CA IDMS™ SQL

User Guide

Release 18.5.00

CA Technologies Product References

This document references the following CA Technologies products:

- CA IDMS™
- CA Visual Express™

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

Welcome to CA IDMS SQL Quick Bridge (or Quick Bridge for short). Quick Bridge makes it easy for CA IDMS programmers and DBAs to define table procedures. Quick Bridge is a graphical user interface (GUI) tool that runs under Windows, using client/server technology. It uses a series of intuitive dialogs that prompt you for information required to define a *table procedure specification* based on records and sets relationships defined in a nonSQL database. The generated table procedure specification creates the SQL table procedure definition; system definitions, and source for the COBOL program that you can upload and store in a CA IDMS dictionary. It also creates SQL table syntax that you can upload to an Ingres II database, allowing you to migrate table definitions and data between CA IDMS and Ingres II databases.

The generated COBOL program provides a functional data access approach using CA IDMS DML. Quick Bridge generates all basic processing logic in a clearly structured, easily readable program. You can modify the program to fine-tune the database navigation or to add additional processing.

This section contains the following topics:

Features

Quick Bridge is a client/server application with the client running under Microsoft Windows on your PC and the server running under a mainframe operating system. The Windows client uses a series of intuitive dialogs and a point and click system to help you create a table procedure specification and generate the output that defines the table procedure.

Quick Bridge offers two methods to establish client/server communications with the mainframe: CA IDMS Server and EDBC for CA IDMS.

Supported Structures

Quick Bridge supports the following non-SQL database structures:

- Multi-level record structures
- Bill-of-material structures
- Repeating element groups using the OCCURS clause
- The lowest level of a group-level field
- One record type of a multi-member set relationship

About This Guide

This guide explains how to create table procedures using the Quick Bridge tool. To use this guide, you should be a CA IDMS DML programmer or database administrator. In addition, you should be familiar with SQL concepts and Windows. Additional knowledge of ODBC and client/server communications would be helpful.

The examples in this guide are based on the CA IDMS Commonweather demo database, defined by schema EMPSCHM. You may find it useful to connect to your demo database to replicate the examples used in this guide.

You may also find it helpful to use the online help provided with Quick Bridge. The online help system provides dialog-level help, as well as procedural help, similar to the information this guide provides.

Related Documentation

For more information about table procedures, see the *CA IDMS SQL Reference* and *CA IDMS SQL Programming Guide*. For more information about defining nonSQL databases and nonSQL programming techniques, see *CA IDMS Database Administration* and the *CA IDMS DML Programming Guide*. For information about defining a CA IDMS system, see *CA IDMS System Generation*.

See the *CA IDMS Server User Guide* if you plan to use the CA IDMS server for client/server communications. If you plan to use EDBC for CA IDMS, see the documentation for that product.

System Requirements

This section describes the system requirements for the client and the server.

For the Client

On the client, you need the following software components:

- Microsoft Windows 95 or 98, Windows NT (service pack 3.0) or higher, Windows 2000, or Windows Millennium
- CA IDMS Server Release 4.0 or greater or EDBC Client 1.0

Hard disk requirements vary, depending on which components you install. The installation program tells you specific hard disk requirements for the components.

For the Server

On the server, you need CA IDMS release 12.01 genlevel 9506 or later, CA IDMS SQL option and CA IDMS server release 2.0 or later or EDBC for CA IDMS 1.0. You also need a COBOL compiler.

Install Quick Bridge

Quick Bridge requires the installation of 2 components. First the PC client software needs to be installed; next the mainframe component needs to be installed.

Installation of the PC Client Component

The PC client component will install the Quick Bridge client software in a folder on the client PC. The client installation will also have created and loaded a subfolder MAINFRAM. This subfolder holds the files that will allow the installation of the mainframe component described following.

1. Insert the Quick Bridge diskette 1 or CD-ROM and use Windows Explorer to locate the READMEQB.TXT file. Use any text editor to review this file. It may contain instructions for specific releases or environments.
2. In Windows Explorer, double-click on INSTALL(.EXE) to run the installation program or use Run from the Start Menu and type A:INSTALL.
3. Follow the instructions during the installation; these are self-explanatory.

Installation of the Mainframe Component

The mainframe component requires the installation of the SQL schema CAAV, the table procedures CAAV.GETREC (CAAVLREC) and CAAV.MODULETEXT (CAAVMTXT).

1. Review the READMEMF.TXT file in the MAINFRAM folder. It may contain instructions for specific mainframe releases or environments.
2. Upload all the files of the MAINFRAM directory to a mainframe source library. The files, which are coded in ASCII must be translated to the mainframe codeset EBCDIC.
3. Process the SQL definitions contained in CAAVQBSD.SQL with IDMSBCF. No configuration of CAAVQBSD.SQL is required, but of course an appropriate catalog must be specified. For more information about the IDMSBCF utility, see the *CA IDMS Command Facility*.
4. The COBOL sources for the table procedures are CAAVLREC.COB and CAAVMTXT.COB. These programs must be compiled and linked in a CA IDMS load library as described in the *CA IDMS DML Reference – COBOL*.
5. Users with an environment that supports fully reentrant COBOL programs in CA IDMS (IBM COBOL II) should use the CAAVSGNR.SGN member as input to RHDCSGEN to update their IDMS system definitions with the table procedure programs. Other users must use the CAAVSGNQ.SGN member. Generate and recycle your IDMS system. For more information about the sysgen utility RHDCSGEN, see *CA IDMS System Generation*.
6. Add an SQL schema for each dbname that will be accessed by Quick Bridge.

   Because Quick Bridge accesses the dictionaries that hold the non SQL schema, subschema, and record definitions through SQL, appropriate SQL schemas must be defined for each dbname containing these dictionary segments.

   For example, if the DBNAME APPLDICT contains the Quick Bridge SQL definitions and will be specified in the ODBC datasource definition, the following SQL syntax will make this dbname accessible by Quick Bridge:

   ```
   CONNECT TO APPLDICT;
   CREATE SCHEMA APPLDICT FOR NONSQL SCHEMA SYSDIRL.IDMSNTWK DBNAME APPLDICT;
   ```

   SYSDIRL is the name of the dbname that has been processed with IDMSDIRL and that contains the IDMSNTWK schema. APPLDICT contains the non SQL schema's, subschema's, and records that Quick Bridge will be using.

   If other dictionary needs to be made accessible within the same ODBC datasource, additional schema's will need to be created in the same manner.

Chapter 2: Creating Table Procedure Specifications

This section contains the following topics:

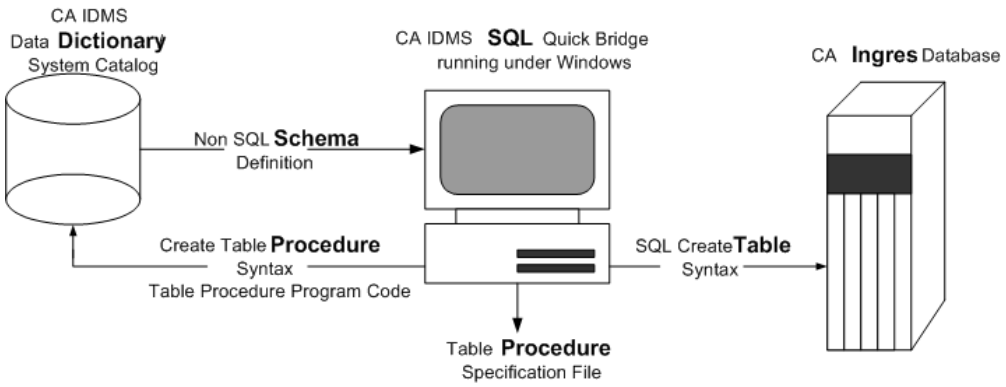
Before You Begin

Before you begin to create a table procedure specification, review the supported structures outlined in the chapter "Introduction," to see whether the definitions in your CA IDMS database can be translated into a table procedure. In addition, you should collect the following information:

- A schema listing and database diagram of the database that you want to access
- The data source name of the CA IDMS system and dictionary that you want to connect to

Understand the Process

This diagram illustrates the process involved with defining a table procedure:



Input for the table procedure definitions comes from the dictionary that contains the nonSQL schema definition of the CA IDMS database you want to access. Using Quick Bridge running under Windows, you define the table procedure specification using a series of dialogs. When you generate the table procedure specification, you create:

- Syntax for defining the table procedure to the CA IDMS system catalog and the COBOL program code for the table procedure
- Syntax for defining the SQL table to Ingres II or some other database

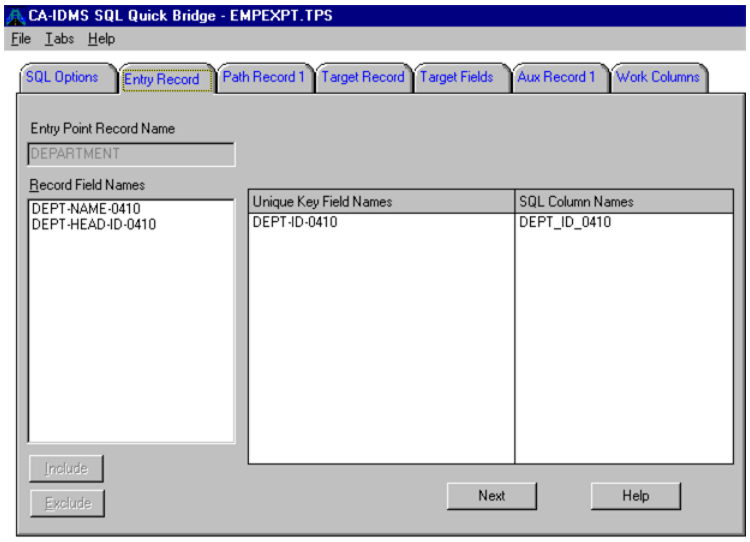
Start and Exit CA IDMS SQL Quick Bridge

To start Quick Bridge, locate the program CA IDMS SQL Quick Bridge in the Windows Start menu or double-click the CA IDMS SQL Quick Bridge icon on your Windows desktop.

The product displays an empty application window. From the File menu, you can connect to a database or exit. You can exit Quick Bridge at any time during your session. If you have not saved your work, Quick Bridge prompts you to save your work before it terminates your session.

Application Window

When you run Quick Bridge, it displays its application window, where you perform the tasks to create a table procedure specification. The illustration below, which contains an existing table procedure specification, displays elements of the application window:



The *title bar* displays the product name and the name of the table procedure specification file. From the *menu bar,* you can pull down menus that help you create and manage table procedure specifications.

The *tabs*, like tabs in a notebook, help you quickly access different parts of the table procedure specification. To move from tab to tab, simply click on it or select it from the Tab menu. If your table procedure specification contains many tabs, Quick Bridge displays *tab scroll arrows* to let you know that there are additional tabs that are not displayed on the screen.

The *split bar* lets you resize the box compartments that contain the unique key field names and SQL column name equivalents. To use the split bar, position the mouse cursor over the split bar in the box heading. When the mouse cursor changes to a double-headed arrow, click the mouse and slide it left or right to resize the box compartments.

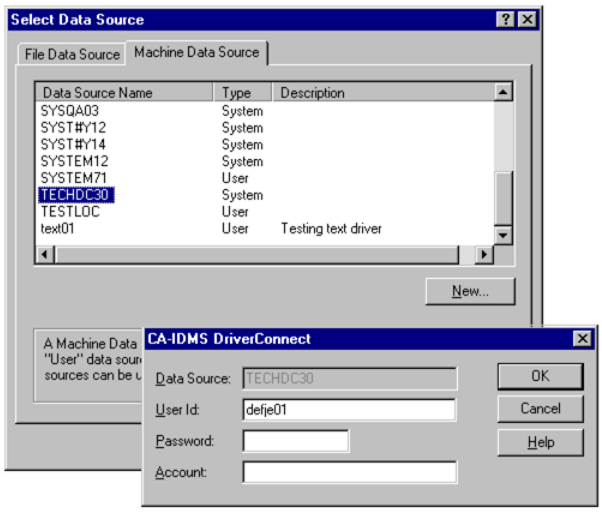
Connect to a CA IDMS Data Source

You must connect to a CA IDMS data source before you can do anything else in Quick Bridge.

To connect to a CA IDMS data source:

1. Select Connect to Database from the File menu.
2. In the SQL Data Sources dialog, select an existing CA IDMS data source or click New to define a new one.

In the following example, user defje01 connects to the TECHDC30 datasource, which uses the CA IDMS Server driver:



The CA IDMS Server requests signon information to complete the connection.

Note: For detailed information about defining data sources using the CA IDMS Server driver, see the *CA IDMS Server User Guide*. If EDBC for CA IDMS is used, see the EDBC product documentation.

Create, Edit, and Save Table Procedure Specification Files

After you connect to a data source, you can create or open a table procedure specification file. To create a new table procedure specification file, select New from the File menu. To open an existing table procedure specification file, select pen from the File menu.

To save an existing table procedure specification, select Save from the File menu. To save a table procedure specification for the first time or to rename an existing table definition, select Save As from the File menu.

The default file name is *external_name*.TPS, which Quick Bridge derives from the external name you type the SQL Options tab. In the Save As dialog, select the drive and directory, and type a file name for the table procedure specification.

Quick Bridge displays the file name (EMPEXPT.TPS) in the title bar of the application window.

Edit SQL Column Names

When you create a table procedure specification, Quick Bridge automatically creates SQL column names for each field (record element) you select for the table procedure. If a synonym for language SQL is defined for the record element, this will be used. Otherwise the default SQL column name is the same as the field name, except that it replaces underscores (_) for hyphens (-). If the default name exceeds 18 characters or is already used elsewhere in the table procedure specification, Quick Bridge notifies you and lets you type an appropriate name.

To edit an SQL column name

1. Double-click the column name.
2. Position the cursor in the column name and edit it. You can also select all or part of the column name and use standard Windows editing keystrokes to cut (Ctrl+X), copy (Ctrl+C), and paste (Ctrl+V) the selected text.
3. To redisplay the original column name, select the column name and press ESC.

To view SQL column names already in use by the table procedure specification

Select the SQL Column Names Currently In Use command on the Help menu. Quick Bridge displays a window, such as the one following that lists the names already in use.

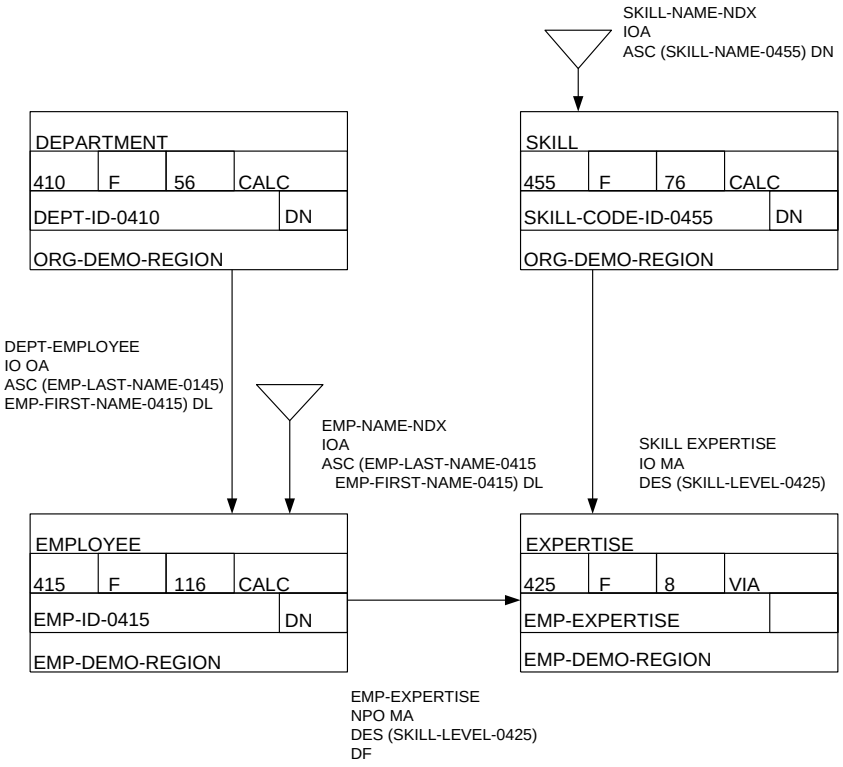
Define a Table Procedure Specification

This section describes the steps required to define a table procedure specification using the Quick Bridge tool. They are:

1. Define basic information about the table procedure, such as its name and the user database schema and subschema it will access.
2. Select the entry record in the path.
3. Select additional records in the primary path.
4. Select the target record and fields for the table procedure.

To follow along using the Commonweather demo database installed with CA IDMS, create a data source that connects to the EMPSS01 subschema of EMPSCHM version 100. The table procedure specification that you will create in this section uses the DEPARTMENT, EMPLOYEE, EXPERTISE, and SKILL database records.

The database diagram following displays these records and their set relationships as defined for the Commonweather demo database:



Define the Basics

When you select New from the File menu, Quick Bridge displays the SQL Options tab in the application window. The SQL Options tab prompts you for basic information to define the table procedure specification, such as a name for the table procedure, the schema and subschema that define the records in the table procedure, and the SQL operations that can be performed on the table procedure.

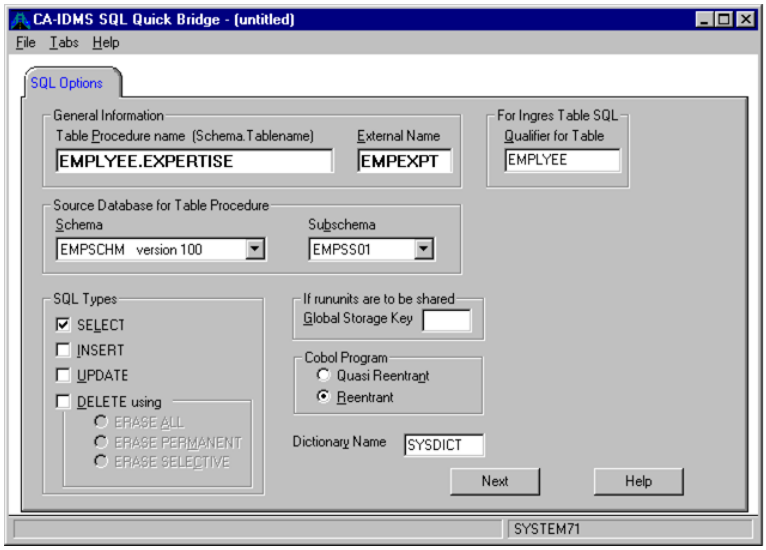
To define basic information about the table procedure

1. Verify that the dictionary you want to access is correct. The default dictionary is the dictionary specified in the ODBC datasource definition. Modify the dictionary by typing the name in the Dictionary Name edit box. This name actually is the name of an SQL schema that must have been defined for the non SQL schema SYSDIRL.IDMSNTWK of the dictionary of interest.
2. Type a name for the table procedure using this format:

   ```
   schema_name.table_name
   ```

   where *schema_name* is the name of an SQL schema defined in the CA IDMS system catalog and *table_name* is a name that conforms to SQL table naming conventions.
3. In the External Name edit box, type a 1- to-8-character external name under which the COBOL program will be linked. Quick Bridge also uses this value in the default naming conventions for files that it creates.
4. In the Schema and Subschema drop-down list boxes, select the nonSQL database schema and subschema that supply the definitions of the data that you want to access.
5. In the SQL Types group, optionally select what SQL operations you want the table procedure to be able to perform on the nonSQL database. SELECT is always an option. If you choose DELETE, you can also select one of the DELETE options, which are described in detail in online Help.
6. If rununits are to be shared. If Global Storage Key is left blank, each generated table procedure will use its own CA IDMS rununit. Otherwise, this table procedure will share a rununit, with other table procedures that use the same Global Storage Key.
7. COBOL Program — Specifies the type of reentrancy of the compiled COBOL program. Choose either Quasi Reentrant or Reentrant. This option drives the syntax in the generated SGN file.
8. In the Qualifier for Table edit box, type the qualifier for the table definition that will be generated.

The following example displays the SQL options tab containing information for the sample table procedure EMPEXPT. The SQL operations selected for the table procedure mean that data in the table procedure can be retrieved, inserted, updated, and deleted.



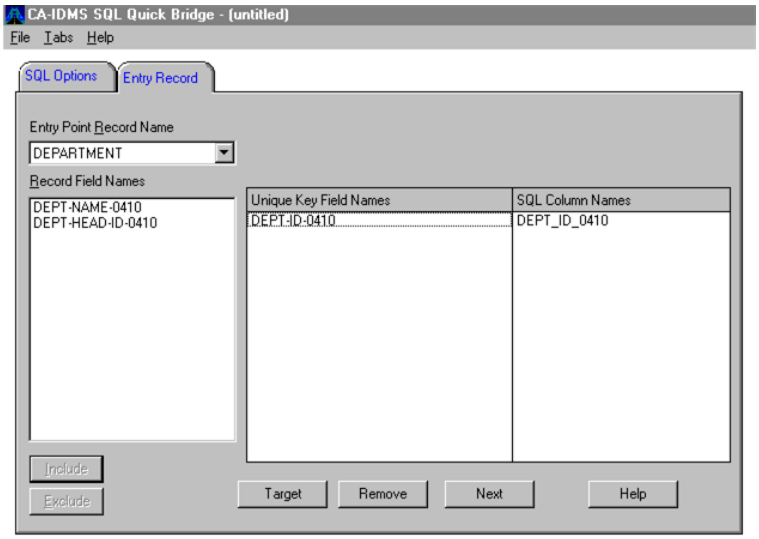
Select the Entry Record

After you complete the SQL Options tab, Quick Bridge displays the Entry Record tab. The *entry record* describes the first record in the primary path that leads to the target record.

To select an entry record

1. Select a record from the drop-down list box. The entry records that Quick Bridge displays in the drop-down list box either have a CALC-key or are members of a system-owned indexed set.
2. Optionally, select the fields that define a unique key for the record. The unique key can be the CALC key, index key, or any other combination of fields. If you do not specify a unique key, CA IDMS accesses the record sequentially. To perform insert operations, you must select a unique key, unless the set that connects the target record with the last path record is not automatic.
3. Click Include to include the selected field names. Each field name appears under the Unique Key Field Name column along side an SQL column name, which you can change to anything you like. For information about SQL column naming conventions and editing, see Editing SQL Column Names.
4. Click Next to add more records to the primary path or Target if the entry record is also the target record of the table procedure. Click Remove if you want to change the basic options you selected in the SQL Options tab.

In the following screen, the DEPARTMENT record is the entry record for the EMPEXPT table procedure specification. The CALC key for the DEPARTMENT record is the DEPT-ID-0410 field, so it is selected as the unique key field name for the entry record:



NOTE: Notice that Quick Bridge automatically replaces hyphens with underscores in the SQL column name.

Add Path Records

After selecting the entry record for the table procedure, you can add more records to the primary path. The Path Record tab is very similar to the Entry Record tab except that it also includes the name of the set that defines the relationship of the path record to the preceding record on the path.

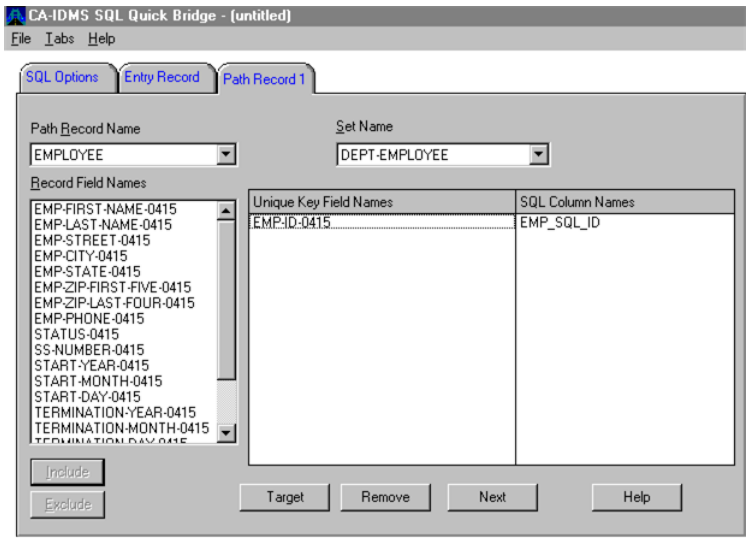
To add a record to the primary path

1. Select a record from the drop-down list box.
2. Select the set relationship that you want to use to access the new path record.
3. Optionally, select the fields that define a unique key for the record. Click Include to include the selected field names. Each field name appears under the Unique Key Field Name column along side an SQL column name, which you can change to anything you like. For information about SQL column naming conventions and editing, see Editing SQL Column Names.
4. Click Next to add more records to the primary path or Target if the path record is also the target record of the table procedure. You can also remove a path record by clicking Remove.

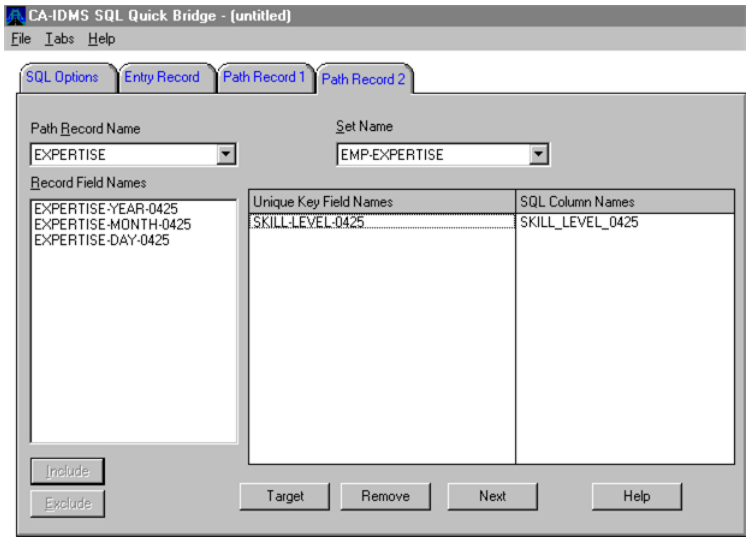
The screens following add the EMPLOYEE record and then the EXPERTISE record to the primary path of the EMPEXPT table procedure specification:

- The EMP-ID-0415 field is the CALC-key field for the EMPLOYEE record, so it is selected as the unique key field.
- The access mode for the EXPERTISE record is via the EMP-EXPERTISE set, which is owned by the EMPLOYEE record. The EXPERTISE record is the member record, which is sorted by skill level, so the SKILL-LEVEL-0425 field is selected as the unique key field.

EMP-ID-0415 is the CALC key for the employee record, the second record in the primary path:



SKILL-LEVEL-0425 is the field used to sort EXPERTISE records owned by EMPLOYEE records. Expertise is the third record in the primary path and also the target:



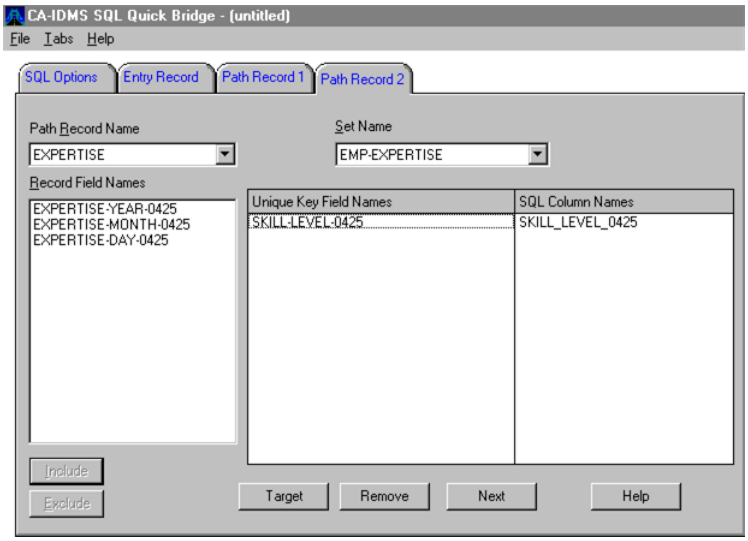
Select the Target Record

The *target record* is the record that you ultimately want to access. Once you create a path to the target record, you can update, delete, and insert target record occurrences. In the EMPEMPT example, the target record is the EXPERTISE record. The Target Record tab is the same as the Path Record tab and appears when you select the Target button in the Entry Record or Path Record tabs.

To create a target record

1. Click the Target button in the Entry Record or Path Record tabs. Be sure you specify a unique key for the record before you click Target.
2. Quick Bridge displays the Target Fields tab. All fields in the target record appear except the unique key fields. Click Include All to include all the fields in the target record or select one or more fields and then Include.

In the following screen, the target record is EXPERTISE and the unique key for the record is SKILL-LEVEL-0425. The remaining fields in the EXPERTISE record are the target fields:



These fields are the non-unique key fields in the target record. Note that the SQL column names have been edited because the name cannot exceed 18 characters.

Define Auxiliary Records

An *auxiliary record* is a non-path record that owns the target record in a set relationship. A target record can have zero or more auxiliary records. Quick Bridge automatically creates an Aux Record tab for each auxiliary record. The auxiliary records that Quick Bridge displays either have a CALC-key or are members of a system-owned indexed set.

CA IDMS needs an auxiliary record when you issue INSERT and sometimes UPDATE statements. In our example, the auxiliary record is the SKILL record, which is the owner record of the SKILL-EXPERTISE set; the member record of that set is the target record, EXPERTISE.

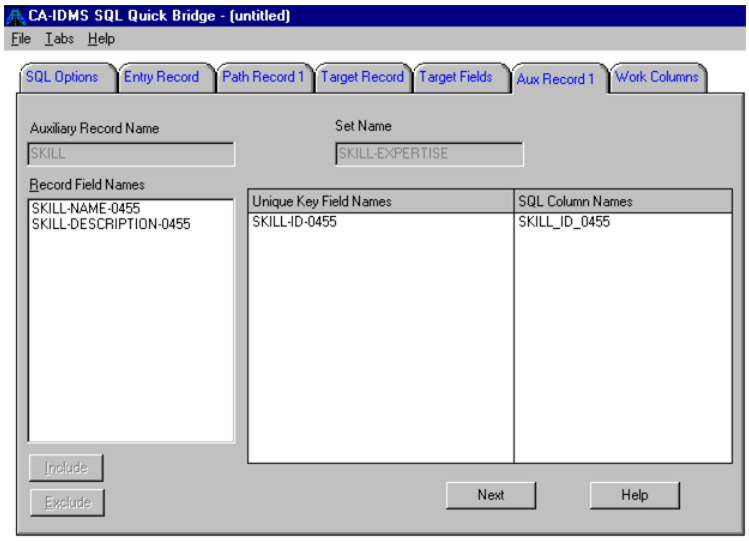
Auxiliary records also automatically define an alternate entry or key into the database. While the key of the primary path is referenced as KEY 1 in the generated COBOL program, the alternate key corresponding to the first auxiliary record is KEY 2, to the second auxiliary record is KEY 3, and so on.

When a row from the table procedure needs to be selected, the generated COBOL program will first check if values for KEY 1 columns are specified, if so KEY 1 will be used to select the row; next it will check if values for KEY 2 are specified and so on. Only if no values have been specified for any key, will the program use sequential access.

To select a unique key for an auxiliary record

1. If an Aux Record tab appears, click it.
2. Optionally, select the fields that define a unique key for the record. The unique key can be the CALC key, index key, or any other combination of fields. If you do not specify a unique key, CA IDMS accesses the record sequentially. To perform insert operations, you must select a unique key, unless the set that connects the target record with the auxiliary record is not automatic.
3. Click Include to create a unique key for the record. Each field name appears under the Unique Key Field Name column along side an SQL column name, which you can change to anything you like. For information about SQL column naming conventions and editing, see Editing SQL Column Names.
4. Click Next or click another Aux Record tab or the Work Columns tab.

In this example, the target record has one auxiliary record, SKILL, which owns the EXPERTISE record using the SKILL-EXPERTISE set. The unique key field for the SKILL record is its CALC key field, SKILL-ID-0455:



Add More Columns

Work columns are additional, non-record columns that you can add to the table procedure specification. They are included in the linkage section of the table procedure program, so that you can use the generated table procedure to pass information back and forth and to perform additional processing. Quick Bridge automatically creates a Work Columns tab when you create a target record.

To add a work column

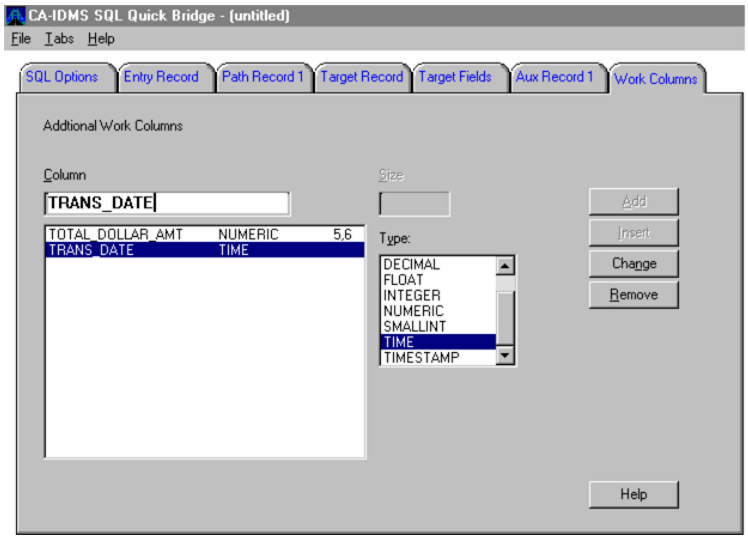
1. Click the Work Columns tab.
2. Type a column name that conforms to SQL naming conventions.
3. Select a data type. If the data type you choose is Char, type an integer value in the Size edit box that represents the column length. If the data type is Numeric or Decimal, type the precision and scale in the Size edit box, using this syntax:

   ```
   precision, scale
   ```

   where *precision* represents the number of digits allowed in the column and *scale* is the number of digits to the right of the decimal point.
4. Select Add to add the column to the bottom of the column list or Insert to insert it before the selected column.

To change the data type or size of a column, select the column name and then click Change after you type the new information. To delete a column, select the column name and click Remove.

The following example shows three columns added to the table definition. If you select Insert, the TRANS_TIME column will be added before the TRANS_DATE column; if you select ADD, it will be added after:



Note: To add work fields to a table procedure specification, define a field and click Add or Insert. Click Change to redefine a field, or Remove to delete it.

Generate a Table Procedure Specification

After you define and save a table procedure specification, you can generate it. Quick Bridge creates files with this default naming convention:

`tps_file_name.file_extension`

where *tps_file_name* is the name that you used to save the table procedure specification file and *file_extension* defines the file contents:

File Extension	Description
.SQ1	CREATE TABLE PROCEDURE syntax to load in the CA IDMS system catalog
.SQ2	CREATE TABLE syntax to load in an Ingres database with Ingres 1.x syntax
.SQ3	CREATE TABLE syntax to load in an Ingres II database with Ingres 2.x syntax
.SGN	ADD PROGRAM system generation syntax to define the COBOL program to the CA IDMS system

File Extension	Description
.COB	The COBOL program that defines that table procedure

Examples of the .SQ1, .SQ2, and .SGN files appear in Appendix A.

To generate a table procedure specification

Select Generate from the File menu. Quick Bridge displays a message box when it is done generating the table procedure specification.

Upload a Table Procedure to CA IDMS

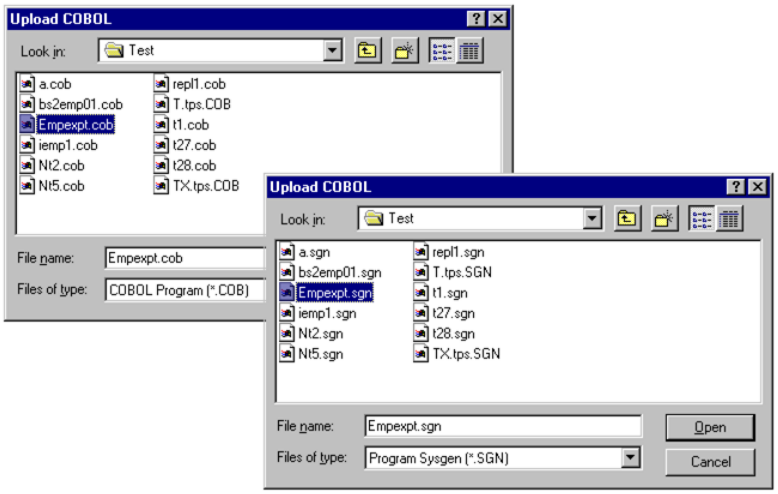
After you generate a table procedure specification, you can upload the CREATE TABLE PROCEDURE syntax and the COBOL program to a CA IDMS dictionary.

To upload table procedure syntax, program syntax, and program code to a CA IDMS dictionary

1. If a table procedure specification is open, select File Close to close it.
2. If necessary, select File Connect to Database to connect to the data source that defines the CA IDMS system and dictionary in which you want to load the table procedure syntax and code.
3. Select Upload SQL from the File menu to upload the CREATE TABLE PROCEDURE syntax to the CA IDMS system catalog. Select a file with file extension .SQ1.

   Note: You can only use the Upload SQL menu item if CA IDMS Server is used for connectivity.
4. Select Upload COBOL from the File menu to upload the table procedure COBOL program. Select a file with file extension .COB and click OK. CA IDMS stores the program in the dictionary as a module with language type COBOL.
5. Select Upload COBOL from the File menu again to upload the system generation PROGRAM statement that defines the COBOL program. Select a file with file extension .SGN and click OK. CA IDMS stores the statement in the dictionary as a module with language type SYSGEN. To include the module in your system startup use the INCLUDE statement as described in CA IDMS System Generation.

Select a .COB file to upload the COBOL program.



Select a .SGN file to upload the PROGRAM system generation statement.

For detailed information about defining and using table procedure programs, see the *CA IDMS SQL Reference*. For instructions to compile and link the COBOL program, see the *CA IDMS DML Reference — COBOL* and *SQL Reference*.

Upload CREATE TABLE Syntax to Ingres II

After you generate a table procedure specification, you can upload the SQL CREATE TABLE syntax to an Ingres II database.

To upload CREATE TABLE syntax to an Ingres II database

1. If a table procedure specification is open, select File Close to close it.
2. If necessary, select File Connect to Database to connect to the data source that defines the Ingres II database.
3. Select Upload SQL from the File menu to upload the CREATE TABLE syntax. Select a file with type "Ingres 1.x table SQL", extension .SQ2 or a file with type "Ingres 2.x table SQL", extension .SQ3.

Appendix A: Table Procedure Examples, Design, and Use

This appendix provides information you can use to:

- Design table procedures and select unique key fields
- Understand how Quick Bridge determines what access strategies to employ to retrieve and update records.
- Study examples of table procedure specifications for a variety of record/set relationships
- Specify a DBName for a table procedure
- Reference authorities required to modify records in the dictionary
- See samples of the output files created by the table procedure you defined in the chapter "Creating Table Procedure Specifications."
- How to assign a data type of date to SQL columns
- Reference messages you might receive when you generate your table procedure

To get the most out of Quick Bridge, you should be familiar with the data structure of the target database and understand how the generated table procedure can be used in your SQL access strategy. To review, a network-type database defines data by elements that are grouped together into record occurrences, which together form a record type. A set defines the relationship between record types. In a relational database, data is represented in columns; a group of columns form a row. Together, the columns and rows form a table. A foreign key defines the relationship between tables. Among other uses, table procedures give users of nonSQL-defined schemas a way to map SQL DML statements to navigational DML statements.

This section contains the following topics:

Identify Record Relationships

Before you create a table procedure specification, invest some time in planning and design. You need to:

- Identify the record type that will become the table in the relational database; this is the target record.
- Identify all the set relationships that the target record participates in. These set relationships form the foreign keys in the new table. To define the foreign keys, you must identify a hierarchy of record key fields that uniquely identify a single record occurrence for each set relationship.
- Identify the network navigational path that leads to the target record. The path can be any one of the network relationships that you already identified. Good candidates include mandatory/automatic sets or a path that can be navigated efficiently. You do not need to define a path if the target record has a CALC access mode or is a member of a system-owned indexed set.

Choose Unique Key Fields

Quick Bridge generates table procedures that use the unique key fields on the primary path to locate a single target record occurrence in the database. This section provides some insight about selecting unique fields for your path and target records. The selection depends, in part, on the type of processing the table procedure will perform: SELECT, UPDATE, INSERT, or DELETE.

SELECT and Searched UPDATE and DELETE Operations

SELECT processing or searched UPDATE or DELETE operations do not require a unique key to locate a row. Searched UPDATEs and DELETEs use database currencies based on search criteria in the Where clause, as described in "Defining and Using Table Procedures" in the *CA IDMS SQL Reference*. The CA IDMS Online Control Facility or Batch Control Facility lets you update multiple rows if the query result returns more than one row. However, not all query tools allow you to update multiple rows.

INSERT Operations

You must define unique keys to perform INSERT processing because only one row will be added to the table and all the owners in the primary path must be identified before the row can be inserted. The target record does not, however, require a unique key.

When a unique key is defined for the target record, the rules for the target record are the same as other path records in regard to "Next Row" processing. But CA IDMS does not determine if the target record is unique unless duplicates are not allowed for the record or set relationship. Therefore, the table procedure returns an IDMS error (1205), not an SQL error. Unique keys are also important to INSERT processing when the target record is inserted in an auxiliary set relationship with an Automatic connect option. Auxiliary records must have unique keys defined for them whenever the relationship is Automatic. Otherwise, IDMS currency is not established for that set, and the INSERT fails.

Join Tables

Unique keys are the only fields retrieved from path and auxiliary records and are, therefore, the only fields that can be used to join tables. In cases where the uniqueness of an Auxiliary record depends on a record above it hierarchically, the table procedure developer should add OBTAIN OWNER code to the COBOL source to retrieve the unique key for that record. Then the developer should update the linkage section of the table procedure program and the CREATE TABLE PROCEDURE statement to include any new columns in the table/linkage section.

Modify Table Procedures

If the code generated by Quick Bridge does not support your exact requirements for "real-world" table procedures, you can modify the table procedure source. If you make changes to the COBOL source, *do not* regenerate the source from Quick Bridge or upload the CREATE TABLE statements again since your changes will be overlaid and lost.

Access Methods for Selected Records

This section describes how Quick Bridge determines how to access entry records, primary path records, and auxiliary records, starting with a description of how it uses unique keys to select record occurrences for each record type on the primary path.

Use Unique Keys to Access Records on the Primary Path

The generated table procedure uses one of two access strategies when it uses unique keys to return a record occurrence for each record type on the primary path:

- **Plural Access** — If the table procedure receives fewer values than the number of defined unique key fields for a record type, it returns all record occurrences for that record type to the SQL engine for the current owner relationship.
- **Unique Access** — If the table procedure receives a value for each defined unique key field for a record type, the table procedure returns a single occurrence of that record type to the SQL engine for the current owner relationship.

Note: If you do not define a unique key for a particular record type, the table procedure uses the plural access method for that record type. The data returned to the SQL engine for a particular record occurrence is subject to additional filtering by the SQL engine. See the CA IDMS SQL Reference to determine when values from an SQL statement will be passed to the table procedure for a particular type of query and how SQL engine filtering may affect the resultant table.

Plural Access

Plural access returns all occurrences of a particular record type to the SQL engine for the current owner occurrence. To satisfy this requirement, table procedures generated by Quick Bridge sequentially process the record type based on its position in the primary path. Entry records that:

- Are members of a mandatory/automatic system-owned index are accessed using that index to find all its member records.
- Do not participate in a mandatory/automatic system-owned index are accessed by performing an area sweep until all record occurrences for that record type are found.

For path records, the set relationship is used to return all members for the current owner occurrence. Target records that are linked with the path through a SET which is not automatic, will always be accessed using an area sweep.

Unique Access

Unique access returns a single occurrence of a particular record type to the SQL engine for the owner occurrence. To satisfy this requirement, generated table procedures process the record type based on its position in the primary path. The following section describes strategies employed to obtain unique occurrences of entry, path, and auxiliary records.

Unique Access Strategies for Entry, Path, and Auxiliary Records[

This section describes strategies Quick Bridge uses to obtain unique record occurrences of entry, path, and auxiliary records.

Unique Access Methods for Entry Records

For entry records, the method used to locate unique record occurrences is determined by which fields were selected as unique keys for that record type. An entry record can be either a CALC type record, a member of a (preferably mandatory/automatic) system-owned index, or both.

When the entry record access mode is CALC and is not a member of a system-owned index:

- The access mode for the record type is CALC if all of the unique keys fields selected in Quick Bridge match the CALC key fields *exactly*
- Otherwise, the record type is accessed using an area sweep

When the entry record is a member of a system-owned index:

- The access mode for the record type is also CALC if the record has a CALC access mode and all fields selected in Quick Bridge match the CALC key fields *exactly*
- Otherwise, the access mode for the record type is INDEX SET USING, if all fields selected in Quick Bridge match the sort elements of the index set exactly. The INDEX SET USING access mode requires that the unique key fields, that make up the index, are specified in the same order as the sort elements of the index set are defined in the network schema. Otherwise, the record type is accessed by walking through the index.

When the record type is accessed using an area sweep or by walking an index, the values passed to the table procedure are compared to the values in the current record occurrence. The area sweep or index walk continues until the values match.

The rules defined on the previous page anchor the query to a single occurrence of the entry record.

Unique Access Method for Path Records

For every other record type in the primary path, the access strategy walks the set relationships from the owner of that relationship to the next member of the relationship. When the next member of the set is obtained, CA IDMS compares the values returned from the current record occurrence to the values supplied to the table procedure in the unique query. An exact match makes the record occurrence unique for that record type for the current owner relationship.

Unique Access Methods for Auxiliary Records

An auxiliary record is a record that participates in a set relationship with the target record, but is not part of the primary path. The generated table procedure locates unique occurrences of auxiliary records in one of two ways:

- In Next Row processing, each auxiliary record is located as owner in the auxiliary relationship. The query returns the auxiliary record's unique key to the SQL engine for filtering.
- In Insert Row and Update Row processing (when the foreign key is changed), the auxiliary record is located by one of four ways.
  - **CALC access mode** — When the unique keys fields selected in Quick Bridge match the CALC key fields exactly, the record type uses CALC access mode
  - **INDEX SET USING access method** — When the auxiliary record is a member of a system-owned index and the unique key fields selected in Quick Bridge match the index key fields exactly, the INDEX SET USING access mode is being used. The INDEX SET USING access mode requires that the unique key fields, that make up the index, are specified in the same order as the sort elements of the index set are defined in the network schema.
  - **Indexed access** — When the auxiliary record is a member of a mandatory/automatic system-owned index, CA IDMS compares all record occurrences in that index to the values passed to the table procedure. The single unique record occurrence with that key becomes the object of the new or updated set relationship.
  - **Area sweep** — When the auxiliary record cannot be accessed by one of the above methods, each record occurrence for that record type in the area is compared to the values passed to the table procedure. The single unique record occurrence with that key becomes the object of the new or updated set relationship.

Modify the Selected Access Method

Sometimes the selected access method does not perform the most "correct" method for your database structure. You can change the table procedure code to reflect your understanding of the database design. For example:

- The auxiliary record is CALC, but the unique keys for that record type contain more than the CALC key fields. This condition makes Quick Bridge perform an area sweep. You could enhance the table procedure code to perform OBTAIN CALC DUPLICATE processing which could be more efficient than an area sweep.
- The auxiliary record is not CALC, but is owned by a CALC record. You could include additional columns in the CREATE TABLE PROCEDURE and table procedure code to support the foreign key to that auxiliary record. Then add the required code to return the data in SELECT processing, and the required code to perform UPDATE processing.

Samples from the Commonweather Database

This section contains a few examples from the Commonweather Demo Database that you can add to the SQL schema EMPDB. The samples include accessing a target record:

- Accessible by CALC key
- That is a member of a set
- With no unique key
- In a bill-of-materials structure

Sample 1: Target Record Accessible by CALC key

Target Record

EMPLOYEE (unique key EMP-ID-0415, the CALC key)

Set Relationships

EPT-EMPLOYEE (owner DEPARTMENT, CALC key DEPT-ID-0410)
OFFICE-EMPLOYEE (owner OFFICE, CALC key OFFICE-CODE-0450)

Analysis

Occurrences of the EMPLOYEE record can be uniquely identified using its CALC key.

To define the table procedure specification

1. After completing the OPTIONS tab, select the EMPLOYEE record type in the Entry Record Name drop down box. Select EMP-ID-0415 as the unique key for this record type and click Target.

   Quick Bridge finds all set relationships for the target record and creates Auxiliary (Aux) tabs for the DEPT-EMPLOYEE and the OFFICE-EMPLOYEE relationships, as well as a tab for the Target Fields and another for Additional Columns.

2. In the Target Fields tab, select all fields you want in the new table. For all but the rare case (field REDEFINES, for example), click Include ALL to include all record fields as table columns. You can create a table view later if you do not want specific columns displayed.

3. The Aux tabs represent the network relationships that will become foreign keys for the new table. For the DEPT-EMPLOYEE Aux tab, choose DEPT-ID-0410 as the unique key. For the OFFICE-EMPLOYEE Aux tab, choose OFFICE-CODE-0450 as the unique key. Save and generate the new table procedure.

Sample 2: Target Record Member of a Set Relationship

Target Record

HOSPITAL-CLAIM (unique key CLAIM-DATE-0430, PATIENT-NAME-0430 within the COVERAGE-CLAIMS set relationship)

Set Relationships

COVERAGE-CLAIMS (owner COVERAGE, unique key, SELECTION-DATE-0400 (made up of Year, Month and Day) combined with TYPE-0400

Path

The table below identifies the primary path and unique keys that define unique occurrences of the HOSPITAL-CLAIM record:

Record	Unique Key	Set Relationship
EMPLOYEE	EMP-ID-0415	Owns COVERAGE via EMP-COVERAGE set
COVERAGE	SELECTION-DATE-0400, TYPE-0400 within EMP-COVERAGE set relationship	Owns HOSPITAL-CLAIM via COVERAGE-CLAIMS set
HOSPITAL-CLAIM	CLAIM-DATE-0430, PATIENT-NAME-0430 within the COVERAGE-CLAIMS set relationship	

Foreign Key for COVERAGE-CLAIMS Relationship

EMP-ID-0415, SELECTION-DATE-0400, TYPE-040

Analysis

The owner of the COVERAGE-CLAIMS set is COVERAGE, which you access via the EMP-COVERAGE set relationship. The unique key for COVERAGE is SELECTION-DATE-0400 (day, month, year) and TYPE-0400. This combination of fields uniquely defines an occurrence of the COVERAGE record within the EMP-COVERAGE relationship. To refine the unique key, you must select the EMPLOYEE record type, which owns the COVERAGE record via the EMP-COVERAGE set. The unique key for EMPLOYEE is EMP-ID-0415. Therefore, the foreign key for the COVERAGE-CLAIMS relationship becomes the EMP-ID-0415, SELECTION-DATE-0400, and TYPE-0400.

To define the table procedure specification

1. Select EMPLOYEE record in the Entry Record tab. Select EMP_ID_0415 for the unique field and then click Next.
2. In the Path Record Tab, select COVERAGE for the path record and include SELECTION_DAY_0400, SELECTION_MONTH_0400, SELECTION_YEAR_0400, and TYPE_0400 for the unique fields. Click Next.
3. In the new Path Record Tab, select the HOSPITAL-CLAIM record and CLAIM_DAY_0430, CLAIM_MONTH_0430, CLAIM_YEAR_0430, PATIENT_FIRST_NAME_0430, and PATIENT_LAST_NAME_0430 as the unique fields. Click Target.
4. In the Target Fields tab, select additional fields for the target record before you save and generate the table procedure.

No auxiliary records exist because the target record does not participate in any other set relationships

Alternative Strategy

Instead of making the HOSPITAL-CLAIM record the target, you can also add a table procedure for the COVERAGE record type. By making COVERAGE the target, you can perform INSERT processing, setting up the foreign keys required to relate the COVERAGE table to the EMPLOYEE table.

If you select COVERAGE to be the target record, then the developer of the COVERAGE table would have to make code changes to the table procedure to support the HOSPITAL-CLAIM record type. At first this may seem an odd way to retrieve the data. But from a table standpoint, this view may be more realistic. The COVERAGE record type can be combined with the HOSPITAL-CLAIM type to form a single view of the two independent tables.

Remember that the COVERAGE-CLAIMS relationship is a multi-member set. You can combine all three member records with the owner into one complete table. To do this, the developer would have to add code into the table procedure to include all of the appropriate fields. The developer would also have to add code to determine what record type to access and how to update and insert records.

Sample 3: Target Record with No Unique Key

Target Record

EXPERTISE (no unique key)

Set Relationships

EMP-EXPERTISE (owner EMPLOYEE, CALC key EMP-ID-0415)
SKILL-EXPERTISE (owner SKILL, CALC key SKILL-ID-0455)

Path

EMPLOYEE, EXPERTISE

Analysis

The EXPERTISE record type does not have a field that can uniquely identify a single occurrence. Both set relationships are mandatory/automatic, but because the EXPERTISE record type is stored VIA the EMP-EXPERTISE set, it makes more sense to choose that relationship, making the owner record, EMPLOYEE, the entry record in the primary path.

To define the table procedure specification

1. In the Entry Record tab, select Employee and EMP_ID_0415 as the unique key field. Click Next.
2. In the Path Record tab, select EXPERTISE. Click Target.
3. In the Auxiliary Record tab, select SKILL-ID-0455 as the unique key for the SKILL record.
4. In the Target Fields tab, select the appropriate columns and then save and generate the table procedure

Sample 4: Target Record Part of a Bill-of-Materials Relationship

Target Record

STRUCTURE (no unique key)

Set Relationships

REPORTS-TO (owner EMPLOYEE)
MANAGES (owner EMPLOYEE)

Path

EMPLOYEE, STRUCTURE

Analysis

Choose EMPLOYEE as the entry record with EMP-ID-0415 as the unique key. Select the MANAGES set relationship for the primary path, making the REPORTS-TO set relationship an auxiliary record.

The only twist here is that the unique keys for both relationships are the same field (EMP-ID-0415) and record type (EMPLOYEE). To avoid problems, you must change the SQL column name that Quick Bridge assigns. For example, you could type MANAGER for the SQL Column Name of the EMP-ID-0415 field in the Entry Record tab and SUBORDINATE in the SQL column name for the same field in the Auxiliary Record tab.

To define the table procedure specification

1. In the Entry Record tab, select Employee and EMP-ID-0415 as the unique key field. Change the SQL column name to something like MANAGER. Click Next.
2. In the Path Record tab, select the STRUCTURE record and the MANAGES set relationship. Click Target.
3. In the Aux Record tab, select EMP-ID-0415 as the unique field for the EMPLOYEE record in the REPORTS-TO set relationship. Change the SQL column name to something like SUBORDINATE.
4. In the Target Fields tab, select the appropriate columns and then save and generate the table procedure.

Sample SQL Statements

This section provides sample SQL Select statements, Update statements, and statements.

Select Statements

Now that we've defined the tables, let's see some possible ways to use them. To display information about an employee, the SQL Select statement for the EMPLOYEE table might look like this:

```
SELECT * FROM EMPDB.EMPLOYEE
    WHERE LAST_NAME_0415 = 'MARKEY';
```

This statement returns all columns for each row found whose last name equals Markey. The result table would also contain the foreign keys for the Department and Office tables. Therefore, you can also write a query that returns rows where employee Markey works for department 411:

```
SELECT * FROM EMPDB.EMPLOYEE
    WHERE LAST\-_NAME_0415 = 'MARKEY'
    AND DEPT_ID_0410 =  411;
```

So far you have only used data from the single table EMPDB.EMPLOYEE. Let's say you want to retrieve the employee Markey who works in the SHIPPING department. The EMPLOYEE table contains the foreign key to the DEPARTMENT table. You can join the two tables to form a query using the department name:

```
SELECT EMP.LAST_NAME_0415, EMP.FIRST_NAME_0415
    FROM EMPDB.EMPLOYEE AS EMP,
    EMPDEMO.DEPARTMENT AS DEPT
    WHERE LAST\-_NAME_0415 = 'MARKEY'
    AND EMP.DEPT_ID_0410 =  DEPT.DEPT_ID_0410
    AND DEPT.DEPT_NAME_0410 = 'SHIPPING';
```

Note: The Department table is from the EMPDEMO schema. You used a table from a non-SQL schema in this query.

Update Statements

The next example is an example of a searched UPDATE statement. Suppose you want to change the last name of employee 517. The Update statement might look like this:

```
UPDATE EMPDB.EMPLOYEE
    SET LAST_NAME_0415 = 'ALLEN'
    WHERE EMP_ID_0415 = 517;
```

The UPDATE statement can change any field in the target record type. UPDATE can also be used to change the foreign keys in the table. This allows the user to change the department or office in which the employee works. Let's say employee Allen moves from the Westwood office, office code WST, to the Mt. Laurel office, office code MTL. An UPDATE statement could be formulated to execute this change:

```
UPDATE EMPDB.EMPLOYEE
    SET OFFICE_CODE_0450 = 'MTL'
    WHERE EMP_LAST_NAME_0415 = 'ALLEN';
```

The table would make sure that there is an office (referential integrity check) with office code MTL before it updates the EMPLOYEE table for employee Allen. If more than one employee has Allen for a last name, the office code for each one is updated. From a table procedure standpoint, the program DISCONNECTs the employee from the OFFICE-EMPLOYEE relationship, then locates office code MTL, and connects the employee to the new office.

Assume that employee Markey does not work in an office, but works from home. The database would not have the employee associated with any office. You can formulate an UPDATE statement that nulls his office code:

```
UPDATE EMPDB.EMPLOYEE
    SET OFFICE_CODE_0450 = NULL
    WHERE EMP_LAST_NAME_0415 = 'MARKEY';
```

The table would now contain a NULL value in the OFFICE_CODE_0450 column of the EMPLOYEE table for all employees with the last name of Markey, which would, from a table procedure view, mean that the EMPLOYEE has been disconnected from the OFFICE-EMPLOYEE set relationship. Note that to change foreign keys values, the set must have an OPTIONAL disconnect option. Because of this rule, the STRUCTURE table does not allow the user to change or null the MANAGER foreign key.

Insert Statements

To UPDATE and SELECT data from a table, the data for that table must already have been added to the table. To add an employee to the EMPLOYEE table, formulate an INSERT statement like this:

```
INSERT INTO EMPDB.EMPLOYEE
    (OFFICE_CODE_0450,
    DEPT_ID_0410,
    EMP_ID_0415, ...)
    VALUES (WST, 411, 512, ...);
```

Because the DEPT-EMPLOYEE and OFFICE-EMPLOYEE set relationships use the Automatic connect option, you must supply the Office Code and Department ID when you add a row to the EMPLOYEE table. The INSERT statement to add a row to the STRUCTURE table would be very similar. Any foreign key that represents an Automatic set relationship (Aux or Path) is required and must be complete before the row can be inserted.

Delete Statements

To remove a row from a table, use the DELETE statement. The following DELETE statement deletes all employees with the last name of Markey:

```
DELETE FROM EMPDB.EMPLOYEE
    WHERE LAST_NAME_0415 = 'MARKEY';
```

This example and other examples that use last name to locate an employee are flawed, in that, all employees with the last name specified in the Where clause are processed by the SQL statement. The user should refine the Where clause to return only the precise row(s) to change. In the case of the EMPLOYEE table, the user could use the EMP_ID_0415 column, which for this table is *the* unique key.

Specify a DBName for Table Procedures

When CA IDMS accesses database records via SQL statements, as is the case for table procedures, the DBNAME value specified in your signon profile has no affect on which database is used. Instead, the DICTNAME value identified in a system DCUF statement or ODBC CONNECT statement identifies the SQL catalog that contains the information needed to locate the target database.

A generated table procedure contains a BIND RUN-UNIT statement that does not include a DBNAME parameter because it assumes that the SQL catalog points to the correct DBNAME. In some cases, this assumption means that the CA IDMS accesses the wrong target database or no database at all (IDMS error status 1491).

If you find that CA IDMS is accessing the wrong target database or no database for a specific table procedure, then:

- Map to the correct subschema in the DBTABLE statement, as described in the *CA IDMS Database Administration Guide*.
- Include a DBNAME in the BIND RUN-UNIT statement in the generated COBOL table procedure using either a hard-coded DBNAME or a GETPROF call to the IDMSIN01 subroutine to return the current DBNAME.

Authority Required to Generate Table Procedures

This section describes three different kinds of authorities.

Authorities to Obtain and Update Dictionary Records

Quick Bridge uses SQL queries to obtain and modify information in the dictionary. The following tables summarize the authorizations required to perform OBTAIN and UPDATE operations on dictionary records:

	GRANT TABLE Privilege		
Table Name	**SELECT**	**INSERT**	**DELETE**
SYSTEM.TABLE	X		
SYSDICT."MODULE-067"	X		
SYSDICT."OOAK-012"	X		
SYSDICT."S-010"	X		

	GRANT TABLE Privilege		
Table Name	**SELECT**	**INSERT**	**DELETE**
SYSDICT."SS-026"	X		
SYSDICT."SSR-032"	X		
SYSDICT."RCDSYN-079"	X		
SYSDICT."NAMESYN-083"	X		
SYSDICT."SDR-042"	X		
SYSDICT."ELEMSYN-085"	X		
SYSDICT."SSMR-068"	X		
SYSDICT."SSOR-034"	X		
SYSDICT."SSCR-070"	X		
SYSDICT."SR-036"	X		
SYSDICT."SRCD-113"	X		
CAAV.GETREC	X		
CAAV.MODULETEXT		X	X

Authorities to Add COBOL Source to the Dictionary

The Upload COBOL procedure uses table procedure CAAV.MODULETEXT to add the COBOL source code directly to the target dictionary. The program CAAVMTXT performs data retrieval and updates to the following records. To perform these operations, you must be authorized to issue OBTAIN and STORE DML statements against these dictionary records:

Record Name	**OBTAIN**	**STORE**
MODULE-067	X	X
TEXT-088	X	X
MODATTR-069	X	X

OOAK-012	X
CLASS-092	X
ATTRIBUTE-093	X

Authorities Needed When Executing CAAVLREC

Quick Bridge uses table procedure CAAV.GETREC to query the target dictionary. The program, CAAVLREC, obtains data from the records listed in the table below; to execute this program, you must be authorized to issue OBTAIN DML statements against the records:

Record Name	OBTAIN
S-010	X
SS-026	X
SSR-032	X
SSOR-034	X
SSMR-068	X
SSAM-066	X
SSA-024	X
SRCD-113	X
SOR-046	X

Sample Files

This section contains the EMPEXPT.TPS table procedure specification file that you saved in Chapter 2, "Creating Table Procedure Specifications." It also contains the files you created by generating the sample table procedure specification, excluding the COBOL program, EMPEXPT.COB, which is too long to include here. To view or print the COBOL program, open the file in a text editor of your choice.

EMPEXPT Table Procedure Specification Code (EMPEXPT.TPS)

This code represents the table procedure specification you defined using the Quick Bridge user interface. The actual code depends on the version of the program and might be slightly different from the one listed here:

```
[CAAV_14.1 CA IDMS SQL Quick Bridge]
[DATES CAAV="Nov  1 1995",TPS="Nov 16 1995"]
CONNECT="Employee"
DICTNAME=SYSDICT
CVNUMBER=0

[TABLE]
DIALOGTYPE=129
TABTEXT="SQL Options"
NAME=EMPLOYEE.EXPERTISE
XNAME=EMPEXPT
DELETE=ALL
UPDATE=YES
INSERT=YES
SCHEMA=EMPSCHM
VERSION=100
SUBSCHEMA=EMPSS01

[EPREC]
DIALOGTYPE=130
TABTEXT="Entry Record"
REC=DEPARTMENT,ORG-DEMO-REGION,56,I
SEQACC=SWEEP,AREA
DIRACC=CALC,CALC
UFLD=DEPT-ID-0410,DEPT_ID_0410,"9(4)"

[PATH]
DIALOGTYPE=132
TABTEXT="Path Record 1"
REC=EMPLOYEE,EMP-DEMO-REGION,116,I
SET=DEPT-EMPLOYEE,YES,AUTO
UFLD=EMP-ID-0415,EMP_ID_0415,"9(4)"

[PATH]
DIALOGTYPE=140
TABTEXT="Target Record"
REC=EXPERTISE,EMP-DEMO-REGION,8,I
SET=EMP-EXPERTISE,NO,AUTO

[TFIELDS]
DIALOGTYPE=16
TABTEXT="Target Fields"
RFLD=EXPERTISE-DAY-0425,EXPERTISE_DAY,"9(2)"
RFLD=EXPERTISE-MONTH-0425,EXPERTISE_MONTH,"9(2)"
RFLD=EXPERTISE-YEAR-0425,EXPERTISE_YEAR,"9(2)"
```

```
[AUX]
DIALOGTYPE=32
TABTEXT="Aux Record 1"
REC=SKILL,ORG-DEMO-REGION,76,I
SET=SKILL-EXPERTISE,NO,AUTO
UPDACC=CALC,CALC
UFLD=SKILL-ID-0455,SKILL_ID_0455,"9(4)"

[COLUMN]
DIALOGTYPE=64
TABTEXT="Work Columns"
CFLD=TOTAL_DOLLAR_AMT,"DECIMAL(6,2)"
CFLD=TRANS_DATE,"DATE"
CFLD=TRANS_TIME,"TIME"

[END]
```

Create Table Procedure Syntax (EMPEXPT.SQ1)

This example shows the CREATE TABLE PROCEDURE syntax, stored in file EMPEXPT.SQ1. You can upload the syntax and store it in the CA IDMS system catalog.

```
*CREATE TABLE PROCEDURE STATEMENT FOR EMPLOYEE.EXPERTISE,EMPEXPT;

CREATE TABLE PROCEDURE EMPLOYEE.EXPERTISE
    (DEPT_ID_0410        UNSIGNED NUMERIC(4,0),
     EMP_ID_0415         UNSIGNED NUMERIC(4,0),
     EXPERTISE_DAY       UNSIGNED NUMERIC(2,0),
     EXPERTISE_MONTH     UNSIGNED NUMERIC(2,0),
     EXPERTISE_YEAR      UNSIGNED NUMERIC(2,0),
     SKILL_ID_0455       UNSIGNED NUMERIC(4,0),
     TOTAL_DOLLAR_AMT    DECIMAL(6,2),
     TRANS_DATE          DATE,
     TRANS_TIME          TIME)
EXTERNAL NAME EMPEXPT
GLOBAL WORK AREA 2048;
CREATE UNIQUE KEY EMPEXPT_KEY
    ON EMPLOYEE.EXPERTISE
    (DEPT_ID_0410        ,
    EMP_ID_0415         );
```

Create Table Syntax (EMPEXPT.SQ2)

This is an example of the CREATE TABLE syntax, stored in file EMPEXPT.SQ2. You can load the table definition into an Ingres II database and use it to migrate and update data stored in a CA IDMS database.

```
*CREATE TABLE STATEMENT FOR EMPLOYEE.EXPERTISE,EMPEXPT;

CREATE TABLE EMPLOYEE.EXPERTISE
    (DEPT_ID_0410        UNSIGNED NUMERIC(4,0) NOT NULL,
     EMP_ID_0415         UNSIGNED NUMERIC(4,0) NOT NULL,
     EXPERTISE_DAY       UNSIGNED NUMERIC(2,0),
     EXPERTISE_MONTH     UNSIGNED NUMERIC(2,0),
     EXPERTISE_YEAR      UNSIGNED NUMERIC(2,0),
     SKILL_ID_0455       UNSIGNED NUMERIC(4,0) NOT NULL,
     TOTAL_DOLLAR_AMT    DECIMAL(6,2),
     TRANS_DATE          DATE,
     TRANS_TIME          TIME);
```

Add Program Syntax (EMPEXPT.SGN)

This syntax, stored in EMPEXPT.SGN, represents the system generation ADD PROGRAM statement that defines the table procedure COBOL program to the CA IDMS system. CA IDMS stores the syntax as a module in the dictionary with language type SYSGEN. o include it at system startup, use the INCLUDE statement, as described in *CA IDMS System Generation*.

```
*SYSGEN STATEMENT FOR EMPEXPT

ADD PROGRAM EMPEXPT
    CONCURRENT
    PROGRAM
    LOADLIB
    LANGUAGE IS COBOL
    NOMAINLINE
    NOPROTECT
    REENTRANT.
```

SQL Data Type Mapping and Defining SQL Columns as Date Columns

The following table gives the mapping of the SQL data types in the generated SQL DDL syntax files. IDMS SQL syntax is generated in the .SQ1 files, Ingres 1.x SQL is generated in the .SQ2 files and Ingres 2.x SQL is generated in the .SQ3 files.

Type	IDMS SQL	Ingres 1.x SQL	Ingres 2.x SQL
B	BINARY(length	BYTE(length)	BYTE(length)
C	CHAR(length)	CHAR(length)	CHAR(length)
DT	DATE	DATE	DATE
D	DECIMAL(precision,scale)	FLOAT8	DECIMAL(precision,scale)
UD	UNSIGNED DECIMAL(precision, scale)	FLOAT8	DECIMAL(precision,scale)
DP	DOUBLE PRECISION	FLOAT8	FLOAT8
F	FLOAT	FLOAT4	FLOAT4
G	GRAPHIC	CHAR	CHAR
I	INTEGER	INTEGER4	INTEGER4
LI	LONGINT	FLOAT8	DECIMAL(19,0)
SI	SMALLINT	SMALLINT	SMALLINT
N	NUMERIC(precision,scale)	FLOAT8	DECIMAL(precision,scale)
UN	UNSIGNED NUMERIC(precision,scale)	FLOAT8	DECIMAL(precision,scale)
R	REAL	FLOAT4	FLOAT4
T	TIME	CHAR(8)	CHAR(8)
TS	TIMESTAMP	CHAR(26)	CHAR(26)

Type represents the data type to be used for the corresponding SQL Column in the generated table and table procedure definitions. The type of a column is determined by Quick Bridge from the element definition in the nonSQL schema. Because date fields cannot be defined as such in a nonSQL schema, one can overwrite the type as derived from the nonSQL schema definition with "DT" to define an SQL column as a date.

The data type can be set to"DT in the same dialogs that allow you to edit the SQL Column Names. To change the data type of a column, make the Type column visible by dragging the SQL Column Names column to the left in the container.

There is no date, time or timestamp data type in a nonSQL schema definition. Dates are represented by fixed length elements that have various formats. In many cases, structures have separate elements for the year, month, day and optional separators are used.

To be able to map to a date SQL column, you must use a nonSQL schema and subschema in which the group element that represents the date is redefined as an elementary field such as PIC 9(6) or PIC 9(8). Quick Bridge will display the date field as a single record field name to be included. Setting the type to DT will result in generated tables and table procedure definitions with the appropriate DATE data types for the date fields. The generated COBOL program will contain comments that indicate where any necessary date conversion statements need to be added.

Messages Returned From a Generated Table Procedure

This section describes the messages you may encounter when you use a client/server query application, such as CA Visual Express, to obtain, update, or delete CA IDMS data, obtained by executing a generated table procedure:

38001--DML STATEMENT stmt-nbr RETURNED AN ERROR STATUS OF idms-status

Reason

An IDMS DML statement returned an unexpected error.

Action

Review the table procedure that was executed at the time of the error using the supplied DML statement number and IDMS status code.

38002--INVALID SQL-COMMAND-CODE

Reason

The value of the SQL-COMMAND-CODE parameter is invalid. One reason for this to happen is when a table procedure is out of sync with its table procedure definition.

Action

Make sure the table procedure program is in sync with the table procedure definition.

38003--UNIQUE KEY(S) FOR ROW NOT SUPPLIED

Reason

The INSERT function performed by the table procedure requires unique keys for all primary path records.

Action

Provide all unique keys required for this function and resubmit the SQL request.

Section

2000-INSERT-ROW

38004--UNIQUE KEY-n FOR ROW NOT FOUND

Reason

The function performed could not find a database record or database relationship that matches the data specified by the provided unique key fields. n specifies the key number with n=1 for the primary key (using the entry and path records) or n>=2 for the first, second, ... auxiliary record.

Action

Provide a set of unique keys that locate the database record or database relationship and resubmit the SQL request. For primary path relationships the key for any member record must be found within owner of the set relationship.

Section

2000-INSERT-ROW
2100-CONNECT-MANUAL-SETS
3200-UPDATE-RELATIONSHIPS
3210-CHANGE-PATH

38005--PRIMARY PATH NOT DISCONNECTABLE

Reason

The primary path relationship from the target record to its owner is a mandatory set relationship.

Action

None. Changing the foreign key for the primary path is not allowed.

Section

3100-DISCONNECT-REQ

38006--*set-name* DISCONNECT KEY-n NOT ALLOWED

Reason

The named auxiliary relationship from the target record to its owner is a mandatory set relationship. n>=2 specifies the key number with n=2 for the first auxiliary record.

Action

None. Changing the foreign key for the auxiliary relationship is not allowed.

Section

3100-DISCONNECT-REQ

38007--SOME, BUT NOT ALL KEY ELEMENTS OF RELATION NULLED

Reason

Some, but not all key elements of a relation are nulled.

Action

Correct the request: specify either values for all key elements or do not specify any values at all.

Section

3100-DISCONNECT-REQ

38008--SQL INSERT NOT IMPLEMENTED FOR THIS TABLE PROCEDURE

Reason

The table procedure does not support INSERT because the INSERT option has not been checked off in the SQL Option dialog.

Action

Do not use INSERT or update the table procedure and check off the INSERT option.

Section

2000-INSERT-ROW

38009--SQL UPDATE NOT IMPLEMENTED FOR THIS TABLE PROCEDURE

Reason

The table procedure does not support UPDATE because the UPDATE option has not been checked off in the SQL Option dialog.

Action

Do not use UPDATE or update the table procedure and check off the UPDATE option.

Section

3000-UPDATE-ROW

38010--SQL DELETE NOT IMPLEMENTED FOR THIS TABLE PROCEDURE

Reason

The table procedure does not support DELETE because the DELETE option has not been checked off in the SQL Option dialog.

Action

Do not use DELETE or update the table procedure and check off the DELETE option.

Section

4000-DELETE-ROW

Index

E

F

G

I

J

K

M

O

P

Q

R

S

T

U

W