CA-IDMS®

Features Summary
Release 14.1

Computer Associates

Contents

Index

Chapter 1 Introduction

Overview

This chapter provides an overview of the CA-IDMS Release 14.1 features, as well as a brief overview of upgrading from Release 14.0 to Release 14.1.

CA-IDMS Release 14.1 Enhancements

CA-IDMS Release 14.1 gives you a wide variety of performance enhancements and utilities that are designed to increase performance and productivity.

These new utilities and enhancements are discussed below:

- Mixed page group support
- Tune Index Utility
- Convert Page Utility
- Callable Restructure Utility
- Enhanced COBOL II performance under LE/370
- Dynamic Front-end System Table
- Exit 24
- Exit 34
- CA-IDMS/DML Online productivity enhancements

Mixed Page Group Support

The new mixed page group support feature gives you the ability to access data from different page groups within a single CA-IDMS rununit.

This feature also permits a single rununit to access a database whose segments have been defined with a different maximum for the number of records that can be stored on a page. You can specify a different maximum for each page group.

This enhancement also allows you to access larger databases than were previously supported. Because you can define up to 32,767 different page groups and a single page group can qualify up to 16 million pages, this feature significantly increases the size of a database accessible from a single rununit. Furthermore, by allowing portions of a database to have different maximum records per page, you can use the available pages more effectively.

Tune Index Utility

The Tune Index Utility walks an index in order to cause the adoption of orphaned indexed records. An *orphaned indexed record* is a record whose index pointer does not point back to the index record (SR8) which contains the record's index entry. Orphans occur as the result of splitting when an existing SR8 is split into two records to accommodate a new entry. As part of the split, some of the entries are moved to a new SR8, but the index pointer in their associated record is not adjusted to reflect the change, thus resulting in "orphaned" records.

By eliminating orphans, runtime database performance is improved when traversing from an indexed record to its associated index entry.

Convert Page Utility

The Convert Page Utility changes the page range for an area or the maximum number of records that can be stored on a page of an area.

You may process an entire segment or one or more individual areas. The utility first identifies the changes that need to be made by comparing old and new definitions for the segment or areas. Once all changes have been identified, it converts one or more files. It is possible to run the utility in parallel to process several files concurrently.

This utility allows you to reassign the page range of an area so you can consolidate free pages. It also allows you to change the maximum number of records on a page, thus permitting more effective use of the space on a page and more flexibility in choosing page sizes.

Callable Restructure Utility

The Callable Restructure Utility (IDMSCRSU) is a version of the CA-IDMS Restructure Segment command that is callable from a user-written program. It uses the standard IDMSRSTT table to control the restructure of the data portion of a database record.

This utility allows you to call Restructure for a particular record image, as compared to the Restructure Segment command, which restructures every record in the segment.

Enhanced COBOL II Performance Under LE/370

Execution of VS COBOL II programs under CA-IDMS/DC using the LE/370 runtime library now results in lower CPU utilization. In most cases, performance will be comparable to using the COBOL II runtime library.

In addition, COBOL II programs running with the LE/370 runtime library may now use storage protection.

Dynamic Front-end System Table

The Dynamic Front-end System Table feature provides an enhancement to 24-hour processing for CA-IDMS/UCF sites.

During startup of the CA-IDMS central version, a front-end system table describing each CA-IDMS UCF front-end is loaded. When you alter this table, the central version has to be stopped and re-started to pick up the changed table.

This feature allows online refresh of the table. New front-end systems may be added or changes to existing front-end systems may be made dynamically. There is no longer a need to stop and re-start the central version, resulting in better 24-hour support.

Exit 24 for Year 2000 Testing

A new numbered exit is provided that is useful for Year 2000 testing of applications by providing a means to return a different value for GET TIME requests than the current value returned by the operating system.

Exit 34 for Unqualified Dbkey Find/Obtain Exit

A new numbered exit is provided for use with the Mixed Page Group Support feature. You can use this exit to help identify and correct applications that may require modification to function correctly when the Mixed Page Group Support feature is enabled.

CA-IDMS/DML Online Productivity Enhancements

The CA-IDMS/DML Online product has been enhanced to provide for improved productivity by reducing the number of required keystrokes.

Upgrading to CA-IDMS Release 14.1

You may upgrade to CA-IDMS Release 14.1 from CA-IDMS Release 10.0, 10.1, 10.2, 10.21, 12.0, or 14.0:

- **Upgrading from Release 10.x:** To upgrade from CA-IDMS Release 10.x you must perform the same conversion steps as if you were upgrading to CA-IDMS Release 12.0 or 14.0. Refer to the *CA-IDMS Release 14.0 Conversion Notebook* for detailed information about this conversion.

  Note: If you are upgrading from CA-IDMS Release 10.0 or 10.1 you cannot use the DMCL Conversion Utility due to internal changes to the DMCL fields in the data dictionary in CA-IDMS Release 10.2. Instead, you must convert your DMCL syntax manually.

- **Upgrading from Release 12.0:** To upgrade from CA-IDMS Release 12.0 you must perform the same conversion steps as if you were upgrading to CA-IDMS Release 14.0. Refer to the *CA-IDMS Release 14.0 Features Guide* for detailed information about this conversion.

  Note: The conversion utilities provided for CA-IDMS Releases 12.0 and 14.0 are included on the CA-IDMS Release 14.1 installation tape for your convenience.

You must perform the following steps to upgrade to CA-IDMS Release 14.1 from CA-IDMS Release 14.0:

1. Install the software and use the new DLODPROT to update the Subschema Control record descriptions in all dictionaries. DLODPROT has been changed to include the new PAGE-INFO field in the subschema control block. It overlays the first 4 bytes of the SSCDBCOM array.
2. Install a new SVC, which is downward compatible with existing versions of the software.

3. Check PIB LI37449 for the optional apars available for use with CA-IDMS Release 14.0 and above.
4. Review the cover letter delivered with the product tape for more information about this upgrade.

Note: You may verify the release that is running by checking the contents of the CSATAPE field. The CSATAPE field is a 6-character string that contains the release number and genlevel. For the initial CA-IDMS Release 14.1 tape, CSATAPE will contain the literal 'E19811' for genlevel 9811. You may use the OPER and DCPROFIL system tasks to display the tape value. Use DCPROFIL to display the current Tools tape value.

Chapter 2 Mixed Page Group Support

Overview

Mixed page group support gives you the ability to access data from different page groups within a single CA-IDMS rununit.

A *page group* is an attribute assigned to a segment of the database when it is defined. The page group serves to qualify the page numbers in the segment, making them unique from similar page numbers in other segments. In previous releases of CA-IDMS, a rununit could access data from only one page group, although it could access any number of segments (provided they were all assigned to the same page group). With this release, mixed page group support allows a rununit to access any segment regardless of its assigned page group.

This feature also permits a single rununit to access a database whose segments have been defined with a different maximum for the number records that can be stored on a page. You can specify a different maximum for each page group.

Benefits

This enhancement allows you to access larger databases than were previously supported. Because you can define up to 32,767 different page groups and a single page group can qualify up to 16 million pages, this feature significantly increases the size of a database accessible from a single rununit. Furthermore, by allowing portions of a database to have different maximum records per page, you can use the available pages more effectively.

Considerations

In order to use the mixed page group support feature, you must decide what portions of the database should be assigned to different page groups. You must define each portion in its own segment or segments, because page group is assigned at the segment level. If there are sets that cross page group boundaries, you must restructure the database to remove the sets. The removal of sets impacts your existing applications.

Once you have defined the segments, you must add them to one or more DMCLs and then regenerate the DMCLs. You must also add them to one or more DBNAMEs and then regenerate DBTABLE. You must specify the DBNAMEs with the new MIXED PAGE GROUP BINDS ALLOWED option.

In some cases, a database might already be comprised of segments residing in different page groups, that, until now, required separate rununits to access. If this is the case, to exploit this feature, you must define a DBNAME (with the new MIXED PAGE GROUP option) that includes all of the segments and define one or more subschemas that include the segments' areas. Now you can develop programs that access all segments using a single rununit.

In other cases, you can use this feature to allow database expansion. If, for example, a database is approaching the 16-million page limit, you can split it into multiple segments with different page groups and continue accessing it through a single rununit. In this case, you have to split the database areas among two or more segments with different page groups. If sets cross the page group boundaries, you have to restructure the database to remove the sets, which, in turn, impacts existing applications. If no restructure is required, existing application programs should require little, if any, modification.

An existing program that binds a rununit using a subschema that includes areas in different page groups may require changes. This is true only if the program retrieves a record by dbkey without specifying the name of the record to be accessed and the current page group for the rununit is not the page group of the record to be retrieved. The current page group for a rununit is the page group of the dbkey that is current of rununit.

Under these conditions, you should change the DML statement to either name the record to be retrieved or specify the page group with which the dbkey is associated. New DML commands and options allow page group information to be retrieved whenever a dbkey is retrieved.

Notes:

- Even if a database contains segments in different page groups, programs whose subschemas include areas from only a single page group will not require changes.

- A new numbered exit, Exit 34, is provided for use with the Mixed Page Group Support feature. You can use this exit to help identify and correct applications that may require modification to function correctly when the Mixed Page Group support feature is enabled. (See Chapter 4, “Other Enhancements,” for more details.)

Usage

The Mixed Page Group Support feature has the following usage rules:

- Dictionaries may not span page groups. This means that the DDLDML, DDLDCLOD, and DDLDCMSG areas must all be in the same page group. The DDLCAT, DDLCATX, and DDLCATLOD areas also must be in the same page group. These two sets of areas may be in different page groups. Dictionaries that share load areas must be in the same page group.
- Sets, indexes, and referential constraints (both linked and unlinked) may not cross page group boundaries.
- Owner and member records for a set must be in the same page group and have the same number of records per page. CA-IDMS Release 14.1 performs a runtime check on update operations to ensure that this rule is observed. If a violation is found the update is not performed and the xx87 minor code error status is returned. If this violation is found for an SQL command the SQLCODE will be set to –4 and SQLCERC to 1031. The message associated with this code will contain the same xx87 code and other related information.
- A record may reside in only one segment. While you may continue to horizontally segment a database; for example, by placing customer information in three segments (CUSTEAST, CUSTWEST, CUSTCENT), you may access only one of these segments at a time from within a rununit. In order to access all customer information, you must bind three rununits (either concurrently or serially). The mixed page group feature does not lift this restriction.
- Multiple page group support is not provided for the following:
  - CA-IDMS Transparency products
  - ASF/IDB
  - WESTI, SHADOW, TASKMASTER, and Intercomm applications
  - Visual Realia Cobol applications
  - Programs written in FORTRAN or RPG
- The following utilities cannot process mixed page groups and will issue an error message if mixed page groups are encountered. You must use multiple invocations of the utility to process different page groups.

  UNLOAD, RELOAD, FASTLOAD, MAINTAIN INDEX, MAINTAIN ASF

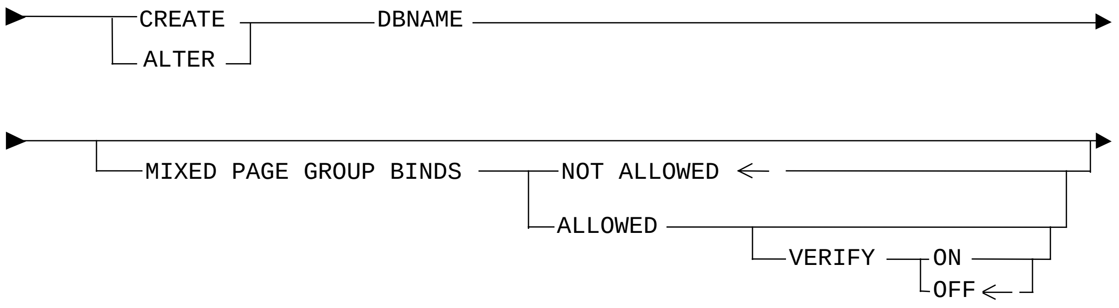
By implication, all other utilities are able to support mixed page groups.

- The following CA-IDMS Tools cannot process mixed page groups and will issue an error if mixed page groups are encountered: CA-IDMS DB/Reorg, CA-IDMS DB/Audit, CA-IDMS DB/Analyzer, and CA-IDMS Database Extractor.

Syntax

For non-SQL defined databases, you must define a database containing segments with different page groups as a DBNAME in the DBNAME table. In the previous release, any attempt to bind a rununit using a subschema whose areas cross page group boundaries failed. A new option on the CREATE/ALTER DBNAME statement now permits such binds to be honored.

DBNAME Syntax



MIXED PAGE GROUP BINDS ALLOWED|NOT ALLOWED

Specifying MIXED PAGE GROUP BINDS ALLOWED on a DBNAME statement allows a rununit accessing the DBNAME to bind to areas with a mixture of page group and radix values. If not explicitly specified, a rununit binding to a DBNAME whose segments have different page groups will fail if the subschema being used includes areas with different page groups. The default is NOT ALLOWED.

Note: This option applies only to non-SQL-defined databases. Mixed page group access is always ALLOWED for SQL-defined databases. To ensure valid results, the SQL-defined database must **not** include constraints or indexes which cross page groups.

```
VERIFY ON|OFF
```

VERIFY ON|OFF specifies whether or not a check will be made at bind rununit time to ensure that no sets included in the subschema cross page group boundaries. If VERIFY OFF is specified, it is your responsibility to ensure that this condition is met. The default for VERIFY is OFF.

Notes:

- This option applies only to non-SQL-defined databases. The VERIFY option is always OFF for SQL-defined databases.
- A runtime check is always performed for update operations to SQL and non-SQL-defined databases to ensure that owner and member records for a set are in the same page group and have the same number of records per page. The VERIFY option setting does not control this runtime check. Refer to the Usage section, above, for more information about this runtime check.

New DML Syntax

When an application program issues a DML request identifying a record, area, or set to be accessed, CA-IDMS automatically determines the target page group based on the name of the entity. The only time you cannot do this is when an unqualified dbkey retrieval is issued. 'Unqualified' means that no record name is specified. In this case, CA-IDMS assumes that the target page group is the same as the one most recently accessed, unless overridden by the application program.

A new field has been added to the subschema control block to hold the page group and dbkey radix of the record that is current of rununit. This is maintained by the DBMS in tandem with the current-of-rununit dbkey. A new command is available for returning the page information (consisting of the page group and dbkey radix) for a record when it is not the current of rununit:

```
ACCEPT pginfo FOR CUSTOMER
```

A new option on the FIND/OBTAIN dbkey statement permits the specification of the page information during unqualified dbkey retrieval:

```
FIND DB-KEY dbkey PAGE-INFO pginfo
```

Mixed page group support provides two new DML commands. It also adds a new field to the Subschema Control.

The new Subschema Control field is:

Field Name	Description
PAGE-INFO	The page information associated with the last record accessed by the rununit. For example, after successful execution of a FIND command, PAGE-INFO is updated with the page information of the located record. Page information is not changed if the call to the DBMS results in a nonzero status condition. Page information is a four-byte field consisting of the following sub-fields: ■ Bytes 1-2: Page group number ■ Bytes 3-4: Dbkey radix The PAGE-INFO field overlays part of the SSCDBCOM area in the subschema control.

Note: Refer to the Dsects #SSCDS (for Release 12.0 and greater format SSC) and #SSC102 (for pre-Release 12.0 format SSC) for more information. Look for field xxxPINFO.

The ACCEPT PAGE_INFO statement moves the page information for a given record to a specified location in program variable storage. Page information that is saved in this manner is available for subsequent direct access by using a FIND/OBTAIN DB-KEY statement.

The dbkey radix portion of the page information can be used in interpreting a dbkey for display purposes and in formatting a dbkey from page and line numbers. The dbkey radix represents the number of bits within a dbkey value that are reserved for the line number of a record. By default, this value is 8, meaning that up to 255 records can be stored on a single page of the area. Given a dbkey, you can separate its associated page number by dividing the dbkey by 2 raised to the power of the dbkey radix. For example, if the dbkey radix is 4, you would divide the dbkey value by 2**4. The resulting value is the page number of the dbkey. To separate the line number, you would multiply the page number by 2 raised to the power of the dbkey radix and subtract this value from the dbkey value. The result would be the line number of the dbkey. The following two formulas can be used to calculate the page and line numbers from a dbkey value:

- Page-number = dbkey value / (2 ** dbkey radix)
- Line-number = dbkey value – (page-number * (2 ** dbkey radix))

Syntax

COBOL:

```
►►──────ACCEPT page-info-location FOR record-name ──────────────────────►◄
```

PL/I:

```
►►─ACCEPT PAGE_INFO RECORD (record-name)INTO (page-info-location) ──────►◄
```

Assembler:

```
►►─@ACCEPT PGINFO=pg-info-v,REC=record-name ────────────────────────────►◄
```

CA-ADS:

```
►►──ACCept PAGE-INFO into page-info-variable FOR record-name ───────────►

►──────┬──────────────────────────┬────── . ────────────────────────────►◄
       └────error-expression ─────┘
```

CA-IDMS/DML Online:

```
►►──────ACCept PGRn FOR record-name ────────────────────────────────────►◄
```

Parameters

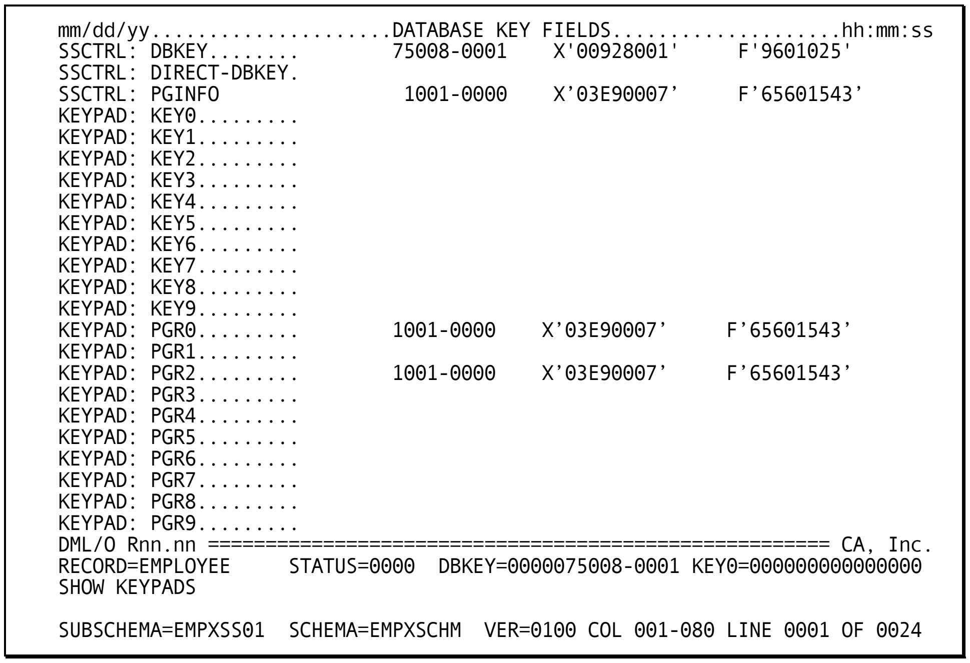
`page-info-location`
`pg-info`
`pg-info-variable`

A four-byte field that may be defined either as a group field or as a fullword field (PIC S9(8) COMP). Identifies the location in variable storage that contains page information for the specified record. Upon successful completion of this statement, the first two bytes of the field contain the page group number and the last two bytes contain a value that may be used for interpreting dbkeys.

`PGR`*n*

PGR*n* always refers to one of the ten CA-IDMS/DML Online PAGE-INFO values.

Use the CA-IDMS/DML Online SHOW KEYPADS screen to display the PAGE-INFO values, in the same manner that you would use it to display the dbkey values:



```
mm/dd/yy....................DATABASE KEY FIELDS...................hh:mm:ss
SSCTRL: DBKEY........        75008-0001    X'00928001'     F'9601025'
SSCTRL: DIRECT-DBKEY.
SSCTRL: PGINFO                1001-0000    X'03E90007'     F'65601543'
KEYPAD: KEY0.........
KEYPAD: KEY1.........
KEYPAD: KEY2.........
KEYPAD: KEY3.........
KEYPAD: KEY4.........
KEYPAD: KEY5.........
KEYPAD: KEY6.........
KEYPAD: KEY7.........
KEYPAD: KEY8.........
KEYPAD: KEY9.........
KEYPAD: PGR0.........        1001-0000    X'03E90007'     F'65601543'
KEYPAD: PGR1.........
KEYPAD: PGR2.........        1001-0000    X'03E90007'     F'65601543'
KEYPAD: PGR3.........
KEYPAD: PGR4.........
KEYPAD: PGR5.........
KEYPAD: PGR6.........
KEYPAD: PGR7.........
KEYPAD: PGR8.........
KEYPAD: PGR9.........
DML/O Rnn.nn ===================================================== CA, Inc.
RECORD=EMPLOYEE     STATUS=0000  DBKEY=0000075008-0001 KEY0=000000000000000
SHOW KEYPADS

SUBSCHEMA=EMPXSS01  SCHEMA=EMPXSCHM  VER=0100 COL 001-080 LINE 0001 OF 0024
```

Note: SHOW KEYPADS is a new CA-IDMS/DML Online command.

`record-name`

Specifies the record whose page information will be placed in the specified location.

Example

The following example retrieves the page information for the DEPARTMENT record and uses the dbkey format information to transform a page number into a dbkey.

```
01 W-PG-INFO.
   02 W-GRP-NUM       PIC S9(4) COMP.
   02 W-DBK-FORMAT    PIC 9(4) COMP.

   ACCEPT W_PG_INFO FOR DEPARTMENT.
   MOVE W-PAGE TO W-DBKEY.
   PERFORM ADJUST-PAGE W-DBK-FORMAT TIMES.

ADJUST-PAGE SECTION.
   MULTIPLY W-DBKEY BY 2.
```

Status Codes

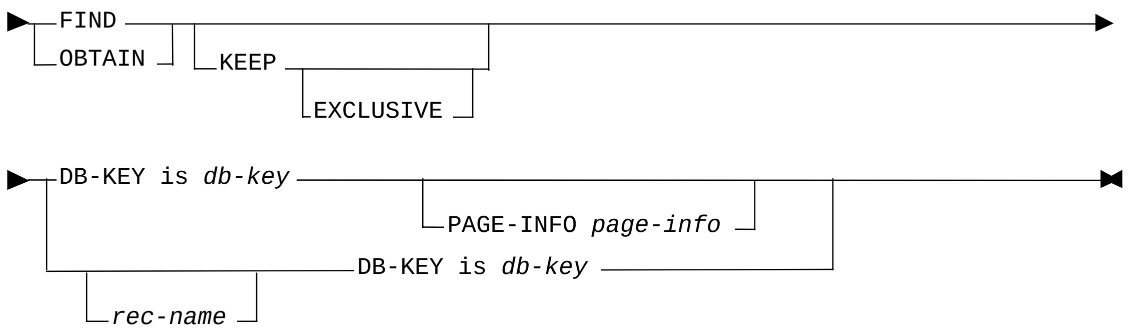
After completion of the ACCEPT PAGE-INFO statement, the ERROR-STATUS field in the CA-IDMS communications block indicates the outcome of the operation:

Status Code	Meaning
0000	The request has been serviced successfully.
1508	The named record is not in the subschema. The program has probably invoked the wrong subschema.

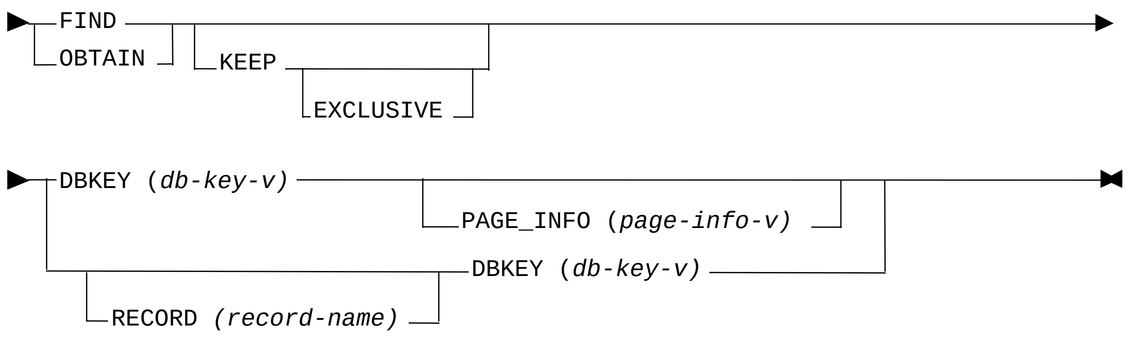
FIND/OBTAIN DB-KEY Syntax

Syntax

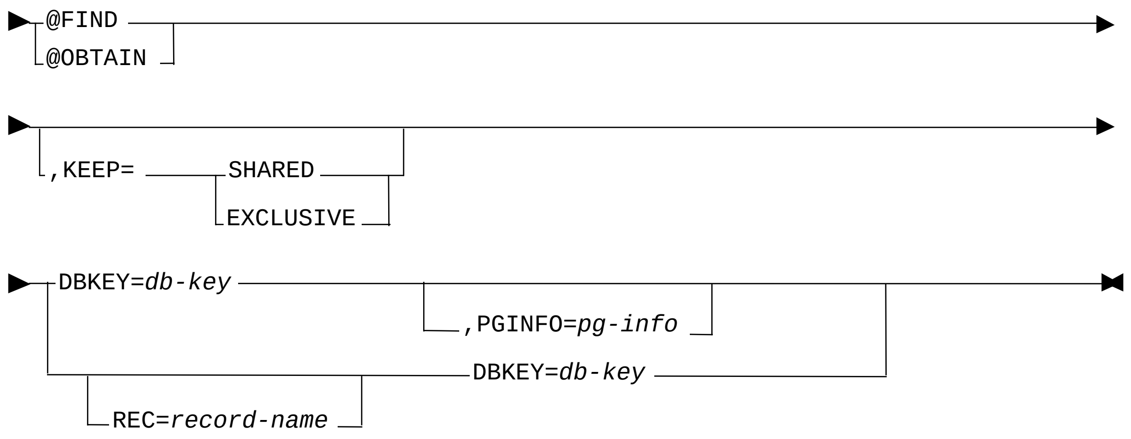
COBOL:



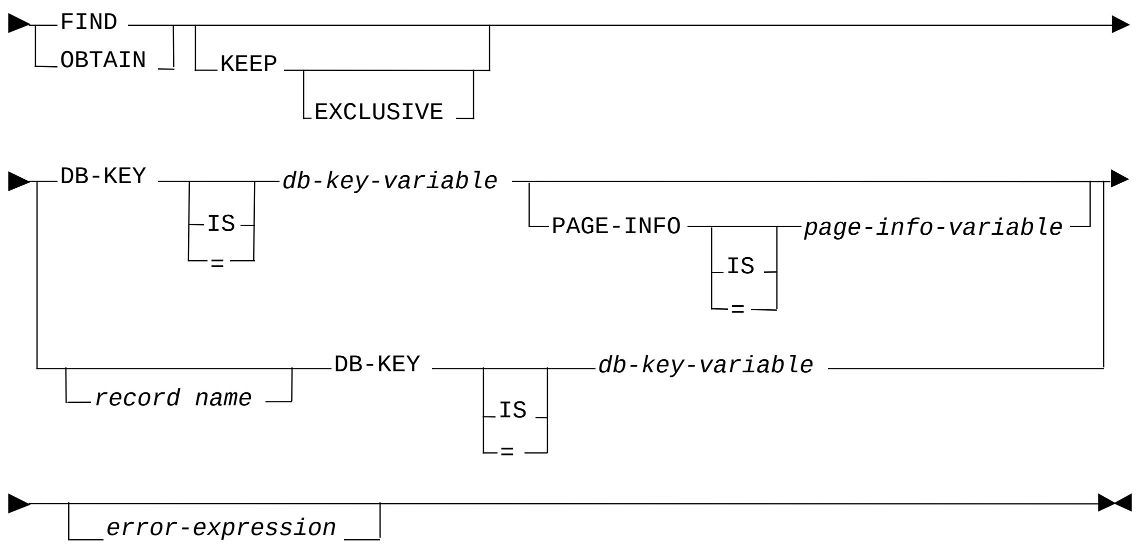
PL/I:



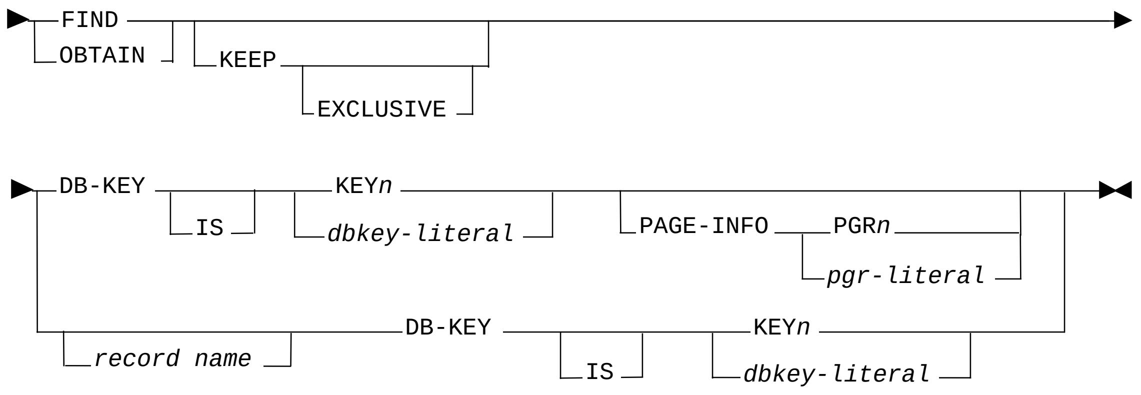
Assembler:



CA-ADS:



CA-IDMS/DML Online:



Parameters

PAGE-INFO

Specifies page information that is used to determine the area with which the dbkey is associated. If not specified, the page information associated with the record that is current of rununit is used.

Note: Page information is only used if the subschema includes areas that have mixed page groups; otherwise, it is ignored.

page-info
pg-info
pag-info-variable

A four-byte field that may be defined either as a group field or as a fullword field (PIC S9(8) COMP). Identifies the location in variable storage that contains the page information previously saved by the program.

Page information is returned in the PAGE-INFO field in the subschema control area if the subschema includes areas in mixed page groups. Page information may also be returned using an ACCEPT PAGE-INFO statement.

db-key

A field that identifies the location within program variable storage that contains a dbkey previously saved by the COBOL program.

rec-name
record-name
record name

The record type of the requested record. Rec-name must name a record that is included in the subschema.

`db-key-v`

A field that identifies the location within program variable storage that contains a dbkey previously saved by the program.

`dbkey`

Identifies the location in program variable storage that contains a dbkey previously saved by the program.

Db-key must identify a binary fullword synchronized field; it can be a register or a user-defined variable.

`db-key-variable`

Specifies the binary fullword in the dialog's record buffers that contains a previously saved database key.

`error-expression`

Specifies the status codes that are returned to the dialog.

`keyn`

One of the ten CA-IDMS/DML Online KEYPAD fields.

`PGR`*n*

PGR*n* always refers to one of the ten CA-IDMS/DML Online PAGE-INFO values.

Use the CA-IDMS/DML Online SHOW KEYPADS screen to display the PAGE-INFO values, in the same manner that you would use it to display the dbkey values:

```
mm/dd/yy....................DATABASE KEY FIELDS...................hh:mm:ss
SSCTRL: DBKEY........        75008-0001    X'00928001'     F'9601025'
SSCTRL: DIRECT-DBKEY.
SSCTRL: PGINFO                1001-0000    X’03E90007’     F’65601543’
KEYPAD: KEY0.........
KEYPAD: KEY1.........
KEYPAD: KEY2.........
KEYPAD: KEY3.........
KEYPAD: KEY4.........
KEYPAD: KEY5.........
KEYPAD: KEY6.........
KEYPAD: KEY7.........
KEYPAD: KEY8.........
KEYPAD: KEY9.........
KEYPAD: PGR0.........        1001-0000    X’03E90007’     F’65601543’
KEYPAD: PGR1.........
KEYPAD: PGR2.........        1001-0000    X’03E90007’     F’65601543’
KEYPAD: PGR3.........
KEYPAD: PGR4.........
KEYPAD: PGR5.........
KEYPAD: PGR6.........
KEYPAD: PGR7.........
KEYPAD: PGR8.........
KEYPAD: PGR9.........
DML/O Rnn.nn ===================================================== CA, Inc.
RECORD=EMPLOYEE     STATUS=0000  DBKEY=0000075008-0001 KEY0=000000000000000
SHOW KEYPADS

SUBSCHEMA=EMPXSS01  SCHEMA=EMPXSCHM  VER=0100 COL 001-080 LINE 0001 OF 0024
```

Note: SHOW KEYPADS is a new CA-IDMS/DML Online command.

`db-key-literal`

The dbkey-literal may be specified as PPPPPP-LLLL (page-line format), X'hhhhhhhh' (hexadecimal format), or F'nnnnnnnnn' (decimal format).

`pgr-literal`

The pgr-literal may be specified as X'hhhhhhhh' (hexadecimal format), or F'nnnnnnnnn' (decimal format).

Chapter

3

New Utilities

Overview

CA-IDMS Release 14.1 provides the following new utilities:

- Tune Index Utility
- Convert Page Utility
- Callable Restructure Utility

Tune Index Utility

The Tune Index Utility walks a sorted index in order to cause the adoption of orphaned indexed records. An *orphaned indexed record* is a record whose index pointer does not point back to the index record (SR8) which contains the record's index entry. Orphans occur as the result of splitting when an existing SR8 is split into two records to accommodate a new entry. As part of the split, some of the entries are moved to a new SR8, but the index pointer in their associated record is not adjusted to reflect the change, thus resulting in "orphaned" records.

By eliminating orphans, runtime database performance is improved when traversing from an indexed record to its associated index entry.

Considerations

This utility eliminates orphans only at the bottom level of the index. After execution, an index may still contain orphans at intermediate levels. There may also be a small number of orphans left at the bottom level.

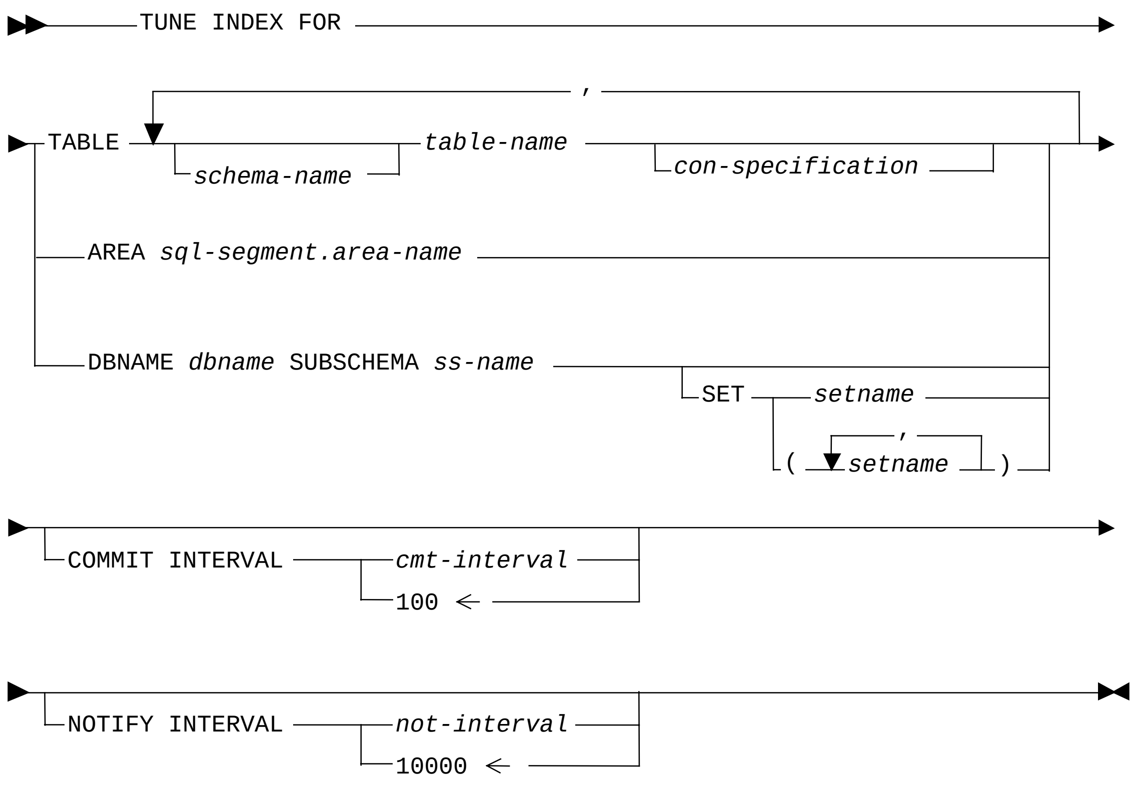
Orphan adoption only occurs for sorted indexed sets with index pointers (referred to as linked indexes). The Tune Index Utility has no affect on unsorted indexes or unlinked indexes. Because all SQL-defined indexes are unlinked, there is no benefit in running this utility on such indexes, and only indexed constraints are processed.

You can execute this utility both online (through the online command facility) and in batch through central version or batch local.

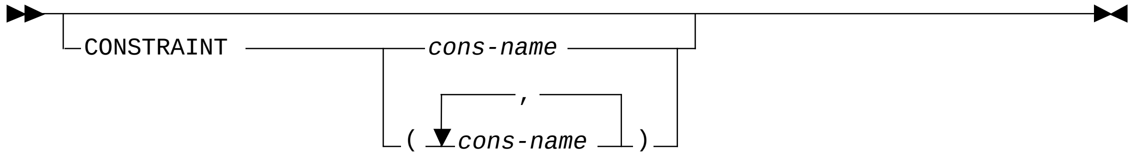
You can specify a commit interval that determines the frequency with which the utility will commit. The interval specifies the number of possible (not actual) updates that may take place before a commit is issued. If a deadlock occurs, the utility automatically restarts execution at the last commit point. You can disable committing and automatic restart by specifying a 0-commit interval.

You can specify a notify interval that determines the frequency with which the utility will write a notification message. The interval specifies the number of records/rows read before a message is written. The message is written to the job log and the operator's console. You can disable notification by specifying a 0-notify interval.

Syntax



`con-specification`



`schema-name`

Identifies the schema that will qualify the table name.

If omitted, the current session associated with the user session schema is used. Schema-name is required if there is no current session schema.

`table-name`

Identifies the table that is constrained by an indexed constraint.

`con-specification`

Identifies constraints on the current table that is being tuned.

If omitted, all indexed constraints on the current table are processed.

`sql-segment.area-name`

Identifies an sql-defined area to be selected for processing.

All tables with indexed constraints in the area are processed.

`dbname`

Identifies the dbname to be used when binding the subschema.

`ss-name`

Identifies the subschema to be used for processing a non-SQL database.

`set-name`

Identifies the indexed sets within the subschema that are to be processed.

If omitted, all linked indexed sets defined in the subschema are processed. (Linked indexed sets are indexed sets with index pointers.)

`cmt-interval`

Specifies the commit interval. After every cmt-interval record read, a commit is issued. If omitted, the default interval is 100. If specified as zero (0), no commits are issued.

`not-interval`

Specifies the notify interval. After every not-interval record read, a message is issued to the console stating how far the job has progressed. If omitted, the default interval is 10,000 records. If specified as zero (0), no notify message is issued.

`cons-name`

Identifies an indexed constraint on the current table that is to be tuned.

Usage

The Tune Index Utility has the following usage considerations:

- In order to use the feature, you must execute the utility specifying one of the following:
  - One or more tables whose indexed constraints are to be tuned
  - One or more areas containing tables whose indexed constraints are to be tuned
  - A subschema and DBNAME and optionally a list of indexed sets to be tuned
- A user must have DBAWRITE authority on all areas processed by the utility.
- If multiple indexes and/or multiple tables are processed in the same area, increasing the number of buffers will further improve performance. Alternatively, restrict processing to a single constraint or indexed set per statement.

Tune Index Utility Output

The following is a sample of a report produced by the Tune Index Utility:

```
IDMSBCF  14.1                            CA-IDMS Batch Command Facility


TUNE INDEX FOR DBNAME EMPDEMO SUBSCHEMA EMPSS01;
Status = 0        SQLSTATE = 00000        Messages follow:
DB002994 C0M333: IDMSTUNE  -  processing started
DB002994 C0M333: IDMSTUNE  -  342 records read for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  20 SR8s adopted for area EMP-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  123 records read for area ORG-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  35 SR8s adopted for area ORG-DEMO-REGION
DB002994 C0M333: IDMSTUNE  -  465 total records read
DB002994 C0M333: IDMSTUNE  -  55 total SR8s adopted
DB002994 C0M333: IDMSTUNE  -  5 indexes/sets processed
DB002994 C0M333: IDMSTUNE  -  processing completed

AutoCommit will COMMIT transaction

Command Facility ended with no errors or warnings
```

Convert Page Utility

The Convert Page Utility changes the page range for an area or the maximum number of records that can be stored on a page of an area.

You may process an entire segment or one or more individual areas using this utility. The utility first identifies the changes that need to be made by comparing old and new definitions for the segment or areas. Once all changes have been identified, it converts one or more files. It is possible to run the utility in parallel to process several files concurrently.

This utility allows you to reassign the page range of an area so that you can consolidate free pages. It also allows you to change the maximum number of records on a page, thus permitting more effective use of the space on a page and more flexibility in choosing page sizes.

Considerations

In order to use the feature, you must define one or more new segments and include them in a DMCL, or alter the definition of an existing segment and include it in a new DMCL. The utility requires both the old and new definitions of the affected segment or areas to be available at runtime.

Once the definitions are available, you can execute the utility. There must be enough disk space available to hold all converted files. The converted files can also be written to tape and then copied back to the original database files. Be sure to back up the original database beforehand.

Once all files have been converted, you must make the new segment or area definitions effective.

You will need DBAWRITE authority on all areas being converted.

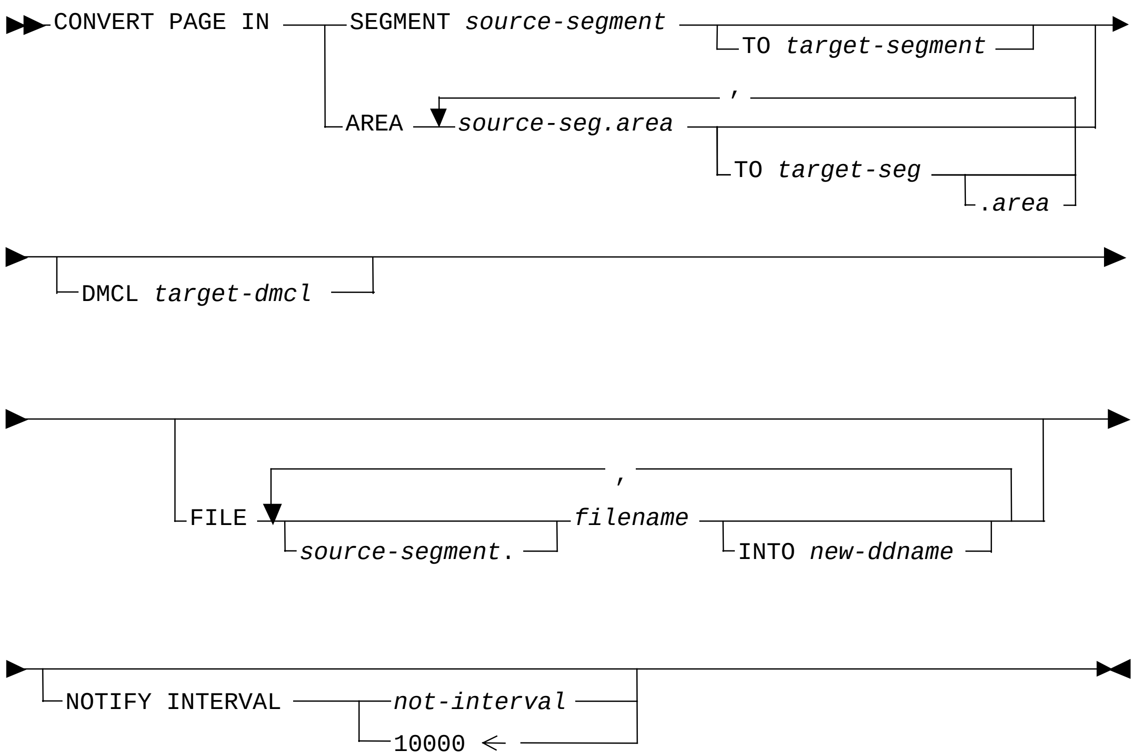
Page range and radix changes are determined by comparing all source and target area definitions. Regardless of which files are selected for conversion, all differences are applied to all files selected for conversion.

File conversion is performed against all files in the specified segment or areas, unless the FILE parameter is specified, in which case only specified files are converted.

File conversion consists of copying a page from the source file to the target file while making any required changes. Page number changes are accomplished by subtracting the old low page and adding the new low page. Maximum records per page changes are effected by breaking up a dbkey into a line and page number using the old information and reassembling it using the new information.

Note: If no changes apply to a selected file it is copied anyway.

Syntax



`source-segment`

Names a segment to be processed. All areas in this segment are compared to matching areas in the target segment for page range and maximum records per page changes. Unless restricted by the FILE parameter, all files in this segment are scanned for page range changes and copied to new files.

`source-seg.area`

Names an area to be processed. All specified areas are compared to the corresponding target areas and checked for page range and maximum records per page changes. Unless restricted by the FILE parameter, all files for this area are scanned for page range changes and copied to new files.

`target-segment`

Names the segment with the new page range and maximum records per page definitions. If you omit target-segment, it defaults to the source segment name. You should use this when the target segment is in the same DMCL as the source.

`target-seg.area`

Names the area that contains the new page range and maximum records per page definitions. If you omit this, it defaults to source-seg.area name. If you omit the area, it defaults to the source area name. You should use this when the target area is in the same DMCL as the source.

`target-dmcl`

Names the DMCL that contains the target segment or area definitions. Use this parameter when the target segment or area is defined in a different DMCL than the current one.

`filename`

Restricts the conversion to the selected source files. If you do not specify the FILE parameter, then all files for selected segments and areas are converted. However, all target file definitions must have ddnames that are different from the source file definitions.

`new-ddname`

Names the JCL ddname used for the output file. This name must be unique in the job step. If you don't specify new-ddname, the ddname in the target file definition is used. In this case the name of the target file must match the source file.

`not-interval`

Specifies the notify interval. After every not-interval record read, a message is issued to the console stating how far the job has progressed. If omitted, the default interval is 10,000 records. If specified as zero (0), no notify message is issued.

Usage

The Convert Page Utility has the following usage considerations:

- All input and output ddnames must be unique within the same job step. If the target DMCL files do not have unique ddnames, use the INTO option for each output file.
- If an area that is being converted has a cross-area set or linked constraint, then you must convert all areas touched by the set or link. You must determine what files are affected, and therefore what files must be converted.

- If you need to convert multiple files, the files do not all have to be converted in the same job.
- As long as all changes are identified to each job, you can convert each affected file separately. It may be desirable to run several jobs in parallel to reduce the time required to convert an area.

  Important! *It is important that you identify all page range changes that affect a file before the file is converted. Depending on the nature of the page range change, it may not be possible to apply a partial change to a previously converted file.*
- If you omit a change identification, then you will need to reapply all changes to the original file, not the converted file.
- You cannot change the number of pages assigned to an area, nor lower the maximum number of records per page to less than the actual number of records stored on any page of an area.
- This utility must run in local mode and with update activity quiesced on the affected areas.
- The source and target files must match in page size and number of blocks. (The source file is copied block for block.) If the target file does not match then the converted file will not be usable by the new file definition.
- JCL must include DD statements for the output files. If a file is not defined to use dynamic file allocation, then DD statements for the input files are also required.

Examples

The EMPDEMO database has three areas: the Emp-demo-region, Org-demo-region, and the Ins-demo-region. There are cross-area sets between the Emp and Org-demo-regions, and between the Emp and Ins-demo-regions, but there are no sets between the Ins-demo-region and the Org-demo-region; therefore, if you need to change the:

- Page range for the Org-demo-region, you must convert both the Emp-demo-region and the Org-demo-region.
- Page range for the Ins-demo-region, you must convert both the Emp-demo-region and the Ins-demo-region.
- Page range for the Emp-demo-region, you need to convert all three areas.

Example 1

The first step in converting an area's page range is to define the new segment with all changes. This new segment could be included in the current DMCL if its name is different from the old segment, if the page group and page ranges are different from the old ones, and if the ddnames are different from the old ones.

After the DMCL is updated, the required syntax to do all conversions would be:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO;
```

The Convert Page Utility scans both segments and determines which areas are being changed. Because the output files have unique ddnames, they do not need to be identified, but all three files that are in the EMPDEMO segment are converted, even if the changes do not affect all files.

To restrict the process, you could code the following:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
   FILE EMPDEMO-FILE, ORGDEMO-FILE;
```

This syntax would only convert the EMPDEMO and ORGDEMO files. If the INSDEMO file were not affected by any changes, then this would be all that would be needed. However, if the INSDEMO file were affected, it could be converted with a separate job as follows:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO NEWDEMO
   FILE INSDEMO-FILE;
```

This works because all changes were identified to both jobs, because all changes are contained in the single segment.

When completed, the DMCL would have to be modified a second time to remove the old segment definition and to possibly rename the new segment to the old one, to possibly change the new page group to the old page group, and to possibly change the ddnames back to the old ones.

Example 2

Another approach is to define a new DMCL with all changes made to the existing segment.

In this case you would not have to modify the DMCL a second time, but more syntax is required for the conversion process, as follows:

```
CONVERT PAGE IN SEGMENT EMPDEMO TO EMPDEMO DMCL NEWDMCL
   FILE EMPDEMO-FILE INTO EMPDDX,
           INSDEMO-FILE INTO INSDDX,
           ORGDEMO-FILE INTO EMPDDX;
```

This syntax requires that you give each output file a unique ddname for the run only, because the defined ddnames in the target DMCL are the same as the source DMCL. However, once converted, the old DMCL is discarded, and the new one is renamed and the old one is replaced.

Again, if all three files are not affected, the number of files converted can be reduced by not naming the unaffected files.

Example 3

If area syntax were being used, the syntax could be as follows:

```
CONVERT PAGE IN AREA
         EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
         EMPDEMO.INS-DEMO-REGION TO NEWDEMO,
         EMPDEMO.ORG-DEMO-REGION TO NEWDEMO;
```

Or if the areas were in a different DMCL:

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                   EMPDEMO.INS-DEMO-REGION,
                   EMPDEMO.ORG-DEMO-REGION
                   DMCL NEWDMCL
                   FILE EMPDEMO-FILE INTO EMPDDX,
                        INSDEMO-FILE INTO INSDDX,
                        ORGDEMO-FILE INTO EMPDDX;
```

This syntax works because all areas affected by the changes are identified. Again, this converts all files even if some are not affected.

Example 4

To only convert affected files, specify only the files that should be converted:

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION TO NEWDEMO,
                   EMPDEMO.INS-DEMO-REGION   TO NEWDEMO,
                   EMPDEMO.ORG-DEMO-REGION   TO NEWDEMO
                   FILE EMPDEMO-FILE, INSDEMO-FILE;
```

Or if the areas were in a different DMCL:

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                   EMPDEMO.INS-DEMO-REGION,
                   EMPDEMO.ORG-DEMO-REGION
                   DMCL NEWDMCL
                   FILE EMPDEMO-FILE INTO EMPDDX,
                        INSDEMO-FILE INTO INSDDX;
```

This syntax would only cause the Emp-demo-region and the Ins-demo-region to be converted.

If you decided that the Org-demo-region also needed to be converted, you could run the following:

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                   EMPDEMO.INS-DEMO-REGION,
                   EMPDEMO.ORG-DEMO-REGION
                   DMCL NEWDMCL
                   FILE ORGDEMO-FILE INTO ORGDDX;
```

This syntax works because all three areas were identified to compare phase. Even though the files were converted in separate jobs, all jobs knew about all changes.

Bad Example 4

The wrong way to do the previous example would be to reduce the areas considered for change, instead of the files to be converted:

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                   EMPDEMO.INS-DEMO-REGION,
                   DMCL NEWDMCL
                   FILE EMPDEMO-FILE INTO EMPDDX,
                        INSDEMO-FILE INTO INSDDX;
```

This syntax works only if the Org-demo-region is unaffected by any changes. Only the Emp-demo-region and the Ins-demo-region are converted. But if it were later discovered that the Org-demo-region required conversion, you would have to reconvert the Emp-demo-region.

```
CONVERT PAGE IN AREA EMPDEMO.EMP-DEMO-REGION,
                   EMPDEMO.INS-DEMO-REGION,
                   EMPDEMO.ORG-DEMO-REGION
                   DMCL NEWDMCL
                   FILE EMPDEMO-FILE INTO EMPDDX,
                        ORGDEMO-FILE INTO ORGDDX;
```

Note that Ins-demo-region did not have to be reconverted because it was unaffected by changes to the Org-demo-region, but because the Emp-demo-region is affected by changes to the Ins-demo-region; Ins-demo-region is included in the list of changed areas.

Callable Restructure Utility

The Callable Restructure Utility (IDMSCRSU) is a version of the CA-IDMS Restructure Segment command that is callable from a user-written program. It uses the standard IDMSRSTT table to control the restructure of the data portion of a database record.

IDMSCRSU is called with the following register values and parms:

On Entry:

- R1 points to a four-word parmlist.
- R13 should point to a 36-word save and work area.
- R14 contains the return address.
- R15 contains the entry point address.

Considerations

IDMSCRSU does not support changes to pointers or calling database procedures. The assumption is the caller has obtained the database record in the old subschema format and now wants to store or modify a record in the new subschema format.

To save a search of the IDMSRSTT table on every call, the caller can save the RREC address returned in R0 after the first call. This address can then be passed instead of the IDMSRSTT address on all subsequent calls for the same record type.

A 36-word register save area and work area is expected to be passed in R13. If called from a DC assembler program, you can use a "#CHKSTK =36" instruction to ensure there is enough room in the stack.

Parameters

`parm-1`

The address of the FSR (SR51) control block from the “old” subschema. This block should describe the record as it exists before the restructure takes place.

`parm-2`

The address of the IDMSRSTT table that describes the changes being made to the database record. This table is generated from the IDMSRSTT macro. (See the *CA-IDMS Utilities Guide*.) The table is searched for the RREC block that matches the FSR name passed in parm-1.

- or -

The address of the RREC control block in the IDMSRSTT table. (No search is done.)

`parm-3`

The address of the data record to be restructured. Only the data should be passed, not pointers or internal fields.

`parm-4`

The address of a buffer where the restructured record will be returned. It should be large enough to hold the maximum size of the record being restructured as described in the ‘new’ subschema.

On exit:

- R0 contains the address of the IDMSRSTT RREC block for the record just restructured, if found.

- R15 contains one of the following return codes:
  - 0: Call successful - The output buffer contains the restructured record.
  - 4: FIELD=ALL was specified for the record in the IDMSRSTT table. There were no changes to the data. The output buffer does **not** contain a copy of the record; the caller should use the input record image.
  - 8: The record was not found in the IDMSRSTT table.

Chapter 4 Other Enhancements

Overview

CA-IDMS Release 14.1 provides the following enhancements:

- Enhanced COBOL II Performance
- Dynamic Front-end System Table Support
- Exit 24
- Exit 34
- CA-IDMS/DML Online productivity enhancements

COBOL II Performance Enhancement Under LE/370

The RHDCLEFE program is provided to improve performance of VS COBOL II programs running under CA-IDMS/DC using the LE/370 runtime library.

To realize the performance improvement, link RHDCLEFE and define it in the sysgen as RESIDENT as follows:

```
ADD PROGRAM RHDCLEFE
   CONCURRENT
   DYNAMIC
   DUMP THRESHOLD IS 0
   ENABLED
   ERROR THRESHOLD IS 5
   ISA SIZE IS 0
   LANGUAGE IS ASSEMBLER
   MPMODE IS SYSTEM
   NOMAINLINE
   NEW COPY IS ENABLED
   NONOVERLAYABLE
   PROGRAM
   NOPROTECT
   REENTRANT
   RESIDENT
   REUSABLE
   NOSAVEAREA
   .
```

This feature also allows COBOL II programs to run with storage protection.

Dynamic Front-end System Table

The Dynamic Front-end System Table feature provides an enhancement to 24-hour processing for CA-IDMS/UCF sites.

During startup of the CA-IDMS central version, a front-end system table describing each CA-IDMS UCF front-end system is loaded. When you alter this table, the central version has to be stopped and re-started to pick up the changed table.

This feature allows online refresh of the table. New front-end systems can be added or changes can be made to an existing front-end system. There is no longer a need to stop and re-start the central version, resulting in better 24-hour support.

To use this feature, create a new or update an existing front-end system table using the procedures described in the *CA-IDMS System Operations Guide*.

Syntax

A new DCMT command will allow reloading the front-end system table:

```
DCMT Vary UCF Front-end System Table New Copy
```

Exit 24 — Year 2000 Testing

A new numbered user exit is provided that is useful for Year 2000 testing of applications. Exit 24 provides a means to return a different value for GET TIME requests than the current value returned by the operating system.

Exit 24 is called whenever a DC GETIME is issued to obtain the time and date from the operating system.

Parameters

To use this feature, you must write a user exit routine with the following attributes:

- System mode
- No storage protect

- Amode 31

On entry to the exit:

- R1 = Address of two (2) word parm list
- +0 = Address of fullword containing exit number
- +4 = Address of a doubleword date and time

The doubleword date and time contains the packed date, and the binary absolute time in 0000 seconds. This is identical to the values normally found in CSATIME and CSADATE fields. The date is expressed as 0nYYDDDF (for 1900 n is 0; for 2000 n is 1).

To set a different time to be returned, simply store a date and time in the doubleword pointed at by R1.

See the *CA-IDMS System Operations Guide* for more information about user exits.

Return Codes

Return codes are ignored.

Exit 34 — Unqualified Dbkey Find/Obtain Exit

A new numbered exit is provided for use with the Mixed Page Group Support feature. Exit 34 helps identify and correct applications that may require modification to function correctly when the Mixed Page Group Support feature is enabled.

Exit 34 is invoked by IDMSDBMS whenever a rununit is enabled for mixed page group processing and a FIND DB-KEY or OBTAIN DB-KEY verb with no record name is issued. The exit is not invoked for rununits accessing the dictionary or catalog.

Exit 34 is provided to allow the runtime detection of unqualified dbkey retrievals when “Mixed Page Group Binds Allowed” is specified for the DBNAME. The exit may be used to display messages on the console and/or abend the task. Furthermore, this exit can provide the correct page group and radix value for the passed dbkey to enable the application to run correctly without requiring source changes.

It is possible that an unqualified FIND DB-KEY or OBTAIN DB-KEY command may not retrieve the desired record when Mixed Page Group Binds are allowed. When this feature is enabled, IDMSDBMS will use the current page and radix value for the dbkey. If the last DML operation referenced a page group other than the one desired, then the wrong record may be retrieved. If the unqualified retrieval is the first DML operation for the rununit, then there is no current of page group and a 0326 status code is returned.

Until you are otherwise notified, Computer Associates recommends that applications with unqualified FIND DB-KEY or OBTAIN DB-KEY commands be enhanced to specify a record name or exploit the PAGE-INFO parameter rather than use this exit.

Considerations

You must write the Exit 34 routine to execute in SYSTEM mode. The #DEFXIT macro that adds the exit routine to a system must:

- Specify MODE=SYSTEM
- Call the routine using either DC or IBM calling conventions
- Call the routine by entry point

Parameters

Four parameters are passed:

- Fullword 1 — Address of a five-word save area. The area will remain consistent for the life of the rununit. Data stored here will remain until the rununit finishes.
- Fullword 2 — Address of the IB50 Control Block. (See macro #FIBDS.)
- Fullword 3 — Address of the Subschema Control Block. (See copy book #SSC120.)

  Fields: SSCPGRUP and SSCRADIX will contain the current page group and radix value for the rununit (these may be changed). Upon return from the exit IDMSDBMS will make the changed values current for the rununit.

Note: Sample exit RHDCUX34 has been supplied as part of CA-IDMS. This program will display a message on the console when called. It also contains examples of abending a task and of modifying the current page group. To use the sample exit as written RHDCUXIT must define a #DEFEXIT as follows:

```
#DEFXIT MODE=SYSTEM,CALL=DC,EP=UX34EP1
```

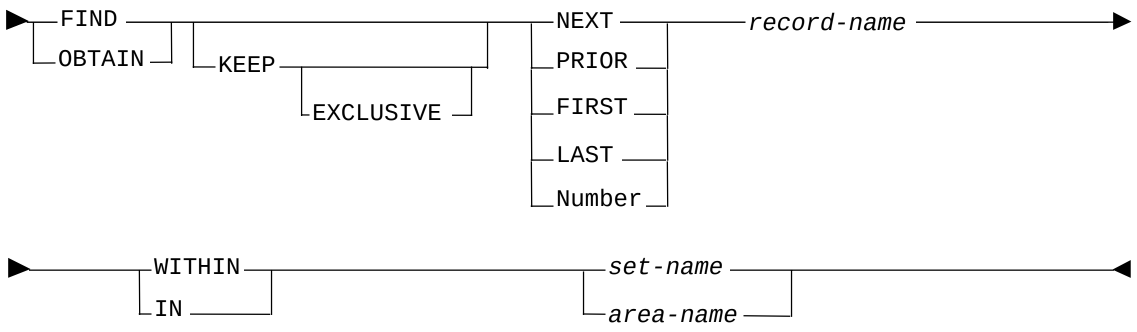
and RHDCUX34 must be linked with RHDCUXIT.

CA-IDMS/DML Online Productivity Enhancements

The CA-IDMS/DML Online product has been enhanced to provide for improved productivity by implementing the following two new features designed to reduce the number of required keystrokes.

Extended FIND/OBTAIN Support

In the Format 3 FIND/OBTAIN statement the clause 'WITHIN set-name/area-name' may be omitted. If omitted, this statement will default to 'WITHIN area-name' and CA-IDMS/DML Online will determine the correct area-name for the record requested.



Changing Enter Key Settings

Use the SET DEFENTK command to change the default settings for the Enter key.

```
SET DEFENTK   'ON /OFF'
```

Setting the command to 'ON' (default) clears all data on the CA-IDMS/DML Online command line when you press the Enter key without entering any data. If DEFENTK is set to 'OFF' the last command on the CA-IDMS/DML Online command line (if any) will be re-executed. This feature is useful, for example, when navigating a database to repeat FIND/OBTAIN NEXT/PRIOR records without having to type over data on the command line.

This value is initially set at install time and may be altered thereafter by changing the parameter module (USDTPARM) source code and then reassembling and relinking it. It can also be changed dynamically at runtime with the above SET DEFENTK command.

Index