

  Note: Scalar functions, however, are allowed.
- A *proc-value-expression* does not allow the use of column names.
- A *proc-value-expression* does not allow the use of host variables.

simple-when-stmt-list

A *simple-when-stmt-list* is a list of simple-when statements, each terminated by a semicolon. For more information, see the example that follows.

searched-when-stmt-list

A *searched-when-stmt-list* is a list of searched-when statements, each terminated by a semicolon. For more information, see the example that follows.

proc-search-condition

The *proc-search-condition* specifies a condition that is true, false, or unknown about a row. The *proc-search-condition* is similar to the *search-condition* described in the *CA Datacom/DB SQL User Guide* with the following modifications that allow its use in SQL Procedures:

- SQL parameter and SQL variable references are allowed instead of column and host-variable references.
- Aggregate functions and any predicate involving a sub-query are not allowed.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon.

Examples

Following is an example that uses a simple-when statement:

```
CASE creditStatus
   WHEN 'CURRENT' THEN
      call creditLimitCheck(clientId, creditAmount, creditLimit, orderStatus);
      IF (orderStatus = 'CREDIT LIMIT EXCEEDED')
         SET orderStatus = orderStatus CONCAT ', DISCUSS CREDIT LINE INCREASE';
      END IF;
   WHEN 'NEW APPLICATION' THEN
      set orderStatus = 'ACCEPT ORDER PENDING ACCOUNT SETUP COMPLETION';
   WHEN 'PAST DUE' THEN
      set orderStatus = 'NOTIFY CUSTOMER, HOLD FOR PAYMENT RECEIPT';
   WHEN 'BAD' THEN
      set orderStatus = 'REJECTED FOR FAILURE TO PAY';
END CASE;
```

Following is an example that uses a searched-when statement:

```
CASE
   WHEN creditStatus = 'CURRENT' OR clientStatus = 'PRE-APPROVED' THEN
      call creditLimitCheck(clientId, creditAmount, creditLimit, orderStatus);
      IF (orderStatus = 'CREDIT LIMIT EXCEEDED')
         SET orderStatus = orderStatus CONCAT ', DISCUSS CREDIT LINE INCREASE';
      END IF;
   WHEN creditStatus = 'NEW APPLICATION' THEN
      set orderStatus = 'ACCEPT ORDER PENDING ACCOUNT SETUP COMPLETION';
   WHEN creditStatus = 'PAST DUE' THEN
      set orderStatus = 'NOTIFY CUSTOMER, HOLD FOR PAYMENT RECEIPT';
   ELSE
      set orderStatus = 'CREDIT IS ' CONCAT orderStatus;
END CASE;
```

Compound Statement

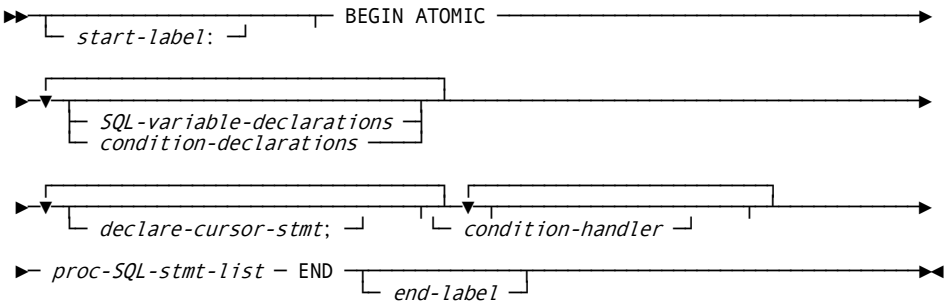
Compound statements implement a small procedural language that provides the building blocks from which SQL Procedures are composed. A compound statement can contain a large number of interacting SQL statements that are limited only by the size of the RWTSA that must transport the syntax to the DBMS. For details about size adjustment to the RWTSA, see the information about the TASKS Multi-User startup option in the *CA Datacom/DB Database* and *System Administration Guide*. in the *CA Datacom/DB Database and System Administration Guide*. When a compound statement is executed, an execution context is created for the statements the compound statement contains. That execution context consists of the following:

- SQL variables and condition handlers for both this compound statement and any statements within which this compound statement is nested
- A save-point token for transaction backout (controlled by the DBMS)
- SQL parameters
- Labels defining SQL statements (used for branching)
- Locally declared cursors

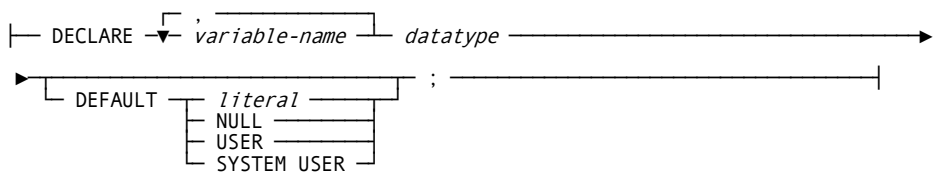
The set of statements to which variables and other context-related information is visible is referred to as scope. Scope consists of all statements and contexts enclosed by a compound statement. Compound statements cannot exist outside of a procedure, but they can be nested. Nested compound statements are most frequently used for condition handling.

Following is the syntax for the compound statement:

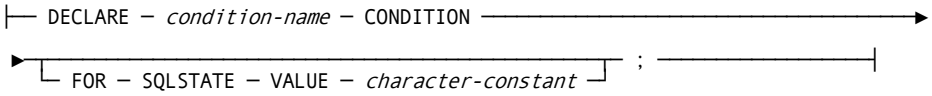
Note: Combinations and multiples of these two optional choices can be used.



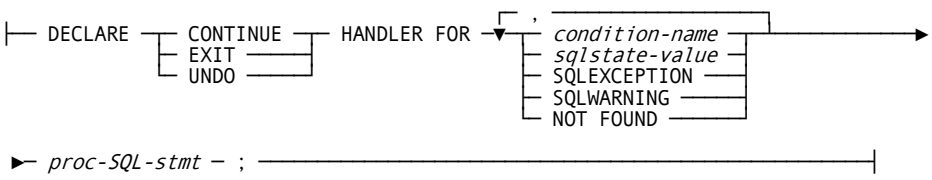
Expansion of Where SQL-variable-declarations are defined as



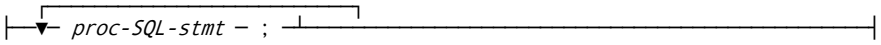
Expansion of Where condition-declarations are defined as



Expansion of Where condition-handler is defined as



Expansion of Where proc-SQL-stmt-list is defined as



Following are descriptions of the parts of the compound statement syntax:

start-label*: / *end-label

(Optional) A *start-label*: is an SQL identifier (followed by a colon) that can be used in various flow-control statements to mark the destination of a branch. When you specify an *end-label* you must also specify a matching *start-label.*

We recommend you do not label a compound statement using the SQL Procedure name. When the SQL Procedure name is used, we cannot guarantee any specific result or its consistency over time.

ATOMIC

Specifying ATOMIC prevents a procedure from containing a COMMIT or a non-savepoint ROLLBACK. Specify ATOMIC after BEGIN and specify an END before the *end-label.*

SQL-variable-declarations

(Optional) In *SQL-variable-declarations* each variable receives the default value on entry to a compound statement. Variables without defaults are assigned NULL values on entry. Each declaration can include a list of variables separated by commas. The *variable-name* is an SQL identifier. For details about SQL identifiers, see the *CA Datacom/DB SQL User Guide*.

The scope (range of visibility) of these variables is the BEGIN-END pair, including any compound statements contained within and any error handlers that execute while the contained compound statement is executing.

Identical variable names can be disambiguated using *start-label.variable-name* where the start label is the *start-label* of a compound statement.

DEFAULT *literal*/NULL/USER/SYSTEM USER

(Optional) The DEFAULT clause, an optional part of the *SQL-var-declarations* clause, specifies a default from among the choices that follow:

literal

Specifies the default to be a literal. Be certain the literal agrees with the data type of the column. The specified literal can be up to 20 bytes long or the length of the column involved, whichever is shorter. You can specify a default value for a character column where the column is greater than 20 bytes long, but the default literal itself is limited to 20 bytes with the remaining bytes padded with blanks by the system.

NULL

Specifies the default as NULL.

USER

Specifies the default to be the current authorization ID.

SYSTEM USER

Specifies the default to be the accessor ID of the currently signed-on user.

condition-declarations

(Optional) Each of the *condition-declarations* relates a name to an SQLSTATE or any user-signaled condition (see the information on the SIGNAL statement on SIGNAL Statement (see page 579)). A condition handler can be written for any declared condition, including those defined by the user and not related to any SQLSTATE. Declared conditions are visible to all compound statements executing within their context. For instructions and examples of conditions and handlers, see Diagnostics and Condition Handling (see page 548).

Note: The names NOT FOUND, SQLWARNING, and SQLEXCEPTION are pre-defined conditions assigned to SQLSTATE subclasses 02, 01, and any other non-00 subclass, respectively.

condition-handler

(Optional) The *condition-handler* defines an SQL statement (a *proc-SQL-stmt* that can be a compound statement) to be executed when a given condition occurs. If a condition occurs for which multiple handlers have been written, SQL chooses and executes the most appropriate handler, after searching available handlers in the inner-most context first, as follows.

1. If an SQLSTATE other than 00000 has occurred and a handler exists in any active context for a condition defined for that specific SQLSTATE, that handler executes. This includes situations where the SQLSTATE was signaled by value or using a *condition-name* associated with a specific SQLSTATE as described in the information about the SIGNAL statement in SIGNAL Statement (see page 579). A *condition-name* is an SQL identifier.
2. If a condition was signaled using a *condition-name* and the first step did not find a match on SQLSTATE, or if there is no SQLSTATE associated with the *condition-name,* a search is done for a handler associated with the specific *condition-name.*
3. If an SQLSTATE other than 00000 has occurred (including being signaled) and a handler for the generalized condition associated with that SQLSTATE, specifically, SQLEXCEPTION, SQLWARNING, or NOT FOUND, exists, that handler is executed.

If no appropriate handler is found, the SQL statement precipitating the error or warning condition aborts with that condition.

Specific SQLSTATEs and condition names are prevented by CA Datacom from being associated with more than one handler, except when the duplicate handler applies only to a generalized condition under which that SQLSTATE is classified, preventing an ambiguity by following the previously given rules. For instructions and examples of conditions and handlers, see Diagnostics and Condition Handling (see page 548).

After execution of the handler-action statement(s) without any errors or warnings, the SQLSTATE is reset to 00000 and procedure execution continues as separately described for each of the handler types, CONTINUE, EXIT, and UNDO.

If an error handler generates an unexpected (rather than signaled) error or warning condition, that condition is handled like any other error. To avoid endless loops in error handling, we only handle errors generated in the top level error handler. Unhandled error handler errors cause a RESIGNAL command to execute, followed by an exit from the handler and execution continuation as prescribed by any handler executed on the RESIGNAL. On exit from the handlers, the DIAGNOSTICS AREA STACK (see GET DIAGNOSTICS in GET DIAGNOSTICS Statement (see page 566)) contains information on any unresolved errors. Unresolved errors cause an abort of the executing procedure statement.

If a SIGNAL, RESIGNAL, or RAISE ERROR statement is executed as part of a handler, CA Datacom/DB assumes that the signaled condition is intended to be seen by the caller of the procedure, and CA Datacom/DB therefore exits with that error intact and unhandled. After successful execution of a CONTINUE type handler, procedure execution resumes with the statement following the one that generated the error.

EXIT handlers operate in a similar way except that when they complete, procedure execution continues after execution of an implied LEAVE statement (see LEAVE Statement (see page 570)) whose target-label is the one attached to the compound statement in which the handler was defined. The result is that the compound statement containing the statement generating the error aborts but with no error.

UNDO handlers start by performing a ROLLBACK that terminates at the save-point that SQL automatically establishes at the start of the compound statement in which the handler was defined, and closing all cursors open in that context. The handler-action statements then execute. If no unhandled conditions occur, then the condition precipitating the handler execution is cleared, and procedure execution continues after execution of an implied LEAVE statement (see LEAVE Statement (see page 570)) whose target label is the one attached to the compound statement under which the handler was defined. In other words, the compound statement containing the statement generating the error aborts but with no error.

Note: SIGNAL, RAISE ERROR, and RESIGNAL statements that execute inside handlers do not activate additional condition handling. They instead cause the handler to be exited with the signaled condition and cause the execution of the compound statement that triggered the handler to abort. In addition, the compound statement that contains the handler definition is aborted if it is different from the compound statement that triggered the handler, the same as would happen if the handler failed to resolve the triggering condition. The SIGNAL, RAISE ERROR, and RESIGNAL statements should therefore be positioned as the last statements in your condition handler.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon.

Example

Following is a compound statement example used in a procedure named creditLimitCheck. In the example, the compound statement begins with line number 000017 and ends with line number 000030:

```
000013 -- Does the customer have enough credit to purchase an ordered part?
000014 CREATE PROCEDURE creditLimitCheck
000015  (INOUT result CHAR(80), IN custId CHAR(5), IN purchaseAmount DECIMAL(15,2))
000016   LANGUAGE SQL
000017   limitCheck: BEGIN ATOMIC
000018      DECLARE creditAvailable DECIMAL(15,2) DEFAULT 0;
000019
000020      SELECT creditMax-creditUsed INTO creditAvailable
000021             FROM credit
000022             WHERE credit.custId =
000023                   CAST(creditLimitCheck.custId AS NUMERIC(5));
000024
000025      IF (purchaseAmount > creditAvailable) THEN
000026         SET result = 'CREDIT LIMIT EXCEEDED';
000027      ELSE
000028         SET result = 'CREDIT APPROVED';
000029      END IF;
000030   END limitCheck@
```

On line 00030 of the previously shown example, the at sign (@), used in conjunction with the TERM=@ parameter in DBSQLPR, enables DBSQLPR to skip over semicolons embedded in statements and instead recognize the @ as the end of the statement.

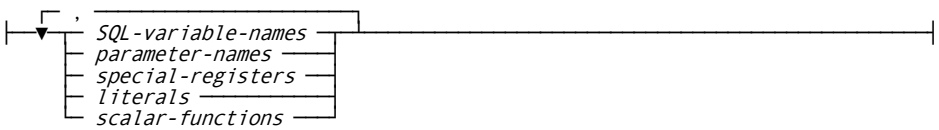
Note: For more information about TERM=, see the information about DBSQLPR in the *CA Datacom/DB SQL User Guide*.

DATACOM DUMP Statement

The DATACOM DUMP statement can be used as a debugging tool for SQL Procedures. This statement causes requested information to be dumped to a specified file and marked by a header and footer as shown in the first and last lines in the example in Example (see page 565). The plan name that is shown in the dump report matches the SQL-name of your procedure.

►►– DATACOM DUMP – *dumpable-expression-list* – TO PXXSQL ————————————►◄

Expansion of Where dumpable-expression-list is defined as



dumpable-expression-list

A list, separated by commas, of *SQL-variable-names, parameter-names, special-registers, literals,* and *scalar-functions.*

TO PXXSQL

PXXSQL is the destination supported by the DATACOM DUMP statement. TO PXXSQL causes output to go to the PXX location to which SQL output is sent. The destination of SQL output is either specified using the Multi-User startup option SYSOUT (see the *CA Datacom/DB Database and System Administration Guide*) or allowed to default.

SQL-variable-names

The *SQL-variable-names* is the name of a variable that is declared within a compound statement inside a SQL Procedure (a LANGUAGE SQL procedure). If the name conflicts with (matches) another SQL variable name from a nested compound statement (for example, a condition handler), or an SQL parameter name, or a column contained within a referenced table or view, the SQL variable name should be qualified using the start-label of the compound statement that immediately contains it.

parameter-names

The *parameter-names* variable refers to any parameter in your procedure.

special-registers

For information about special registers, see the *CA Datacom/DB SQL User Guide*.

literals

For information about literals, see the *CA Datacom/DB SQL User Guide*.

scalar-functions

For information about scalar functions, see the *CA Datacom/DB SQL User Guide*.

Note: When using scalar functions, operands can include SQL variables and parameters but not column names. Fetching the value of a column into a variable is a suggested way to use a scalar function.

Example

If you add a tracesOn input parameter or SQL variable to your procedure, you can use it to turn DATACOM DUMP statements on and off, preventing your having to recode the statements if procedure modifications require debugging. For example:

```
IF tracesOn = 'TRUE' THEN
  DATACOM DUMP dumpable-expression-list TO PXXSQL;
END IF;
```

DATACOM DUMP produces output that goes to the PXXSQL and is printed in the detailed format shown in the following sample. In this context, the dumping of constants, for example VALUES ON EXIT in the sample, can help find the output on your test or development MUFs. We reserve the right, however, to change the format of dumps without notice. For example, a less detailed, summary style dump that produces simpler output could be provided at a later time, a summary dump style that could even, at that time, become the default style.

Following is the DATACOM DUMP statement that produces the sample output that follows:

```
DATACOM DUMP 'VALUES ON EXIT:', result, sqlstateLocal,
                                        errorMessage TO PXXSQL;
```

```
000017 S0001 -- DCM DUMP PLAN SYSADM.SUPPLYHOSTVARSFORC STMT 5 LUW 0000000000000001 REQ 305 TASK 1 8/3/2007 17:39:02 --
000018 S0001   LVL 2 NODE 3684D16C OP=00000000 NXT=3684D1A8  QCB=00000000 VAL=3684D15C NAM=00000000
000019 S0001   ... . PROCTYPE=0x0000  STEP=-01  OPER=000    FLAGS=0x0,0x0
000020 S0001   ... . LITERAL (#0021:00000.00015): CHAR (15) NOT NULL
000021 S0001   DCM . CONTENTS= VALUES ON EXIT:.
000022 S0001   LVL 2 NODE 3684D1A8 OP=00000000 NXT=3684D266 QCB=3684D068 VAL=3684254F NAM=00000000
000023 S0001   ... . PROCTYPE=0x0000  STEP=01  OPER=000    FLAGS=0x0,0x0
000024 S0001   ... . LCLVARREF (#1027:00000.00081): CHAR (80)
000025 S0001   DCM . NAME: ..RESULT
000026 S0001   DCM . CONTENTS= CREDIT LIMIT EXCEEDED
000027 S0001   LVL 2 NODE 3684D266 OP=00000000 NXT=3684D324  QCB=3684D068 VAL=36842722 NAM=00000000
000028 S0001   ... . PROCTYPE=0x0000  STEP=01  OPER=000    FLAGS=0x0,0x0
000029 S0001   ... . LCLVARREF (#1025:00000.00006): CHAR (5)
000030 S0001   DCM . NAME: ..SQLSTATELOCAL
000031 S0001   DCM . CONTENTS= NULL
000032 S0001   LVL 2 NODE 3684D324 OP=00000000 NXT=3684D446L QCB=3684D068 VAL=3684244B NAM=00000000
000033 S0001   ... . PROCTYPE=0x8000  STEP=01  OPER=000    FLAGS=0x0,0x0
000034 S0001   ... . LCLVARREF (#1026:00000.00081): CHAR (80)
000035 S0001   DCM . NAME: ..ERRORMESSAGE
000036 S0001   DCM . CONTENTS= N/A
000037 S0001 -- END DCM DUMP PLAN SYSADM.SUPPLYHOSTVARSFORC STMT 5 LUW 0000000000000001 REQ 305 TASK 1 8/3/2007 17:39:02 --
```

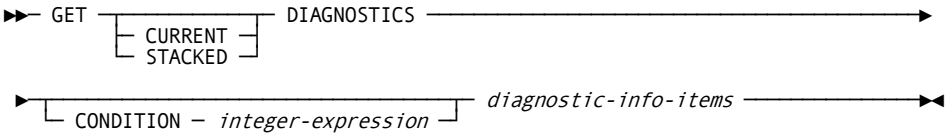
See SIMULATE DATACOM PROCEDURE Statement (see page 582) to see how a DATACOM DUMP statement that produced the previously shown sample report appears within a SIMULATE PROCEDURE statement.

GET DIAGNOSTICS Statement

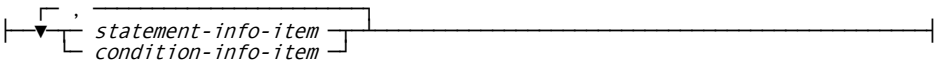
The GET DIAGNOSTICS statement retrieves exception or completion condition information from a Diagnostics Area (for information, see Diagnostics Area (see page 548)).

Note: CURRENT and STACKED are optional choices for specifying which diagnostics area to use.

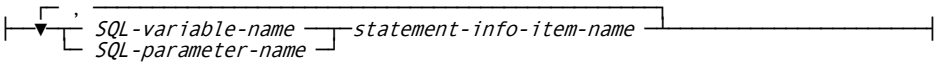
CONDITION followed by an integer is an optional specification for which condition area to use. If you do not specify CONDITION, a default of CONDITION 1 is used (see the description following the syntax).



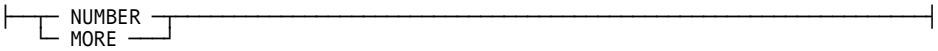
Expansion of Where diagnostic-info-items are defined as



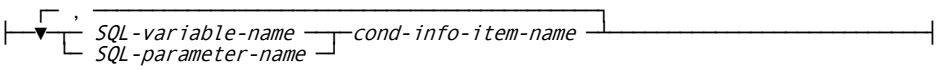
Expansion of Where statement-info-item is defined as



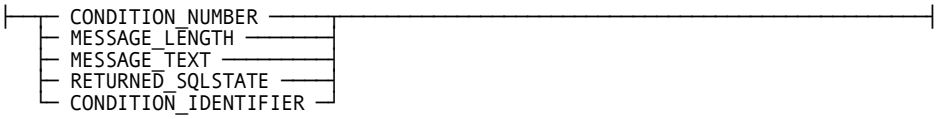
Expansion of Where statement-info-item-name is defined as



Expansion of Where condition-info-item is defined as



Expansion of Where cond-info-item-name is defined as



CURRENT

(Optional) Specifying CURRENT (or omitting this optional specification), to indicate which diagnostics area to use, causes information to be retrieved on the SQL error (or other condition) generated by the SQL statement that was just executed.

STACKED

(Optional) Specifying STACKED inside an error handler retrieves the Diagnostics Area that is related to the condition that caused the handler to be executed. This original Diagnostics Area has been pushed in order to supply an empty current Diagnostics Area for use by any conditions that can be caused by the error handler itself.

CONDITION *integer-expression*

(Optional) CONDITION followed by an integer that can range in value from one (1) to the number of condition areas available, specifies which condition area to retrieve. Within each Diagnostics Area lies a stack of condition information areas, allowing each Diagnostics Area to store information related to any sequence of errors or conditions that could have occurred during the execution of a single statement.

The number of condition areas available is specified by using the Multi-User startup option SQL_COND_INFO_AREAS (for Multi-User startup option information, see the *CA Datacom/DB Database and System Administration Guide*).

If you do not specify CONDITION, the default is CONDITION 1, which specifies using the most recent error condition, that is, the error condition that occurred during the last non-diagnostics related SQL statement that executed.

diagnostic-info-items

The *diagnostic-info-items* variable specifies the information you want to store and where to store it.

statement-info-item

The *statement-info-item* contains information about the statement that was executing when the condition occurred.

statement-info-item-name

A *statement-info-item-name* (the data type listed in parentheses must be compatible with the data type of the variable or parameter into which the value is being stored) can be one of the following:

NUMBER (SMALLINT) NUMBER indicates the number of condition areas that are in use.

MORE (CHAR(1)) A letter Y (for YES) indicates that more conditions than condition areas occurred.

condition-info-item

The *condition-info-item* variable contains information about the condition that occurred.

cond-info-item-name

A *cond-info-item-name* (the data type listed in parentheses must be compatible with the data type of the variable or parameter into which the value is being stored) can be one of the following:

CONDITION_NUMBER (SMALLINT) indicates the condition for which you want to retrieve information. Specifying 1 (a number one) indicates the most recent condition, a 2 specifies the second most recent condition, and so on.

MESSAGE_LENGTH (SMALLINT) indicates the length of the message text information item.

MESSAGE_TEXT (VARCHAR(128)) indicates the text of the error message or signal condition.

RETURNED_SQLSTATE (CHAR(5)) indicates the condition that occurred.

CONDITION_IDENTIFIER (VARCHAR(128)) gives the name, if one was signaled, of a named condition.

Example

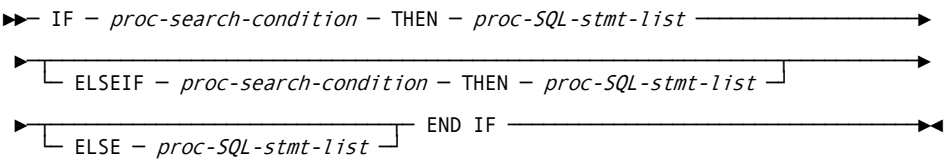
A GET DIAGNOSTICS statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
    .
    .
    .
    DECLARE errMsg varchar(128);
    DECLARE sqlStateLocal char(5) default '00000';
    DECLARE continue HANDLER FOR sqlexception, sqlwarning, not found
       GET stacked DIAGNOSTICS
          sqlStateLocal = RETURNED_SQLSTATE, errMsg = MESSAGE_TEXT;
    .
    .
    .
 startRepeat:
   REPEAT
      FETCH orderCrs INTO orderId, custId, creditReqAmt;
      .
      .
      .
   UNTIL sqlStateLocal <-> '00000'
   END REPEAT DATACOM LOOPLIMIT 100;
   .
   .
   .
```

IF-THEN Statement

The IF-THEN statement controls logic execution flow in a compound statement, that is, it allows you to specify what logic to use, based on some condition that you specify.

Following is the syntax for the IF-THEN statement:



proc-search-condition

The *proc-search-condition* specifies a condition that is true, false, or unknown about a row. The *proc-search-condition* is similar to the *search-condition* described in the *CA Datacom/DB SQL User Guide* with the following modifications that allow its use in SQL Procedures:

- SQL parameter and SQL variable references are allowed instead of column and host-variable references.
- Aggregate functions and any predicate involving a sub-query are not allowed.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon. For information about a *proc-SQL-stmt,* see CREATE PROCEDURE Syntax and Description (see page 538).

Example

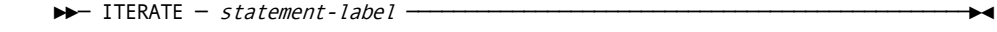
An IF-THEN statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
.
.
.
IF sqlStateLocal <-> '00000' THEN
   LEAVE loopExample;
END IF;
.
.
.
```

ITERATE Statement

The ITERATE statement controls looping logic in a compound statement. Use the ITERATE statement to specify when to skip to the next iteration of a loop.

Following is the syntax for the ITERATE statement:



statement-label

The *statement-label* specifies the *start-label* of a loop, that is, it relates the ITERATE statement to the position at which execution continues.

Example

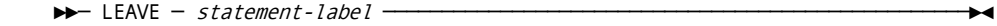
An ITERATE statement example follows. This example skips rows we are not interested in. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
    .
    .
    .
 startWhile:
    WHILE sqlStateLocal <-> '00000' DO
       FETCH orderCrs INTO orderId, custId, creditReqAmt;
       .
       .
       .
       IF creditReqAmt = 0 THEN
          ITERATE startWhile;
       END IF;
       .
       .
       .
    END WHILE endWhile DATACOM LOOPLIMIT 10000;
    .
    .
    .
```

LEAVE Statement

The LEAVE statement in a compound statement specifies that you want to exit the specified statement, which must contain the LEAVE statement.

Following is the syntax for the LEAVE statement:



statement-label

The *statement-label* specifies the *start-label* of the statement that you want to exit.

Example

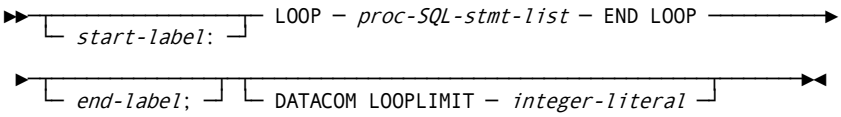
A LEAVE statement example follows. This example exits the loop after a non-zero SQLSTATE. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
      .
      .
      .
      DECLARE continue HANDLER FOR sqlexception, sqlwarning, not found
         GET stacked DIAGNOSTICS
            sqlStateLocal = RETURNED_SQLSTATE, errMsg = MESSAGE_TEXT;
      .
      .
      .
loopExample:
   LOOP
      FETCH orderCrs INTO orderId, custId, creditReqAmt;
      IF sqlStateLocal <-> '00000' THEN
         LEAVE loopExample;
      END IF;
      .
      .
      .
   END LOOP loopExample DATACOM LOOPLIMIT 100;
   .
   .
   .
```

LOOP Statement

The LOOP statement controls looping logic in a compound statement. It provides an unconditional loop that continues until there is an explicit or condition-based (with a carefully designed EXIT or UNDO condition-handler) loop exit. You must therefore be certain to include a LEAVE statement or SIGNAL statement in your design logic. In addition, we recommend that you use the optional DATACOM LOOPLIMIT clause.

Following is the syntax for the LOOP statement:



start-label*: / *end-label

(Optional) A *start-label*: is an SQL identifier (followed by a colon) that can be used in various flow-control statements to mark the destination of a branch. When you specify an *end-label* you must also specify a matching *start-label.* The labels must match.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon. For information about a *proc-SQL-stmt,* see CREATE PROCEDURE Syntax and Description (see page 538).

DATACOM LOOPLIMIT

We recommend that you use DATACOM LOOPLIMIT. DATACOM LOOPLIMIT is a CA Datacom extension that we provide to allow your proactive avoidance of endless loops that might be caused by faulty logic, unexpected or missing column values, or other unforeseen conditions. The Multi-User startup option SQL_DATACOM_LOOPLIMIT can be used to force a default limit on each looping statement. An error is produced if the limit is exceeded.

integer-literal

Defines how many loops are too many.

Example

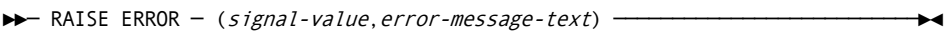
A LOOP statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
     .
     .
     .
     DECLARE continue HANDLER FOR sqlexception, sqlwarning, not found
        GET stacked DIAGNOSTICS
           sqlStateLocal = RETURNED_SQLSTATE, errMsg = MESSAGE_TEXT;
     .
     .
     .
 loopExample:
    LOOP
       FETCH orderCrs INTO orderId, custId, creditReqAmt;
       IF sqlStateLocal <-> '00000' THEN
          LEAVE loopExample;
       END IF;
       .
       .
       .
    END LOOP loopExample DATACOM LOOPLIMIT 100;
    .
    .
    .
```

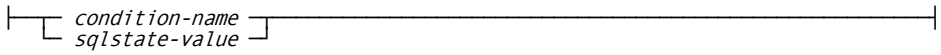
RAISE ERROR Statement

The RAISE ERROR statement is used to signal an error or other condition in the manner of the SIGNAL statement. You must provide a valid SQLSTATE value when using the RAISE ERROR statement.

Note: SIGNAL, RAISE ERROR, and RESIGNAL statements that execute inside handlers do not activate additional condition handling. They instead cause the handler to be exited with the signaled condition and cause the execution of the compound statement that triggered the handler to abort. In addition, the compound statement that contains the handler definition is aborted if it is different from the compound statement that triggered the handler, the same as would happen if the handler failed to resolve the triggering condition. The SIGNAL, RAISE ERROR, and RESIGNAL statements should therefore be positioned as the last statements in your condition handler.

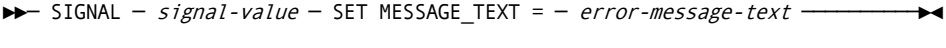


Expansion of Where signal-value is defined as

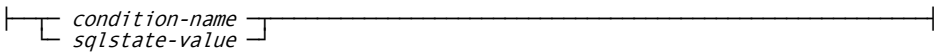


The RAISE ERROR statement functions identically to the following SIGNAL statement syntax:

Note: For more information on the SIGNAL statement, see SIGNAL Statement (see page 579).



Expansion of Where signal-value is defined as



signal-value

(Optional) When a *signal-value* is specified, existing Condition Areas within the current Diagnostics Area are pushed down in the Condition Area stack, that is, Condition Area number one becomes Condition Area number two. Condition Area number one is then populated with any supplied or implied RETURNED_SQLSTATE (supplied using *sqlstate-value*) and (or) CONDITION_IDENTIFIER (supplied using *condition-name*) in addition to any other information supplied in the SET *signal-information* clause. The *statement-information-items* are modified to indicate that a RESIGNAL statement was the last to modify the Diagnostics Area.

condition-name

See the *condition-name* information in the compound statement section that begins in Compound Statement (see page 558).

sqlstate-value

See the *sqlstate-value* information in the compound statement section that begins in Compound Statement (see page 558).

error-message-text

The text of an error message whose occurrence you want to signal an error.

Example

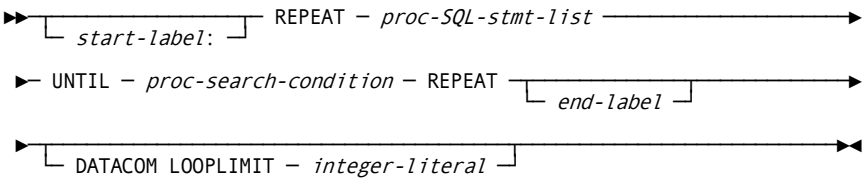
A RAISE ERROR statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
 .
 .
 .
 IF creditReqAmt > creditAvailable
  RAISE ERROR ('2FS04', 'CUSTOMER HAS INSUFFICIENT CREDIT LIMIT TO COMPLETE
PURCHASE');
 END IF;
 .
 .
 .
```

REPEAT-UNTIL Statement

The REPEAT-UNTIL statement defines conditional looping logic in a compound statement. It provides a means to loop until some predicate becomes true. We recommend that you use the optional DATACOM LOOPLIMIT clause with this statement.

Following is the syntax for the REPEAT-UNTIL statement:



start-label*: / *end-label

(Optional) A *start-label*: is an SQL identifier (followed by a colon) that can be used in various flow-control statements to mark the destination of a branch. When you specify an *end-label* you must also specify a *start-label,* and the labels must match.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon. For information about a *proc-SQL-stmt,* see CREATE PROCEDURE Syntax and Description (see page 538).

proc-search-condition

The *proc-search-condition* specifies a condition that is true, false, or unknown about a row. The *proc-search-condition* is similar to the *search-condition* described in the *CA Datacom/DB SQL User Guide* with the following modifications that allow its use in SQL Procedures:

- SQL parameter and SQL variable references are allowed instead of column and host-variable references.
- Aggregate functions and any predicate involving a sub-query are not allowed.

DATACOM LOOPLIMIT

We recommend that you use DATACOM LOOPLIMIT. DATACOM LOOPLIMIT is a CA Datacom extension that we provide to allow your proactive avoidance of endless loops that might be caused by faulty logic, unexpected or missing column values, or other unforeseen conditions. The Multi-User startup option SQL_DATACOM_LOOPLIMIT can be used to force a default limit on each looping statement. An error is produced if the limit is exceeded.

integer-literal

The *integer-literal* specifies the maximum number of times the loop is allowed to execute.

Example

A REPEAT-UNTIL statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

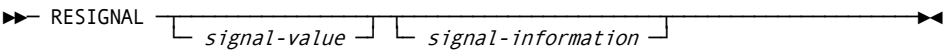
```
    .
    .
    .
    DECLARE continue HANDLER FOR sqlexception, sqlwarning, not found
       GET stacked DIAGNOSTICS
          sqlStateLocal = RETURNED_SQLSTATE, errMsg = MESSAGE_TEXT;
    .
    .
    .
 startRepeat:
    REPEAT
       FETCH orderCrs INTO orderId, custId, creditReqAmt;
       .
       .
       .
    UNTIL sqlStateLocal <-> '00000'
    END REPEAT DATACOM LOOPLIMIT 100;
    .
    .
    .
```

RESIGNAL Statement

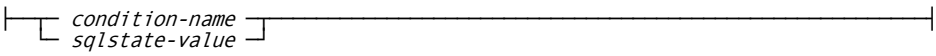
The RESIGNAL statement is generally used within a handler to either add information to the condition information items in a Diagnostics Area that represents an error being currently handled, or to fill in subsequent Condition Areas within a Diagnostics Area to indicate that a subsequent error occurred during error handling. The most appropriate handler, if one exists, is also triggered. A RESIGNAL that specifies an SQLSTATE that has no handler can also be used to alter or customize the error information seen by the end-user.

If a handler executes, SQL statement execution might not continue following completion of the SIGNAL statement. The SQL statement that is executed after a SIGNAL completes is dependent upon how the handler was declared and whether that handler succeeds. For example, if a CONTINUE type handler executes and succeeds, execution continues as if no error occurred. For information about other possibilities such as UNDO and EXIT, see Diagnostics and Condition Handling (see page 548).

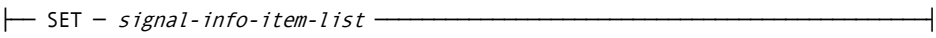
Note: SIGNAL, RAISE ERROR, and RESIGNAL statements that execute inside handlers do not activate additional condition handling. They instead cause the handler to be exited with the signaled condition and cause the execution of the compound statement that triggered the handler to abort. In addition, the compound statement that contains the handler definition is aborted if it is different from the compound statement that triggered the handler, the same as would happen if the handler failed to resolve the triggering condition. The SIGNAL, RAISE ERROR, and RESIGNAL statements should therefore be positioned as the last statements in your condition handler.



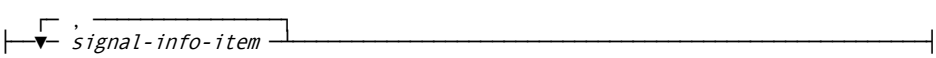
Expansion of Where signal-value is defined as



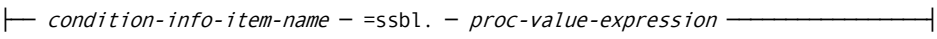
Expansion of Where signal-information is defined as



Expansion of Where signal-info-item-list is defined as



Expansion of Where signal-info-item is defined as



signal-value

(Optional) When a *signal-value* is specified, existing Condition Areas within the current Diagnostics Area are pushed down in the Condition Area stack, that is, Condition Area number one becomes Condition Area number two. Condition Area number one is then populated with any supplied or implied RETURNED_SQLSTATE (supplied using *sqlstate-value*) and (or) CONDITION_IDENTIFIER (supplied using *condition-name*) in addition to any other information supplied in the SET *signal-information* clause. The *statement-information-items* are modified to indicate that a RESIGNAL statement was the last to modify the Diagnostics Area.

signal-information

(Optional) If *signal-information* is supplied without a *signal-value,* that information is copied into Condition Area number one to represent the error condition being handled. The most appropriate handler, if one exists, is then triggered.

condition-name

See the *condition-name* information in the compound statement section that begins in Compound Statement (see page 558).

sqlstate-value

See the *sqlstate-value* information in the compound statement section that begins in Compound Statement (see page 558).

SET *signal-info-item-list*

The SET *signal-info-item-list* specification is used to set a *condition-info-item* such as the error message.

condition-info-item-name

For information about the *condition-info-item-name,* including a list of the condition information items that can be set.

proc-value-expression

For information about the *proc-value-expression* see CASE Statement (see page 555).

Example

A RESIGNAL statement example follows. In this example, SQLSTATE 'S0480' remains unhandled and can be seen by the end user. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
.
.
.
declare continue handler for sqlexception
errHandler1: begin atomic
   set errCount = errCount + 1;
   get STACKED diagnostics
      sqlStateLocal = returned_sqlstate, errMsg = message_text;
   insert into errTable values
      (orderId,errCount,"sqlState",errMsg);
   if sqlStateLocal = 'S0480'
      RESIGNAL
   endif
end errHandler1;
.
.
.
```

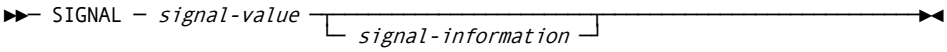
SIGNAL Statement

The SIGNAL statement signals an error or other condition, populates the first Diagnostics Area (see Diagnostics and Condition Handling (see page 548)), and triggers the execution of the appropriate condition handler, if one exists. The information supplied by the SIGNAL statement is available for retrieval by the GET DIAGNOSTICS statement until the error condition is either resolved or a subsequent statement is executed.

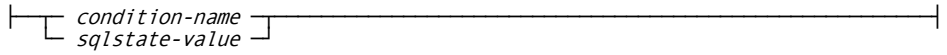
The first Diagnostics Area is cleared and the RETURNED_SQLSTATE and/or CONDITION_IDENTIFIER are set in the first condition information area of that Diagnostics Area. Certain statement information items such as NUMBER and MORE are also filled in. Any signal information supplied is also moved to the condition information area. The most appropriate error/condition handler (if one exists) is then triggered.

If a handler executes, SQL statement execution might not continue following completion of the SIGNAL statement. The SQL statement that is executed after a SIGNAL completes is dependent upon how the handler was declared and whether that handler succeeds. For example, if a CONTINUE type handler executes and succeeds, execution continues as if no error occurred. For information about other possibilities such as UNDO and EXIT, see Diagnostics and Condition Handling (see page 548).

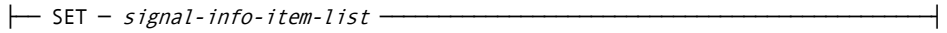
Note: SIGNAL, RAISE ERROR, and RESIGNAL statements that execute inside handlers do not activate additional condition handling. They instead cause the handler to be exited with the signaled condition and cause the execution of the compound statement that triggered the handler to abort. In addition, the compound statement that contains the handler definition is aborted if it is different from the compound statement that triggered the handler, the same as would happen if the handler failed to resolve the triggering condition. The SIGNAL, RAISE ERROR, and RESIGNAL statements should therefore be positioned as the last statements in your condition handler.



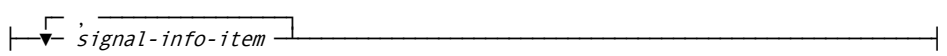
Expansion of Where signal-value is defined as



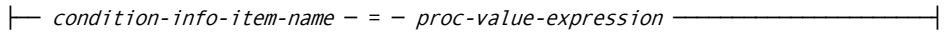
Expansion of Where signal-information is defined as



Expansion of Where signal-info-item-list is defined as



Expansion of Where signal-info-item is defined as



signal-value

(Optional) When a *signal-value* is specified, existing Condition Areas within the current Diagnostics Area are pushed down in the Condition Area stack, that is, Condition Area number one becomes Condition Area number two. Condition Area number one is then populated with any supplied or implied RETURNED_SQLSTATE (supplied using *sqlstate-value*) and/or CONDITION_IDENTIFIER (supplied using *condition-name*) in addition to any other information supplied in the SET *signal-information* clause. The *statement-information-items* are modified to indicate that a SIGNAL statement was the last to modify the Diagnostics Area.

signal-information

(Optional) The supplied *signal-information* is copied into Condition Area number one to represent the condition being signaled. The most appropriate handler, if one exists, is then triggered.

condition-name

See the *condition-name* information in the compound statement section that begins in Compound Statement (see page 558).

sqlstate-value

See the *sqlstate-value* information in the compound statement section that begins in Compound Statement (see page 558).

SET *signal-info-item-list*

The SET *signal-info-item-list* specification is used to set a *condition-info-item* such as the error message.

condition-info-item-name

For information about the *condition-info-item-name,* including a list of the condition information items that can be set, see GET DIAGNOSTICS Statement (see page 566).

proc-value-expression

For information about the *proc-value-expression* see CASE Statement (see page 555).

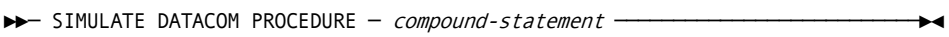
Example

A SIGNAL statement example follows. Another SIGNAL statement example can be found in the section labeled SIGNAL order approved in Sample Procedure 2 (see page 585). In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
.
.
.
IF creditReqAmt > creditAvailable
 SIGNAL SQLSTATE '2FS04'
  SET MESSAGE_TEXT = 'CUSTOMER HAS INSUFFICIENT CREDIT LIMIT TO COMPLETE PURCHASE';
END IF;
.
.
.
```

SIMULATE DATACOM PROCEDURE Statement

(DBSQLPR only) The SIMULATE DATACOM PROCEDURE statement provides support in DBSQLPR for SQL variables, user-controlled cursors, and complex program logic.



compound-statement

The compound statement is described in Compound Statement (see page 558). The start-label of the compound statement is required in the context of the SIMULATE DATACOM PROCEDURE statement and becomes the SQL-name of an actual procedure that is created, executed, and then dropped.

To simulate a procedure that itself has parameters, either simulate the parameters by declaring SQL variables and assigning values or issue a CREATE PROCEDURE followed by a simulated call to that procedure as shown in the following example.

In the following example, SQL-names that do not consist of delimited identifiers are converted to upper-case by SQL. Also, if you need confirmation of the values returned into the ordersAccepted and ordersRejected variables, use the DATACOM DUMP statement (see DATACOM DUMP Statement (see page 563)) or insert the values into a table, for retrieval outside of the procedure, by subsequent SELECT statements. The ORDERREVIEW procedure called from this example is created by the example SQL procedure that begins in Examples (see page 585).

```
SIMULATE DATACOM PROCEDURE
supplyVarsForCall: begin atomic
   declare ordersAccepted, ordersRejected int;
   CALL ORDERREVIEW(ordersAccepted, ordersRejected);
end supplyVarsForCall@
```

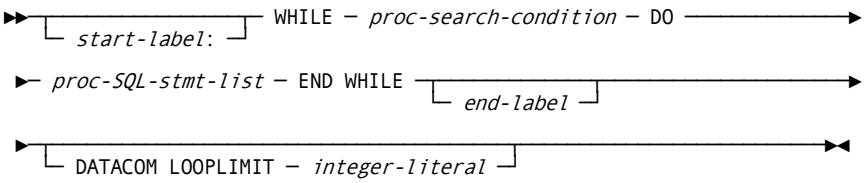
Following is a SIMULATE PROCEDURE statement that calls the procedure created by the compound statement example in Example (see page 563). This SIMULATE PROCEDURE statement contains the DATACOM DUMP example shown in Example (see page 565).

```
000034 simulate datacom procedure
000035 supplyHostVarsForCall: begin atomic
000036    declare errorMessage, result char(80) default 'N/A';
000037 -- The default value for SQL vars is NULL
000038    declare sqlstateLocal char(5);
000039
000040    declare continue handler for not found
000041          set errorMessage = 'CUSTOMER HAS NO CREDIT RECORD';
000042    declare continue handler for sqlexception, sqlwarning
000043          get diagnostics sqlstateLocal = returned_sqlstate,
000044                          errorMessage  = message_text;
000045
000046    datacom dump 'VALUES ON ENTRY:', result,errorMessage to pxxsql;
000047    call creditLimitCheck(result, '00001', 999.99);
000048    datacom dump 'VALUES ON EXIT:', result,sqlstateLocal,errorMessage to pxxsql;
000049 --
000050 -- In a more comprehensive example, procedure "creditLimitCheck" could have
000051 -- returned an error message and sqlstate through the parameter list, with
000052 -- the inclusion of an SQLEXCEPTION error handler containing a GET DIAGNOSTICS
000053 -- call.  This would prevent the caller from having to code their own
000054 -- GET DIAGNOSTICS statement to retrieve the information into SQL variables.
000055 end supplyHostVarsForCall@
```

WHILE Statement

The WHILE statement defines conditional looping logic in a compound statement. It provides a means to loop while some predicate remains true. We recommend that you use the optional DATACOM LOOPLIMIT clause with this statement.

Following is the syntax for the WHILE statement:



start-label*: / *end-label

(Optional) A *start-label*: is an SQL identifier (followed by a colon) that can be used in various flow-control statements to mark the destination of a branch. When you specify an *end-label* you must also specify a *start-label,* and the labels must match.

proc-search-condition

The *proc-search-condition* specifies a condition that is true, false, or unknown about a row. The *proc-search-condition* is similar to the *search-condition* described in the *CA Datacom/DB SQL User Guide* with the following modifications that allow its use in SQL Procedures:

- SQL parameter and SQL variable references are allowed instead of column and host-variable references.
- Aggregate functions and any predicate involving a sub-query are not allowed.

proc-SQL-stmt-list

A *proc-SQL-stmt-list* is a list of *proc-SQL-stmt* statements, each terminated by a semicolon. For information about a *proc-SQL-stmt,* see CREATE PROCEDURE Syntax and Description (see page 538).

DATACOM LOOPLIMIT

We recommend that you use DATACOM LOOPLIMIT. DATACOM LOOPLIMIT is a CA Datacom extension that we provide to allow your proactive avoidance of endless loops that might be caused by faulty logic, unexpected or missing column values, or other unforeseen conditions. The Multi-User startup option SQL_DATACOM_LOOPLIMIT can be used to force a default limit on each looping statement. An error is produced if the limit is exceeded.

integer-literal

The *integer-literal* specifies the maximum number of times the loop is allowed to execute.

Example

A WHILE statement example follows. In the example that follows, logic that is not shown (for space considerations) is indicated by three vertically arranged periods.

```
   .
   .
   .
   DECLARE continue HANDLER FOR sqlexception, sqlwarning, not found
      GET stacked DIAGNOSTICS
         sqlStateLocal = RETURNED_SQLSTATE, errMsg = MESSAGE_TEXT;
   .
   .
   .
 startWhile:
    WHILE sqlStateLocal <-> '00000' DO
       FETCH orderCrs INTO orderId, custId, creditReqAmt;
       .
       .
       .
    END WHILE endWhile DATACOM LOOPLIMIT 10000;
    .
    .
    .
```

Examples

Following are examples that show the use of SQL Procedures.

Sample Procedure 1

In this example, SQL names that do not consist of delimited identifiers are converted to uppercase by SQL. Also, notice that the at sign (@), used in conjunction with the TERM=@ parameter in DBSQLPR, enables DBSQLPR to skip over semicolons embedded in statements and instead recognize the @ as the end of the statement.

Note: For more information about TERM=, see the information about DBSQLPR in the *CA Datacom/DB SQL User Guide*. For z/OS and z/VSE JCL samples, see the information about DBSQLPR in the *CA Datacom/DB SQL User Guide*.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards. See Listing Libraries for CA Datacom Products.

```
//jobname     See the note above.
//       REGION=1024K
//JOBLIB  your DD statements go here
//SQLEXEC  EXEC PGM=DBSQLPR,
//       PARM='authid=sysadm,prtWidth=1500,inputWidth=80,term=@'
//*CAOESTOP DD  DUMMY
//SYSUDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//OPTIONS  DD  *
/*
//SYSIN    DD  *
-- Supply customer credit limits for procedure creditLimitCheck below.
create table credit
   (custId numeric(10),
    creditMax  decimal(15,2),
    creditUsed decimal(15,2)) @
insert into credit values (1, 2000., 1000.)@
commit@
--
-- Does the customer have enough credit to purchase an ordered part?
create procedure creditLimitCheck
   (inout result varchar(25), in custId char(5),
    in purchaseAmount decimal(15,2), in turnDebugDumpsOn int)
   language sql
   limitCheck: begin atomic
      declare creditAvailable decimal(15,2) default 0;

--    Compute available credit.
      select creditMax-creditUsed into creditAvailable
             from credit
             where credit.custId =
                   cast(creditLimitCheck.custId as numeric(5));

      if (purchaseAmount > creditAvailable) then
         set result = 'CREDIT LIMIT EXCEEDED';
      else
         set result = 'CREDIT APPROVED';
      end if;

      if (turnDebugDumpsOn = 1) then
         datacom dump purchaseAmount, creditAvailable, result,
                     'CREDITLIMITCHECK CUSTID=' || custId
                to pxxsql;
      end if;
   end limitCheck@

--
--  Use SIMULATE to supply a variable for the INOUT "result" parameter
```

```
--  SIMULATE was invented to provide procedure-like functionality to DBSQLPR.
simulate datacom procedure
supplyVarsForCall: begin atomic
   declare errorMessage char(80) default 'N/A';
   declare result varchar(25) default 'N/A';
-- The default value for SQL variables is NULL
   declare sqlstateLocal char(5);
--
-- Note that proc. "creditLimitCheck" could have contained the error handlers
-- directly and returned an error message and sqlstate through the parameters.
   declare continue handler for not found
         set errorMessage = 'CUSTOMER HAS NO CREDIT RECORD';
   declare continue handler for sqlexception, sqlwarning
         get diagnostics sqlstateLocal = returned_sqlstate,
                         errorMessage  = message_text;

   datacom dump 'VALUES ON ENTRY:', result, errorMessage to pxxsql;
-- Note that SQL variable "result" is being used like a host variable here.
   call creditLimitCheck(result, '00001', 999.99, 1);
   datacom dump 'VALUES ON EXIT:', result,sqlstateLocal,errorMessage to pxxsql;
end supplyVarsForCall@
--
-- The above procedure, coded using a SELECT rather than a compound statement:
--commit@
-- Does the customer have enough credit to purchase an ordered part?
create procedure creditLimitCheckV2
   (inout result varchar(80), in custId char(5),
    in purchaseAmount decimal(15,2))
   language sql
   select case
          when creditMax-creditUsed >= purchaseAmount then
             'CREDIT APPROVED'
          else
             'CREDIT LIMIT EXCEEDED'
          end
      into result
      from credit
      where credit.custId =
            cast(creditLimitCheckV2.custId as numeric(5))@
--
--  Use SIMULATE to supply a host variable for the INOUT "result" parameter
simulate datacom procedure
supplyVarsForCall: begin atomic
   declare errorMessage, result char(80) default 'N/A';
-- The default value for SQL vars is NULL
   declare sqlstateLocal char(5);

   declare continue handler for not found
         set errorMessage = 'CUSTOMER HAS NO CREDIT RECORD';
```

```
    declare continue handler for sqlexception, sqlwarning
          get diagnostics sqlstateLocal = returned_sqlstate,
                          errorMessage  = message_text;

    datacom dump 'VALUES ON ENTRY:', result, errorMessage to pxxsql;
    call creditLimitCheckV2(result, '00001', 999.99);
    datacom dump 'VALUES ON EXIT:', result,sqlstateLocal,errorMessage to pxxsql;
 end supplyVarsForCall@
/*
```

Sample Procedure 2

In this example, SQL names that do not consist of delimited identifiers are converted to uppercase by SQL. Also, notice that the at sign (@), used in conjunction with the TERM=@ parameter in DBSQLPR, enables DBSQLPR to skip over semicolons embedded in statements and instead recognize the @ as the end of the statement.

Note: For more information about TERM=, see the information about DBSQLPR in the *CA Datacom/DB SQL User Guide*. For z/OS and z/VSE JCL samples, see the information about DBSQLPR in the *CA Datacom/DB SQL User Guide*.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards. See Listing Libraries for CA Datacom Products.

```
//jobname     See the note above.
//       REGION=1024K
//JOBLIB  your DD statements go here
-- Regarding TERM=@ in the following, DBSQLPR normally uses a semicolon to detect
-- the end of an SQL statement, but SQL statements, as sub-statements within
-- compound-statements, are also required to be terminated with semicolons.
-- This presents a challenge for DBSQLPR, because DBSQLPR must detect the end of a
-- compound-statement (without the problem-prone addition of an extraneous SQL parser
-- outside SQL-proper) despite the appearance of additional semicolons. We circumvent
-- this inherent ambiguity by using the TERM=@ specification to provide a different
-- terminating character for the compound-statement. The specification applies to
-- the entire input file (SYSIN).
//SQLEXEC  EXEC PGM=DBSQLPR,PARM='AUTHID=SYSADM,TERM=@,prtwidth=500'
//SYSUDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//OPTIONS  DD  *
inputwidth=80
/*
//SYSIN    DD  *
-- Set up tables and data for use by the procedure
drop procedure orderReview CASCADE@
drop table errTable@
drop table newOrder@
drop table warehouseToDo@
commit@
create table errTable
   (orderId     int,
    errCountCol int,
    sqlStateCol char(5),
    errMsgCol   varchar(128))@
create table newOrder
   (orderId      int,
    custId       numeric(10),
    creditReqAmt decimal(15,2),
    status       char(25))@
create table credit
   (custId       numeric(10),
    creditMax    decimal(15,2),
    creditUsed   decimal(15,2))@
create table warehouseToDo
   (orderId        integer,
    instructions   char(500),
    whenInstructed timestamp)@
commit@
delete from neworder@
delete from credit@
insert into newOrder values(1, 1, 501.01, 'NOT REVIEWED')@
insert into newOrder values(2, 2, 902.02, 'NOT REVIEWED')@
```

```
insert into credit values(1,750.01, 100.01)@
insert into credit values(2, 750.02, 100.02)@
commit@
--
CREATE PROCEDURE ORDERREVIEW
   (out ordersAccepted int, out ordersRejected int)
   language sql
-- The rest of this procedure consists of a single compound statement
-- labelled "orderReviewMain".
   orderReviewMain:
   begin atomic
--    Non-delimited variables will be uppercased so ABC matches abc.
      declare statusLocal char(25);
      declare creditReqAmt, creditAvail decimal(15,2);
      declare custId numeric(10);
      declare errCount, orderId, I_want_to_leave_the_loop_NOW,
              I_want_to_re_loop_NOW int;
      declare errMsg varchar(128);
      declare sqlStateLocal char(5);
--
      declare orderCrs cursor for
              select orderId, custId, creditReqAmt
              from newOrder
              where status = 'NOT REVIEWED';
--    Conditions and Condition Handler(s)
--
--    This condition is signaled when an order is approved.
      declare orderApproved condition;
--
--    Handlers may consist of a compound or non-compound statement.
--    Non-compound handler examples:
--
      declare continue handler for orderApproved
            insert into warehouseToDo values (orderReviewMain.orderId,
                                    'ORDER APPROVED.  BEGIN MAKE-READY',
                                    current timestamp);

      declare undo handler for sqlstate '44444', sqlstate value '55555'
            set I_want_to_leave_the_loop_NOW = 1;

      declare exit handler for sqlstate '66666'
            set sqlStateLocal = '66666';

      declare continue handler for not found
            set sqlStateLocal = '02000';
--
--    Compound handler example:
--       Note the use of "errCount" even though it's declared outside the
--       scope of the errHandler1 compound statement.  If errHandler1 had an
```

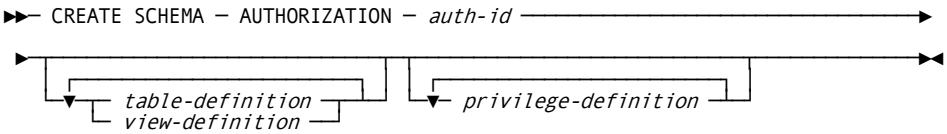
```
--       errCount variable, we'd have to specify "orderReviewMain.errCount"
--       to reference the variable from the outer compound statement.  Also,
--       a locally declared copy of "errCount" would have to have its' value
--       reset on each entry to the handler, hence the "outer" reference.
--
--       Note that there ARE duplicate "errMsg" variables in other scopes
--       but we'll resolve to the one inside the error handler's context.
--
--       SQL exceptions '44444', '55555', and '66666' would trigger the
--       handlers for those specific SQLSTATEs.  But all other SQLSTATEs
--       qualifying as SQLEXCEPTIONs will trigger this handler.
--
      declare continue handler for sqlexception, sqlstate '12345',
                                   sqlwarning
      errHandler1: begin atomic
            set errCount = errCount + 1;
            get STACKED diagnostics
                  sqlStateLocal = returned_sqlstate,
                  errMsg     = message_text;
            insert into errTable values
                  (orderId,errCount,sqlStateLocal,errMsg);
      end errHandler1;
--    Main Logic Start      --
      set errCount = 0;
      set ordersAccepted = 0;
      set ordersRejected = 0;
      set I_want_to_leave_the_loop_NOW = 0;
      set I_want_to_re_loop_NOW = 0;
      delete from errTable;
      set sqlStateLocal = '00000';
--    Loop thru new orders and approve or reject each.
      open orderCrs;
      start_while:
      while (sqlStateLocal = '00000') do
--
--       Note references to SQL variables
         fetch orderCrs into orderId, custId, creditReqAmt;
--       Note usage of SQL variable orderWatchMain.custId in query
         select creditMax-creditUsed into creditAvail
            from credit
            where credit.custId = orderReviewMain.custId;

         if (creditReqAmt > creditAvail) then
            set statusLocal = 'CREDIT LIMIT EXCEEDED';
            set ordersRejected = ordersRejected + 1;
         else
            set statusLocal = 'CREDIT APPROVED';
            update credit set creditUsed = creditUsed+creditReqAmt
               where custId = orderReviewMain.custId;
```

```
            set ordersAccepted = ordersAccepted + 1;
            SIGNAL orderApproved;
         end if;

-- Statement included only to demonstrate usage

         if (I_want_to_re_loop_NOW = 1)
            then iterate start_while;
         end if;
         update newOrder set status = statusLocal
            where current of orderCrs;

-- Statement included only to demonstrate usage

         if (I_want_to_leave_the_loop_NOW = 1)
            then leave start_while;
         end if;

      end while end_while_label datacom looplimit 100;
      close orderCrs;
   end
   orderReviewMain@
COMMIT@
-- Confirm contents
SELECT * from newOrder@
DELETE from errTable@
drop procedure supplyHostVarsForCallStmt@
commit@
SIMULATE DATACOM PROCEDURE
supplyHostVarsForCallStmt: begin atomic
   declare ordersAccepted, ordersRejected int;
   CALL ORDERREVIEW(ordersAccepted, ordersRejected);
   datacom dump ordersAccepted, ordersRejected to pxxsql;
end supplyHostVarsForCallStmt@
--
-- Confirm results
SELECT * from newOrder@
select * from errTable@
select * from warehouseToDo@
/*
```

CREATE RULE

For information about the CREATE RULE statement, see CREATE TRIGGER/RULE (see page 616).

CREATE SCHEMA



CREATE SYNONYM

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CREATE SYNONYM	YES	YES	YES

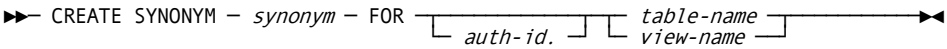
Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The CREATE SYNONYM statement is a CA Datacom/DB extension. CREATE SYNONYM defines an alternative name for a table or view. You can use the synonym to reference a table in another schema without having to enter the qualified name. You can also define synonyms for tables or views in your default schema. When a CREATE SYNONYM statement successfully executes, a SYNONYM entity-occurrence is defined in Datadictionary in PRODuction status. The synonym is added to your default schema.

With regard to table partitioning, CREATE statements may not be issued against a table which is partitioned nor against a partition.

Note: For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

Following is the syntax diagram for the CREATE SYNONYM statement:



Description

synonym

The alternative name you want to use when referring to the table or view. The synonym name must not be identical to another synonym or to the unqualified name of a table or view in your default schema. The synonym name can be 1 to 18 characters if you specify the Preprocessor option SQLMODE=ANSI or SQLMODE=FIPS. If SQLMODE=DATACOM, the synonym name can be 1 to 32 characters.

FOR

Introduces the qualified name of the table or view for which you are creating the synonym.

auth-id.

Specify an authorization ID if you are creating a synonym for a table or view in a schema other than your default schema. Use a period (.) to concatenate the authorization ID to the table or view name (for example, *auth-id.table-name*).

table-name* or *view-name

Must name a table or view described in the Datadictionary. The synonym is defined only for your authorization ID, that is to say, the authorization ID of the CREATE SYNONYM statement when the statement is prepared.

Example

The following example defines an alternative name, OUTDEPTS, for the table DEPTTBL. KWAN is the authorization ID of the user who owns DEPTTBL.

```
CREATE SYNONYM OUTDEPTS
FOR KWAN.DEPTTBL
```

CREATE TABLE

In the following table, YES indicates a valid execution method for this statement.

This SQL statement can be executed in the following ways:	**Through the Datadictionary Interactive SQL Service Facility *(interactive)***	**In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)***	**By using CA Dataquery *(SQL & Batch Modes)***
CREATE TABLE	YES	YES	YES

Note: To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

Note: If you are using CA Datacom/DB as part of the CA Datacom/AD environment, you cannot use the CREATE TABLE statement.

CREATE TABLE defines a table. In this statement, you must specify:

- The name of the table, and
- The column definition for the table.

Optional specifications include:

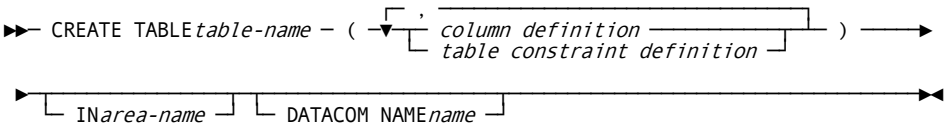
- The table constraint definition for the table.
- Designating more than one column definition.
- Designating one or more table constraint definitions.
- Naming the data area where the table data is to reside.

When a CREATE TABLE statement successfully executes, a TABLE entity-occurrence is defined in Datadictionary in PRODuction status, cataloged to the CA Datacom/DB Directory (CXX), and is ready to be populated with data. See the *CA Datacom/DB SQL User Guide* for information about the results in Datadictionary of using the CREATE TABLE statement.

With regard to table partitioning, CREATE statements may not be issued against a table which is partitioned nor against a partition. An SQL integrity constraint cannot reference a partitioned table, nor a partition of a partitioned table. That is to say, constraints and partitioned tables are mutually exclusive.

Note: For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

Following is the syntax diagram for the CREATE TABLE statement:



Note:

- IN area-name is a CA Datacom/DB extension.
- DATACOM NAME name is a CA Datacom/DB extension.

Description

table-name

The name of the table you are creating. The name you supply, including the implicit or explicit qualifier, must not identify a table, view or synonym already described in the Datadictionary.

If you specify SQLMODE=ANSI or SQLMODE=FIPS in the Preprocessor options, the table name can be 1 to 18 characters in length.

If you specify SQLMODE=DATACOM for extended mode in the Preprocessor options, the table name can be 1 to 32 characters in length.

The qualified form is the name preceded by an authorization ID and a period, for example, auth-id.table-name (but do *not* use SYSADM for the auth-id). If you qualify the table-name, the qualifier designates the schema of the table. If you do not qualify the table-name, the default authorization ID is used as the qualifier. If the CREATE TABLE statement is embedded within a CREATE SCHEMA statement, the authid that qualifies the table must be the same as the authid that follows the AUTHORIZATION keyword.

column definition

See Column Definition (see page 597) for information on the column definition.

table constraint definition

See Table Constraint Definition (see page 602) for information on the table constraint definition.

IN *area-name*

Use this CA Datacom/DB extension to specify the name of the area in which the table data is to reside. The name you specify for the area must be the SQL name, not the Datadictionary occurrence name. The area must already exist in the Datadictionary and be cataloged to the Directory (CXX). If you do not specify the area-name, the table is placed in the default area. If you need to specify an area other than the default, or if the default area is full, see your Database Administrator for names of other areas you can specify when creating a table.

DATACOM NAME *name*

Use this CA Datacom/DB extension to specify a three-character DATACOM NAME name (a TABLE entity-occurrence in Datadictionary) for the table. If you do not specify a DATACOM NAME name, Datadictionary generates a name for you. (This is not the Datadictionary entity-occurrence name.)

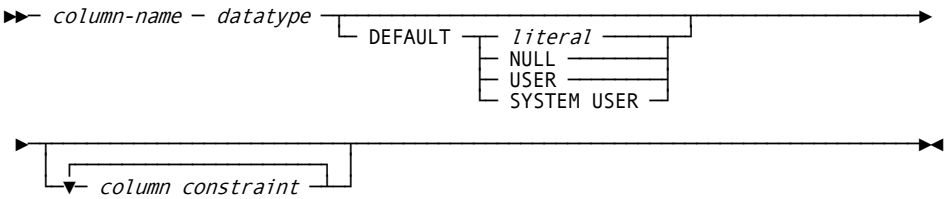
Privileges

If the Multi-User Facility is secured, whether the creator of the table may access that table depends on the security model of the table. Under the SQL Security Model, when a table is created the user who executes the CREATE TABLE statement is automatically granted all privileges on the table. Under the CA Datacom/DB External Security Model, all access rights, including those for the creator of a table, must be defined through the external security product. See the *CA Datacom Security Reference Guide* for more information.

The CREATE TABLE statement cannot create a remote, partitioned or replicated table. For remote tables, a complete duplicate definition of a remote table in Datadictionary is required for SQL access, and no version control enforcement is available to ensure that remote definitions are synchronized with the local active definition.

Column Definition

Following is the syntax diagram for the column definition:



Note: SYSTEM USER is a CA Datacom/DB extension.

Description

column-name

The name of a column of the table. Do not use the same name for more than one column in this table.

If you specify SQLMODE=ANSI or SQLMODE=FIPS in the Preprocessor options, the column name can be 1 to 18 characters in length.

If you specify SQLMODE=DATACOM for extended mode in the Preprocessor options, the column name can be 1 to 32 characters in length.

The number of columns you can define for a single table is limited only by the maximum size of the physical record. The sum of the byte counts of the columns must not be less than 1 or greater than 32720.

If you use the UNIQUE (column-list) constraint in a table constraint definition (see Table Constraint Definition (see page 602)), you must separate the last column definition from the UNIQUE (column-list) constraint with a comma.

If you specify a ref-col-name in the column constraint definition (see Column Constraint Definition (see page 599)), the data type, length, and scale of the column-name specified here in the column definition must be identical to those of the ref-col-name.

The first column you name in the CREATE TABLE statement becomes the CA Datacom/DB Master and Native Key for the table except when a primary key has been defined. Datadictionary automatically generates this KEY entity-occurrence, which has the same name as the column.

Note: The DUPE-MASTER-KEY and CHNG-MASTER-KEY attributes of the TABLE entity-occurrence are set to Y, indicating that the value of the Master Key can be duplicated and/or changed. The 5-character CA Datacom/DB name of the key is SQnnn, where nnn is a sequential number unique to each database.

datatype

See Data Types (see page 609).

Note: If NOT NULL WITH DEFAULT is specified as a constraint with DATE, TIME, or TIMESTAMP, the default value is the current date, current time, or current timestamp.

DEFAULT

Used to specify a default. If DEFAULT is specified, you cannot use a column constraint of WITH DEFAULT.

literal

Specifies a literal as the default. The literal you specify must be consistent with the data type of the column. A user-supplied DEFAULT literal can be up to 20 bytes long, or the length of the column involved, whichever is shorter. A default value may be specified for a character column where the column is greater than 20 bytes long, but the default literal itself is limited to 20 bytes, with the remaining bytes padded with blanks by the system.

NULL

Specifies NULL as the default. If NULL is specified, you cannot use a column constraint of NOT NULL.

column constraint

See the column constraint definition in the following.

USER

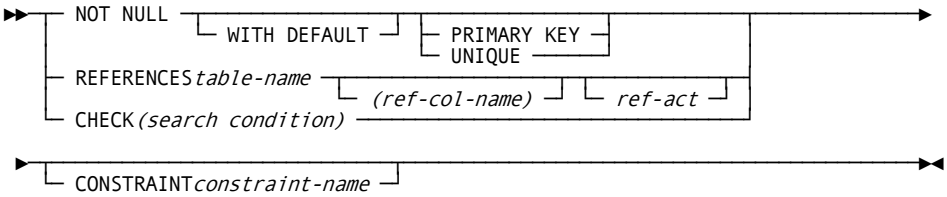
Specifies the current authorization ID as the default.

SYSTEM USER

This CA Datacom/DB extension specifies the accessor ID of the currently signed-on user as the default.

Column Constraint Definition

Following is the syntax diagram for the column constraint definition:



Note: WITH DEFAULT is a CA Datacom/DB extension.

Description

NOT NULL

Specifies that the column value cannot be NULL. If NOT NULL is not specified, the column may contain NULL values. On INSERT, a non-null value must be supplied, or a DEFAULT must be specified. On UPDATE, the column cannot be set to the null value. If NOT NULL WITH DEFAULT is specified as a constraint with DATE, TIME, or TIMESTAMP, the default value is the current date, current time, or current timestamp. You must specify NOT NULL to use the column-level UNIQUE constraint. If you specify NOT NULL, you cannot specify NULL as the column default.

UNIQUE

Specifying the column-level UNIQUE constraint after an individual column name indicates that the value for the column is to be unique for each row of the table.

You must specify NOT NULL to use the UNIQUE column-level constraint.

The UNIQUE column-level constraint restricts update or insertion of a row if it contains a column value which has been previously assigned. For example, if COLUMN_A has been assigned the value '75252' for one row in the table, you cannot insert a row or update another row to assign that same value to the column.

You can specify the column-level UNIQUE on a maximum 50 individual columns per table. The column length must not exceed 180 bytes if you specify this constraint on the column.

When you use the column-level UNIQUE constraint, Datadictionary generates a KEY entity-occurrence with the UNIQUE attribute to enforce uniqueness. This key may not be deleted directly. You must DROP the UNIQUE constraint to delete the key. The Datadictionary name for the key is the same as the column. The 5-character CA Datacom/DB name of the key is SQnnn, where nnn is a sequential number unique to each database.

PRIMARY KEY

Same as UNIQUE, except for the following:

- Only one primary key is allowed per table, whether specified at the column or table level.
- When the referential constraint ref-col-list is omitted, the columns of the primary key are implied by default.

WITH DEFAULT

This is a CA Datacom/DB extension. Specifying WITH DEFAULT means that if you do not specify a value, blanks is used if the data type is CHARACTER. If the data type is not CHARACTER, zero (0) is used of the appropriate type, length, and scale.

If NOT NULL WITH DEFAULT is specified as a constraint with DATE, TIME, or TIMESTAMP, the default value is the current date, current time, or current timestamp.

You must specify NOT NULL to use the WITH DEFAULT column-level constraint.

Do not use WITH DEFAULT if you specify a DEFAULT in the column definition.

REFERENCES *table-name*

Specify the name of the table you want to reference. The table-name must identify a table described in the Datadictionary other than a Datadictionary table.

(ref-col-name)

If you specify a ref-col-name, the data type, length, and scale of the column-name specified in the column definition (see Column Definition (see page 597)) must be identical to those of the ref-col-name.

When no ref-col-name is specified, the table-name must have a PRIMARY KEY, and its column-name is the column-name specified in the column definition.

The referenced column must be a UNIQUE or PRIMARY KEY in the referenced table. If the referenced column is not a UNIQUE or PRIMARY KEY, CA Datacom/DB issues a -169 SQL return code.

ref-act

Specifies the referential action to be taken when a row in the referenced table is deleted or updated. See Description (see page 604) for a detailed description.

CHECK (*search condition*)

Allows you to check domain constraints specified in the search condition.

The search condition must be enclosed in parentheses.

When you check a constraint through the CREATE TABLE or ALTER TABLE statements, do not include a function, special register, host variable, subquery, or external table reference in the search condition. If you do, CA Datacom/DB issues an error message informing you that none of those are allowed.

Note: For more information on the search condition, see the *CA Datacom/DB SQL User Guide*.

CONSTRAINT *constraint-name*

Allows you to specify a constraint-name. Constraint names must be unique within all constraints defined in the same schema. A fully qualified constraint name contains the schema ID, that is to say, the authorization ID or creator ID, and a 32-byte name.

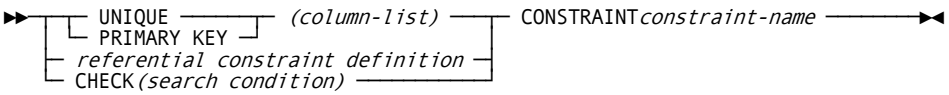
If the name you specify already exists, CA Datacom/DB issues an error message. Query the SYSCONSTRSRC table (see the *CA Datacom/DB SQL User Guide*) to see which names are already in use.

If you do not specify a constraint-name, the system generates a name in the form CONSTRAINT_nnnn where nnnn is 0001 to 9999. For example, the first constraint-name generated by the system would be CONSTRAINT_0001, the second would be CONSTRAINT_0002, and so on.

Constraint names are returned in the SQLCA Error Message to indicate which of possibly several constraints were violated, and in the ALTER TABLE DROP statement to specify which constraint is to be dropped.

Table Constraint Definition

Following is the syntax diagram for the table constraint definition:



Description

UNIQUE

The UNIQUE table-level constraint specifies that the combination of values assigned to all the listed columns is to be unique.

Individual columns on which the column-level UNIQUE is specified can also be included in the column list for a table-level UNIQUE.

If you use the table-level UNIQUE constraint, Datadictionary generates a KEY entity-occurrence with the UNIQUE attribute to enforce uniqueness. This key may not be deleted directly. You must DROP the UNIQUE constraint to delete the key. The Datadictionary name for the key is the same as the table, followed by a number which makes the name unique. The 5-character CA Datacom/DB name of the key is SQnnn, where nnn is a sequential number unique to each database.

Note: Uniqueness is enforced at the key level, not at the column level, that is to say, UNIQUE forces unique values for the entire key and not for the individual columns making up the key.

PRIMARY KEY

Same as UNIQUE, except for the following:

- Only one primary key is allowed per table, whether specified at the column or table level.
- When the referential constraint ref-col-list is omitted, the columns of the primary key are implied by default.

(*column-list*)

Specify the column-list for the table-level UNIQUE or PRIMARY KEY constraint by enclosing one or more column names in parentheses. Use a comma to separate column names.

For example, if you specify UNIQUE (COLUMN_A, COLUMN_B) when you create a table, the following value assignments are valid because the combination of values for these two columns are unique in each row.

	COLUMN_A	**COLUMN_B**
Row 1	DALLAS	TEXAS
Row 2	PARIS	TEXAS
Row 3	PARIS	FRANCE

The total length of all columns listed in each table-level constraint cannot exceed 180 bytes.

referential constraint definition

See Referential Constraint Definition (see page 604).

CHECK (*search condition*)

Allows you to check domain constraints specified in the search conditions.

The search condition must be enclosed in parentheses.

When you define a constraint through the CREATE TABLE or ALTER TABLE statements, do not include a function, special register, host variable, subquery, or external table reference in the search condition. If you do, CA Datacom/DB issues an error message informing you that none of those are allowed.

You may define multiple CHECK constraints to indicate exactly which constraint has been violated, since the constraint name is returned in the SQL Error Message.

Note: For information on the search condition, see the *CA Datacom/DB SQL User Guide*.

CONSTRAINT *constraint-name*

Allows you to specify a constraint-name. Constraint names must be unique within all constraints defined in the same schema. A fully qualified constraint name contains the schema ID, that is to say, the authorization ID or creator ID, and a 32-byte name.

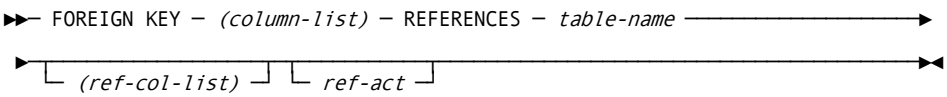
If the name you specify already exists, CA Datacom/DB issues an error message. Query the SYSCONSTRSRC table (see the *CA Datacom/DB SQL User Guide*) to see which names are already in use.

If you do not specify a constraint-name, the system generates a name in the form CONSTRAINT_nnnn where nnnn is 0001 to 9999. For example, the first constraint-name generated by the system would be CONSTRAINT_0001, the second would be CONSTRAINT_0002, and so on.

Constraint names are returned in the SQLCA Error Message to indicate which of possibly several constraints were violated, and in the ALTER TABLE DROP statement to specify which constraint is to be dropped.

Referential Constraint Definition

Following is the syntax diagram for the referential constraint definition:



Description

FOREIGN KEY

Allows you to specify a foreign key.

Foreign keys define relationships between tables. The column(s) of a foreign key in one table are related to the primary or unique key of some table. That related primary or unique key may not be defined on the same table as the foreign key.

Every foreign key is related to a primary or unique key. A primary or unique key may be related to zero, one, or many foreign keys.

Each foreign key defines a referential integrity constraint. Every foreign key value (the column values of the foreign key's columns from a single row of the table) must exactly match a primary or unique key value of the foreign key's related primary or unique key, except when at least one foreign key column value is NULL. When a foreign key value exists which does not match any primary or unique key value of its related primary or unique key, the referential integrity constraint is violated.

The foreign key constraint does *not* restrict the number of rows with the same foreign key value, that is to say, the constraint is "1-to-many," one referenced table row may match multiple referencing rows. You may define a "1-to-1" constraint by also defining the foreign key columns in a primary key or unique constraint.

An attempt to INSERT or UPDATE a foreign key value that does not exist in the referenced primary/unique key is rejected as a foreign key *value* error. An attempt to DELETE or UPDATE a primary/unique value that is referenced by one or more foreign key values is rejected as a foreign key *reference* error.

When the referenced table is loaded or recovered, the integrity of the foreign key is in doubt, and the table is placed in a CHECK-RELATED check state. A table in a check state cannot be opened. You must execute the CONFIRM function of DBUTLTY to confirm that all referenced values still exist. The check state is reported in the Directory (CXX) report.

Note: Tables defined using the SQL CREATE TABLE statement have the RECOVERY attribute-type of the TABLE entity-occurrence set to Y (for yes) in Datadictionary.

A "physical" key is *not* generated for a foreign key. However, if you define a key, it is used when checking the DELETE or UPDATE of the referenced primary/unique key. For best performance, the key should include all columns of the foreign key as the only or leading columns. Other columns may be included following the foreign key columns. The order in which foreign key columns are specified in the key does not affect performance.

A foreign key is rejected if the table involved has more than one foreign key and the delete or update actions of any two of those foreign keys conflict. A -36 SQL return code is issued if a conflict occurs. For more information see Referential Actions That Conflict (see page 607).

(*column-list*)

Use the column-list to specify one or more column names. The column names must be separated by commas and the list must be enclosed in parentheses.

The number of columns in the column-list and their data type, length, and scale must be identical to those specified in the ref-col-list or the default ref-col-list.

When at least one of the columns is not defined with NOT NULL, that is to say, it may contain the null value, the reference is said to be "optional." When at least one column is the null value, no check is made for a matching primary/unique value.

REFERENCES *table-name*

Specify the name of the table you want to reference. The table-name must identify a table described in the Datadictionary other than a Datadictionary table. When no ref-col-list is specified, the table-name must have a PRIMARY KEY, and its column(s) is the default ref-col-list. The referenced table cannot be a remote table. Reference to a table in a different schema is permitted.

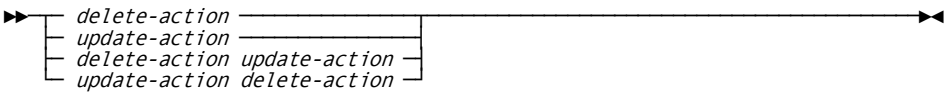
(*ref-col-list*)

Use the column-list to specify one or more column names. The column names must be separated by commas and the list must be enclosed in parentheses. When you are specifying a ref-col-list, specify column(s) of a UNIQUE or PRIMARY KEY constraint on the table-name.

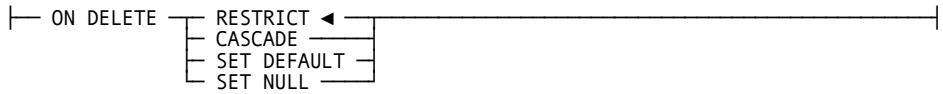
ref-act

Specifies the referential action to be taken when a row in the referenced table is deleted or updated.

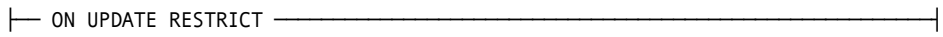
Following is the syntax diagram for ref-act (referential action):



Expansion of Where delete-action is defined as



Expansion of Where update-action is defined as



ON DELETE

Specifies which delete-action is to be taken. Errors that occur during delete-actions cause the original DELETE (and all propagated delete-actions) to be aborted.

Delete-actions may be one of the following:

RESTRICT

ON DELETE RESTRICT is the default delete-action. It specifies that there shall be no matching rows when a row is deleted from the referenced table. Matching rows are (for a given row in the referenced table) all rows in the referencing table whose referencing columns equal the corresponding referenced column. If referencing rows exist, the DELETE is rejected with a -175 return code.

CASCADE

Specifies that referencing rows are deleted after any references to the child table are followed and those referential actions performed.

SET DEFAULT

Specifies that each column in the referencing foreign key is set to its default value for every referencing row. If ON DELETE SET DEFAULT is specified, all of the columns in the foreign key must have defaults defined.

SET NULL

Specifies that nullable columns in the referencing foreign key are set to NULL for all referencing rows. At least one column is guaranteed to be nullable in this foreign key. If ON DELETE SET NULL is specified, at least one of the foreign key columns must be nullable.

ON UPDATE

Specifies which update action is to be taken. RESTRICT is the only update action supported. RESTRICT is the default and is enforced even if not specified.

ON UPDATE RESTRICT specifies that there shall be no matching rows when a row is updated in the referenced table. Matching rows are (for a given row in the referenced table) all rows in the referencing table whose referencing columns equal the corresponding referenced column. If referencing rows exist, the UPDATE is rejected with a -175 return code.

Referential Actions That Conflict

A foreign key is rejected if the table involved has more than one foreign key and the referential actions of any two of those foreign keys conflict. One of two conditions are generally responsible for conflicts between referential actions:

1. A delete on a referenced table can propagate conflicting referential actions to a row of a referencing table. For example, if two foreign keys on table A reference table B, and one foreign key specifies ON DELETE CASCADE but the other specifies ON DELETE SET DEFAULT, then it becomes impossible to update table A in a way that satisfies both foreign keys.
2. The order in which foreign key references are followed and satisfied affects the resulting actions taken on referencing rows. It might initially seem that all situations described in the first condition (previously described) would also be included here. For example, if the SET DEFAULT described in the first condition (previously described) were executed before the CASCADE, the stored default values could effectively sever the row's other foreign key connection to the deleted row in table B, in which case the ON DELETE CASCADE would never be executed. But this second condition can occur even if the delete actions agree.

Consider a case in which both foreign keys on table A specify ON DELETE SET NULL and the columns used in the two keys partially overlap. Setting any of the overlapping columns to NULL would sever both foreign keys' connections, because a NULL in any column of a foreign key severs the connection to the parent row. The first foreign key that is executed would therefore have its columns set to NULL, but the second foreign key would never be executed.

A table listing referential actions that conflict is provided in the following. When referring to the table, please note that it should be interpreted with regard to the following:

- An update action only conflicts with a delete action if the first condition on the previous page is violated, that is to say only if a DELETE on some indirect parent of a table can propagate both an ON UPDATE RESTRICT and a conflicting delete action to the table using two separate foreign key paths at the same time. A DELETE can propagate ON UPDATE RESTRICT because when an ON DELETE SET DEFAULT or ON DELETE SET NULL is executed on a table that is in turn referenced by another foreign key, the update actions on a referencing child table are executed instead of the delete actions.
- Referential actions that would normally conflict are allowed if the foreign keys involved have disjoint sets of parent tables (including both direct and indirect parents), or if the sets of columns affected by the conflicting rules are disjoint. SET NULL and SET DEFAULT actions that are not listed as being in conflict in the following table are, however, rejected if the second condition described on the previous page exists. But that second condition does not apply if the two foreign keys involved contain either totally disjoint or completely matching sets of columns on the child table.

In the following table, the abbreviations describe **CONFLICTS** and represent the following:

ODR

Represents **ON DELETE RESTRICT.**

ODC

Represents **ON DELETE CASCADE.**

ODSD

Represents **ON DELETE SET DEFAULT.**

ODSN

Represents **ON DELETE SET NULL.**

OUR

Represents **ON UPDATE RESTRICT.**

Note: The asterisk (*) means it does not conflict if defaults for all columns involved are NULL.

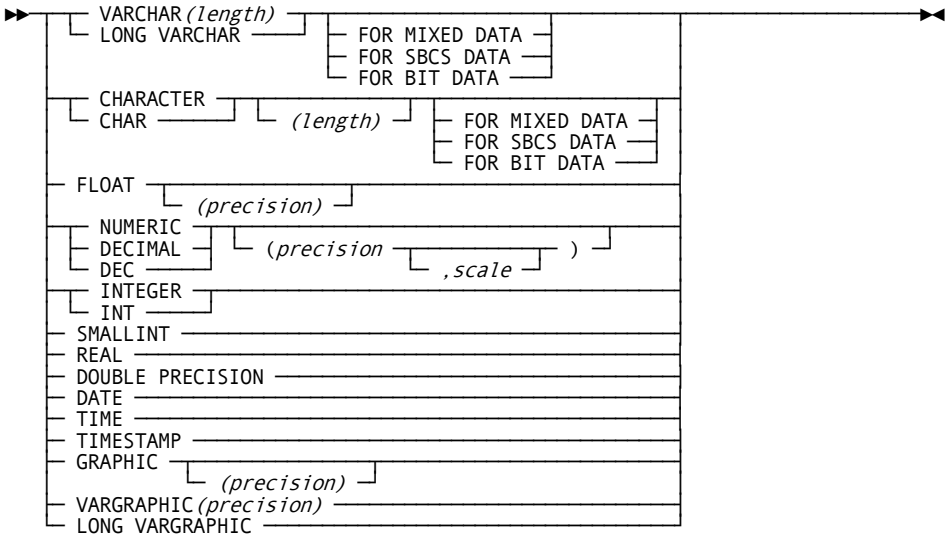
REFERENTIAL ACTIONS:	ODR	ODC	ODSD	ODSN	OUR
ON DELETE RESTRICT		X	X	X	
ON DELETE CASCADE	X		X	X	X
ON DELETE SET DEFAULT	X	X		X*	X

REFERENTIAL ACTIONS:	ODR	ODC	ODSD	ODSN	OUR
ON DELETE SET NULL	X	X	X*		X
ON UPDATE RESTRICT		X	X	X	

Note: Foreign keys can also be rejected for reasons not mentioned here. For a complete list of reasons, see the SQL return codes section in the *CA Datacom/DB Message Reference Guide*.

Data Types

Following is the syntax diagram for the valid SQL data types:



Note:

- VARCHAR and LONG VARCHAR are CA Datacom/DB extensions.
- FOR MIXED DATA, FOR SBCS DATA, and FOR BIT DATA are CA Datacom/DB extensions.
- DATE, TIME, and TIMESTAMP are CA Datacom/DB extensions.
- GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC are CA Datacom/DB extensions.

The following table summarizes information about the valid SQL data types. For more information on data types, see the *CA Datacom/DB SQL User Guide*.

Valid SQL Data Types:

Data Type	Explanation
CHARACTER (*length*) CHAR (*length*) VARCHAR (*length*) LONG VARCHAR	CHARACTER or CHAR specifies a character string. The optional *length* must be an integer designating the number of characters in the string and must be enclosed in parentheses. The default value for the CHARACTER data type length is one byte. VARCHAR specifies a varying-length character string of length 1 to maximum row size. The *length* is required, must be an integer designating the number of characters in the string, and must be enclosed in parentheses. LONG VARCHAR specifies a varying-length character string whose maximum length is determined by the amount of space available in a block. Use of the LONG VARCHAR Data Type creates rows that are as long as the block size of the containing area. If LONG VARCHARs are to be used, this block size should be taken as the row size to be used in calculating the Log Area (LXX) block size (the block size of the longest block sized area containing LONG VARCHARs should be used). Three semantic types are allowed: **FOR MIXED DATA, FOR SBCS DATA,** and **FOR BIT DATA.** FOR MIXED DATA means that DBCS characters are allowed in values stored in the column, in addition to EBCDIC (Single-Byte Character Set (SBCS)) characters. This is relevant when SQL is processing a value in the column, since it must recognize the Shift-Out and Shift-In characters that delimit DBCS substrings. FOR SBCS DATA means that DBCS characters are not allowed in the column. FOR BIT DATA means that the data is a string of binary data rather than a string of characters. If a semantic type is not specified when the column is created, the default is the semantic type that was specified on the CXXMAINT OPTION=ALTER,DBCS=xx option. If xx was IS or FS, the default is FOR SBCS DATA. If xx was IM or FM, the default is FOR MIXED DATA.
FLOAT (*precision*)	Specifies a 64-bit floating-point number. The optional *precision* of the number is the total number of digits. The *precision* can range from 1 to 15. The *precision* must be enclosed in parentheses.

Data Type	Explanation
NUMERIC (*precision,scale*)	Specifies a zoned decimal number. The optional *precision* of the number is the total number of digits. The *precision* can range from 1 to 31. If you specify *precision,* you have the option of also specifying *scale*. The optional *scale* of the number is the number of digits to the right of the decimal point. The *scale* can range from 0 to the precision. If you specify both *precision* and *scale,* you must separate them with a comma and enclose them in parentheses. When specifying a *scale* of zero, however, you can use (precision) for (precision,0). If you do not specify *precision* and/or *scale,* the default value used is a *precision* of 5 and a *scale* of zero (5,0).
DECIMAL (*precision,scale*) DEC (*precision,scale*)	Specifies a packed decimal number. The optional *precision* of the number is the total number of digits. The *precision* can range from 1 to 31. If you specify *precision,* you have the option of also specifying *scale.* The optional *scale* of the number is the number of digits to the right of the decimal point. The *scale* can range from 0 to the *precision.* If you specify both *precision* and *scale,* you must separate them with a comma and enclose them in parentheses. When specifying a *scale* of zero, however, you can use (precision) for (precision,0). If you do not specify *precision* and/or *scale,* the default value used is a *precision* of 5 and a *scale* of zero (5,0).
INTEGER INT	Specifies a large binary integer with a *precision* of 31 bits.
SMALLINT	Specifies a small binary integer with a *precision* of 15 bits.
REAL	Specifies a 64-bit floating-point number.
DOUBLE PRECISION	Specifies a 64-bit floating-point number.
DATE	Specifies a date.
TIME	Specifies a time.
TIMESTAMP	Specifies a timestamp.

Data Type	Explanation
GRAPHIC (*precision*) VARGRAPHIC (*precision*) LONG VARGRAPHIC	GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC specify Double-Byte Character Set (DBCS) characters. Each character is two bytes long. The *precision* of a DBCS column (optional for GRAPHIC, required for VARGRAPHIC) is the maximum number of two-byte characters that can be stored, not the physical length. DBCS characters in SQL statements must be delimited by the Shift-Out and Shift-In characters. Shift characters are either IBM-defined (X'0E' and X'0F') or Fujitsu-defined (X'28' and X'29'). Shift characters are specified in CXXMAINT OPTION=ALTER,DBCS=xx in the CA Datacom/DB Utility (DBUTLTY). If xx is IS or IM, the Shift characters are the IBM-defined characters. If xx is FS or FM, the Shift characters are the Fujitsu-defined characters. the information about GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC in the section on character strings within the data types section in the basic language elements chapter in the *CA Datacom/DB SQL User Guide*.

Example 1

Some examples of using the CREATE TABLE statement are:

Create a table with the following specifications:

1. Authorization ID: JOE

   The example assumes that JOE is the default authorization ID in effect when the statement is executed. In this case, it is not necessary to specify the authorization ID.

2. Table name: DEPTTBL
3. Column names: DEPTNO, DEPTNAME, MGRNBR, ADMDEPT
4. Data type: CHAR for all columns
5. Area name: CASQLDEFAULT

   The default area name is specified, although it is not necessary. If you do not specify an area name, the table is automatically placed in the default area. You must specify the area name if you want to place the table in an area other than the default.

```
CREATE TABLE DEPTTBL
     (DEPTNO CHAR(2),
      DEPTNAME CHAR(24),
      MGRNBR CHAR(6),
      ADMDEPT CHAR(2))
IN CASQLDEFAULT
```

Example 2

Create a table with the following specifications:

1. Authorization ID: TED

   In this example, TED is not the default authorization ID. It is therefore necessary to specify the authorization ID.

2. Table name: DEPTTBL

3. Column names: DEPTNO (value to be unique), DEPTNAME, MGRNBR, ADMDEPT

4. Data type: CHAR for all columns

```
CREATE TABLE TED.DEPTTBL
       (DEPTNO CHAR(2) NOT NULL UNIQUE,
        DEPTNAME CHAR(24),
        MGRNBR CHAR(6),
        ADMDEPT CHAR(2))
```

Example 3

Create a table with the following specifications:

1. Authorization ID: JOE

2. Table name: PROJECTTBL

3. Column names (CHARACTER data): PROJECTNUM (value to be unique), PROJECTNAME (value to be unique), DEPTNO, EMPNUM, MAJPROJECT

4. Column names (DECIMAL data): PROJSTAFF, PRSTARTDATE, PRENDDATE

```
CREATE TABLE JOE.PROJECTTBL
       (PROJECTNUM CHAR(6) NOT NULL UNIQUE,
        PROJECTNAME CHAR(24) NOT NULL UNIQUE,
        DEPTNO CHAR(3),
        EMPNUM CHAR(6),
        PROJSTAFF DECIMAL(5,2),
        PRSTARTDATE DECIMAL(6),
        PRENDDATE DECIMAL(6),
        MAJPROJECT CHAR(6))
```

Example 4

Create a table with the same specifications as Example 2, but specify that the DEPTNO and MGRNBR columns are to have a combined value which is unique for this table.

Note: The last column definition is separated from the table-level UNIQUE constraint by a comma.

```
CREATE TABLE TED.DEPTTBL
     (DEPTNO CHAR(2) NOT NULL UNIQUE,
      DEPTNAME CHAR(24),
      MGRNBR CHAR(6),
      ADMDEPT CHAR(2),
      UNIQUE (DEPTNO, MGRNBR))
```

Example 5

Create a table to keep track of customers who pay their bills on time. Then create another table to keep track of current orders. Design the tables to enforce the following business rules:

1. Never accept an order from a customer who does not pay his bills.
2. Do not finance orders of less than $1,000.00 worth of merchandise.
3. Always get a down payment of at least 25 percent on financed orders.
4. Never negotiate interest rates to below 11 percent.
5. Ensure maximum performance when retrieving customer orders based on the assigned "order ID."

Note: The following example is intended to show the specification of as many different data types and options as possible. For this reason, proper data base design and table normalization have been ignored. Proper design and normalization of your databases is imperative for maximum system performance.

```
CREATE TABLE ACCOUNTING.PAYING_CUSTOMERS
 (TOTAL_NUM_ORDERS    INTEGER NOT NULL,
  FIRST_ORDER         DATE,
  LAST_ORDER          DATE,
  COMPANY_NAME        CHAR(40) NOT NULL,
  TOTAL_GROSS_ORDERS  DECIMAL(12,2) NOT NULL,
  AVG_GROSS_PER_ORDER DECIMAL(9,2) NOT NULL,
  PRIMARY KEY (COMPANY_NAME));
```

(Example 5 continued on next page)

```
CREATE TABLE ORDERS
 (ORDER_ID_NUMBER     INTEGER NOT NULL PRIMARY KEY,
  CUSTOMER_NAME       CHAR(40) DEFAULT 'INTERNAL ORDER',
  SHIPMENT_ID_NUMBER  DECIMAL(6,0) NOT NULL UNIQUE,
  GROSS_AMOUNT        NUMERIC(8,2) NOT NULL,
  PERCENT_DOWN_PMT    REAL DEFAULT 100.0
                      CHECK (PERCENT_DOWN_PMT >= 25.0),
  NUM_PAYMENTS        SMALLINT DEFAULT NULL,
  INTEREST_RATE       DOUBLE PRECISION CHECK (INTEREST_RATE >= 11.0)
                      CONSTRAINT ORDERS_MIN_INTEREST_RATE,
  DATE_PROMISED       DATE,
  TIME_PROMISED       TIME,
  DATE_AND_TIME_SHIPPED TIMESTAMP NOT NULL WITH DEFAULT,
  AVG_ITEM_QUANTITY   FLOAT,
  AVG_NUM_ITEMS       FLOAT(6),
  TOTAL_PIECES        INTEGER NOT NULL WITH DEFAULT,
  FOREIGN KEY (CUSTOMER_NAME) REFERENCES
              ACCOUNTING.PAYING_CUSTOMERS(COMPANY_NAME),
  CHECK (GROSS_AMOUNT > 1000.0 OR PERCENT_DOWN_PMT = 100.0)
        CONSTRAINT ORDERS_MIN_AMT_FINANCED);
```

How the Tables Enforce the Business Rules:

The following numbers correspond to the numbers of the business rules specified at the beginning of this example.

1. The foreign key on ORDERS.CUSTOMER-NAME prevents orders from being inserted into the table ORDERS unless the customer's name appears in the PAYING-CUSTOMERS table in column COMPANY-NAME. Any attempt to insert an order for a nonpaying customer is rejected.
2. Constraint ORDERS-MIN-AMT-FINANCED prevents an order from being inserted if an attempt is being made to finance an order of less than $1000.00.
3. CHECK (PERCENT-DOWN-PMT >= 25.0) stops orders that are being financed with a down payment of less than 25 percent. Since the user did not specify a constraint name, the system generates a unique name. It is recommended that you give meaningful names to all constraints so that error messages are more meaningful when a constraint name is in the message.
4. Constraint ORDERS-MIN-INTEREST-RATE stops orders with an interest rate below 11 percent.
5. Making ORDER-ID-NUMBER the primary key causes the system to generate an index based on this column for quick access to the data records.

CREATE TRIGGER/RULE

Note: For an overview and examples of procedures and triggers, see the *CA Datacom/DB SQL User Guide*.

In the following table, YES indicates a valid execution method for this statement.

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CREATE TRIGGER/RULE	YES	YES	YES

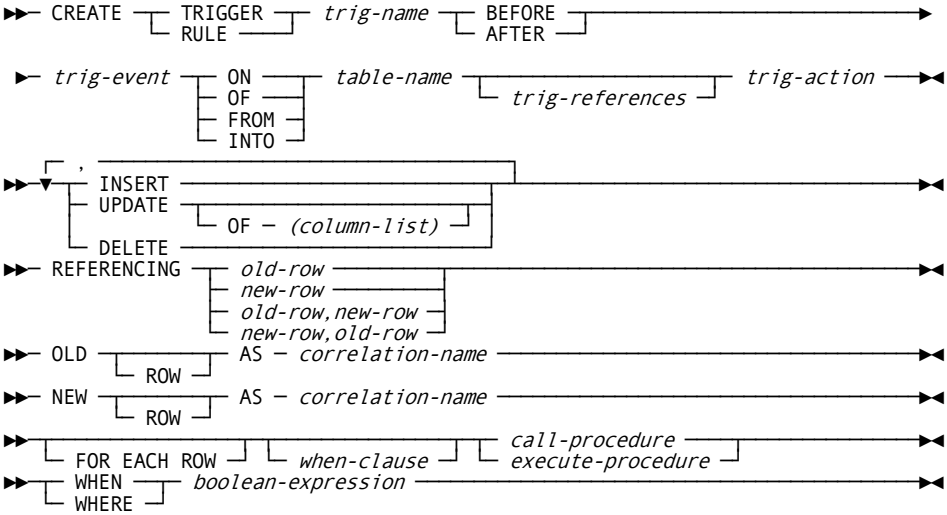
The CREATE TRIGGER/RULE statement creates a user-specified set of instructions whose execution is triggered by a specified maintenance request against a certain base table. See the *CA Datacom/DB SQL User Guide* for information about when a created trigger is recognized by applications that are currently executing.

Note:

- Statement-level rules are not supported by CA Datacom/DB, that is, a TRIGGER/RULE can only be invoked for each row affected, not once for a statement that can affect many rows.
- Use of host variables, and therefore OUT and INOUT parameters, is forbidden in a procedure call that is part of a triggered action.
- To allow the trigger to be recognized during the execution of the plan, plans containing SQL statements that reference the table to which the trigger is being added are marked invalid and automatically rebound the next time they are executed.
- A single event can invoke multiple triggers. When this occurs, the triggers are invoked according to the date and time created with the most recently created being invoked last.

With regard to table partitioning, CREATE statements may not be issued against a table which is partitioned nor against a partition.

Note: For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.



CREATE TRIGGER/RULE

(Required) Specify for the CREATE either a TRIGGER or a RULE.

trig-name

Specify the SQL-name of the trigger. SQL-names are SQL-identifiers.

BEFORE/AFTER

Trigger BEFORE or AFTER the CA Datacom maintenance occurs.

trig-event* ON/OF/FROM/INTO *table-name

Specify a trigger event, one of the listed options, and a table name. Options ON, OF, FROM, and INTO are supplied for syntax compatibility.

INSERT/UPDATE/DELETE

Part of the *trig-event* syntax. You can use up to three comma-separated occurrences of these three maintenance types (INSERT, UPDATE, DELETE), but each maintenance type can only be used once in the *trig-event* definition.

OF

(Optional) Part of the *trig-event* syntax. Supplied for syntax compatibility.

(column-list)

is a list of columns whose value-change triggers a trigger. Omission implies all columns are triggers.

REFERENCING

Part of the *trig-references* syntax.

OLD

Part of the old-row syntax. Refers to the row prior to UPDATE or DELETE.

NEW

Part of the new-row syntax. Refers to the row that results from execution of the INSERT or UPDATE.

ROW

(Optional) Part of the old-row and new-row syntax.

AS *correlation-name*

Part of the old-row and new-row syntax. The *correlation-name* is the name which is to be used to reference a column from the before-maintenance or after-maintenance image of the row being maintained. For example, if ACCOUNT_NUMBER is a column name, OLD_ACCOUNT_ROW is the before-image *correlation-name,* and NEW_ACCOUNT_ROW, then OLD_ACCOUNT_ROW.ACCOUNT_NUMBER can be used to refer to the account number prior to an UPDATE (or DELETE), and NEW_ACCOUNT_ROW.ACCOUNT_NUMBER may be used to refer to the account number after an UPDATE (or INSERT) occurs. The use of this name is, of course, limited to the CREATE statement itself.

FOR EACH ROW

(Optional) Part of the *trig-action* syntax. Denotes this trigger as one that executes once per maintained row rather than once per SQL statement. ROW level is used (the default) even if you do not specify this optional clause.

call-procedure

Part of the *trig-action* syntax.

As a convenience to users of CREATE TRIGGER who need to pass large base table rows to a triggered procedure, a special DATACOM_WHOLE_ROW column is available. DATACOM_WHOLE_ROW is only visible during execution of the CREATE TRIGGER statement. When you pass the special DATACOM_WHOLE_ROW column name to your procedure, the procedure receives a CHAR parameter containing the entire row in CA Datacom/DB format. Following is an example of the use of DATACOM_WHOLE_ROW:

```
CREATE TRIGGER myTrigger AFTER DELETE FROM myTable
  REFERENCING OLD ROW AS old_row
  CALL handleMaintenance('DELETE', old_row.datacom_whole_row)
```

We recommend that you use this special DATACOM_WHOLE_ROW column name only if your parameter list becomes unwieldy because of an excessive number of columns being passed.

execute-procedure

Part of the *trig-action* syntax.

As a convenience to users of CREATE TRIGGER who need to pass large base table rows to a triggered procedure, a special DATACOM_WHOLE_ROW column is available. DATACOM_WHOLE_ROW is only visible during execution of the CREATE TRIGGER statement. See the information previously given in the *call-procedure* description about DATACOM_WHOLE_ROW.

WHEN/WHERE *boolean-expression*

This is the when-clause syntax. The when-clause executes before the triggered action (the procedure call) is taken and cancels execution of the procedure if the *boolean-expression* evaluates to UNKNOWN or FALSE. The *boolean-expression* is similar to the WHERE clause of the SELECT statement but may not include subqueries or functions of any type. The *correlation-name* (see previous description) can be used to reference values in the row being maintained.

The trigger only activates on rows for which the *boolean-expression* evaluates to TRUE. If the WHEN clause containing the *boolean-expression* is omitted, the trigger always triggers.

CREATE VIEW

In the following table, YES indicates a valid execution method for this statement.

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
CREATE VIEW	YES	YES	YES

Note: To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The CREATE VIEW statement creates a view (derived table) of one or more tables or views.

When a CREATE VIEW statement successfully executes, a VIEW entity-occurrence is defined in Datadictionary in PRODuction status.

All of the tables and views specified in the CREATE VIEW statement must be in databases of the same security type, that is to say, either the CA Datacom/DB External Security Model or the SQL Security Model. See the *CA Datacom Security Reference Guide* for more information about security models.

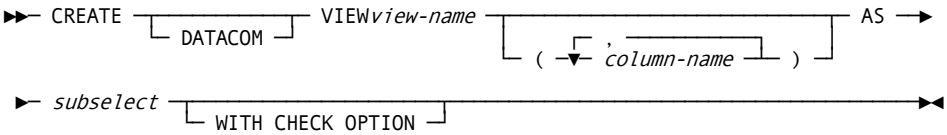
Privileges

If the Multi-User Facility is secured, whether the creator of the view may access that view depends on the security model of the view. Under the SQL Security Model, when a view is created the user who executes the CREATE VIEW statement is automatically granted all privileges on the view. Under the CA Datacom/DB External Security Model, all access rights, including those for the creator of a view, must be defined through the external security product. See the *CA Datacom Security Reference Guide* for more information.

With regard to table partitioning, CREATE statements may not be issued against a table which is partitioned nor against a partition.

Note: For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

Following is the syntax diagram for the CREATE VIEW statement:



Description

DATACOM

(Optional) See the information about DATACOM VIEWs in the *CA Datacom/DB SQL User Guide*.

view-name

The unqualified or qualified name of the view. This name, including the implicit or explicit qualifier, must not name a table, view or synonym already described in the Datadictionary.

If you specify SQLMODE=ANSI or SQLMODE=FIPS in the Preprocessor options, the view name can be 1 to 18 characters in length.

If you specify SQLMODE=DATACOM for extended mode in the Preprocessor options, the view name can be 1 to 32 characters in length.

The qualified form is the name preceded by an authorization ID and a period, for example, auth-id.view-name.

If you qualify the view-name, the qualifier designates the schema of the view. If you do not qualify the view-name, the default authorization ID is used as the qualifier.

If the CA Datacom/DB Security Facility is installed, the creator of the view is automatically granted the SELECT privilege on the view, and any other privilege that applies to a view and is a privilege that the creator has on the tables or views identified in the FROM clause of the SELECT statement. A privilege acquired by the creator is grantable only if the privilege from which it is derived is grantable by the creator.

column-name

A list of one or more names for columns in the view. The column names must be separated by commas and the list must be enclosed in parentheses.

If you specify SQLMODE=ANSI or SQLMODE=FIPS in the Preprocessor options, the column name can be 1 to 18 characters in length.

If you specify SQLMODE=DATACOM for extended mode in the Preprocessor options, the column name can be 1 to 32 characters in length.

If you do not give other names to the columns, the columns of the view have the same names as the columns of the result table of the SELECT statement.

If the result of the SELECT statement has duplicate column names, or a column derived from a function, literal, or arithmetic expression, you must give names to all the columns. Give a name for each column and do not use the same name more than once.

AS *subselect*

The subselect defines the view. At any time, the view consists of the rows that result if the SELECT statement is executed.

The subselect must be the subselect form of a SELECT statement that does not reference host variables. See Subselect (see page 647) for the subselect syntax diagram and information about the subselect parameters.

WITH CHECK OPTION

Specifies that all inserts and updates against this view are checked to ensure that the newly inserted or updated row satisfies the view definition.

For example, if you use the view to insert a row, the row can be inserted only if you can also view it using this view, that is to say, the data being inserted must be within the bounds specified in the view's definition.

The WITH CHECK OPTION can be specified only if the view is updateable and the view definition does not include a nested subquery.

See Example 3 at the end of this chapter for a view definition using the WITH CHECK OPTION.

Processing

A view is read-only and cannot be updated if its definition includes any of the following phrases:

- A FROM clause naming more than one table or view (a join)
- The keyword DISTINCT
- A GROUP BY clause

Keywords and phrases which cannot be used in the definition of a view are:

- FOR UPDATE OF
- ORDER BY
- UNION

Note: A view whose definition involves either a GROUP BY clause or a column function cannot be named in any FROM clause that contains another table or view, that is to say, the view cannot be joined. This is not a restriction on the content of the CREATE VIEW statement itself, but on queries that reference the view. An error message is issued if you violate this rule. Also, a view whose definition contains a SELECT DISTINCT may only be joined if the join itself has a SELECT DISTINCT. Again, this is a restriction on queries that reference the view, not a restriction on the view itself.

Example 1

Create a read-only view with the specifications listed in the following:

1. View name: PROJV1
2. Column names: PROJNO, PROJNAME, PROJDEP, RESPEMP, EMPNO, FIRSTNME, MIDINIT, LASTNAME
3. The view joins the tables PROJTBL and NAMETBL where a value in the RESPEMP column is equal to a value in the EMPNO column.

   Note: Since this view joins two tables, it is a read-only view.

```
CREATE VIEW PROJV1
     (PROJNO, PROJNAME, PROJDEP, RESPEMP,
      FIRSTNME, MIDINIT, LASTNAME)
AS SELECT ALL
     PROJNO, PROJNAME, DEPTNO, EMPNO,
     FIRSTNME, MIDINIT, LASTNAME
FROM PROJTBL, NAMETBL
WHERE RESPEMP = EMPNO
```

Example 2

Create an updateable view with the specifications listed in the following:

1. View name: SUPPLY
2. Column names: SNUM, SNAME, STATUS, CITY
3. Search condition: The value of STATUS must be greater than 10

This view is a row and column subset view of the table CASUPL which contains columns SNUM, SNAME, STATUS, CITY, STATE and ZIP. View SUPPLY can be used to update values for the four specified columns. However, this view would not be used to insert new rows into CASUPL since some of the columns are not included in the view definition.

```
CREATE VIEW SUPPLY
     (SNUM, SNAME, STATUS, CITY)
AS SELECT ALL
     SNUM, SNAME, STATUS, CITY
FROM CASUPL
WHERE STATUS > 10
```

Example 3

Create an updateable view with the specifications listed in the following:

1. View name: SUPPLY
2. Column names: SNUM, SNAME, STATUS, CITY, STATE, ZIP
3. Search condition: The value of STATUS must be greater than 10
4. Use the WITH CHECK OPTION

This view is a row subset view of the table CASUPL which contains columns SNUM, SNAME, STATUS, CITY, STATE and ZIP. View SUPPLY can be used to update values for all the columns, and to insert new rows into CASUPL where the value for STATUS is greater than 10. If you attempt to insert a row where the value of STATUS is less than or equal to 10, the row is not inserted because the WITH CHECK OPTION does not allow you to view any row where STATUS is not greater than 10.

```
CREATE VIEW SUPPLY
     (SNUM, SNAME, STATUS, CITY, STATE, ZIP)
AS SELECT ALL
     SNUM, SNAME, STATUS, CITY, STATE, ZIP
FROM CASUPL
WHERE STATUS > 10
WITH CHECK OPTION
```

DELETE

This SQL statement can be executed in the following ways:	**Through the Datadictionary Interactive SQL Service Facility *(interactive)***	**In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)***	**By using CA Dataquery *(SQL & Batch Modes)***
DELETE (positioned)		YES	

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
DELETE (searched)	YES	YES	YES

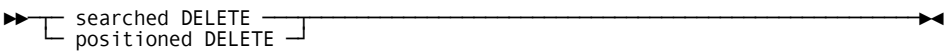
Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The DELETE statement deletes rows from a table or view. Deleting a row from a view deletes the row from the table that the view is based on.

You must specify ISOLEVEL=C (isolation level C) in the Preprocessor options when using the DELETE statement (see the *CA Datacom/DB SQL User Guide*).

Note: When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, a DELETE statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force.

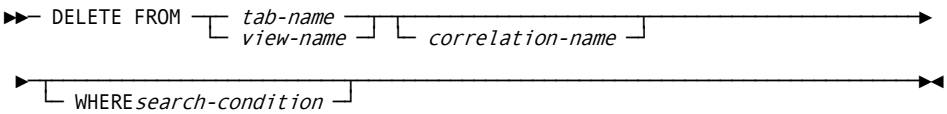
Following is the syntax diagram for a DELETE statement:



Note: The positioned DELETE is not used by CA Dataquery.

Searched DELETE

Following is the syntax diagram for the searched DELETE statement:



Description

FROM *tab-name* or FROM *view-name*

Name the table or view from which you want to delete. The table or view must be described in the Datadictionary, but must not be a Datadictionary table or a read-only view.

correlation-name

You can specify a 1- to 18-byte correlation-name to be used within the search-condition to designate the table or view.

WHERE

Introduces a condition that specifies what rows are to be deleted. You can omit the clause, give a search condition or name a cursor. If you omit the clause, all rows of the table or view are deleted.

search-condition

Each column-name in the search condition must name a column of the table or view, and the table or view must not be referenced in the FROM clause of any subselect in the search condition. See the *CA Datacom/DB SQL User Guide* for the search-condition syntax diagram.

Each column-name in the search condition which is not in a subquery must name a column of the table or view being updated. In subqueries, the table or view being updated must not be named in any FROM clause.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a row. The result of the subquery is used in applying the search condition. In actuality, the subquery is executed for each row only if it contains a correlated reference to a column of the table or view.

CURRENT OF *cursor-name*

Not used in CA Dataquery.

Example 1

Delete the row from table DEPTTBL where the value of DEPTNO equals D1.

```
DELETE FROM DEPTTBL
WHERE DEPTNO = 'D1'
```

Example 2

Delete several rows from table NAMETBL, specifically, those for all employees in Department E1 or D2.

```
DELETE FROM NAMETBL
WHERE WORKDEPT = 'E1' OR WORKDEPT = 'D2'
```

DROP

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
DROP INDEX	YES	YES	
DROP SYNONYM	YES	YES	YES
DROP TABLE	YES	YES	YES
DROP VIEW	YES	YES	YES
DROP PROCEDURE	YES	YES	YES
DROP TRIGGER/RULE	YES	YES	YES

Note: YES indicates a valid execution method for this statement.

Important! There is also a DROP PLAN statement that can only be submitted through DBSQLPR. See the information about DBSQLPR in the CA Datacom/DB SQL User Guide.

Note: To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The DROP statement is a CA Datacom/DB extension. DROP removes an SQL-accessible object. The object's description is removed from the Datadictionary. Any application plans that reference the object are invalidated.

Important! When an object is dropped, any objects that are directly or indirectly dependent on that object are either dropped or marked for rebind.

When a table or view is dropped, the prepared statements in plans that reference the table or view are marked non-executable until rebound. An attempt to execute these statements invokes automatic rebind, and the rebind fails. If the plans containing referencing statements are currently executing or binding, the DROP request is aborted. This is always the case when the plan containing the DROP statement references the table or view, because this is an attempt to invalidate its own plan and is not allowed. Dropping an index causes all plans dependent on the indexed table to be marked invalid. You can run a Datadictionary Relationship Report to find out what plans are dependent on a table. See the *CA Datacom Datadictionary Batch Reference Guide* for information on Relationship Reports.

The DROP statement does not process and you receive a -118 return code when the Datadictionary entity-occurrence definition of the table, view, synonym, or index you specify is protected with a password or a Lock Level 1 or 2. The error message also includes a Datadictionary Service Facility (DSF) return code. The DSF return codes are:

- IPW (for password protected)
- IOR (for Lock Level 1 protected)
- NTF (for Lock Level 2 protected)

See the *CA Datacom/DB SQL User Guide* for more information., and see the Datadictionary documentation for information on passwords and lock levels.

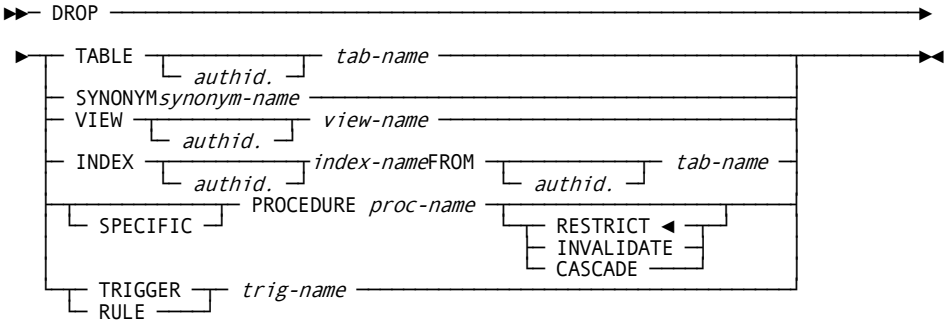
Note: When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, a DROP statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force.

Also note that the DROP statement is rejected if other DML statements (see the *CA Datacom/DB SQL User Guide*) are in the same plan or if the table being dropped is in use by other plans.

With regard to table partitioning, DROP statements may not be issued against a table which is partitioned nor against a partition.

Note: For more information about table partitioning, see the *CA Datacom/DB Database and System Administration Guide*.

Following is the syntax diagram for the DROP statement:



Note: DROP INDEX is not used by CA Dataquery.

Description

authid.

An optional identifier of the schema for the table, view, or index. If specified for the index-name, it must be the same as the authorization ID of the table you have specified with the tab-name in the FROM tab-name clause. Use a period (.) to concatenate the authorization ID to the table or view name, for example, authid.tab-name.

TABLE

Indicates that the object you are dropping is a table.

tab-name

Specify the name of the table you want to drop. The tab-name must identify a table described in the Datadictionary, other than a Datadictionary table.

The DROP TABLE statement removes the table from the schema, obsoletes the table from the Datadictionary, and removes the table from the CA Datacom/DB Directory (CXX).

Important! The table data is deleted and the space is reclaimed when you drop a table.

All views and synonyms based on the table are dropped. All application plans and statements that reference the table are invalidated. If the table definition exists in TEST and/or HISTory status in Datadictionary, those status/versions are deleted at the same time as the PRODuction status.

All columns (columns appear as FIELD occurrences in Datadictionary), keys, elements, and support data related to the table are also obsoleted. The support data includes aliases, descriptors, relationship definitions, and text (SQL comments).

SYNONYM

Indicates that the object you are dropping is a synonym.

synonym-name

Specify the name of the synonym you want to drop. Dropping a synonym does not affect the table, view or synonym referenced by the dropped synonym.

VIEW

Indicates that the object you are dropping is a view.

view-name

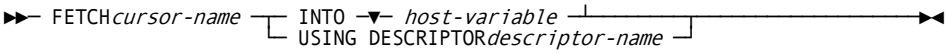
Specify the name of the view you want to drop. The view name must identify a view described in the Datadictionary. The definition of the view is removed from the Datadictionary. The definition of any view that is directly or indirectly dependent on that view is also removed. When the definition of a view is removed from the Datadictionary, all privileges on that view are also dropped.

INDEX

Not used in CA Dataquery.

EXECUTE PROCEDURE

For information about the EXECUTE PROCEDURE statement, see CALL/EXECUTE PROCEDURE (see page 527).



GRANT

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
GRANT	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

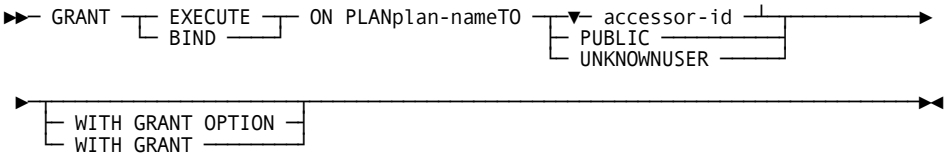
The GRANT statement grants privileges on tables and views in databases that are secured under the SQL Security Model. All of the tables and views specified in the GRANT statement must be in databases secured under the SQL Security Model. If tables and views are in databases secured under the CA Datacom/DB External Security Model, the GRANT statement is rejected with an SQL error code -273. See the *CA Datacom Security Reference Guide* for more information about security models.

Plan Security

SQL plans are securable. With plan security you can create a plan such that, in order to execute the plan, an accessor ID must have the plan EXECUTE privilege for that plan. The plan EXECUTE privilege can be granted with the GRANT statement and revoked with the REVOKE statement. Also see the information on the REVOKE statement and the CHECKPLAN=, CHECKWHEN=, CHECKWHO=, and SAVEPLANSEC= options in the *CA Datacom/DB* SQL *User Guide*). For detailed information about plan security, see the *CA Datacom Security Reference Guide*.

Note: To grant a plan privilege you must possess that privilege WITH GRANT OPTION or be a Global Owner. To revoke a plan privilege you must have granted the privilege or be a Global Owner. See the *CA Datacom Security Reference Guide* for more information on Global Owners.

Following is the syntax diagram for the plan security version of the GRANT statement:



Description of Plan Security Diagram

EXECUTE

Grants the PLAN EXECUTE privilege.

BIND

Grants the PLAN BIND privilege.

ON PLAN *plan-name*

Specifies the name of the plan to which the PLAN EXECUTE or PLAN BIND privilege is to be granted.

TO *accessor-id*

Specify the accessor ID of one or more users to whom you are granting the privileges. This is a *user's* ID, not a schema auth-id. Do not specify your own accessor ID, that is to say, you cannot grant privileges to yourself.

TO PUBLIC

Specify PUBLIC when you are granting the specified privileges to all users. A new user automatically has any privileges previously granted to the public.

TO UNKNOWNUSER

Specify UNKNOWNUSER when you are granting the specified privileges to users whose identities cannot be determined by CA Datacom/DB security.

WITH GRANT OPTION

Specify this option if you want the user to whom you have granted the privilege to be able to grant it to another user. WITH GRANT OPTION cannot be used with PUBLIC.

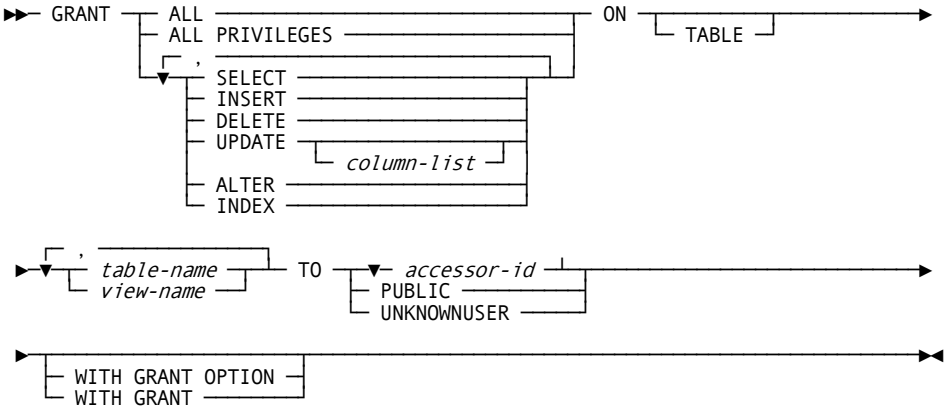
WITH GRANT

Specify WITH GRANT if you want the user to whom you have granted the privilege to be able to grant it to another user. WITH GRANT cannot be used with PUBLIC or with UNKNOWNUSER.

Following is the non-plan security syntax diagram for the GRANT statement:

The following are CA Datacom/DB extensions.

- ALL
- TABLE
- ALTER
- INDEX
- UNKNOWNUSER
- WITH GRANT (without the keyword OPTION)



Description of Non-Plan Security Diagram

ALL or ALL PRIVILEGES

Grants all privileges (excluding ALTER and INDEX) for which you have GRANT authority on all tables or views named in the ON clause. GRANT ALL is a CA Datacom/DB extension.

If you do not use ALL, you must use one or more of the following keywords. Each keyword grants the privilege described, but only as it applies to the tables or views named in the ON clause.

SELECT

Grants the privilege to use the SELECT statement.

INSERT

Grants the privilege to use the INSERT statement.

DELETE

Grants the privilege to use the DELETE statement.

ALTER

Grants the privilege to use the ALTER statement. GRANT ALTER is a CA Datacom/DB extension.

INDEX

Grants the privilege to execute the CREATE INDEX and DROP INDEX statements. GRANT INDEX is a CA Datacom/DB extension.

UPDATE

Grants the privilege to use the UPDATE statement.

UPDATE (*column-list*)

Grants the privilege to update only the named columns. Each column-name must belong to every table or view named in the ON clause. The column names must be separated by commas and the list must be enclosed with parentheses.

ON or ON TABLE

Introduces a list of table and/or view names. ON TABLE is a CA Datacom/DB extension.

table-name* or *view-name

Specify the name of one or more tables or views on which you are granting the privileges. The list can be a list of table names or view names, or a combination of the two. The names must be separated by commas.

If you name specific privileges, then for each table or view you name, you must have all those privileges with GRANT authority. If you use GRANT ALL, then for each table or view you name, you must have at least one privilege with GRANT authority.

TO *accessor-id*

Specify the accessor ID of one or more users to whom you are granting the privileges. This is a *user's* ID, not a schema auth-id. If listing more than one accessor ID, separate them with commas. Do not specify your own accessor ID, that is to say, you cannot grant privileges to yourself.

TO PUBLIC

Specify PUBLIC when you are granting the specified privileges to all users. A new user automatically has any privileges previously granted to the public.

TO UNKNOWNUSER

Specify UNKNOWNUSER when you are granting the specified privileges to users whose identities cannot be determined by CA Datacom/DB security. UNKNOWNUSER is a CA Datacom/DB extension.

WITH GRANT OPTION

Specify this option if you want the user to whom you have granted the privilege to be able to grant it to another user. The WITH GRANT OPTION cannot be used with PUBLIC.

WITH GRANT

Specify WITH GRANT if you want the user to whom you have granted the privilege to be able to grant it to another user. WITH GRANT cannot be used with PUBLIC or with UNKNOWNUSER. WITH GRANT is a CA Datacom/DB extension.

Example 1

Grant SELECT privileges on table TEMPL to user PULASKI.

```
GRANT SELECT
    ON TABLE CA.TEMPL
    TO PULASKI
```

Example 2

Grant UPDATE privileges on columns EMPNO and WORKDEPT in table TEMPL to all users.

```
GRANT UPDATE (EMPNO, WORKDEPT)
    ON TABLE CA.TEMPL
    TO PUBLIC
```

Example 3

Grant all privileges on table TEMPL to users KWAN and THOMPSON, including the WITH GRANT OPTION.

```
GRANT ALL PRIVILEGES
    ON TABLE CA.TEMPL
    TO KWAN, THOMPSON
    WITH GRANT OPTION
```

IF-THEN Statement

For details about this statement, see IF-THEN Statement (see page 569).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

INSERT

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
INSERT	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

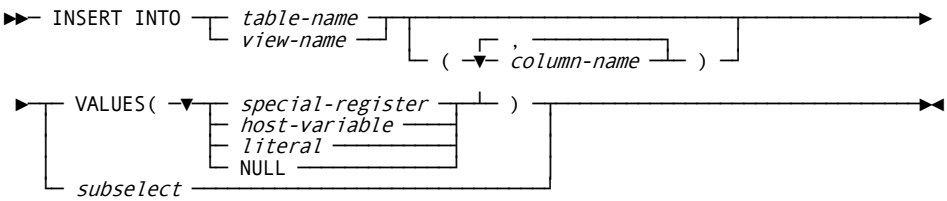
The INSERT statement inserts rows into a table or view. Inserting a row into a view inserts the row into the table upon which the view is based. You must specify ISOLEVEL=C (isolation level C) in the Preprocessor options when using the INSERT statement (see the *CA Datacom/DB SQL User Guide*).

Note: When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, an INSERT statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force.

Also note that constraints that existed in versions before r10 can act differently in Release 10.0 and above because, in Release 10.0 and above, columns containing a NULL rather than a value do *not* cause a CHECK constraint to be violated. Constraints are considered, in Release 10.0 and above, to be satisfied unless the predicates evaluate explicitly to FALSE. That is, CHECK constraints whose predicates evaluate to UNKNOWN rather than TRUE or FALSE are considered, with Release 10.0 and above, to have been satisfied. Therefore, INSERT and UPDATE statements, from versions before r10, that resulted in constrained columns being nulled are, when used in Release 10.0 and above, successful for the first time.

Following is the syntax diagram for the INSERT statement. See the *CA Datacom/DB SQL User Guide* for the special-register diagram. See Subselect (see page 647) for the subselect's syntax diagram.

Note: The special-register is a CA Datacom/DB extension.



Description

table-name or view-name

Specify the name of the table or view into which you want to insert the row(s). The table or view must be described in the Datadictionary, and must not be a Datadictionary table or any of the following types of views:

- A read-only view.
- A view of a Datadictionary table.
- A view with a column that is derived from a literal or an arithmetic expression.
- A view with two columns derived from the same column of the underlying table.

column-name

Specify the name of one or more columns for which you provide insert values. You can name the columns in any order. Each column must belong to the table or view you specified, and you cannot name the same column more than once. The column names must be separated by commas and the list must be enclosed with parentheses.

VALUES

Introduces one row of values to be inserted. The values of the row are the values of the keywords, literals, or host variables specified in the clause. The values must be separated by commas and the list must be enclosed with parentheses.

When the statement is executed, the number of values in the VALUES clause must equal the number of names in the column list. The first value is inserted in the first column in the list, the second value in the second column, and so on.

special-register

See the *CA Datacom/DB SQL User Guide* for the special-register diagram. This is a CA Datacom/DB extension.

host-variable

Specify a host-variable which must be a variable that is described in your program in accordance with the rules for declaring host variables.

literal

Specify a literal consistent with the data type of the column.

NULL

Use the NULL keyword to specify that the value(s) being inserted are null values.

subselect

If you specify a subselect, the rows of the subselect's result table are inserted into the table or view you specified in the INSERT statement. The result can be one row, more than one row or no rows. If no rows are inserted, SQLCODE is set to +100. See Subselect (see page 647) for the subselect's syntax diagram.

The table or view named after INSERT INTO must not be named in a FROM clause of the subselect or the FROM clause of any subquery in the subselect.

The number of columns in the result table must equal the number of names in the column list. The value of the first column of the result is inserted in the first column in the list, the second value in the second column, and so on.

Rules for Inserting

Insert values must satisfy the following rules:

Default values:

The value inserted in any column not in the column list is the default value of the column. Columns without a default value must be included in the column list.

If you insert into a view, the default value is inserted into any column of the base table that is not included in the view.

All columns of the base table that are not in the view must have default values.

Length:

If the insert value of a column is a number, the column must be a numeric column with the capacity to represent the integral part of the number.

If the insert value of a column is a string, the column must be a string column with a length attribute at least as great as the length of the string.

If insert values do not adhere to the previously given rules, or if any other error occurs during the execution of the INSERT statement, no rows are inserted.

Processing

Rows can be inserted that do not conform to the definition of a view. These rows cannot appear in the view, but are inserted into the base table of the view.

Unless appropriate locks already exist, one or more exclusive locks are acquired at the execution of a successful INSERT statement. Until the locks are released, an inserted row can be accessed only by the unit of recovery in which the insert was performed.

Example 1

Insert a row into table EMP.

```
INSERT INTO EMP
     VALUES('000315', 'JOHN', 'T', 'SMITH', 'A1', '1234',
             870110, 32, 19, 'M', 550422, 16325)
```

Example 2

Load the temporary table TEMPEMP with data from table EMP.

```
INSERT INTO SMITH.TEMPEMP
     SELECT *
     FROM EMP
```

Example 3

Load the temporary table TEMPEMP with data from Department E1 from EMP.

```
INSERT INTO SMITH.TEMPEMP
     SELECT *
     FROM EMP
     WHERE WORKDEPT = 'E1'
```

ITERATE Statement

For details about this statement, see ITERATE Statement (see page 570).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

LEAVE Statement

For details about this statement, see LEAVE Statement (see page 570).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

REPEAT-UNTIL Statement

For details about this statement, see REPEAT-UNTIL Statement (see page 575).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

REVOKE

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
REVOKE	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

Important! If CA Datacom/DB security is not installed at your site, the REVOKE statement is rejected with an SQL error code -559.

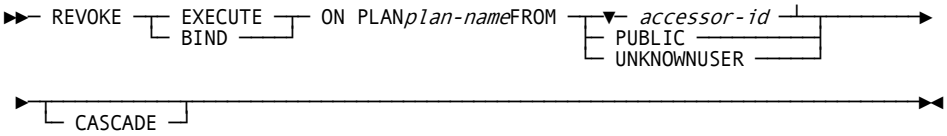
CA Datacom/DB provides the REVOKE statement as an extension so you can revoke privileges on tables and views for which privileges have been granted through the GRANT statement. All tables and views in the statement must belong to databases secured under the SQL Security Model. If tables and views are in databases secured under the CA Datacom/DB External Security Model, the GRANT and REVOKE statements are rejected with an SQL error code -273. See the *CA Datacom Security Reference Guide* for more information about security models.

Plan Security

SQL plans are securable. With plan security you can create a plan such that, in order to execute the plan, an accessor ID must have the plan EXECUTE privilege for that plan. The plan EXECUTE privilege can be granted with the GRANT statement and revoked with the REVOKE statement. Also see the information on the REVOKE statement and the CHECKPLAN=, CHECKWHEN=, CHECKWHO=, and SAVEPLANSEC= options in the *CA Datacom/DB* SQL *User Guide*). For detailed information about plan security, see the *CA Datacom Security Reference Guide*.

Note: To grant a plan privilege you must possess that privilege WITH GRANT OPTION or be a Global Owner. To revoke a plan privilege you must have granted the privilege or be a Global Owner. See the *CA Datacom Security Reference Guide* for more information on Global Owners.

Following is the syntax diagram for the plan security version of the REVOKE statement:



Description of Plan Security Diagram

EXECUTE

Revokes the PLAN EXECUTE privilege.

BIND

Revokes the PLAN BIND privilege.

ON PLAN *plan-name*

Specifies the name of the plan from which the PLAN EXECUTE or PLAN BIND privilege is to be revoked.

FROM *accessor-id*

Specify the accessor ID of a user from whom you are revoking the privileges that were granted with a GRANT statement. This is a *user's* ID, not a schema auth-id.

FROM PUBLIC

Specify PUBLIC when you are granting or revoking the specified privileges to or from all users. A new user automatically has any privileges previously granted to the public.

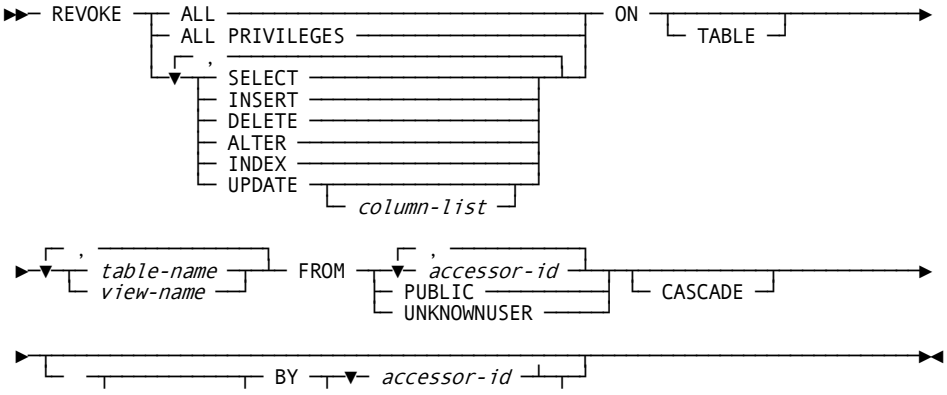
FROM UNKNOWNUSER

Specify UNKNOWNUSER when you are revoking the specified privileges from users whose identities cannot be determined by the CA Datacom/DB security.

CASCADE

If CASCADE is specified, any other dependent privileges that have been granted to others (through the GRANT statement) are also revoked. If a REVOKE is issued without CASCADE and the grantee granted privileges to other users, the REVOKE is not permitted. The CASCADE option of REVOKE does not block the cascading effect of a revoke but operates instead as a fail-safe device. Specifying CASCADE simply acknowledges your understanding that there are cascading effects.

Following is the non-plan security syntax diagram for the REVOKE statement:



Description of Non-Plan Security Diagram

ALL or ALL PRIVILEGES

Revokes all privileges (excluding ALTER and INDEX) which the executor of the REVOKE statement has previously granted to the specified users.

If you do not use ALL, you must use one or more of the following keywords. Each keyword revokes the privilege described, but only as it applies to the tables or views named in the ON clause.

SELECT

Revokes the privilege to use the SELECT statement.

INSERT

Revokes the privilege to use the INSERT statement.

DELETE

Revokes the privilege to use the DELETE statement.

ALTER

Revokes the privilege to use the ALTER statement.

INDEX

Not used in CA Dataquery.

UPDATE

Revokes the privilege to use the UPDATE statement.

UPDATE (*column-name*)

Revokes the privilege to update only the named columns. Each column-name must belong to every table or view named in the ON clause. The column names must be separated by commas and the list must be enclosed with parentheses.

ON or ON TABLE

Introduces a list of table and/or view names.

table-name* or *view-name

Specify the name of one or more tables or views on which you are revoking the privileges. The list can be a list of table names or view names, or a combination of the two. The names must be separated by commas.

For each table or view you identify, you (or the indicated grantors) must have granted (using the GRANT statement) at least one of the specified privileges on that table or view to all identified users (including PUBLIC, if specified).

FROM *accessor-id*

Specify the accessor ID of one or more users from whom you are revoking the privileges that were granted with a GRANT statement. This is a *user's* ID, not a schema auth-id. If listing more than one accessor ID, separate them with commas.

FROM PUBLIC

Specify PUBLIC when you are revoking the specified privileges from all users.

FROM UNKNOWNUSER

Specify UNKNOWNUSER when you are revoking the specified privileges from users whose identities cannot be determined by the CA Datacom/DB security.

CASCADE

If CASCADE is specified, any other dependent privileges that have been granted to others (through the GRANT statement) are also revoked. If a REVOKE is issued without CASCADE and the grantee granted privileges to other users, the REVOKE is not permitted. The CASCADE option of REVOKE does not block the cascading effect of a revoke but operates instead as a fail-safe device. Specifying CASCADE simply acknowledges your understanding that there are cascading effects.

GRANTED

Allows you to specify that you are revoking privileges that were GRANTED BY another user(s). The accessor ID(s) specified in GRANTED BY must therefore have previously granted the specified privileges to the grantee. This form of the REVOKE statement may only be executed by a global database owner. For information about global database owners, see the *CA Datacom Security Reference Guide*.

BY *accessor-id*

The BY indicates that the person revoking the privileges is doing so on behalf of another user. Specify the accessor ID of the person who granted the privileges you are revoking. The accessor IDs must be separated by commas.

BY ALL

Specify ALL when revoking privileges granted by all other users to the user identified in the FROM clause of the REVOKE statement.

Example 1

Revoke SELECT privileges on table TEMPL from user PULASKI.

```
REVOKE SELECT
    ON TABLE CA.TEMPL
    FROM PULASKI
```

Example 2

Revoke UPDATE privileges on table TEMPL previously granted to all users.

Note: Grants to specific users are not affected.

```
REVOKE UPDATE
    ON TABLE CA.TEMPL
    FROM PUBLIC
```

Example 3

Revoke all privileges on table TEMPL from users KWAN and THOMPSON.

```
REVOKE ALL PRIVILEGES
    ON TABLE CA.TEMPL
    FROM KWAN, THOMPSON
```

SELECT

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
select-into statement		YES	
select-statement	YES	(use DECLARE CURSOR)	YES
full-select statement	(part of the select-statement)	(part of the select-statement)	(part of the select-statement)
subselect	(part of full-select statement)	(part of full-select statement)	(part of full-select statement)

Note: YES indicates a valid execution method for this statement.

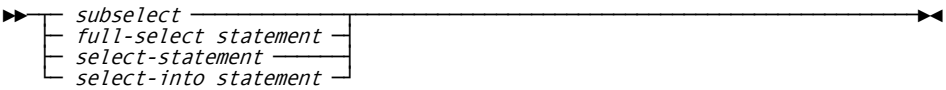
Information about the DECLARE CURSOR statement can be found in the *CA Datacom/DB SQL User Guide*.

Note: The subselect and full-select forms of the SELECT statement are not executable directly, because they are components of other statements. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

The SELECT statement specifies a result table.

All of the tables and views specified in the SELECT statement must be in databases of the same security type, that is to say, either the CA Datacom/DB External Security Model or the SQL Security Model. See the *CA Datacom Security Reference Guide* for more information about security models.

Following is the syntax diagram for a SELECT statement:



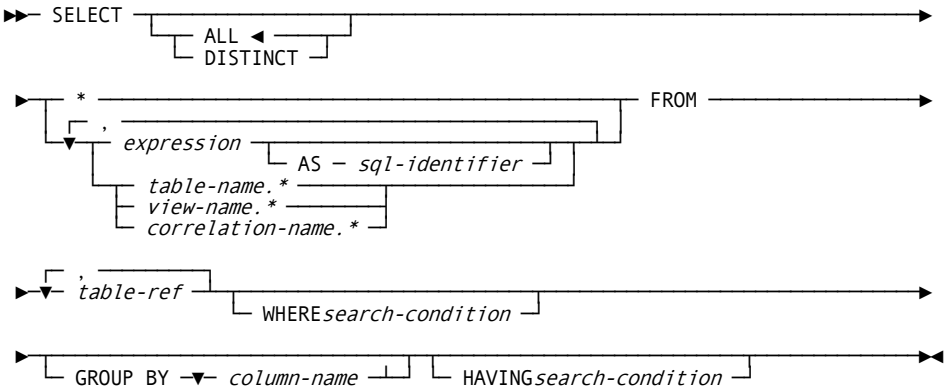
Note: The select-into statement is not used by CA Dataquery.

Subselect

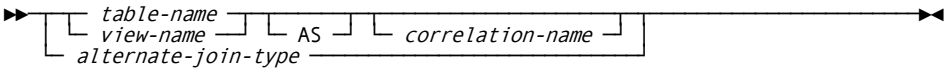
The subselect is a component of:

1. The full-select statement.
2. The CREATE VIEW statement.
3. The INSERT statement.
4. Certain predicates, which, in turn, are components of a subselect. A subselect that is a component of another subselect is called a subquery.

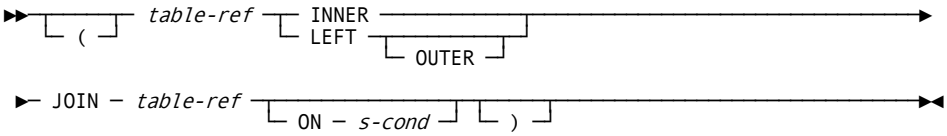
Following is the syntax diagram for the subselect statement:



The table-ref shown in the syntax box immediately preceding the following one has syntax as follows:



The *alternate-join-type* shown in the syntax box immediately preceding the following one has syntax as follows:



Note: The *s-cond* (search-condition) specified in the optional ON clause differs from the one in the WHERE clause in that the ON clause defines the join conditions that determine which rows contain nulls, as opposed to the WHERE clause, which eliminates rows from the result entirely. Also note that if you use the optional parentheses, they must be balanced. That is, if you use an open parenthesis, you must also use a close parenthesis.

The previously shown JOIN syntax is compatible with Ingres, DB2, and ANSI SQL3 Core SQL.

Description

A subselect specifies a result table derived from the tables or views identified in the FROM clause. The derivation can be described as a sequence of operations in which the result of each operation is input for the next.

The sequence of the hypothetical operations is:

1. FROM clause
2. WHERE clause
3. GROUP BY clause
4. HAVING clause
5. SELECT clause

Descriptions of these clauses appear in the order as previously listed.

FROM Clause

FROM table-ref

Where table-ref can be a table-name, view-name, or alternate-join-type. If table-ref is a table-name or view-name, it names:

- A single table or view.
- Several tables and/or views to produce an intermediate result table.

If table-ref is an alternate-join-type, see the information about joins in the *CA Datacom/DB SQL User Guide*.

You can reference up to 20 tables in a FROM clause of a query when you are performing a join. For example, if a view is based on five tables, you can name that view in the FROM clause and up to 15 other tables. The names must be separated by commas.

The intermediate result table contains all possible combinations of the rows of the named tables or views. Each row of the result is a row from the first table or view concatenated with a row from the second table or view, concatenated in turn with a row from the third, and so on. The number of rows in the result is the product of the number of rows in all the named tables or views.

The list of names in the FROM clause must conform to these rules:

1. Each table-name and view-name must name a table or view described in the Datadictionary.
2. If the FROM clause is in a subquery of a basic predicate, no view named can use either GROUP BY or HAVING.
3. In other cases, if a named view uses GROUP BY, HAVING, or a column function, no other table or view can be named. The subselect statement is processed as if it contained the GROUP BY, HAVING, or column function used in the definition of the view.

correlation-name

The 1- to 18-byte *correlation-name* applies to the table or view named by the immediately preceding table-name or view-name. The correlation name can be used elsewhere in the statement to designate that table or view. Use a blank to separate the correlation name from the table-name or view-name.

WHERE Clause

WHERE *search-condition*

Produces an intermediate result table by applying the search-condition to each row of table R, which is the result of the FROM clause. The result table contains the rows of R for which the search condition is true.

Note: If using alternate-join-type, see the additional information about WHERE clauses in the *CA Datacom/DB SQL User Guide*. Also, in addition to the information about search conditions that follows, see the additional alternate-join-type search condition information in Subselect (see page 647).

The search-condition describes a search condition that conforms to the following rules:

1. The condition is formed (see the *CA Datacom/DB SQL User Guide*).
2. Each column-name in the search-condition either unambiguously identifies a column of R, or is a correlated reference. A correlated reference is allowed only in a subquery.
3. A column-name in the search-condition does not identify a column that is derived from a function or a grouping column. A column of a view can be derived from a function or a grouping column.
4. The search condition does not include a function unless the argument of the function is a correlated reference. This is only possible in a subquery of a HAVING clause.

Any subquery in the search-condition is effectively executed for each row of R and the results used in the application of the search-condition to the given row of R. A subquery is actually executed for each row of R only if it includes a correlated reference to a column of R.

GROUP BY Clause

GROUP BY

Produces an intermediate result table by grouping the rows of R, where R is the result of the previous clause.

column-name

The column-name must unambiguously name a column of R. Each column named is called a group column. The names must be separated by commas.

The result of GROUP BY is a set of groups of rows. In each group of more than one row:

1. All values of each grouping column are equal.
2. All rows with the same set of values of the grouping columns are in the same group.

For the purpose of grouping, all null values within a grouping column are considered equal.

Because every row of a group contains the same value of any grouping column, the name of a grouping column can be used in a search condition in a HAVING clause or an expression in a SELECT clause. In each case, the reference specifies only one value for each group.

GROUP BY must not be used in:

- A subquery of a basic predicate.
- A subselect whose FROM clause names a view that used GROUP BY or HAVING.

HAVING Clause

HAVING *search-condition*

Produces an intermediate result table by applying the search-condition to each group of R, where R is the result of the previous clause. If that clause is not GROUP BY, all rows of R are considered as one group. The result table contains those groups of R for which the search condition is true.

The search-condition describes a search condition that conforms to the following rules:

1. The condition is formed (see the *CA Datacom/DB SQL User Guide*).
2. Each column-name in the search-condition must:
   a. Unambiguously identify a grouping column of R, or
   b. Be a correlated reference, or
   c. Be specified within a function.

A group of R to which the search condition is applied supplies the argument for each function in the search condition, except for any function whose argument is a correlated reference.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a group of R, and the results used in applying the search condition. In actuality, the subquery is executed for each group only if it contains a correlated reference.

A correlated reference to a group of R must either identify a grouping column or be contained within a function.

The HAVING clause must not be used in:

- A subquery of a basic predicate.
- A subselect whose FROM clause names a view that used GROUP BY or HAVING.

SELECT Clause

SELECT

Produces a final result table by selecting only the columns indicated by the select list from R, where R is the result of the previous clause.

Note: We do not impose an arbitrary limit on the number of columns you can select in a query. We are limited only by environmental factors. Some of these factors are the size of your work area as specified by the *size* parameter of the TASKS Multi-User startup option (see the *CA Datacom/DB Database and System Administration Guide*), your column sizes, and limits placed by other products such as the CA Datacom Server.

ALL

Retains all rows of the final result table and does not eliminate redundant duplicates. ALL is the default.

DISTINCT

Eliminates all but one of each set of duplicate rows of the final result table.

DISTINCT must not be used more than once in a subselect. This restriction includes:

1. Functions in the SELECT list of the subselect.
2. Functions in a HAVING clause of a subselect.
3. Functions which are specified in a subquery of the HAVING clause and contain a correlated reference to groups of the subselect.

***

The asterisk (*) represents a list of names that identify the columns of R. The first name in the list identifies the first column of R, the second name identifies the second column of R, and so on. The list of names is established when the program containing the SELECT clause is prepared.

name*.

The asterisk (*) represents a list of names that identify the columns of R. The **name** can be a table-name, view-name, or correlation-name, and must designate a table-name or view-name in the FROM clause. The first name in the list identifies the first column of R, the second name identifies the second column, and so on. The list is established at preparation time and does not represent any columns that have been added later. The names must be separated by commas.

expression

Commonly, the expressions used in a SELECT statement include column names. Each column name used in the select list must unambiguously identify a column of R. Multiple expressions must be separated by commas.

The number of columns in the result of SELECT is the same as the number of expressions in the operational form of the select list, that is to say, the list established at preparation time.

The result of a subquery must be a single column unless the subquery is used in the EXISTS predicate.

Applying the Select List

Some of the results of applying the select list to R depend on whether GROUP BY or HAVING is or is not used.

If neither GROUP BY nor HAVING is used:

The select list must either include no functions, or be entirely a list of functions.

If the select includes no functions, then the select list is applied to each row of R, and the result contains as many rows as there are rows in R.

If the select list is a list of functions, then R is the source of the arguments of the functions, and the result of applying the select list is one row.

If GROUP BY or HAVING is used:

Each column-name in the select list must either identify a grouping column or be specified within a function.

The select list is applied to each group of R, and the result contains as many rows as there are groups in R. When the select list is applied to a group of R, that group is the source of the arguments of the functions in the select list.

In either case the nth column of the result contains the values specified by applying the nth expression in the operational form of the select list.

A result column derived from a column name acquires the unqualified name of that column. All other result columns have no names.

Each column of the result of SELECT acquires a data type from the expression from which it is derived.

Example 1

Show all rows of the employee table TEMPL.

```
SELECT * FROM CA.TEMPL
```

Example 2

Show the job code, maximum salary and minimum salary for each group of rows of the table TEMPL with the same job code, but only for groups with more than one row and with a maximum salary less than $50,000.

```
SELECT JOBCODE, MAX(SALARY), MIN(SALARY)
FROM CA.TEMPL
GROUP BY JOBCODE
HAVING COUNT(*) > 1 AND MAX(SALARY) < 50000
```

Example 3

Show all rows of the employee-to-project-activity table TEMPRAC for Department E11, as determined by the employee table TEMPL.

```
SELECT *
FROM CA.TEMPRAC
WHERE EMPNO IN (SELECT EMPNO
                FROM CA.TEMPL
                WHERE WORKDEPT = 'E11')
```

Example 4

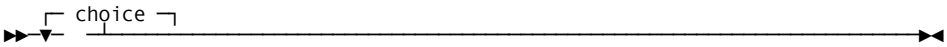
Show the department number and maximum departmental salary for all departments whose maximum salary is less than the average salary for all employees. The information is in the employee table TEMPL. In this example, the subselect would be executed only once.

```
SELECT WORKDEPT, MAX(SALARY)
FROM CA.TEMPL
GROUP BY WORKDEPT
HAVING MAX(SALARY) < (SELECT AVG(SALARY)
                      FROM CA.TEMPL)
```

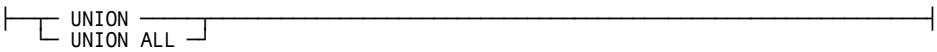
Full-Select Statement

A full-select statement specifies a result table. If UNION is not used, the result of the full-select is the result of the specified subselect.

Following is the syntax diagram for the full-select statement:



Expansion of Where choice is as follows



Description

UNION

Derives a result table by combining two other result tables. The set of rows in the UNION of result tables R1 and R2 is the set of rows in either R1 or R2, with redundant duplicate rows eliminated. Each row of the UNION table is either a row from R1 or a row from R2.

The columns of the result table are not named.

UNION ALL

As in UNION, derives a result table by combining two other result tables. The set of rows in the UNION ALL of result tables R1 and R2 is the set of rows in either R1 or R2, but UNION ALL specifies not to eliminate duplicate rows when deriving the result table. Each row of the UNION ALL table is either a row from R1 or a row from R2.

The columns of the result table are not named.

subselect

Specify a subselect. For more information about the subselect see Subselect (see page 647).

Important! If more than two subselects are used, you may not mix UNION with UNION ALL.

Duplicate Rows

Two rows are duplicates of one another only if each value in the first row is equal to the corresponding value of the second row.

All but one row of each set of duplicates are eliminated by a UNION. The number of rows in the UNION table is the sum of the number of rows in R1 and R2, less the number of duplicates eliminated.

If ALL is specified, duplicate rows are not eliminated.

Rules for Columns

Result tables R1 and R2 must have the same number of columns.

When UNION is performed, the corresponding columns of the SELECT lists need only be of compatible data types.

Note: In CA Datacom/DB Release 8.0 the corresponding columns were required to be exactly the same data type, that is, the same data type, precision, length and scale.

Example 1

Show all the rows from the EMP table.

```
SELECT *
FROM EMP
```

Example 2

List the employee numbers of all employees:

- Whose department number begins with D or,
- Who are assigned to projects whose project number begins with AD.

```
SELECT EMPNO
FROM EMP
WHERE WORKDEPT LIKE 'D%'
UNION
SELECT EMPNO
FROM ASSIGNTBL
WHERE PROJNO LIKE 'AD%'
```

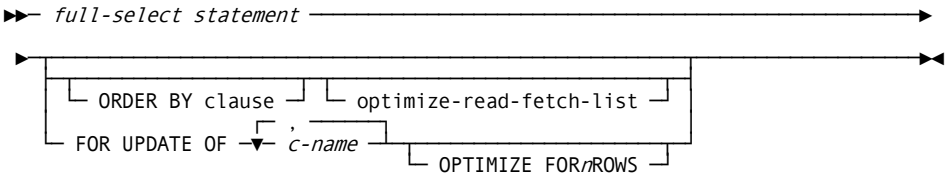
Select-Statement

The select-statement is the form of a query that you can issue in interactive SQL (using Datadictionary or CA Dataquery online) or in static SQL (embedded in a DECLARE CURSOR statement in a preprocessed batch program), but not in dynamic SQL. The result table returned by a select-statement is the result of the full-select statement.

Following is the syntax diagram for the select-statement:

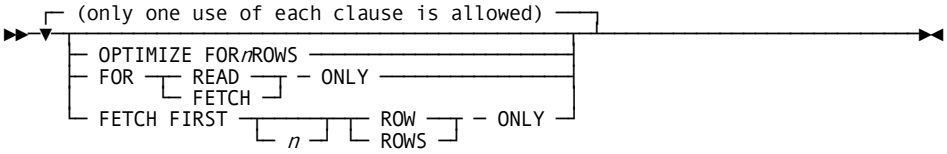
Note: For the syntax of the full-select statement, see Full-Select Statement (see page 654). the *CA Datacom/DB SQL User Guide*.

FOR UPDATE OF is a CA Datacom/DB extension.



Note: OPTIMIZE FOR n ROWS is a CA Datacom/DB extension.

FOR READ/FETCH ONLY is a CA Datacom/DB extension. FETCH FIRST is a CA Datacom/DB extension.



Description

ORDER BY *clause*

You can specify the order for rows in the result table. For information on using the ORDER BY clause, see ORDER BY Clause (see page 658).

FOR UPDATE OF *c-name*

This CA Datacom/DB extension is provided for compatibility with other SQL implementations. The functionality of update indication is actually accomplished by other means in CA Datacom/DB. The column(s) must belong to the table or view named in the FROM clause of the SELECT statement. Column names (*c-name* in the diagram) must be separated by commas.

OPTIMIZE FOR *n* ROWS

This CA Datacom/DB extension tells the optimizer how many rows are normally fetched before the cursor is closed. This is useful when there is a choice between a key that would build a temporary table and give a lower cost if all rows are fetched, and a key that does **not** build a temporary table and would take longer if all rows are fetched, but takes less time when only a few rows are fetched.

Note: The number n is **not** a limit (as it is, for example, in FOR FIRST n). The complete set of rows can be fetched. The number n in this clause is merely input to the optimizer.

FOR READ/FETCH ONLY

This clause is provided for compatibility with other SQL implementations, but if you use it and then try to do an update or delete on the current cursor, you receive an SQL return code -130 CURSOR NOT UPDATABLE (SQLSTATE 24S05).

FETCH FIRST *n* ROW/ROWS ONLY

The FETCH FIRST clause allows you to control how many rows are retrieved by a query, giving you the ability to limit the number of fetches that are done. FETCH FIRST can therefore be used to prevent runaway queries.

Note: Queries that require the resulting set to be sorted have to perform the sort on the entire resulting set. For example, if a non-indexed column is being ordered by the use of an ORDER BY clause, to return the correct *n* rows the entire resulting set would need to be generated and sorted. In this case, the savings would be minimal even though CA Datacom/DB would prevent an excessive number of FETCHes from occurring.

The *n* can be optionally used to specify the number of rows to be fetched. When a specified number of rows have been fetched, a SQL return code number +100 NO ROW FOUND (SQLSTATE 02000) is received. If *n* is not specified, the resulting set is limited to one row.

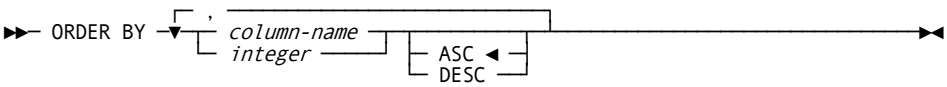
Either ROW or ROWS must be specified.

ONLY must be specified. ONLY is provided for compatibility with other SQL implementations.

ORDER BY Clause

The ORDER BY clause is a component of a select-statement.

Following is the syntax diagram for the ORDER BY clause:



Description

ORDER BY

Puts the rows of the result table in order by the values of the columns you identify. If you identify more than one column, the rows are ordered first by the values of the column you identify first, then by the values of the column you identify second, and so on. If you do not specify the ORDER BY clause, the rows of the result table have an arbitrary order.

colume-name

Each column-name specified must unambiguously identify a column of the result table. The sum of the lengths of the columns must be less than 32720. The column names must be separated by commas.

integer

The integer n identifies the nth column of the result table. Each integer specified must be greater than 0 and not greater than the number of columns in the result table. The integers must be separated by commas.

A named column can be identified by an integer or a column-name.

An unnamed column must be identified by an integer. A column is unnamed if it is derived from:

- A literal
- An arithmetic expression
- A function

If the full-select statement includes a UNION operator, every column of the result table is unnamed.

ASC

Places the values of the column in ascending order. ASC is the default order.

DESC

Places the values of the column in descending order.

Ordering is performed in accordance with the comparison rules. The null value is higher than all other values. If your ordering specification does not determine a complete ordering, rows with duplicate values of the last identified column have an arbitrary order.

Processing

A cursor defined with an ORDER BY clause cannot be used for update.

Examples

In this example, the ORDER BY clause is used to order the results of the SELECT. The SELECT retrieves the number and name of all employees hired before 1980 in order of seniority. HIREDATE is in the form yymmdd.

```
SELECT EMPNO, LASTNAME, FIRSTNME, HIREDATE
FROM EMP
WHERE HIREDATE < 800000
ORDER BY HIREDATE
```

Example

This example shows an SQL query, using an ORDER BY clause, that runs against system tables (SYS authorization ID) to display keys, where TBL and FLD are correlation names.

```
SELECT DISTINCT TBL.TABLE_SQLNAME,
   FLD.DBID,
   FLD.KEY_NAME,
   FLD.OCCURRENCE,
   FLD.KEY_FIELD_SEQ,
   FLD.FIELD_LENGTH,
   FLD.FIELD_UNIQUE
FROM SYS.DIR_TABLE TBL,
           SYS.DIR_KEY_FIELD FLD
WHERE (TBL.TABLE_NAME = FLD.TABLE_NAME)
AND
      (TBL.TABLE_SQLNAME = 'PSN_AST' AND TBL.STATUS = 'P')
ORDER BY FLD.DBID, FLD.KEY_NAME, FLD.KEY_FIELD_SEQ;
```

UPDATE

This SQL statement can be executed in the following ways:	Through the Datadictionary Interactive SQL Service Facility *(interactive)*	In an application program prepared using a CA Datacom/DB SQL Preprocessor *(embedded)*	By using CA Dataquery *(SQL & Batch Modes)*
UPDATE (positioned)		YES	
UPDATE (searched)	YES	YES	YES

Note: YES indicates a valid execution method for this statement. To learn about using SQL keywords in CA Dataquery, see the *CA Dataquery User Guide*. For information about the access rights required to execute this statement, see the *CA Datacom/DB Database and System Administration Guide*.

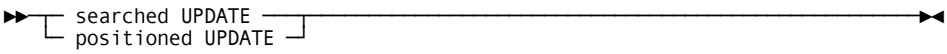
The UPDATE statement updates the values of specified columns in rows of a table or view. Updating a row of a view updates a row of the base table from which the view is derived.

You must specify ISOLEVEL=C (isolation level C) in the Preprocessor options when using the UPDATE statement (see the *CA Datacom/DB SQL User Guide*).

Note: When the NOMAINT option of the CA Datacom/DB Utility (DBUTLTY) ACCESS function is in force, an UPDATE statement receives a CA Datacom/DB return code 94(87), where 87 is a decimal internal return code (hex 57) that tells you no maintenance statements are allowed while NOMAINT is in force.

Also note that constraints that existed in versions before r10 can act differently in Release 10.0 and above because, in Release 10.0 and above, columns containing a NULL rather than a value do *not* cause a CHECK constraint to be violated. Constraints are considered, in Release 10.0, to be satisfied unless the predicates evaluate explicitly to FALSE. That is, CHECK constraints whose predicates evaluate to UNKNOWN rather than TRUE or FALSE are considered, starting with Release 10.0, to have been satisfied. Therefore, INSERT and UPDATE statements, from versions before r10, that resulted in constrained columns being nulled are, when used in Release 10.0 and above, successful for the first time.

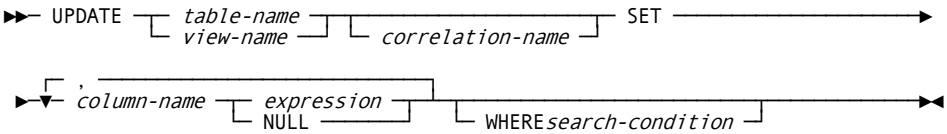
Following is the syntax diagram for an UPDATE statement:



Note: The positioned UPDATE is not used by CA Dataquery.

Searched UPDATE

Following is the syntax diagram for the searched UPDATE statement:



Description

table-name* or *view-name

Specify the name of the table or view you want to update. You must name a table or view described in the Datadictionary, but not a Datadictionary table, a view of a Datadictionary table, or a read-only view.

correlation-name

You can specify a 1- to 18-byte *correlation-name* (correlation-name) to be used within the search-condition to designate the table or view.

SET

Introduces a list of column names and values.

column-name

Specify the names of the columns you want to update. The names must be separated by commas. You must name a column of the table or view you specified. Do not specify the name of the same column more than once and do not specify a column of any of the following types:

- A view column that is derived from a literal.
- A view column that is derived from an arithmetic expression.

expression

Tells the new value of the column. The expression cannot include a function. See the *CA Datacom/DB SQL User Guide* for the expression's syntax diagram.

A column-name in an expression must name a column of the named table or view. For each row that is updated, the value of the column in the expression is the value of the column in the row *before* the row is updated. The following table shows a SET clause, the value of the columns in the row (before) and the resulting value (after) for each column:

SET clause	Value of X Before	Value of Y Before	Value of X After	Value of Y After
SET X = 10, Y = X + 1	1 3	1 5	1 10	1 4

NULL

Use the NULL keyword to specify that the value for the column is to be set to a null value.

WHERE

Introduces a condition that tells what rows are updated. You can omit the clause, give a search condition, or name a cursor. If you omit the clause, all rows of the table or view are updated.

search-condition

The search-condition is applied to each row of the table or view and the updated rows are those for which the result of the search-condition is true. the *CA Datacom/DB SQL User Guide* for the search-condition syntax diagram.

Each column-name in the search condition must name a column of the table or view, and the table or view must not be referenced in the FROM clause of any subselect in the search condition.

If the search condition contains a subquery, the subquery can be thought of as being executed each time the search condition is applied to a row, and the results used in applying the search condition. In actuality, the subquery is executed for each row only if it contains a correlated reference to a column of the table or view.

Processing

Update values are assigned to columns in accordance with the following assignment rules:

If update value is:	The column must be:
A number	A numeric column with the capacity to represent the integral part of the number

If update value is:	The column must be:
A character string	A character string column with a length attribute that is not less than the length of the string

The updated row must conform to any constraint imposed on the table (or on the base table of the view) by any unique index on an update column. If an update value violates any of those constraints, or if any other error occurs during the execution of the UPDATE statement, no rows are updated. If an error occurs during the execution of UPDATE that makes the position of a cursor unpredictable, the cursor is closed. Rows can be changed so that they no longer conform to the definition of the view. Such rows are updated in the base table of the view and no longer appear in the view.

Unless appropriate locks already exist, one or more exclusive locks are acquired by the execution of a successful UPDATE statement. Until the locks are released, an updated row can be accessed only by the unit of recovery that performed the update.

Example 1

For employee 000190, change the employee's telephone number in the TEMPL table.

```
UPDATE CA.TEMPL
SET PHONENO = '3565'
WHERE EMPNO = '000190'
```

Example 2

Increase the job code by 10 members in Department D11.

```
UPDATE CA.TEMPL
SET JOBCODE = JOBCODE + 10
WHERE WORKDEPT = 'D11'
```

WHILE Statement

For details about this statement, see WHILE Statement (see page 583).

Note: This statement is executed from within the Compound Statement (see Compound Statement (see page 558)).

Appendix A: Limitations

This appendix lists CA Dataquery limitations. Also provided is information about some variable limitations that are dependent on system size, parameters set up at installation, or the number of other variables you use.

Personal Tables

Review the limitations in this section when using the CA Dataquery Personal Database Facility.

- Only relational tables can be used. No repeating fields or groups, or redefines can be created or used.
- The maximum number of columns in a Personal Database table is 99.
- No compound fields can be used.
- No null valued columns can be created or maintained by PDB.
- Maximum numeric precision is 15 digits.
- Maximum column length is 254 bytes.
- SQL DATE, TIME, and TIMESTAMP data types cannot be created or maintained by the Personal Database Facility.

SQL Mode Limitations

Review the limitations in this section when using CA Dataquery in SQL Mode.

- The sum of the lengths of all the columns in an SQL query cannot exceed the DQESIZE parameter in the System Option Table.
- An SQL query report cannot be formatted until the SQL query is validated.
- SQL language verbs cannot be translated.
- The CANCEL FIND function cannot be used to cancel an SQL query.
- A joined view cannot be updated.
- The number of tables/views in a single query cannot exceed 10.
- Terms cannot be used.
- The length of an authorization ID cannot exceed 18 characters.

- An internal table may be exceeded if excessive amounts of the following (or combination of the following) have been specified in a query:
  - Number of columns selected in a query
  - Number of alternate headings in a report format
  - Number of control breaks in a report format
  - Number of column functions in a report format
  - Number of long literals specified for column values in an insert query
- Only one SQL command is allowed per query.

DQL Mode Limitations

Review the limitations in this section when using CA Dataquery in DQL Mode. This section describes the items that are internally monitored by CA Dataquery in DQL Mode. The following subsections are organized by DQL keywords, such as syntax, DRAW command, Guided Query, and dialogs.

Internal CA Dataquery Table

An internal table is used to keep track of the following:

- Columns and SET statements in an INSERT query.
- Columns and keys referenced in a query.
- Titles.
- Alternate headings (whether from DD or specified in the query).
- All the simple columns that make up a key.
- All column entries. Only columns with class S (simple), C (compound), or F (filler) can be accessed in DQL Mode.
- Repeating field subscripts. (A maximum of two levels is allowed.)
- Totals and column functions.
- Restricted conditions.
- Edit patterns (whether from DD or specified in the query).

An excess of one or more of these items can make this internal table overflow.

Syntax Limitations

In some cases, you must limit the numbers of certain items in some DQL Mode statements and clauses. This section lists the most important limitations to observe when using FIND, SET, SORT, WITH, UDATE, INSERT, ERASE, WHEN/DO, sets, and when nesting parentheses.

FIND Limitations

The following are important limitations to observe when constructing a FIND or COUNT statement.

- A maximum of 17 tables can be related or used in a FIND or COUNT statement. The number of tables is based on the following considerations:
  1. The total number of tables is developed after *Join Optimization* processes the statement.
  2. Repeating relationships or complex WITH statements can cause an increase in the total number of tables used by the query.
- Three parameters in the System Option Table limit the execution of a FIND or COUNT statement:
  - FNDBLKS: Maximum number of logical blocks allocated in the work table for this query.
  - MFTIME: Maximum number of seconds for the FIND or COUNT verb to execute.
  - SRCHLIM: Maximum number of DATACOM/DB physical I/Os allowed during the execution of a FIND or COUNT statement.

SET Limitations

Following are important limitations to observe when constructing a SET statement.

- Only simple fields may be referenced in a SET statement.
- To determine how many SETs can be in a query, use the following formula:

  (18 x Number of Tables Joined) + (Length of SORT key)

  + (17 x Number of SET Statements) + 6

  = (A Number That is Less Than 760)
- There is an absolute maximum of 43 SETs for a query with a FIND on one table and no SORT statement.
- The Found Table record size (768) limits the number of combined SET statements and sort keys that can be used in a query.
- SQL DATE, TIME and TIMESTAMP cannot be used with DQL SET, SORT and WITH.

- A numeric column whose value can be zero should not be used as a divisor in a SET statement or an arithmetic expression.
- The maximum numeric precision is 18 digits.
- An arithmetic expression cannot have more than five levels of nested parentheses.
- A SET statement cannot have more than 10 levels of nested parentheses.

SORT Limitations

Following are important considerations to observe when constructing SORT statements.

- Ten control breaks in a SORT statement is the maximum allowed.
- The total length of columns/keys used in a SORT statement cannot exceed 100 characters. (An extra character is added for each signed numeric column.)
- SQL DATE, TIME and TIMESTAMP cannot be used with DQL SET, SORT and WITH.

WITH Limitations

Following are important limitations to observe in constructing WITH clauses. Also see DQL Mode Nesting Limitations (see page 669).

- Two columns or keys that are not the same type cannot be compared.
- Columns that are greater than 240 characters in length cannot be used in a WITH clause or in a SET statement in an INSERT or UPDATE query.
- The maximum number of literal values per WITH statement is 15.
- Columns used in a WITH clause as part of an arithmetic expression cannot be qualified.
- The maximum number of OR entries in a WITH statement is 15. (That is, a maximum of 14 ORs are allowed.)
- Values following an EQ (=) or NE (,=) cannot be separated by AND or OR.
- SQL DATE, TIME and TIMESTAMP cannot be used with DQL SET, SORT and WITH.
- A numeric column whose value can be zero should not be used as a divisor in a SET statement or an arithmetic expression.
- The maximum numeric precision is 18 digits.
- An arithmetic expression cannot have more than five levels of nested parentheses.

UPDATE/INSERT/ERASE Limitations

Following are important limitations to observe when constructing an UPDATE, INSERT, or ERASE query.

- An UPDATE or ERASE query cannot contain a RELATED BY clause.
- Columns that are greater than 240 characters in length cannot be used in a WITH clause or in a SET statement in an INSERT or UPDATE query.

WHEN/DO Limitations

The following is an important limitation to observe when constructing a WHEN/DO statement.

- Legends in DO statements cannot exceed 64 characters.

Exported, Saved, Kept, and Extracted Sets

Following are important limitations to observe when using sets in DQL Mode.

- The total length of all columns used in the PRINT statement for an Export cannot exceed 4088 characters.
- The maximum number of Saved Sets for a single user is determined by the MXSETS parameter in the System Option Table.
- An Extracted Saved Set cannot be used by CA Dataquery.
- The maximum size of an extracted file cannot exceed the XTRBLKS parameter in the System Option Table.

DQL Mode Nesting Limitations

- Terms cannot be nested more than four levels deep.
- An arithmetic expression cannot have more than five levels of nested parentheses.
- A SET statement cannot have more than ten levels of nested parentheses.

DRAW Command

Following are limitations on the use of the DRAW command in either mode.

- Simple columns which repeat are included. Compound fields which repeat are shown with *all.*
- Only simple columns will be included in the query expansion, unless the word ALL follows the table-name in the DRAW command.

Guided Query Limitations

Following are limitations on the use of GUIDE to construct queries in either mode:

- Up to two tables can be queried.
- A dialog query cannot be built.
- Absolute GUIDE limitations are as follows:
  - 15 conditions per table
  - 10 sort keys
  - 10 temporary results
  - 10 columns in a calculation to compute a temporary result
  - 95 columns to be printed or displayed
  - 95 column functions and/or legends (95 for both)
- An excessive use of a combination of these can also exceed GUIDE limitations.

Dialogs

Following are limitations on construction of dialog queries in either mode:

- A list of values can be used only with a character variable.
- A range of values can be used only with a numeric variable.
- 25 dialog variables is the maximum allowed in a single dialog.
- A dialog variable can be no longer than 77 characters.
- A variable ID used more than once must have the same default value each time it is used in the dialog text.

Maximums for Both Modes

Following are miscellaneous maximums of interest to users and CA Dataquery Administrators in either mode.

- The maximum numeric precision is 18 digits.
- The maximum number of pages a user may print on a system printer is determined by the PRTCTL parameter in the System Option Table.
- The logical page size (all page segments without the title and PF Key lines) cannot exceed 64K.

- The maximum lines of query and JCL text cannot exceed 24 times the QPAGES parameter in the System Option Table. The minimum number of lines is 24; the maximum is 96. If the GUIDE query is to be used, QPAGES should be set to 2 (48 lines) as a minimum.
- The maximum row size of a table that can be accessed by CA Dataquery is determined by the BUFSIZE parameter in the System Option Table.
- The maximum length for a data name is 32 characters.
- A maximum of 17 tables can be related or used in a FIND or COUNT statement.

Miscellaneous

- Public members can be updated only by a CA Dataquery Administrator if the library is partitioned (QRYGRPS=YES in the System Option Table).
- A system or network printer can be used only for columnar output.

Batch CA Dataquery

The number of partitions requested for the batch work table (DQW) cannot exceed 47.

Appendix B: Reserved and Ignored Words

Using words that have special meaning to CA Dataquery or words that are ignored by CA Dataquery in places where CA Dataquery does not expect them causes problems during execution of any query or dialog containing them.

Do Not Use:

When assigning names to such things as tables, sets, queries, columns, keys, temporary results, terms, and so on, do not use words from these categories:

- CA Dataquery system commands
- CA Dataquery Editor commands
- DQL and SQL keywords
- DQL and SQL ignored words
- Translations of reserved words

Translations:

Translations of CA Dataquery with the CA Dataquery Language Facility are also affected by reserved words. Any translation of a reserved word is also a reserved word.

DQL Ignored Words:

The following table shows the words ignored by CA Dataquery:

ALL AND ASCENDING	BREAK BREAKS BY	COLUMN COLUMNS	FIELD FIELDS FILE FILES FROM
IN INTO	KEY KEYS	OF ON	RECORD RECORDS ROW ROWS
TABLE TABLES THAN THE THEM THEN TO	UP	VIA	

CA Dataquery Reserved Words

Do not use the following words when working in DQL Mode. We recommend avoiding DQL and SQL words even if you normally work in only one mode.

+nnnn -nnnn	ADMIN ASC AUTHID AVG	BOTTOM	CANCEL CHANGE CNT CONDITIONS CONTAINING COPY COUNT CREATE
DESC DESCENDING DETAIL DIAGNOSTICS DIRECTORY DISPLAY DO DOWN DQL DRAW	EDIT ENCLOSE EQ EQUAL ERASE EXECUTE EXTRACT	FIND FIRST	GT GTE GREATER
HELP	INSERT	JCL JOIN JOIN-AS-IS JOINED	KEEP
LANGUAGE LESS LIBRARY LIST LOCATE LTE	MAX MENU MIN MOVE MSG	NOT NO-TOTALS NOWRAP NULL	OFF OR ORDER

PDB	RELATED	SCROLL	THROUGH
PFn	REPEAT	SECURITY	TIME
PIC	RESTRICTIONS	SEND	TITLE
PICTURE		SET	TITLE1
PRINT		SETS	TITLE2
PRINTER		SHIFT	TOP
PROFILE		SORT	TOTALS
PROJECT		SPLIT	
		SQL	
		STATS	
		STALL	
		STORE	
		SUBMIT	
		SUM	
UPDATE	WHEN-ONLY		
USER	WHEN		
	WHERE		
	WITH		
	WRAP		

SQL Reserved Words

If your site uses SQL at all, we recommend that you do not use SQL reserved words for tables, columns, and so forth. We recommend avoiding DQL and SQL words even if you normally work in only one mode. The following table shows the words ignored by CA Dataquery in SQL Mode:

In the following table, where more than one word is listed on a line, those words as a group are reserved, not necessarily the individual words that make up the group unless that word is listed separately on its own line. Words followed by an asterisk (*) indicate that word is reserved only in the COBOL language. We reserve the right to add or change reserved keywords as needed.

SQL Reserved Words

ADD	BEFORE	CALL	DATA
AFTER	BEGIN	CASE	DATACOM
ALL	BETWEEN	CAST	DATACOM DUMP
ALTER	BIT	COALESCE	DATACOM
AND	BIT_ADD	COLUMN	LOOPLIMIT
ANY	BIT_AND	COBOL	DATACOM TSN
ARRAY	BIT_NOT	CONCAT	DELETE
AS	BIT_OR	CONDITION	DESCRIPTOR
ASSEMBLER	BIT_XOR	CONTINUE HANDLER	DETERMINISTIC
ASENSITIVE	BY	CONTAINS	DISTINCT
ATOMIC	BYREF	CONVERSION	DO
		COUNT	DROP
		CURRENT	
		CURSOR	
EACH	FIRST	GENERAL	HANDLER
ELSE	FOR	GET CURRENT DIAGNOSTICS	HAVING
ELSEIF	FROM	GET DIAGNOSTICS	
END		GET STACKED	
END-EXEC*		GET STACKED DIAGNOSTICS	
EXECUTE		GRANT	
EXISTS		GROUP	
EXIT HANDLER			
EXTERNAL			

IF	JOIN	KEY	LANGUAGE
IMMEDIATE			LEADING
IN			LEAVE
INDEX			LEFT
INNER			LIKE
INOUT			LOOP
INPUT			LOWER
INSENSITIVE			LOWERCASE
INSERT			LTRIM
ITERATE			
INTEXTRACT			
INTO			
INVALIDATE			
IS			
MODIFIES	NEW	OF	PARAMETER
MUF_NAME	NEWFUN1	OLD	PLI
	NEWFUN2	ON	PRIVILEGES
	NEWFUN3	OPTIMIZE	PROCEDURE
	NO	OPTION	PROGRAM
	NOT	OPTIONS	
	NOT FOUND	OR	
	NULL	ORDER	
	NULLIF	OUT	
	NULLS	OUTER	
RAISE ERROR	SELECT	TABLE	UNION
READS	SENSITIVE	THEN	UNDO HANDLER
REFERENCING	SET	TO	UNTIL
REPEAT	SIGNAL	TO PXXSQL	UPDATE
RESIGNAL	SOME	TO SYSOUT	UPPER
RETURN	SPECIFIC	TRAILING	UPPERCASE
RTRIM	SQL	TRIGGER	USER
RULE	SQLEXCEPTION	TRIM	USING
RUN	SQLSTATE	TSN	
	SQLWARNING		
	SQUEEZE		
	STATEMENT		
	STRIP		
	STYLE		
	SUBSTRING		
	SYNONYM		
VALUES	WHEN	XMLATTRIBUTES	
VARCHAR	WHERE	XMLCONCAT	
VIEW	WHILE	XMLELEMENT	
	WITH	XMLFOREST	
	WITHOUT	XMLSERIALIZE	

Appendix C: Data Types

This chapter provides information about which data types are used by CA Dataquery DQL Mode and the Personal Database Facility.

Note: For complete details about CA Datacom/DB data types, including SQL data types, see the *CA Datacom/DB SQL User Guide* chapter on basic language elements.

DQL Mode Data Types

Following is a list of CA Dataquery statements and the data types which they can use.

WITH,

CHAR, NUM, DEC, BIN, 1/2 BIN, FUL BIN, DBL CHAR, KANJI

RELATED

CHAR, NUM, DEC, BIN, 1/2 BIN, FUL BIN, DBL CHAR, KANJI

SET

NUM DEC, BIN, 1/2 BIN, FUL BIN, DBL CHAR, KANJI

SORT

NUM DEC, BIN, 1/2 BIN, FUL BIN, DBL CHAR, DBL MIXED, KANJI

PRINT/DISPLAY

CHAR NUM, DEC, BIN, FLOAT-PT SHRT, FLOAT-PT LONG, EXT, 1/2 BIN, FUL BIN, DBL CHAR, DBL MIXED, KANJI

DO

CHAR NUM, DEC, BIN, FLOAT-PT SHRT, FLOAT-PT LONG, EXT, 1/2 BIN, FUL BIN, DBL CHAR, DBL MIXED, KANJI

Note: For details about these data types, see the *CA Datacom/DB SQL User Guide*.

DATE, TIME, or TIMESTAMP

DQL Mode queries cannot include columns defined as data types DATE, TIME, or TIMESTAMP. These data types are reserved for SQL Mode queries. These data types are listed as character types on Extended Column Display panels because SQL Mode queries return character-type data.

To test a column that is suspected of being one of these types before selecting it for a DQL Mode query, follow these steps:

1. Look at the length of the column. SQL-DATE length is always 10, SQL-TIME is always 8, and SQL-TIMESTAMP is always 26.
2. If the data type is suspect, create a query that finds one row and PRINT the output.

Use the PF key for Extended Column Definitions to check the data type of the column you wish to use, and correct your query.

Personal Database Data Types

The Personal Database data types are:

CHARACTER

Character string

NUMERIC

Zoned decimal

DECIMAL

Packed decimal

SMALLINT

Small integer

INTEGER

Large integer

FLOAT

Floating-point