IBM InfoSphere Optim for z/OS
Version 11 Release 3

# *Common Elements*

IBM InfoSphere Optim for z/OS
Version 11 Release 3

# *Common Elements*

**Note**
Before using this information and the product it supports, read the information in "Notices" on page 441.

**Version 11 Release 3**

This edition applies to version 11, release 3 of IBM InfoSphere® Optim for z/OS and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# About this publication

This document provides information on using the elements common to the components of Optim™.

For detailed information on each component, refer to the appropriate user manual.

**Note:** Certain aspects of an element common to all Optim components do not apply to all components. For example, all Optim components use Access Definitions. However, only Archive supports Archive Actions, which can be included in an Access Definition. Any features that do not apply to all installations of Optim are clearly noted.

# Chapter 1. Introduction

IBM® Optim for z/OS® manages enterprise data throughout every stage of the information life cycle. Optim enables you to assess, classify, subset, archive, store, and access enterprise application data.

Optim uses the relationships defined in the DB2® Catalog, where available, and supplements these relationships with those defined in the Optim Directory. Optim runs as a TSO/ISPF application and incorporates familiar ISPF commands. Optim handles any number of tables and any number of relationships, regardless of the complexity.

Optim helps you achieve these benefits with the following components: Access, Archive, Move, and Compare. You can use these Optim components for test data management, data privacy, data retention, application retirement, and data growth management. This manual describes the use of elements that are common to all Optim components.

## Test Data Management

The Optim test data management capabilities provide an efficient alternative to database cloning, allowing you to create development and testing environments that are sized appropriately. For information about the test data management functions of Optim, see the *Move User Manual*, the *Access User Manual*, and the *Compare User Manual*.

## Data Privacy

Data privacy is a licensed function of test data management.

For information about the general test data management functions of Optim, see the *Move User Manual*, the *Access User Manual*, and the *Compare User Manual*. Data transformations for privacy are accomplished through the use of Optim column maps. For information needed to transform data using a column map, see Chapter 5, "Column Maps," on page 155.

## Data Retention, Application Retirement, and Data Growth Management

You can use the archiving features in Optim to perform the following tasks:

- Isolate historical data from current activity and safely remove it to a secure archive.
- Access archived data easily, using familiar tools and interfaces.
- Restore archived data to its original business context when it requires additional processing.

## Optim Components

Your site may be licensed for one, all, or a combination of the following Optim components. The **Main Menu** provides release and copyright information for Optim.

- Access is used to browse and edit sets of relationally intact data in multiple database tables. This tool allows you to edit database data, review logical application paths, and browse data after testing applications to ensure that results are as expected. You can use Access to support rapid development of applications, analyze the structure of your database, and browse precisely-defined segments of relational data.
- Archive enhances database performance by facilitating the removal of infrequently referenced data. This component allows you to identify and archive sets of relationally intact data before removing selected data from your database. Archived data is indexed and stored. You can use Archive to browse, search, or restore selected subsets of archived data.

- Compare facilitates comparisons of sets of relationally intact data. This component allows you to compare data resulting from application tests to the original data, highlighting all differences. You can use Compare to rapidly analyze the effects of application software or modifications to it.
- Move facilitates the extraction and migration of sets of relationally intact data. Move allows you to create test databases that are referentially intact subsets of a production database. You can use Move to copy sets of related data to a file and transform the data as you migrate it to a test database.

Menu-driven prompt screens or panels are used to specify which data to obtain and how to handle that data. Intelligent screen handling allows simultaneous display of multiple tables, pop-up windows, context-sensitive online help, and tutorials.

## Common Elements and Facilities

Features common to all or most of the Optim components are discussed in this book. To carry out their functions, the Optim components rely on user-defined objects that supplement objects defined to the database (for example, tables, primary keys, relationships, stored procedures). These user-defined objects (collectively, Optim objects) are stored in the Optim Directory.

### Optim Objects

The Optim Directory is a set of database tables used by Optim to track processing status and store the objects needed for processing. Objects that are common to the Optim components include the following:

- **Access Definitions**
  An Access Definition identifies a set of related data to be processed by Optim. It references the database tables and their relationships, and provides criteria to select specific rows within tables.
- An Access Definition is required for an Archive or Extract Process and is sometimes used for a Compare or Restore Process, or an Access edit or browse session.
- **Primary Keys**
  Values in primary key columns uniquely identify each row in a database table. A primary key can be used to create an Optim relationship, and is required for a table that is changed by a Delete, Insert, or Restore Process or a table that is visited more than once in an Extract or Archive Process. A primary key is also required to enable the row selection (Point-and-Shoot) feature for an Access Definition or an Archive or Extract Process.
- Optim uses primary keys that are defined to the DB2 Catalog or, for Optim Move or Compare for IMS, VSAM, and Sequential Files, in an IBM IMS™ DBD. You can define Optim primary keys to supplement those in the database.
- **Relationships**
  A relationship is a defined connection between the rows of two tables that determines the parent or child rows to be processed and the order in which they are processed. A relationship can be defined in the DB2 Catalog, the Optim Directory, or to IMS.
- Optim uses relationships to determine the data to be retrieved from related tables. Generally, a relationship is needed in a process that uses an Access Definition. Using Optim, you can create or modify Optim relationships, browse DB2 relationships, or list IMS relationships.
- **Column Maps**
  A Column Map provides specifications needed to pair columns between two tables referenced in a Table Map. Also, a Column Map can be used to transform data, age dates in tables, and exclude one or more columns from processing.
- A Column Map is used for a single table Compare Process and can be referenced in a Table Map used for a Compare, Convert, Insert, Load, or Restore Process. (Rules for Column Maps used with Archive and Move are different from those for Compare. Features in a Column Map created with Archive or Move may not be available for Compare.)

- **Column Map Procedures**
  A Column Map Procedure contains either a complex expression (unstructured set of Lua statements) or a set of standard functions (structured set of Lua statements) used to transform or mask data.
- **Table Maps**
  A Table Map identifies and matches two tables or sets of tables in a process and can be used to exclude one or more tables from processing.
- A Table Map is required for a Compare, Convert, Insert, Load, or Restore Process. (Rules for Table Maps used with Archive and Move are different from those for Compare.)
- **Archive Collections**
  An Archive Collection is a mainframe object that contains one or more Archive Files. When an ODM connection is made using a collection name, the Archive Files in the collection are unioned and presented to the user as though a single Archive File was being accessed.
- For example, say you archive the customer database 50 times a year, and all 50 Archive Files are assigned to the same collection. Each Archive File has selection criteria for a single, unique state. When accessed individually, each Archive File holds only one state's worth of data. However, when accessed via the collection, data from all 50 states is available for retrieval. A collection can included any number of Archive Files, and a given Archive File may be included in more than one collection, but each Archive File can be included only once in a given collection.

**Note:** Legacy Tables, IMS Environment Definitions, and IMS Retrieval Definitions are available to a site licensed for *Move* or *Compare for IMS, VSAM, and Sequential Files*. These objects are discussed in *Move User Manual*, Definitions and *Compare for IMS, VSAM, and Sequential Files*.

## Utilities

Utilities common to the Optim components include the following:

- **Export/Import**
  The Export and Import Processes copy Optim Directory objects (e.g., Access Definitions, Table Maps, etc.) in the current DB2 subsystem to a dataset, so the objects can be imported to an Optim Directory in any DB2 subsystem.
- **Convert**
  A Convert Process transforms data in an Extract or Archive File. The converted file can be used as input to most Optim processes. You can use the Convert Process to mask sensitive data or to convert data to a Comma Separated Values format to be used with any application that supports CSV files.
- **Retry/Restart**
  You can retry an Insert or Restore Process when one or more rows are discarded during the process, or restart an Insert, Restore, or Delete Process if the process does not execute to completion.
- **Browse**
  During an Extract File, Archive File, or Compare File browse session or a Point-and-Shoot session, you can use the Browse facility to join data in related tables and use various techniques to manage the display while viewing the data.
- **Batch Utilities**
  Batch utilities allow you to maintain the Optim environment and automate Optim processes. Use maintenance utilities to retrieve information from the Optim Directory, maintain objects in the Optim Directory, migrate Optim objects, and manage Archive File entries and files. Use processing utilities to automate Optim processes, such as Archive, Restore, Delete, Search, and Extract. (See the *Batch Utilities Guide* for detailed information on using the batch utilities.)

## Options

Several types of options are available to manage the functions performed while using Optim. These options include

- User Options

- Editor Options
- Job Card and Print Options
- Compare Options (Compare only)
- Archive Options (Archive only)
- Legacy Options (*Move* or *Compare for IMS, VSAM, and Sequential Files* only).

**Note:** Site Options are only available to users who have authorized passwords. Site Options are discussed in the *Customization Guide*, Customize the Optim Site Options.

## Basic Screen Format and Handling

Optim emulates ISPF commands, screen format and handling, and provides an extensive set of primary and line commands. Many commands are comparable to ISPF commands.

When the syntax is identical to ISPF (e.g., UP and DOWN), the functions are identical. However, many commands, such as FIND, have been extended with operands and functionality to suit Optim.

The functioning of several primary commands differs depending on the current Optim activity. Information about these differences is included with the commands described in this manual. For detailed information about all Optim primary commands, see the *Command Reference Manual*, Primary Commands.

### Common Panel

Facilities that are common to all Optim components are documented in this manual. For the most part, figures in this manual present the panels as they appear when all Optim components are installed. The facilities available exclusively with Archive, Access, Compare, or Move are noted in this manual and documented in the appropriate user manual.

### Pop-up Windows

Optim incorporates a unique pop-up window facility that is independent of the current ISPF release. A pop-up window facilitates the current function, without terminating it.

Some pop-up windows provide selection lists. For example, when adding table names to an Access Definition, you can display a pop-up window with a list of available tables, and select the desired names directly from the list. When the pop-up window is terminated, the selected tables are automatically listed in the Access Definition and you can continue editing it.

For sites using DB2 version 8 or later, pop-up windows are also used to enter Long Object Names (LONs) of up to 128 characters for objects such as tables and creator IDs. Pop-up windows for LONs are displayed using the EXPAND command. See “Long Object Names (LONs)” on page 8 for further information.

The characters used to draw the box around a pop-up window are site⌂defined. User options allow you to change these characters. The figures in this manual show pop-up windows delimited by dashes and vertical bars.

### Function Keys

Function keys are handled as in ISPF. The keys provide a simple way to execute primary commands. When a function key is pressed, it is evaluated as if the command assigned to the key were typed in the primary command area of the screen. (See PF Keys in the *Command Reference Manual* for information on assigning values to function keys.)

The ISPF command PFSHOW is available to display the function key assignments on the screen. The ISPF command KEYS is available to list your current function key assignments. This list is modifiable.

Changes to the PF key assignments during an Optim session affect the assignments only while using Optim. The changes do not affect assignments for other ISPF applications.

Several functions are frequently assigned to function keys. The following are of note:

**HELP** Displays Help information for the active Optim panel. This information explains the purpose and use of the current panel. It may also list the commands available on the current panel. You can scroll certain Help panels, using the function keys F10 and F11. HELP is usually assigned to PF1.

**END** Process input and return to the previously displayed panel. END is usually assigned to PF3.

### Scrolling

All ISPF scrolling functions are supported. Vertical and horizontal scrolling is coordinated with the scroll value specified in SCROLL. This value can be specified as:

**blank** Cursor location determines the scroll amount.

**PAGE** Full page scroll so that the line or column following the last line or column on the current page is the first line or column of the next page.

**DATA** Full page scroll so that the last line or column on the current screen is the first line or column on the next screen.

**HALF** Half page scroll.

**n** Specific number of rows or columns to scroll.

**MAX** Depending on direction, either the first full screen of data or the last full screen of data is displayed.

### Scrolling LONs

Another type of scroll functionality is available for viewing Long Object Names (LONs) that exceed the display area on a given panel. Table names, for example, can consist of up to 128 characters, which is larger than the display area allocated for those names on Optim panels. In such cases, you can use the LEFT and RIGHT function keys (typically F10 and F11) to scroll through the entire LON within the display field. For example, if the Table Name area on a panel is 25 characters in length, only the first 25 characters of a 128-character LON are displayed. Press the RIGHT function key to display the next 25 characters, until the entire LON has been displayed; conversely, press the LEFT function key to display the previous 25 characters.

If you need to view the entire LON in a single viewing, you can use the EXPAND command to display the entire LON in a pop-up or LON panel. See “Long Object Names (LONs)” on page 8 for more information on entering and displaying LONs.

## Processing Order

The CANCEL, HELP, and RESET primary commands take precedence over all other processing and are performed first; otherwise, information is processed in the following order:

1. All editing is evaluated.
2. Line commands are processed.
3. Function key requests and primary commands are processed.

If editing introduces any syntax errors, all command processing is suspended and an appropriate error message is displayed. You must correct the error to resume processing.

A syntax error in a line command causes all command processing to be suspended with an appropriate error message. Correct or delete the line command in error to resume processing. If scrolling is performed to complete a block command specification or enter a destination, errors in the pending block command or Move/Copy command must be resolved prior to processing other line commands.

If a primary command is in error, all commands before the error are processed. All other processing is suspended and an appropriate error message is displayed. Correct or delete the faulty command to resume processing.

## Naming Conventions

In naming objects, you must avoid the use of DB2 reserved words.

Long Object Names (LONs) are supported for objects such as table names and creator IDs, as described in "Long Object Names (LONs)" on page 8.

A Retrieval Definition name consists of the Environment Definition name and the IMS DBD name. The maximum length for a DBD name is 8 characters. Names of other objects unique to Optim follow DB2 conventions. These names, and the maximum length of each, include:

**Access Definition base name**
12

**Column Map base name**
12

**Column Map Procedure ID**
8

**Column Map Procedure Name**
8

**Compare Definition base name**
12

**Environment Definition name**
8

**Group ID**
8

**Legacy Table base name**
18

**Map ID**
8

**Table Map base name**
12

**User ID**
8

**(Archive) Collection ID**
8

**(Archive) Collection Name**
12

## Delimited Identifiers

A DB2 delimited identifier is a sequence of one or more characters enclosed within quotation marks. Delimited identifiers are required when non-standard names are used, such as names with embedded spaces or periods. DB2 delimited identifiers can be included in the name of any DB2 object.

Also, delimited identifiers, with delimiters, can be supplied as the operands of primary commands. However, delimited identifiers are *not* supported for Access Definition, Table Map, and Column Map names.

The use of delimited identifiers may require more space than provided on a panel. If so, the entry is truncated and cannot be edited. Use line commands to delete the entry. When line commands are not available, the entry area is large enough to support the maximum width so you can edit the entry.

To conserve space, the delimiters are not included with the delimited identifier when the name is displayed in the heading of a panel. Delimiters are also omitted when delimited names are presented in selection lists.

## DB2 LIKE Syntax

Standard DB2 naming conventions are supported in Optim, including the ability to use DB2 LIKE syntax. Characters that have special meaning are:

% Represents any number of characters.

_ Represents any single character.

Use DB2 LIKE syntax for the operand on a primary command to display a matching selection list. For example, to obtain a selection list of tables with names that begin with the letters PST, enter:

LIST TABLES PST%.%

Note that when the '%' character is used in conjunction with the '_' character, the '_' is treated as a 'DB2 like' character, and not as a literal.

## File Names

Archive, Compare, and Move use external files to store information necessary for processes and to document those processes. For these files, you can provide the name explicitly by enclosing it in single quotes (‘ '), as shown in this example:

```
'SAMPLE.CONTROL.FILE'
```

Alternatively, you can provide a name without quotes so that the default prefix selected on the User Options panel is automatically added to the file name. For example, if you provide the file name

```
SAMPLE.CONTROL.FILE
```

and the current default prefix is FOPDEMO, the resulting file name is

```
FOPDEMO.SAMPLE.CONTROL.FILE
```

To obtain a selection list, specify an asterisk, alone or as the last character in the dataset name. For example, to obtain a selection list of names that begin with FOPDEMO.SAMP, specify:

```
'FOPDEMO.SAMP*'
```

## Long Object Names (LONs)

Optim includes support of up to 128 characters for LON-eligible objects. Shown here are the names of LON-eligible fields, followed by the maximum number of bytes allowed:

Creator IDs - 128
Column Names - 30
Table Names - 128
View Names - 128
Index Names - 128
Aliases - 128
Synonyms - 128
Relationship Names - 128
Correlation Names - 128
Stored Procedure Names - 128
User-Defined Function Names - 128
User-Defined Data Type Names - 128
Triggers - 128
Schema Names - 128
Storage Group Names - 128

LONs are allowed for DB2 objects only; they are not allowed for legacy tables, which do not support the use of Long Object Names.

Areas for LON-eligible names are identified on Optim panels by one or two arrows (> or >>). There are two methods of entering a LON: a separate LON panel or a LON pop-up. One arrow (>) means the LON panel is used, two arrows (>>) means the LON pop-up is used. In either case, the EXPAND command is used to enter the LON, described as follows:

1. Type the EXPAND command.
2. Position the cursor in the area in which you want to enter a LON and press **Enter** to display the LON panel or pop-up.

**Note:** If you use LONs frequently, you might want to assign a function key to the EXPAND command, as described in the *Command Reference Manual* under the heading PF Keys. You may then skip steps 1 and 2 and display the LON panel or pop-up by pressing the assigned PF key while the cursor is any LON-eligible field.

1. Type or paste the Long Object Name in the panel or pop-up.

**Note:** If you paste the LON from a file, be sure the cursor is in the first character position on the panel or pop-up to prevent LON from being formatted incorrectly. In the LON pop-up, each line cannot exceed 65 characters. In the LON panel, the first line cannot exceed 73 characters.

1. Use the END command to return to the panel, which will now display the first portion of the LON.

**Scrolling through a LON**

When only a portion of a LON is displayed on a panel, one or more arrows will appear at the end of the area to indicate that there is more data horizontally in either direction:

**> or >>**
there is more data after

**< or <<**
there is more data before

**<> or <<>>**
there is more in both directions

If a LON is displayed on a panel, you can scroll through it using the LEFT and RIGHT PF keys, typically assigned to F10 and F11. Use F10 to scroll to the LEFT through the LON, and use F11 to scroll to the RIGHT. In most cases, the entire LON may also be displayed and edited using the EXPAND command. In some cases, however, such as in display-only tables, you can only scroll through the LON.

**Note:** We recommend that you delete or modify all LONs using the LON panel or pop-up — ***do not*** edit the LON on the main panel, which can display only a portion of the overall LON — to avoid deleting a portion of a LON or incorrectly modifying a LON.

When a LON is truncated on a panel because of space limitations, a plus sign (+) at the end of the displayed portion of the LON indicates that only a portion of the LON is displayed.

## Sample Database

A sample database is distributed with Optim. The sample database is created during the installation of Optim. The sample database provides data for training and allows you to experiment with Optim without fear of disrupting your production database.

**Note:** A detailed description of the sample database is included in Appendix A, "Sample Database Tables and Structure," on page 401.

This database includes the following tables (names are prefixed with the Creator ID FOPDEMO):
- OPTIM_CUSTOMERS
- OPTIM_ORDERS
- OPTIM_DETAILS
- OPTIM_SALES
- OPTIM_FEMALE_RATES
- OPTIM_SHIP_INSTR
- OPTIM_ITEMS
- OPTIM_SHIP_TO
- OPTIM_MALE_RATES
- OPTIM_STATE_LOOKUP

**Note:** In some examples, the table names from the sample database are shown without the OPTIM_ prefix.

The database also includes the following Legacy Tables (for *Move* or *Compare for IMS, VSAM, and Sequential Files*only):
- DEPARTMENT
- VENDITEM
- EMPLOYEE
- VENDOR
- POSITION
- JOBCODE

**Note:** Relationships may have been added to the sample database at your site during training or other activities.

## Help and Tutorial Facility

Optim provides an online help facility that is available from any panel. Also, Option T TUTORIAL on the **Main Menu** invokes a tutorial that explains how to use Optim.

From each panel, use the HELP command (usually assigned to PF1) to obtain an overview for the panel. If a numbered list of commands available from that panel is included, enter the listed number associated with the command to obtain command information and syntax.

When an error condition occurs during a session, a short message is displayed on the panel.

## Main Menu

When you invoke Optim, the **Main Menu** is displayed, as in the following figure. The options available on the **Main Menu** will vary based on which Optim components are installed at your site. Any option marked with an asterisk is not available for your installation.

```
---------------------------- IBM's InfoSphere Optim ----------------------------
OPTION  ===>

0  OPTIONS         - Site and User Options      SQLID  ===> PSTDEMO
1  BROWSE TABLE    - Browse a DB2 Table         SUBSYS ===> TDB2
2  EDIT TABLE      - Edit a DB2 Table           LOCATION ===>
3  BROWSE USING AD - Browse DB2 Tables Using Access Definition
4  EDIT USING AD   - Edit DB2 Tables Using Access Definition
5  ADS             - Create or Modify Access Definitions
6  DEFINITIONS     - Maintain InfoSphere Optim Definitions (Keys, Maps, ...)
7  MIGRATION       - Data Migration - Extract, Insert, Update, ...
8  COMPARE         - Compare Two Sets of Data
9  ARCHIVE         - Archive and Restore Data

T  TUTORIAL        - Information About IBM's InfoSphere Optim
C  CHANGES         - Changes from Prior Release(s)
X  EXIT            - Terminate Product Use
P  LICENSING       - Product Licensing Modification
```

*Figure 1. Main Menu*

### Panel Options

To select an option, type the corresponding one-character identifier. Refer to the User Manual for the indicated Optim component for more information. The options are:

**0** OPTIONS

Specify Optim options, including site and user options. Administrator privileges are required to define site options. For details, see Chapter 12, “Options,” on page 371.

**1** BROWSE TABLE

Browse the data in a DB2 table. This facility is documented in the *Access User Manual*, Editing and Browsing DB2 Data.

**2** EDIT TABLE

Edit the data in a DB2 table. This facility is documented in the *Access User Manual*, Editing and Browsing DB2 Data.

**3** BROWSE USING AD

Browse DB2 data defined by an Access Definition. This facility is documented in the *Access User Manual*, Using an Explicit Access Definition.

**4** EDIT USING AD

Edit DB2 data defined by an Access Definition. This facility is documented in the *Access User Manual*, Using an Explicit Access Definition.

**5** ADS

Create and maintain Access Definitions. For more information, see Chapter 2, "Access Definitions," on page 21.

**6** DEFINITIONS

Invoke the Choose a Definition Option menu, described in "Choose a Definition Option." Options from this menu allow you to define or edit Access Definitions, Optim primary keys, Optim relationships, Table Maps, Column Maps, and Archive Collections, or to invoke utilities to export or import these objects.

**7** MIGRATION

Perform the Move or Compare processes. These facilities are documented in the *Move User Manual*, Data Migration, and the *Compare User Manual*, Extract Data.

**8** COMPARE

Compare one set of tables with another and browse the results. This facility is documented in the *Compare User Manual*, Compare Process.

**9** ARCHIVE

Perform the Archive processes for archiving data, browsing and searching the archives, and restoring archived data. This facility is documented in the *Archive User Manual*, Session Overview.

**T** TUTORIAL

Display the online Tutorial.

**C** CHANGES

Display a list of enhancements for the current release.

**X** EXIT

Terminate Optim.

**P** LICENSING

Display a list of Optim components and their releases. The status for each component is identified as "In Evaluation: n Days Left" or "Not Installed." Administrator privilege is required to enable or disable a component. This facility is documented in the *Customization Guide*, Enable and Disable Products.

## Choose a Definition Option

Use the **Choose a Definition Option** menu to create and maintain primary keys, relationships, Column Maps, Table Maps, and Archive Collections stored in the Optim Directory, and display primary keys and relationships from the DB2 Catalog.

When you select Option 6 Definitions on the **Main Menu**, the **Choose a Definition Option** menu is displayed, as in the following figure. This menu also allows you to export objects defined in the Optim Directory of one DB2 subsystem and import them to a Directory in another DB2 subsystem.

```
------------------------- Choose a Definition Option -------------------------
OPTION ===>
                                              SQLID  ===> FOPDEMO
1  PRIMARY KEYS    - Maintain Primary Keys     SUBSYS ===> TDB2
2  RELATIONSHIPS   - Maintain Relationships    LOCATION ===>
3  COLUMN MAPS     - Maintain Column Maps
4  TABLE MAPS      - Maintain Table Maps
5  ADS             - Maintain Access Definitions
6  LEGACY TABLES   - Maintain Legacy Tables for Non-DB2 Data
7  IMS ENVIRONMENT - Maintain IMS Environment Definitions
8  IMS RETRIEVAL   - Maintain IMS Retrieval Definitions
9  COLLECTIONS     - Maintain Archive Collections
A  PROCEDURES      - Maintain Column Map Procedures

E  EXPORT          - Export Optim Object Definitions
I  IMPORT          - Import Optim Object Definitions
```

*Figure 2. Choose a Definition Option*

## Panel Options

The available options are:

**1 - PRIMARY KEYS**

Primary keys are columns that contain values that uniquely identify each row in a table.

Use Option 1 to create, modify, or delete primary keys in the Directory. You can also browse DB2 and IMS primary keys. See Chapter 3, “Primary Keys,” on page 115 for detailed information about this option.

**2 - RELATIONSHIPS**

Relationships are the sets of columns from each of two tables used to define a correspondence between the tables.

Use Option 2 to create, modify, or delete Optim relationships. You can also browse DB2 relationships and list IMS relationships. See Section 4. Relationships for detailed information about this option.

**3 - COLUMN MAPS**

Column Maps are used by Optim to map source columns to destination columns, or to transform the data for a destination column as part of a Restore, Insert, Load, or Convert Process. Column Maps are also used to exclude columns from participation in a process.

Use Option 3 to create, modify, or delete a Column Map. See Chapter 5, “Column Maps,” on page 155 for detailed information about this option.

**4 - TABLE MAPS**

Table Maps are used by Optim to map the source tables to their corresponding destination tables, so that tables with different names in the source and destination can be mapped and tables in the source can be excluded from the process.

Use Option 4 to create, modify, or delete a Table Map. See Chapter 7, “Table Maps,” on page 249 for detailed information about this option.

**5 - ADS**

Access Definitions are used by Optim to specify the related data to process. You can specify the set of tables, selection criteria, relationships, and other criteria to define the desired set of data.

Use Option 5 to create, modify, or delete an Access Definition. This option is the same as Option 5 ADS on the **Main Menu**. For information about creating or modifying Access Definitions, see Chapter 2, “Access Definitions,” on page 21.

**6 - LEGACY TABLES**

Option 6 is displayed only if *Move* or *Compare for IMS, VSAM, and Sequential Files*is installed. A Legacy Table allows you to incorporate legacy data into Move or Compare processes. The Legacy Table describes the legacy data and maps it to a DB2 table format.

Use Option 6 to create, modify, or delete a Legacy Table. For information about this option, see the *Move User Manual*, Legacy Tables or *Compare for IMS, VSAM, and Sequential Files*.

**7 - IMS ENVIRONMENT**

Option 7 is displayed only if *Move* or *Compare for IMS, VSAM, and Sequential Files*is installed. An IMS Environment Definition allows you to define the information needed to access IMS data.

Use Option 7 to create, modify, or delete an IMS Environment Definition. For information about this option, see the *Move User Manual*, IMS Environment Definition or *Compare for IMS, VSAM, and Sequential Files*.

**8 - IMS RETRIEVAL**

Option 8 is displayed only if *Move* or *Compare for IMS, VSAM, and Sequential Files*is installed. An IMS Retrieval Definition provides the information necessary to dynamically process IMS data.

Use Option 8 to create, modify, or delete an IMS Retrieval Definition. For information about this option, see the *Move User Manual*, IMS Retrieval Definition or *Compare for IMS, VSAM, and Sequential Files*.

**9 - COLLECTIONS**

An Archive Collection is an object that references one or more Archive Files. When an ODM connection is made using a collection name, the data in the Archive Files is unioned and presented to the user as though a single Archive File was being accessed.

Use Option 9 to create, modify, or delete an Archive Collection. For information about this option, see the *Archive User Manual*, Archive Collections.

**A - PROCEDURES**

A Column Map Procedure contains either a complex expression (unstructured set of Lua statements) or a set of standard functions (structured set of Lua statements) used to transform or mask data.

**E - EXPORT**

The Export Process is used to copy objects specific to Optim from the Optim Directory and store them in an external file. This file may then be used by the Import Process to add the objects to an Optim Directory in another DB2 subsystem.

Use Option E to export objects from the Optim Directory. See “Use the Export Process” on page 279 for detailed information about this option.

**I - IMPORT**

The Import Process is used to import previously exported Optim objects. The imported objects are stored in the Optim Directory. You can also import primary keys and relationships described in CREATE TABLE and ALTER TABLE JCL statements, allowing you to import definitions stored in an external data modeling tool.

Use Option I to import objects to the Optim Directory. See “Use the Import Process” on page 286 for detailed information about this option.

## Panel Prompts

Using prompts on the **Main Menu** or the **Choose a Definition Option** menu, you can change the SQLID, DB2 subsystem, or remote location. This is especially useful for accessing the objects for a specific database, or when exporting and importing, to connect to the desired target database. Values for prompts are profiled.

**SQLID**
The current SQLID. Modify this value to connect using a different SQLID.

**SUBSYS**
The current DB2 subsystem. Modify this value to connect to a different DB2 subsystem.

When connecting to a remote subsystem, this value should be the local subsystem where the remote location is defined.

**LOCATION**
The remote location. This prompt is displayed if remote access is available. Specify a value to connect to a remote DB2 subsystem. You can use a percent sign (%) to obtain a selection list of available locations. If the connection fails, the Optim session is restarted and the **Main Menu** is redisplayed.

**Note:** If you leave this prompt blank, the local subsystem is assumed.

## Object Selection List Functions

From a selection list of Optim objects (e.g., Access Definitions), use the Select line command, S, to choose an object.

In the following figure, the formatted Access Definition selection list is displayed and the last definition on the list, GRP.USER.ADSAMPLE, is selected.

```
------------------------ Select Access Definitions --------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 of 8

    ----- Access Definition ------ ------ Last Modified -------
Cmd  Group     User       Name        By           Date
--- -------- -------- ------------ -------- ------------------
    ****************************** TOP ***********************************
___ ADMIN    JAA      EMPL01       ALLEGRA  2005-09-21 16.36.59
___ ADMIN    JAA      PAYROLL      LISAC    2005-09-22 09.51.12
___ DVLMT01  FOPDEMO  SAMPLTST     DCOHEN   2005-10-23 19.12.23
___ DVLMT01  FOPDEMO  TEST04       DCOHEN   2005-10-13 14.52.49
___ DVLMT02  FOPDEMO  TEST05       KEBLERD  2005-10-06 16.37.59
___ DVLMT02  FOPDEMO  TEST06       KEBLERD  2005-10-13 14.52.49
___ GRP      USER     ADSAMP       SENTNER  2005-11-02 13.23.14
S__ GRP      USER     ADSAMPLE     ALLEGRA  2005-10-26 12.14.39
     ***************************** BOTTOM *********************************
```

*Figure 3. Select an Optim Object*

### Selection List Line Commands

The line commands for a selection list of Optim objects are:

**S** Select an object for processing.

**D** Delete an Optim object. After deleting, "*DELETED" is displayed in the selection list.

**C** Copy an Optim object to create a new Optim object. After copying, "*COPIED" is displayed in the selection list.

**R** Change the name of an existing Optim object. After renaming, "*RENAMED" is displayed in the selection list.

**AT** Modify the attributes of a Optim object. The Object Attributes panel allows you to edit the description and security status of an Optim object.

The description is 1 to 40 characters. A user option determines whether the description is displayed on selection lists. (See "User Options" on page 372 for more information.)

**I** Display information about an Optim object.

## Copy an Optim Object

To copy an Optim object from a selection list, use the C line command next to the name of the object to be copied. The following figure shows the Copy Access Definition panel.

```
+-------Copy Access Definition-------+
|                                     |
| Existing Name: GRP.USER.ADSAMPLE    |
|                                     |
| New Name:                           |
|                                     |
|    Group ===> GRP                   |
|    User  ===> USER                  |
|    Name  ===> ADSAMPLE              |
+-------------------------------------+
```

*Figure 4. Copy an Optim Object*

The **Copy** panel displays the name of the object and prompts for a name for the new, copied object. After you enter the new name, press ENTER to copy the object under the specified name.

## Rename an Optim Object

To rename an Optim object from a selection list, use the R line command next to the name of the object to be renamed. The following figure shows the Rename Access Definition panel.

```
+------Rename Access Definition------+
|                                     |
| Existing Name: GRP.USER.ADSAMPLE    |
|                                     |
| New Name:                           |
|                                     |
|    Group ===> GRP                   |
|    User  ===> USER                  |
|    Name  ===> ADSAMPLE              |
+-------------------------------------+
```

*Figure 5. Rename an Optim Object*

The **Rename** panel displays the current name of the object and prompts for a new name. After you enter the new name, press ENTER to rename the object.

## Object Attributes

To modify the description and security status attributes of an Optim object from a selection list, use the AT line command for the object. The description and security status are specified on the **Object Attributes** panel.

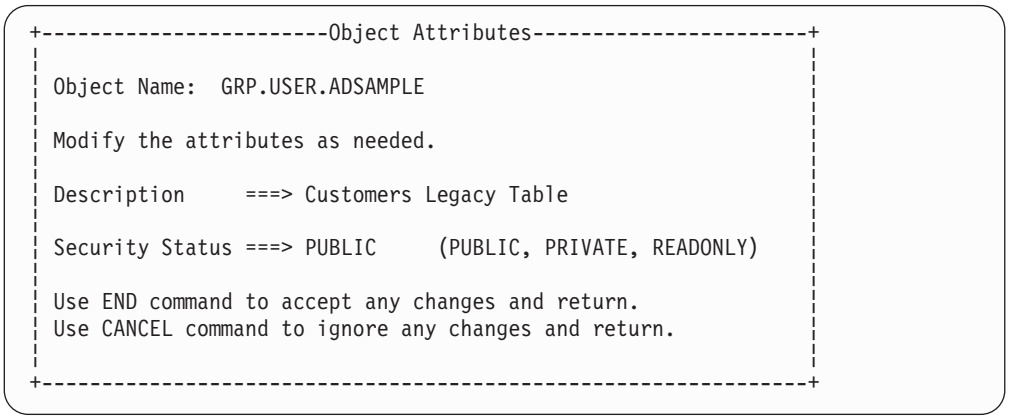



*Figure 6. Object Attributes*

The Object Attributes panel provides a 40-character area to display and edit the description. An 8-character area is also available to specify one of the following security statuses:

**Public** Anyone can edit and use.

**Private**
Only the owner can edit and use.

**Readonly**
Anyone can use, only the owner can edit.

A site option determines whether security status is available. If it is not available, **Security Status** is not displayed. For additional information about the Object Attributes panel, see "Specify Description and Security Status" on page 380.

## Scrolling

You can use standard ISPF scrolling facilities to scroll a selection list.

## Primary Commands

The following primary commands are available for a selection list of Optim objects. For detailed information about these commands, see the *Command Reference Manual*, Primary Commands.

BOTTOM
CANCEL
DOWN
END
FIND
LIST TABLES
LOCATE
OPTIONS
RESET
RFIND
SELECT
SHOW
SORT
TOP
UP

**Note:**

- FIND locates a character string anywhere in the list. RFIND repeats the FIND operation. (This command is usually assigned to PF5.)
- LIST TABLES is available for selection lists of primary keys or relationships only.
- LOCATE locates and scrolls to an object name that matches or is greater than the search value.
- SORT arranges the list in ascending or descending order by values under a column heading (such as SORT DATE).
- SHOW limits the selection list to objects for which a specific value is displayed.

## Double-Byte Character Set Support

Optim processes Double-Byte Character Set (DBCS) data and provides DBCS users with all Optim capabilities that are available to Extended Binary Coded Decimal Interchange Code (EBCDIC) users.

Optim DBCS support assumes EBCDIC DBCS data. EBCDIC DBCS requires exactly two bytes for every DBCS character. Optim DBCS support assumes a single DBCS language, including pure DBCS, and Mixed Single-Byte Character Set (SBCS) and DBCS.

In mixed character strings (strings that contain both SBCS and DBCS characters), two special control characters indicate the start and end of a DBCS substring. Shift Out (X'0E') indicates the start and Shift In (X'0F') indicates the end of the DBCS substring.

DB2 translates all DB2 character data between the internal DB2 table Coded Character Set Identifier (CCSID) and the external application (Optim) CCSID. All data that Optim processes remains in the external Optim CCSID.

**Note:** To eliminate problems with round-trip character translation, the Optim installer should ensure that the external Optim CCSIDs match the internal DB2 CCSIDs for DBCS data.

Optim Archive and Extract Files preserve the encoding scheme and CCSID. Optim warns users if incompatible CCSIDs could cause problems in storing data. Site and User options (“Allow Mismatched CCSIDs”) indicate the action Optim should take when the CCSID of a source column does not match that of a target column and when the CCSID of the terminal does not match that of the DB2 subsystem.

### DBCS Data Types

Optim supports the following DBCS data types. The conversion and mapping rules for the DBCS data types are the same as the corresponding DB2 rules.

- **GRAPHIC** - A fixed-length data type used to store a graphic string, that is, a string consisting of double-byte EBCDIC characters that are not stored with Shift Out and Shift In characters. The maximum length of a GRAPHIC string is 254 bytes.
- **VARGRAPHIC/LONGVARGRAPHIC** - A varying-length data type used to store a variable-length graphic string. The maximum length of a VARGRAPHIC/LONGVARGRAPHIC string is 32704 bytes.
- **CHAR or CLOB defined with “FOR MIXED DATA”** - Used to store mixed data, that is, Multi-byte Character Set (MBCS) data (the MIXED=YES DB2 install option is required). Shift Out and Shift In characters are required to identify the double-byte data in a mixed character string.
- **DBCLOB** - A double-byte character large object. The maximum length of a DBCLOB is 1,073,741,823 DBCS characters.

### Functions that Support DBCS Data

Optim supports DBCS data in the following functional areas:

- The Optim ISPF interface accepts and displays DBCS data.

- The Optim Directory accepts DBCS characters in object names (for example, column names, table names, Access Definition names, Compare Definition names, relationship names). When DBCS characters are used for an object name, they must be enclosed in double quotes.
- Keywords, commands, and control statements remain in EBCDIC SBCS. Batch Optim Directory control statements (for example, EXPORT, IMPORT) support DBCS characters.
- Access displays and allows entry of DBCS data.
- Archive and Extract processes save and restore DBCS data in Archive and Extract Files. (Archive and Extract Files save data in the external (Optim) application CCSIDs.)

  **Note:** To eliminate problems with round-trip character translation, the Optim installer should ensure that the external Optim CCSIDs match the internal DB2 CCSIDs for DBCS data.
- Compare supports DBCS data.
- Browse/Edit mode presents DBCS data and allows insert and updates. It also supports DBCS data in Selection Criteria and SQL WHERE clauses.
- All Column Map functions support DBCS literals and SUBSTR operations with DBCS data.
- Optim checks for data type compatibility between source and destination columns during Insert, Restore, and Convert processes.
- Functions that display table names, column names, and relationships support DBCS data.
- Export/Import and substitution variables support DBCS data.
- Process reports accommodate DBCS data.

### Definitions

The following definitions explain terms used in this section:

**Coded Character Set Identifier (CCSID)**
A 16-bit number that uniquely identifies a coded representation of graphic characters. It designates an encoding scheme identifier and one or more pairs that consist of a character set identifier and an associated code page identifier.

**Double-Byte Character Set (DBCS)**
A set of characters, which are used by national languages such as Japanese and Chinese, that have more symbols than can be represented by a single byte. Each character is 2 bytes in length.

**Graphic String**
A string consisting of double-byte EBCDIC characters that are not stored with Shift Out and Shift In characters.

**Multi-Byte Character Set (MBCS)**
A character set that represents single characters with more than a single byte. UTF-8 is an example of an MBCS. Characters in UTF-8 can range from 1 to 4 bytes in DB2.

**Single-Byte Character Set (SBCS)**
A set of characters in which each character is represented by a single byte.

## ASCII and Unicode Support

While the Optim plan must be bound with an EBCDIC encoding scheme, Optim is still able to process data in ASCII and Unicode DB2 tables. When Optim is reading data from an ASCII or Unicode DB2 table, DB2 will convert characters from the table's encoding scheme to the target Optim EBCDIC encoding scheme. When Optim is writing data to an ASCII or Unicode DB2 table, DB2 will convert characters from the Optim EBCDIC encoding scheme to the target table's encoding scheme.

Characters that cannot be represented in the target encoding scheme will not be preserved.

## Mismatched CCSIDs

A discrepancy in the Coded Character Set Identifier (CCSID) can lead to unexpected data conversion, affecting any characters that do not map to the same code point in two CCSIDs.

Subsequent behavior is determined according to the value of the **Allow Mismatched CCSIDs** option. Refer to "User Options" on page 372 for more information regarding this option.

Optim detects and reports the following discrepancies in CCSIDs:

- The CCSID of the source column does not match the CCSID of the target column.
- The TSO terminal does not match the CCSID of the DB2 subsystem where Optim is running.

Possible reasons why Optim may report a CCSID mismatch when you believe the CCSIDs should match include:

- Optim cannot load module DSNHDECP because the SDSNEXIT and SDSNLOAD DB2 libraries are not allocated and available to the Optim environment.
- The SDSNEXIT and SDSNLOAD DB2 libraries that are allocated and available to the Optim environment do not match the DB2 subsystem to which Optim is connecting.
- The order of DB2 library concatenation has been reversed. The SDSNEXIT library should be concatenated ahead of SDSNLOAD.

# Chapter 2. Access Definitions

An Access Definition defines a set of related data to be processed by Optim.

Use an Access Definition to select a set of related data that

- Archive copies to an Archive File or restores to a database.
- Move or Compare copies to an Extract File for processing.
- Access displays for browsing or editing.

The components of an Access Definition are

- A list of tables or views (i.e., the Table List).
- Selection criteria in the form of WHERE predicates, substitution variables, Point-and-Shoot, and or row list designations.
- Archive Actions to be performed before or after a row is archived, deleted, or restored.
- Processing criteria in the form of row limits and selection factors.
- A list of relationships among the listed tables and instructions for traversing them.
- Optional instructions for displaying data for Point-and-Shoot, browsing, or editing.
- Parameter settings to establish how the Access Definition is applied and to limit modifications to it during use.

In addition, the Access Definition editor includes several features to help you analyze and tune processing.

- Show Steps lets you analyze a process before running it to ensure that the data processed and the traversal paths used to process the data are as you expect.
- The Relationship Index Analysis feature analyzes DBMS indexes for selected relationships in the Access Definition, allowing you to create needed indexes to enhance performance.

You can define a temporary or permanent Access Definition for processing data. A permanent Access Definition is stored in the Optim Directory. For an Extract or Archive Process, a temporary Access Definition is discarded after the data is extracted. For a Compare Process, a temporary Access Definition is a "LOCAL" definition stored with the current Compare Definition, and it is available only for that Compare Definition.

## Creating an Access Definition

A series of panels guides you through the process of creating or modifying an Access Definition. These panels are referred to collectively as the Access Definition editor. You can invoke the Access Definition editor independently, by selecting Option 5 (or 6.5) ADS from the **Main Menu**, or while preparing to execute an Archive, Compare, or Extract Process. (Each process is invoked from the **Main Menu**. For details about each process option, see the appropriate *User Manual*.)

The features and methods used to create an Access Definition are the same as those used to modify an existing Access Definition. You can initiate the creation of an Access Definition by providing a name that is not in use when displaying the Access Definition editor. Alternatively, you can edit an existing Access Definition in the Access Definition editor and save it under a new name.

## Consider Purpose

When creating an Access Definition, it is important to consider the purpose of the Access Definition and who is to use it. An Access Definition can be used with Archive to archive data, with Access to browse or

edit data, with Move to extract data, or with Compare to select data for comparison. However, if Access Definitions are shared among the Optim components, care must be taken to avoid changes that inadvertently affect other users of Optim. For example, if an Access user modifies an Access Definition established by a Move user, the Move user might then extract an unexpected set of data.

In addition, an Access Definition used by Move, Compare, or Archive does not allow duplicate references to a table, whether direct or indirect (for example, a reference to a view, synonym, or alias for a listed table). Access permits duplicate references.

## Select an Access Definition

Optim allows you to invoke the Access Definition editor and create or edit an Access Definition, without having to initiate a process. Moreover, you can save the Access Definition in the Optim Directory for later use with an online or batch process.

Select Option 5 ADS from the **Main Menu** or Option 5 ADS from the **Choose a Definition Option** menu (accessed by selecting Option 6 DEFINITIONS on the **Main Menu**) to display the Choose an Access Definition panel. (For information about the **Choose a Definition Option** menu, see "Choose a Definition Option" on page 11.)

```
------------------------- Choose an Access Definition -----------------------
Command ===>
                                            SQLID  ===> FOPDEMO
Access Definition:                          SUBSYS ===> DSNA
  Group ===> GRP                            LOCATION ===>
  User  ===> USR
  Name  ===>

Use '_' for DB2 LIKE character ===> NO    (Y-Yes, N-No)


To limit selection list to Access Definitions with certain start tables, enter
the start table name.  A wild card is allowed at the end of each part.

  Start Table Creator ID ===>                                        >
  Start Table Name       ===>                                        >
```

*Figure 7. Choose an Access Definition*

### Panel

The prompts on the Choose an Access Definition panel are:

**SQLID**
The current SQLID. Modify this value to connect using a different SQLID.

**SUBSYS**
The current DB2 subsystem. Modify this value to connect to a different DB2 subsystem.

When connecting to a remote subsystem, this value should be the local subsystem where the remote location is defined.

**LOCATION**
The remote location. This prompt is displayed if remote access is available. Specify a value to connect to a remote DB2 subsystem. You can use a percent sign (%) to obtain a selection list of available locations. If the connection fails, the Optim session is restarted and the Main Menu is redisplayed. If you leave this prompt blank, the local subsystem is assumed.
The Choose an Access Definition panel does not include prompts for SQLID, SUBSYS, and LOCATION when displayed from the **Choose a Definition Option** menu. You must return to the previous panel to change these entries.

**Access Definition:**
Together, Group, User, and Name make up the fully qualified name of the Access Definition. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Group** The Group ID for the Access Definition. Enter 1 to 8 characters. The default is the previously entered value, or the TSO ID of the current user if a **Group** has never been specified.

**User** The User name for the Access Definition. Enter 1 to 8 characters. The default is the previously entered value, or the DB2 SQLID for the current user if a **User** has never been specified.

**Name** The Name of the Access Definition. Enter 1 to 12 characters.

**Use '_' for DB2 LIKE character**
This entry determines the use of the underscore ( _ ) character. Type **Y** if the underscore is used as a DB2 LIKE character, or **N** if it is used literally as part of the name.

**Note:** When the '_' character is used in conjunction with the '%' DB2 LIKE character, the '_' is treated as a 'DB2 like' character, and not as a literal.

For example, depending upon the use of the underscore character, A_B is a three-character name containing the characters 'A_B', as entered, or a three-character name that begins with "A" and ends with "B" with any valid character in the middle.

**Start Table**
You can list Access Definitions that reference a specific Start Table by specifying Start Table criteria. Enter an explicit value, wild card (%), or blanks for these prompts in any combination.

**Creator ID**
The Creator ID for the Start Table. Enter 1 to 128 characters. The default is the previously entered value.

**Table Name**
The Name for the Start Table. Enter 1 to 128 characters. The default is the previously entered value.

**Note:** See "Long Object Names (LONs)" on page 8 for information on entering Creator IDs and Table Names of up to 128 characters.

## Specify Name Explicitly

When you enter a fully qualified name explicitly, you are presented with the Select Tables/Views for AD panel. If the explicitly named Access Definition exists, you can modify it and save it under the same or a different name. If the explicitly named Access Definition does not exist, you can enter the desired information and save the newly defined object.

## Obtain Selection List of Names

You can enter DB2 LIKE syntax or leave one or more prompts blank to obtain a selection list of Access Definitions. For example, you can obtain a selection list of all Access Definitions, regardless of the **Group**, **User**, or **Name** by leaving those three prompts blank.

To obtain a selection list of all Access Definitions in any **Group** beginning with 'A' for the **User** identified as JAA, you would specify:

Group A%
User JAA
Name blank or %

If no Access Definitions match the criteria, the message “NO MATCHES” is displayed.

### Start Table Criteria

You can further limit an Access Definition selection list by entering Start Table criteria. Specify a fully-qualified Start Table name or use DB2 LIKE syntax to obtain a selection list of Access Definitions that reference a Start Table that matches the specified criteria.

If no Access Definitions match the Start Table criteria, the message “START TABLE NOT FOUND” is displayed.

### Subsequent Use

The specified **Group**, **User**, and **Name** values are profiled and displayed the next time you invoke this panel. You can change any displayed value.

### Primary Commands

Use END or CANCEL to return to the main menu or ENTER to display the next panel, whether selection list or Access Definition editor.

## Access Definition Selection List

A selection list of Access Definitions based on the entries in **Group**, **User**, and **Name** is displayed on the Select Access Definitions panel. The names in the selection list appear in alphabetical order by **Group**, **User**, and **Name**.

Use the **S** line command to select one or more Access Definitions for modification. You can also use the SELECT command to select a single Access Definition, by providing the fully qualified name as an operand. The following figure shows the formatted list. In the example, the Access Definition GRP.FOPDEMO.AD is selected, as indicated by the S in the Cmd column for that AD.

```
------------------------ Select Access Definitions --------------------------
Command ===>                                                Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 of 4

    ----- Access Definition ------ ------ Last Modified -------
Cmd  Group     User       Name        By           Date
--- -------- -------- ------------ -------- ------------------
    ****************************** TOP ***********************************
___ ADMIN    FOPDEMO  ARCHIVE      FOPSJD   2005-04-08 16.31.55
___ DVLMT01  FOPDEMO  ARCHIVE      FOPSJD   2005-03-24 12.45.38
___ DVLMT01  FOPDEMO  EXTRACT      FOPSJD   2005-03-22 14.40.48
_S_ GRP      FOPDEMO  AD           FOPSLN   2005-08-03 13.24.33
     ***************************** BOTTOM ***********************************
```

*Figure 8. Select Access Definitions*

### Description

A User Option setting determines whether the description of each Access Definition is displayed on this panel. See “User Options” on page 372 for information about the **Selection List Format** option.

### Line Commands

The **Cmd** area of the panel is used to enter line commands.

**Cmd** The following line commands are available:

**S** Select an Access Definition.

**D** Delete an Access Definition.

**C** Copy an Access Definition.

**R** Rename an Access Definition.

**AT** Modify the attributes of an Access Definition.

**I** Display information about an Access Definition.

See "Object Selection List Functions" on page 14 for more information about these commands.

## Access Definition Attributes

To display the attributes of an Access Definition from the selection list, type I in **Cmd** next to the name of the Access Definition. The following figure shows the read-only Access Definition Attributes panel.

```
------------------------- Access Definition Attributes  ----------------------
Command ===>

  Group              : GROUP
  User               : USER
  Name               : ADSAMPLE

  Description        : Sample Access Definition
  Security Status    : PUBLIC
  Last Modified By   : FOPDEMO
  Modified On        : 2005-01-12 11.06.42

  Number of Tables   : 6
  Start Table        : CUSTOMERS                                          >
  Default Creator ID : FOPDEMO                                            >

  Access Definition Parameters
    Dynamically Add New Tables     :  Yes
    Modify Selection/Sort Criteria :  Yes
    Begin Table Display with       :  Selection Criteria for Start Table
    Changes to AD During Edit      :  Permanent
    Use NEW Relationships          :  Yes
    Apply Crit in Self Reference   :  Yes
```

*Figure 9. Access Definition Attributes*

## Return Display

After selecting one or more Access Definitions from the Select Access Definitions panel, press ENTER to display the first selection in the Access Definition editor. After modifying or reviewing an Access Definition selected from the list, use END to display the next selection or redisplay the Select Access Definitions panel if no selections remain. If you selected Access Definitions using a line command, the redisplayed list is scrolled to show the last selection on the first line of the display screen.

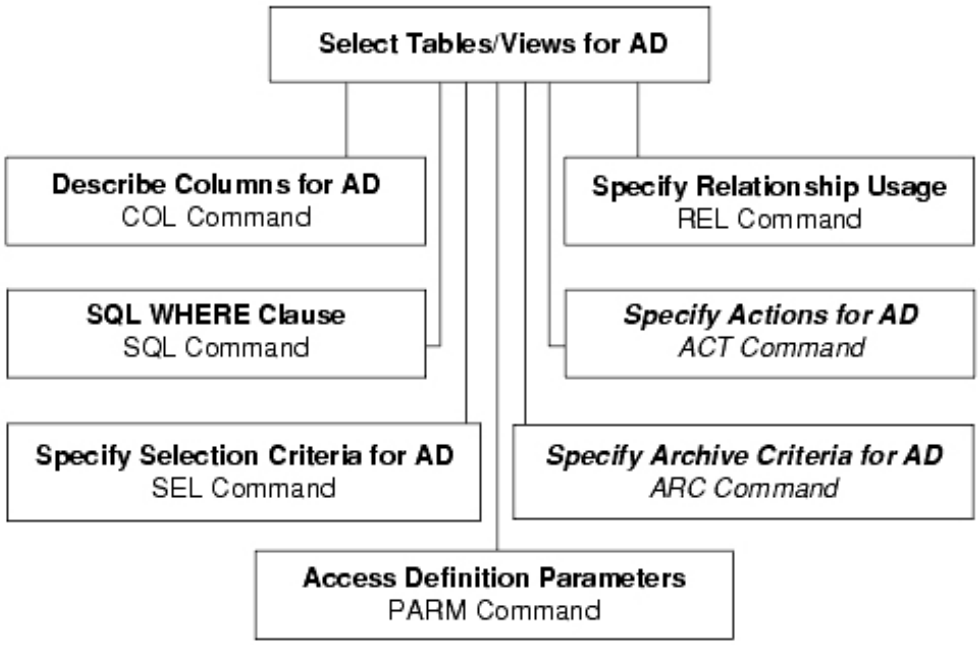
## Terminate List

Use END or CANCEL to return to the Choose an Access Definition panel.

## Components of an Access Definition

Optim displays a series of panels to prompt for the components of an Access Definition. These panels make up the Access Definition editor and the first panel, Select Tables/Views for AD, is used to specify the list of tables (Table List). All other Access Definition panels are accessible from this panel.

The name of each panel is shown in boldface in the following diagram. The command used to display the panel is also shown in each box.




**Note:** The ARC and ACT commands are available only for Access Definitions used in an Archive Process.

The Access Definition panels are available from both the **EXTRACT Process** and **ARCHIVE Process** menus, during a Compare Process, and by selecting Option 5 ADS from the **Main Menu**. For example, Option 1 TABLES on the **ARCHIVE Process** menu displays the Select Tables/Views for AD panel. Option 2 PATHS on the **ARCHIVE Process** menu invokes the Specify Relationship Usage panel.

Each panel and its use in defining the set of data to be processed are discussed in the following sections.

## Table/View Selection

Use the Select Tables/Views for AD panel to list the names of tables that are accessed in an Archive, Compare, or Extract Process or during an edit or browse session. This panel is displayed when you select an Access Definition from the Select Access Definitions panel or explicitly name one on the **Choose an Access Definition** panel. The Select Tables/Views for AD panel is also displayed when you select Option 1 TABLES from the **ARCHIVE Process** menu or the **EXTRACT Process** menu, or if you specify an Access Definition for either source in a Compare Process. (For details, see the appropriate *User Manual*.)

Unless indicated otherwise in this discussion, references to tables include Legacy Tables (for *Move* or *Compare for IMS, VSAM, and Sequential Files* only) and Materialized Query Tables. The use of a Legacy Table or Materialized Query Table in an Access Definition is virtually identical to that of a table. Exceptions are noted, where appropriate.

## Two Pages

When Access and Archive, Compare, or Move are available, this panel is presented on two panels, horizontally first and second "pages." To indicate the presence of the relevant horizontal page, "MORE" with an appropriate direction arrow is presented on the panel. The direction of the arrow indicates whether the remaining prompts are first or second horizontally. Use the LEFT and RIGHT primary commands or the assigned function keys to scroll the pages horizontally.

**Note:** If you are licensed only for Access, only the first horizontal page is available. If Access is not licensed, then only the second horizontal page is available.

For this manual, licensing of all Optim components is assumed.

## Initial Display

If you provide the name of an existing Access Definition on the Choose an Access Definition panel, the list of tables for that Access Definition is shown on the initial display of the Select Tables/Views for AD panel. If you are creating a new Access Definition, the initial display provides space to enter the name of a table. There are a variety of methods and commands available to help populate this list. For example, you can type the name of a single table and use the GET TABLES RELATED ALL command to populate the list with the names of all tables related to the first table.

## Sample Display

The following figure shows the first horizontal page of the Select Tables/Views for AD panel for a hypothetical Access Definition named AD. Note the fully qualified name of the Access Definition on the title line, which is GRP.FOPDEMO.AD in the example.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD ---------------------------------
Command ===>                                                Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),OR(A),DP(A),OP(A),
           DC(A),OC(A),EXP,ARC,ACT,STA
                                                  Table 1 of 1       MORE>>
Default Creator ID ===> FOPDEMO                                         >>
Start Table        ===> ORDERS                                          >>
Start Table Options   : None
                                          ---- Access Rights ----
Cmd   Status    (CreatorID.)Table/View Name (S)el/(U)pd/(I)ns/(D)el  Type
--- ----------- ------------------------>> ---------------------- -------
*** ******************************** TOP ********************************
___             ORDERS                          D            TABLE
*** ***************************** BOTTOM ********************************
```

*Figure 10. Select Tables/Views for AD – Access Rights*

## Panel

The prompts on the Select Tables/Views for AD panel are:

**Default Creator ID**

The default Creator ID automatically prefixed to any **Table/View Name** specified without one. In the preceding figure, for example, the Creator ID "FOPDEMO" would be prefixed to the table name "ORDERS" to form the fully qualified name "FOPDEMO.ORDERS".

Specify a 1 to 128 character Creator ID. When creating an Access Definition, the default Creator ID is the DB2 SQLID for the current user.

**Start Table**

The name of the table from which data is selected first. Specify the fully qualified name if the Creator ID is different from the **Default Creator ID**. You can specify a partial name, with wildcard characters, to display a table name list from which you can select the appropriate name.

The Start Table must be on the Table List. If you leave Start Table blank, the first name entered on the Table List is automatically inserted as the default Start Table. Conversely, if you enter a name in **Start Table** that is not listed in the Access Definition, the name is added to the Table List.

**Start Table Options**

Start Table selection criteria methods that apply. This is a display-only area that contains one or more of the following designations:

**None** No special Start Table criteria have been specified.

**Row List (POINT)**
Primary key values have been stored in a Row List and are available for any Archive or Extract Process.

**Temp. Row List (POINT)**
Primary key values have been stored in a Row List and are available for the current Archive or Extract Process only.

**Group Selection (GROUP)**
Group selection processing has been specified. (Group selection processing is available for Move and Compare only.)

**Note:** Start Table Options are relevant for Archive, Compare, and Move. If only Access is licensed, the value is always None.

See "The Point-and-Shoot Facility" on page 72 for information about Row Lists. See "Group Selection Processing" on page 85 for information about group selection processing.

**Cmd** Line command entry area. Use the following line commands to complete the Table List and specify or remove criteria for the listed tables.

**ACT** Display the Select an Action To Be Defined panel. (This command is available if Archive is licensed.)

**ALL** Remove or drop any selection criteria, SQL WHERE clauses, or archive criteria for the specified table, so that all rows are selected.

**ARC** Display the Specify Archive Criteria for AD panel. (This command is available if Archive is licensed.)

**COL** Display the Describe Columns for AD panel.

**EXP** Display the Expand - (Creator.ID)Table/View Name pop-up, which shows the table's full name. This command is useful in displaying Long Object Names (LONs) that cannot be displayed in full on the panel.

**GR(n)** Add names of tables that are parents or children to the selected table.

**Note:** A number or the letter "A" with the GR, GP, GC, DR, DP, DC, PR, PP, and PC commands indicate the number of levels or generations you want included. Specify a number from 1 through 9 to indicate the number of levels you want included, such as GR3, or specify the letter "A" to indicate that you want ALL levels included, such as GRA. If a number or "A" is not specified, the default is one level.

**GP(n)** Add names of tables that are parents to the selected table.

**GC(n)** Add names of tables that are children to the selected table.

**DR(n)** Using DB2 relationships, add names of tables that are parents or children to the selected table.

**DP(n)** Using DB2 relationships, add names of tables that are parents to the selected table.

**DC(n)** Using DB2 relationships, add names of tables that are children to the selected table.

**OR(n)** Using Optim relationships, add names of tables that are parents or children to the selected table.

**OP(n)** Using Optim relationships, add names of tables that are parents to the selected table.

**OC(n)** Using Optim relationships, add names of tables that are children to the selected table.

**LR** Generate a selection list of tables related to the selected table.

**SEL** Display the **Specify Selection Criteria for AD** panel.

**SQL** Display the SQL WHERE Clause panel.

**STA** Display the Criteria in Effect pop-up window, which lists the types of criteria in effect for the specified table.

**Note:** ISPF-like line commands are also available to copy, move, repeat, insert, or delete table names.

**Status** One or more status indicators for the table:

**ACT** Archive Actions are specified.

**ARC** Archive index or date criteria are specified.

**blank** All columns are selected with default display formatting. Default.

**COL** Column attributes are specified.

**REMOTE**
The named table is an alias to another DB2 subsystem. Remote tables are not supported. You must remove the name of a table in REMOTE status to save the Access Definition.

If remote access is available, you can change the LOCATION to access remote tables.

**SEL** Selection criteria are specified.

**SQL** An SQL WHERE clause is specified.

**TEMPTABLE**
The table is a DB2 Temporary Table. It contains no data and must be defined as a reference table. It also cannot be used as a Start Table, joined in a browse or Point-and-Shoot session, or selected for a browse or edit session.

**UNKNOWN**
The named table is not known to DB2. UNKNOWN status indicates the table does not exist. Possible reasons for UNKNOWN status are:

- The table name is mistyped.
- The named table does not exist or has been dropped.
- Prefixing the name with the default Creator ID results in a name for a table that does not exist.

You can save and use an Access Definition with any table or tables, other than the Start Table, in UNKNOWN status.

**VIEWERR**
The named view is not usable. Generally, this status indicates a conflict between the view definition and the underlying base tables. You can save an Access Definition with any view or views, other than the Start Table, in VIEWERR status.

**Note:** The **Status** area may not accommodate all the indicators that apply. See "Table Status Indicators" on page 32 for more information.

**Table/View Name**
The names of tables and views to be processed (i.e., the Table List). Specify the fully qualified name if the Creator ID is different from the **Default Creator ID**.

The Table List must include at least one table or view name. If you delete all names and exit the Access Definition editor, the Access Definition is deleted.

A table can be referenced only once in an Access Definition; therefore, the Table List cannot reference both a table and one or more views, aliases, or synonyms for the table. (This restriction does not apply to Access.)

Generally, a Temporary Table on the list is ignored by Optim. For Move, however, you can specify a temporary table, but it must be a reference table and cannot be used as the Start Table. Since there is no data in the table, you cannot join to it during a browse session.

**Access Rights**
Access privileges. (Access only) The Access Rights privilege applies if more restrictive for the user than the DB2 authorization; otherwise, the authorization in the DB2 Catalog is used. Use Access Rights to restrict DB2 privileges for a user when editing or browsing data with the Access Definition.

**S** Select only.

**U** Update or select.

**I** Insert, update, or select.

**D** Delete, insert, update, or select.

The implied hierarchy is S, U, I, and D in order of increasing privilege. If no value is specified, D is assumed.

**Type** The type of object. Possible values are:

**TABLE**
Table name

**S-MQT**
System-maintained Materialized Query Table

**U-MQT**
User-maintained Materialized Query Table

**VIEW** View name

**S-TABLE**
Synonym for a table

**S-VIEW**
Synonym for a view

**A-TABLE**
Alias for a table

**A-VIEW**
Alias for a view

**LEGACY**
Legacy Table name

When you scroll horizontally, the second page of the Select Tables/Views for AD panel is displayed.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD --------------------------------
Command ===>                                                 Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                      Table 1 of 1 <<MORE
Default Creator ID ===> FOPDEMO                                        >>
Start Table        ===> ORDERS                                         >>
Start Table Options   : None
                                              R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name F A EveryNth RowLimit  Type
--- ----------- ------------------------------>> - - -------- -------- -------
*** ********************************* TOP ***********************************
___              ORDERS                         N   ____   ________ TABLE
*** ******************************** BOTTOM ***********************************
```

*Figure 11. Select Tables/Views for AD – Criteria Portion*

## Panel

Many prompts on this panel are also displayed on the left-hand page and are described previously in this section. The following prompts are displayed only on the right-hand portion.

Optim displays this page first when you select Option 1 TABLES from the **ARCHIVE Process** or **EXTRACT Process** menu.

**RF** Identifier for a reference table, a table for which all rows are selected, unless selection criteria are specified for the table. Specify:

**N** Select only the related rows.

**Y** Use as a reference table.

Any table on the Table List, except the Start Table, can be treated as a reference table. Overtype the **RF** value with the desired designation or use the REF primary command to specify N or Y for all listed tables, other than the Start Table. Relationships for a reference table are not displayed on the Specify Relationship Usage panel because the relationships are irrelevant when processing reference table data. This label is **Ref Tbl** when the Select Tables/Views for AD panel is invoked during an Extract, Archive, or Compare Process.

**DA** Indicator for the deletion of database rows after they are archived. (This prompt is displayed only if Archive is installed.) Specify:

**N** Retain source rows after archiving.

**Y** Delete source rows after archiving.

You can select any table on the Table List. Overtype the **DA** value with the desired designation or use the DAA primary command to specify N or Y for all listed tables.

**Note:** This label is DAA when the Select Tables/Views for AD panel is invoked from the **ARCHIVE Process** menu.

**EveryNth**
A numeric value used as a factor for selecting rows from the table. Specify any value from 1 through 65,535. Specify “10” for example, to select every 10th row.

This entry is unavailable when the Select Tables/Views for AD panel is invoked from the **ARCHIVE Process** menu, or if Move or Compare is not installed.

**RowLimit**

The maximum number of rows that can be selected from the table. Specify any value from 1 through your site maximum value.

You can use **Row Limit** as a check when processing data. For example, if you expect to archive fewer than 1000 rows from a table, a row limit causes the Archive Process to terminate if the number of archived rows exceeds 1000. The termination allows you to troubleshoot the specifications for the process. Row Limit is also useful to extract a limited set of data for a test database.

Beginning and end markers on both horizontal pages of the panel indicate the first and last table names in the list.

## Table Status Indicators

This section describes the Status area on the Select Tables/Views for AD panel.

The **Status** area on the Select Tables/Views for AD panel contains:

- ACT when Archive Actions have been specified.
- ARC when archive indexing or date criteria have been specified.
- COL when specifications have been made for the columns of the selected table.
- SEL when selection criteria have been specified.
- SQL when an SQL WHERE clause has been specified.
- Blank when no criteria are specified.

In the following figure, archive criteria, column specifications, selection criteria, and an SQL WHERE clause have been defined for the ORDERS table.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD ---------------------------------
Command ===>                                                Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                      Table 1 of 4 <<MORE
Default Creator ID ===> FOPDEMO                                       >>
Start Table        ===> ORDERS                                        >>
Start Table Options   : None
                                          R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ----------------------------->> - - -------- -------- -------
*** ********************************* TOP ***********************************
___ ARC/COL/... ORDERS                          N   ____   ________ TABLE
___             CUSTOMERS                     N N   ____   ________ TABLE
___             DETAILS                       N N   ____   ________ TABLE
___             ITEMS                         N N   ____   ________ TABLE
*** ******************************** BOTTOM ***********************************
```

*Figure 12. Multiple Status Indicators*

### STA Command

**Status** is not large enough to display all four status indicators, ARC, COL, SEL, and SQL. Use the STA line command to display the Criteria in Effect pop-up window, which lists the types of criteria in effect for the table on the Select Tables/Views for AD panel, as shown in the following figure.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD --------------------------------
Command ===>                                                 Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                    Table 1 of 4 <<MORE
Default Creator ID ===> FOPDEMO                                      >>
Start Table        ===> ORDERS                                       >>
Start Table Options   : None
                                                R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ------------------------------>> - - -------- -------- -------
--- ----------- ----------                      - - -------- -------- -------
*** ********************** +--Criteria in Effect-+ ***************************
STA ARC/COL/... ORDERS     ¦                      ¦ N    ____   ________ TABLE
___             CUSTOMERS  ¦  Archive Criteria    ¦ N    ____   ________ TABLE
___             DETAILS    ¦  Column Criteria     ¦ N    ____   ________ TABLE
___             ITEMS      ¦  Selection Criteria  ¦ N    ____   ________ TABLE
*** ********************** ¦  SQL Criteria        ¦ ***************************
                           +----------------------+
```

*Figure 13. Criteria in Effect*

## Description

Before saving an Access Definition, you can provide a description to be displayed on a selection list. Use the ATTRIBUTES command to display the Object Attributes panel, which provides a 40-character area to display and edit the description. (Site management determines whether this panel also displays a prompt for Security Status.) For additional information about the Object Attributes panel, see "Specify Description and Security Status" on page 380.

## Available Commands

The following commands are available when the **Select Tables/Views for AD** panel is displayed, except as noted.

- ACTIONS[2]
- ALL
- ARCHIVE[2]
- ATTRIBUTES
- BOTTOM
- BROWSE
- CANCEL
- COLUMNS
- CREATE PRIMARY KEY
- CREATE REL
- DAA[2]
- DOWN
- EDIT[1]
- END
- EXPAND *or* EXP
- FIND
- GET TABLES RELATED
- GROUP[3]

- INDENT
- LEFT
- LIST
- LIST SUBS
- OPTIONS
- PARAMETERS
- POINT[4]
- REF[4]
- REL[4]
- RESET
- RFIND
- RIGHT
- SAVE[1]
- SEL CRITERIA
- SORT
- SQL
- STA (Status)
- TOP
- UP

**Note:**

1. Access only.
2. Archive only.
3. Move and Compare only.
4. Move, Compare, and Archive only.

## List Table Names

At least one table must be referenced by the Access Definition. You can type the table names on the panel, or you can select the appropriate names from a selection list or use commands to let Optim insert the table names automatically.

### Use a Selection List

The LIST commands provide a variety of selection lists.

#### LIST TABLES

Use the LIST TABLES command to display an alphabetical selection list of tables. If you do not include a name pattern operand, the list includes all tables for the default Creator ID. Specify a name pattern, using DB2 LIKE syntax, to limit the selection list. For example, to list all table names that begin with "A" for the default Creator ID, specify:

```
LIST TABLES A%
```

You can include additional operands (DB, TS, PL, or PKG) to limit the list to tables in a specific database, tablespace, plan, or package. These operands also support DB2 LIKE syntax. For example, to list all table names that begin with "A" for the default Creator ID in a database named FOP1, specify:

```
LIST TABLES A% IN DB FOP1
```

### LIST VIEWS

Use the LIST VIEWS command to display an alphabetical selection list of views. A name pattern may be specified.

### LIST ALIASES

Use the LIST ALIASES command to display an alphabetical selection list of aliases. A name pattern may be specified.

### LIST SYNONYMS

Use the LIST SYNONYMS command to display an alphabetical selection list of synonyms for the current SQLID. (This command does not support DB2 LIKE syntax.)

### LIST Example

As an example, assume you are creating a new Access Definition with ORDERS as the Start Table. Type the LIST TABLES command and press ENTER to display the names of all tables for the default Creator ID, FOPDEMO.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD ---------------------------------
Command ===>                                                 Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                  Table 1 of 1 <<MORE
Default Creator ID ===> FOPDE                                        >>
Start Table        ===> ORDER +----Select One or More Tables----+    >>
Start Table Options   : None  | Cmd CreatorID.TableName  1 OF 8 |
                              | --- --------------------        | ms--
Cmd   Status      (CreatorID. | ********* TOP **********       | imit  Type
--- ----------- ------------- | _S_ FOPDEMO.CUSTOMERS           | ---- -------
*** ************************* | ___ FOPDEMO.DETAILS             | ************
___              ORDERS       | ___ FOPDEMO.FEMALE_RATES        | ____ TABLE
*** ************************* | ___ FOPDEMO.ITEMS               | ************
                              | ___ FOPDEMO.MALE_RATES          |
                              | ___ FOPDEMO.SALES               |
                              | ___ FOPDEMO.SHIP_INSTR          |
                              | ___ FOPDEMO.SHIP_TO             |
                              | ******** BOTTOM ********        |
                              +---------------------------------+
```

*Figure 14. Table Selection List*

Use the S command to select any number of tables individually, or use the SEL ALL command to select all listed tables. You can select a maximum of 99 tables from the list at one time. Replace the S with a blank to "unselect" a table. When you scroll the list, each selection is processed to show an uppercase S in **Cmd**. Selected tables are listed in the Access Definition when you press the ENTER key or use END to terminate selection list processing.

### Table Selected

In Figure 14, the CUSTOMERS table is selected by typing S in **Cmd**. Use END or press ENTER to redisplay the Select Tables/Views for AD panel with the name of the selected table added to the list, as shown in the following figure.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD --------------------------------
Command ===>                                                Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                      Table 1 of 2 <<MORE
Default Creator ID ===> FOPDEMO                                       >>
Start Table        ===> ORDERS                                        >>
Start Table Options   : None
                                              R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ------------------------------>> - - -------- -------- -------
*** ********************************* TOP ************************************
___             ORDERS                           N    ____   ________ TABLE
___             CUSTOMERS                      N N    ____   ________ TABLE
*** ******************************** BOTTOM ***********************************
```

*Figure 15. Selected Table Included in Access Definition*

Redisplay the selection list as needed. The selection list excludes any tables already listed in the Access Definition.

### Selection List of Related Tables

A selection list is also available to display all tables that are related to a specific table. Use the LIST TABLES RELATED primary command or the LR line command to display a list of related tables. The LIST TABLES RELATED command requires a table name operand. For example, to display a list of tables with the default Creator ID that are related to the CUSTOMERS table, enter:

```
LIST TABLES RELATED TO CUSTOMERS
```

You can display the same list by specifying:

```
LIST TABLES RELATED
```

and positioning the cursor on the line for CUSTOMERS and pressing Enter. You can also type the LR line **Cmd** (command) on the CUSTOMERS line and press ENTER to list all tables related to that table.

The selection list shows the names of the related tables, the source of the relationship definition, and the role of a table in the relationship as either a parent or a child. The source of the relationship is indicated as DB2 for relationships in the DB2 Catalog, OPT for those defined to the Optim Directory, or IMS for those defined in an IMS DBD (*Move* or *Compare for IMS, VSAM, and Sequential Files* only).

Select tables in the same manner as from the LIST TABLES list. You can redisplay this list as needed.

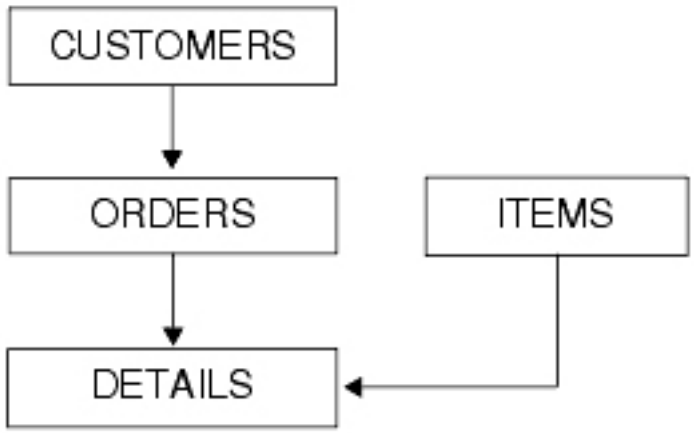
## Insert Table Names Automatically

Use the GET TABLES RELATED command to insert the names of related tables in the Access Definition Table List. The names are the same as those provided on the selection list obtained with the LIST TABLES RELATED command. You can delete any unwanted names that have been inserted.

The **Relationship Processing** User Option determines the type of relationship (OPT, DB2, or both) used to populate the Table List. This option also allows you to use the **Get Related Processing Options** pop-up window to determine the type of relationship. You can override the User Option by specifying the relationship type in the command.

Use the ALL operand with the command to obtain names of the entire set of tables, including all parents, grandparents, children, grandchildren, and so on, related to the named table.

For example, assume the four tables in the following diagram are related, as shown. (DB2 relationships are selected in the **Relationship Processing** User Option.) The arrows indicate the direction of the relationship from parent to child.




Specifying GET TABLES RELATED ALL for any table in the diagram populates the Table List with all table names. You can use the D line command to remove any unwanted names from the list.

Use operands to insert only the names of related parents or related children for a specified number of levels. Note the following examples:

**GET TABLES RELATED TO CUSTOMERS CHILD**
This command inserts the names of children directly related to CUSTOMERS. In this example, only ORDERS is inserted.

**GET TABLES RELATED TO CUSTOMERS 2 CHILD**
This command inserts the names of children at two levels: the child and grandchild levels. In this example, ORDERS and DETAILS are inserted.

Conversely, the command can be directed to include names of parent tables. Note the following examples:

**GET TABLES RELATED TO DETAILS PARENT**
This command inserts the names of parents of the DETAILS table only. In this example, the ORDERS table and the ITEMS table are parents.

**GET TABLES RELATED TO DETAILS 2 PARENT**
This command inserts the names of parents at two levels: the parent and grandparent levels. In this example, the CUSTOMERS table, as the parent of the ORDERS table, is inserted, in addition to the ORDERS table and the ITEMS table, because CUSTOMERS is the grandparent of the DETAILS table.

You can also specify which relationships to use, that is, either DB2 or OPT. Specifying a relationship type overrides the **Relationship Processing** User Option. Note the following examples:

**GET TABLES RELATED TO DETAILS PARENT OPT**
This command inserts the names of parents from OPT relationships with the DETAILS table.

**GET TABLES RELATED TO DETAILS 2 PARENT OPT**
This command inserts the names of parents at two levels – the parent and grandparent levels – from OPT relationships with the DETAILS table.

Similar line commands (GR, GP, GC, DR, DP, DC, PR, PP, and PC) are also available. A number from 1 through 9 indicates the number of levels or generations to include (for example, GP2). Use the letter "A" to indicate that ALL levels are included (for example, GRA). If a third character is not specified, only one level is included. Enter the command in the line command area of any table name on the Select Tables/Views for AD panel.

A table designated as a reference table cannot be the subject of the GET TABLES RELATED primary or line command. By definition, all rows in a reference table are selected (subject to selection criteria), regardless of relationships with other tables on the Table List.

### Get Related Processing Options

If the **Relationship Processing** User Option is set to R (prompt), the **Get Related Processing Options** pop-up window displays when you enter the GET TABLES RELATED command.

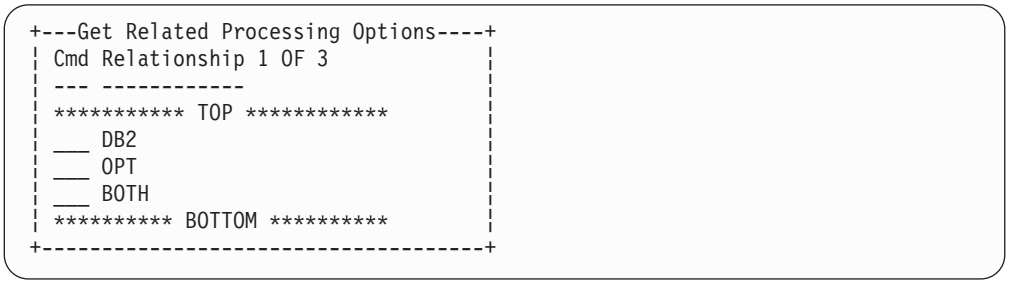



*Figure 16. Get Related Processing Options*

In the window, use the S line command to select the relationship type (DB2, OPT, or BOTH), and use END when you have made your selection, or use the CANCEL command to abandon any unsaved changes and return to the Select Tables/Views for AD panel.

For more information about User Options, see "User Options" on page 372.

### List Tables with Different Creator IDs

The names of related database or Legacy Tables may not be prefixed with the Default Creator ID. If so, you must use a more specific GET TABLES RELATED ALL or GRA syntax to list these tables in the Access Definition. For example, if the Default Creator ID is FOPDEMO and you want to insert the names of Legacy Tables prefixed with FOPLEG that are related to the DB2 table FOPDEMO.CUSTOMERS, the command syntax is:

GET TABLES FOPLEG.% RELATED TO CUSTOMERS

The resulting Table List includes all tables with the prefix FOPLEG that are related to FOPDEMO.CUSTOMERS. In this example FOPLEG.ORDERS, FOPLEG.ITEMS, and FOPLEG.DETAILS are selected.

```
-- Select Tables/Views for AD: FOPDEMO.FOPLEG.NEWLEGREL -----------------------
Command ===>                                                 Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                    Table 1 of 7 <<MORE
Default Creator ID ===> FOPDEMO                                      >>
Start Table        ===> CUSTOMERS                                    >>
Start Table Options   : None
                                                R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ------------------------------>> - - -------- -------- -------
*** ********************************* TOP ************************************
___             CUSTOMERS                        N    ____   ________ TABLE
___             FOPLEG.ORDERS                  N N    ____   ________ LEGACY
___             FOPLEG.ITEMS                   N N    ____   ________ LEGACY
___             FOPLEG.DETAILS                 N N    ____   ________ LEGACY
```

*Figure 17. Legacy Tables in Table/View Name List*

For more information about the GET TABLES command, see GET TABLES RELATED in the *Command Reference Manual.* For more information about Legacy Tables, see the *MOVE User Manual*, Legacy Tables or *Compare for IMS, VSAM, and Sequential Files*.

## Supporting Facilities

When constructing an Access Definition, you may find that certain application-enforced relationships are not defined to DB2. You can create or analyze relationships, if needed.

### Create Relationships

If an expected table is not included when you use GET TABLES RELATED or LIST TABLES RELATED, the necessary relationship does not exist. You can use the CREATE RELATIONSHIP command on the **Select Tables/Views for AD** panel to display the panels needed to define OPT relationships. (Option 6 DEFINITIONS on the **Main Menu** also invokes these panels. Information about defining relationships is provided in Chapter 4, “Relationships,” on page 125.)

When you have defined the relationship, use END or CANCEL to return to the Select Tables/Views for AD panel. After you create the needed relationship, the GET TABLES RELATED or LIST TABLES RELATED command uses the new relationship to populate the Table List.

### Displaying Relationships

If a GET TABLES RELATED or LIST TABLES RELATED command returns an unexpected table name, you can display the list of tables in an indented format that illustrates the relationships. The indented list allows you to analyze the relationships to determine the tables and relationships that should be included in the Access Definition. Use the INDENT command from the Select Tables/Views for AD panel or the Specify Relationship Usage panel to display the indented list.

For example, if the CUSTOMERS, ORDERS, DETAILS, and ITEMS TABLES shown in the diagram on page “Insert Table Names Automatically” on page 36 are listed in an Access Definition, the INDENT command will result in the following **Indented Table Display**.

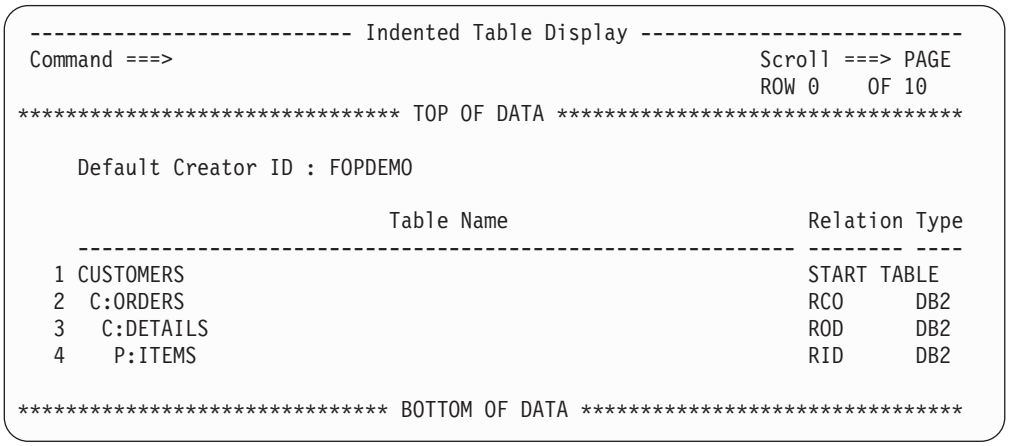



*Figure 18. Indented Table Display*

## Panel

The Indented Table Display panel includes:

**Default Creator ID**
The default Creator ID, as specified on the Select Tables/Views for AD panel.

**Table Name**
The name of each table. If the Creator ID is different from the **Default Creator ID**, it is included with the name.

The Start Table is listed first, followed by the other table names, and the list is formatted to reflect the relationships. The names of parent tables are followed by the names of children. Within this constraint, tables are listed in the order of the Access Definition Table List.

Information about each relationship includes:

**Parent/Child Indicators**
Except for the Start Table, a table name is prefixed to indicate its relationship with the table under which it is indented. The prefix is either:
**C** (Child) or **P** (Parent)

**Multiple Reference Indicator**
The second and subsequent references to a table (e.g., a table that is the child of three parents) are noted by the suffix (*n*) where *n* is the line number for the first reference. The first reference is not displayed with a multiple reference indicator and is the only one to show the names of the tables indented under it.

**Cycle Indicator**
Any table in an RI cycle is noted by the suffix (CYCLE:*n*) where *n* indicates the line numbers of the other tables in the cycle.

**Relation**
The name of the relationship between the table and the table under which it is indented. START TABLE is always specified for the Start Table.

**Type** Indicator for the type of relationship. Possible values are:

**DB2** Relationship is defined in the DB2 Catalog.

**OPT** Relationship is defined in the Optim Directory.

**Note:** The relationship type may be shown as 'OPT', 'FOP', or 'PST' in a batch process report or ISPF panel.

**IMS** Relationship is defined within the referenced IMS DBD. (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**IML** Relationship is defined in the Optim Directory, but is based on an existing Logical IMS Relationship. (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

If there is insufficient space for the full table name, the name is preceded with a level indicator (L*n* where *n* is the level number) and the indentation is eliminated to provide more room on the line.

Any reference tables are not included in the indented list but are noted at the end of the list. Any table that does not have a relationship with any other table on the list is also noted at the end of the **Indented Table Display**.

Use END to return to the Select Tables/Views for AD panel.

## Handling the Table List

On the Select Tables/Views for AD panel, you can scroll the list of tables using UP, DOWN, TOP, and BOTTOM. You can also use the FIND command to locate a table name, specifying all or part of the name with that command.

### Line Commands

To edit the list of tables, you can type over a table name or use line commands. The line commands available for editing include standard ISPF-like commands to change the order of the table names and to add new rows to the Table List. Rearranging the names on the Table List has no effect on the sequence in which the tables are accessed during a process.

The available line commands are:

**Copy** C or CC

**Delete** D or DD

**Insert** I

**Move** M or MM

**Repeat**
R or RR

**Note:** Use A or B to indicate the destination of a copy or move operation.

Each table name must be unique. If editing results in duplicate table names, the Access Definition must be corrected before you can save or use it.

### Delete Names

To remove a table name, use the Delete line commands, space over the name, or use the Erase EOF key. Although any number of names can be deleted from the Access Definition, at least one name must remain. If all table names are removed from the Access Definition when you exit the Select Tables/Views for AD panel, the Access Definition is automatically deleted. Use the **Confirm on Deletes** user option to be prompted for confirmation before the Access Definition is deleted.

### Multiple Use

You can use a single Access Definition to process more than one version of your database tables. For example, if you modify only the Creator ID portion of a listed table name, any column specifications and SQL WHERE Clauses are retained in the Access Definition. Also, any relationships that do not apply to

the newly listed table remain on the Specify Relationship Usage panel with a status of UNKNOWN, while any relationships for the newly listed table are added to the **Specify Relationship Usage** list. You can explicitly remove UNKNOWN relationships, or leave them in the Access Definition to be used when you reinstate the original Creator ID.

However, if you modify the base name entry for a table, any column specifications and SQL WHERE Clauses are removed from the Access Definition, and any relationships for the previously listed table are also removed.

## Completed Table Selection

Use the END command to save the Access Definition specifications in the Optim Directory. After saving the Access Definition, Optim redisplays the panel from which the Select Tables/Views for AD panel was invoked.

### Error Handling

Any error conditions must be resolved. Some errors prevent you from proceeding with the Access Definition; other errors must be resolved before you can save the Access Definition.

The following errors must be resolved to proceed to another panel in the Access Definition editor.

- Invalid line command entry.
- Invalid characters.
- Syntax errors in a command entry.

The following errors must be resolved to save the Access Definition, and browse or edit data.

- A duplicate table name is specified (Archive, Move, and Compare only).
- No table names are specified.
- The designated Start Table is removed from the list.
- A remote table name is specified.
- DB2 cascade delete restrictions apply to one or more relationships (i.e., a table selected for a delete operation is the parent of a table that is not in the Table List).

### SAVE Command

Use the SAVE command to save the Access Definition under the current name or a new name. For more information, see the *Command Reference Manual*, Primary Commands.

### CANCEL

Use the CANCEL command to ignore any unsaved changes and redisplay the previous panel.

## Associate Legacy Tables with Data Sources

For *Move* or *Compare for IMS, VSAM, and Sequential Files*, a Legacy Table referenced in an Access Definition must be associated with a specific data source to be used in an Extract Process. Each time you exit the Access Definition editor using the END command, the Associate Legacy Tables with Data Sources panel is displayed. The panel contents vary based on whether the Legacy Tables in the Access Definition describe IMS data, VSAM or sequential files, or both.

In the following example, the Access Definition references two Legacy Tables: FOPLEG.LORDERS, which describes data in a VSAM file, and FOPIMS.I$ITEMS, which describes IMS data.

```
----------------- Associate Legacy Tables with Data Sources -------------------
Command ===>                                                 Scroll ===> PAGE

Overriding Dataset Prefix ===>                                        1 of 2

                           File-Dataset Name
Legacy Table               IMS--Segment    DBD      PSB    PCB IMSID DBRC LOG
-------------------------- ---- -------- -------- -------- --- ----- ---- ---
******************************** TOP ***************************************
FOPLEG.LORDERS             File 'FOPRT.LEGACY.ORDERS'
FOPIMS.I$ITEMS             IMS  ITEMS    ITEMHDAM ITEHDAMA 1          N    N
******************************* BOTTOM **************************************
```

*Figure 19. Associate Legacy Tables with Data Sources*

**Note:** The labels displayed on this panel vary based on what type of legacy tables are displayed (i.e., IMS, VSAM, or sequential files). The following labels are displayed ***for IMS files only***: IMS--Segment, DBD, PSB, PCB, IMSID, DBRC, and Log. The following labels are displayed ***for VSAM or sequential files only***: Overriding Dataset Prefix and File-Dataset Name (or just Dataset Name if no IMS files are displayed). If more than one file type is displayed, a combination of labels is displayed.

## Panel

The Associate Legacy Tables with Data Sources panel includes:

**Overriding**
**Dataset Prefix**
An optional prefix for names of VSAM or sequential datasets associated with Legacy Tables referenced in the Access Definition. Specify 1 to 8 characters.

**Legacy Table**
The read-only name of each Legacy Table referenced in the Access Definition.

**File-Dataset Name**
If the Legacy Table describes data in a VSAM or sequential file (i.e., a VSAM Legacy Table), the designation **File** is displayed with any default data source or previously supplied **Dataset Name**. You can edit any supplied dataset name and if this area is blank, you must provide a dataset name in order to save the Access Definition.

Enclose the complete dataset name in single quotes, or it will be prefixed with the **Overriding Dataset Prefix** or the default prefix specified in the User Options.

If several Legacy Tables are associated with the same dataset or a similarly named dataset, you can type the dataset name for the first relevant Legacy Table and enter the equal sign (=) for subsequent Legacy Tables to duplicate the entry.

**IMS--Segment**
If the Legacy Table describes IMS data (i.e., an IMS Legacy Table), the designation **IMS** is displayed with the read-only name of the Segment associated with the Legacy Table.

**DBD** The read-only name of the DBD associated with the IMS Legacy Table.

**PSB** The name of the PSB for access to IMS services. **PSB** is populated with the Default PSB Name from the Retrieval Definition, if any, or with any previously entered PSB name for the Legacy Table; otherwise, it is blank.

A valid PSB name is required to save the Access Definition. To enter or replace the name, specify the 1- to 8-character name of a PSB that is in a library referenced by the Environment Definition for the Legacy Table. Enter an asterisk to generate a selection list.

**PCB** The relative number of the PCB within the PSB. **PCB** number is populated with the PCB number from the Retrieval Definition, if any, or with any previously entered PCB number for the Legacy Table; otherwise, it is blank.

To enter or replace the number, specify the number for a PCB within the PSB. The PCB must grant Move the authorization to manipulate the data. Enter an asterisk to generate a selection list of PCB numbers for the PSB.

**IMSID**

The IMS ID required to access the IMS data when allocated to a control region (i.e., the data is online to IMS). **IMSID** is populated with the IMS System ID from the Environment Definition, unless overridden by the Default IMS ID from the Retrieval Definition, if any.

**DBRC** This entry is valid only for IMS processing in DL/I mode (i.e, when an IMS ID is not specified). If appropriate, enter **Y** for yes to use Database Recovery Control (DBRC) to control logging and perform database recovery; otherwise enter **N** for no. IMS uses the online log datasets (OLDS) if the database is accessed in BMP or DBB mode.

The default for a HALDB (High Availability Large DataBase), is **Y**, and that entry cannot be changed.

DBRC use is optional for a non-HALDB, such as HIDAM, HDAM, HISAM, etc. Thus, you may specify **Y** for a non-HALDB, but it is not required.

**LOG** This entry is valid only for IMS processing in DL/I mode (i.e, when an IMS ID is not specified). If appropriate, enter **Y** for yes to use an IMS log to perform database recovery; otherwise enter **N** for no.

If you specify Y, you must specify a dataset name for the IMS log on the Associate IMS Segments with IMS Database Datasets panel. The DD Name “IEFRDER” is used to allocate the log dataset on that panel.

If a PSB with a Processing Option (PROCOPT) other than G (Get) is used while accessing a HALDB in DL/I mode (i.e., an IMS region name is not specified), you must specify the name of the dataset to be allocated for DD Name IEFRDER on the Associate IMS Segments with IMS Database Datasets panel.

After you specify the IEFRDER dataset name and exit the Associate IMS Segments with IMS Database Datasets panel, the Allocate Dataset panel automatically displays. You must provide sufficient Primary and Secondary space units on that panel to allocate the IEFRDER dataset. Failing to do so will cause IMS to abort processing and lock the database from further updates until a recover/rollback is done.

Use the END command when you are finished making changes to this panel, or use the CANCEL command to abandon any unsaved changes and return to the **Select Tables/Views for AD** panel.

If one or more Legacy Tables describe IMS data and an IMS ID was not specified, the Associate IMS Segments With IMS Database Datasets panel displays. Otherwise, the previous panel displays, saving any changes to the Access Definition.

## Associate IMS Segments with IMS Database Datasets

When you use END to exit the Associate Legacy Tables with Data Sources panel and one or more Legacy Tables in the Access Definition reference IMS data and an IMS ID was not specified, Optim displays the Associate IMS Segments With IMS Database Datasets panel.

Use this panel to associate the Legacy Table with the IMS Database Dataset to be processed when the Access Definition is used.

```
------------- Associate IMS Segments With IMS Database Datasets ---------------
Command ===>                                                 Scroll ===> PAGE

Overriding Dataset Prefix ===>                                        1 of 7


  DBD    Segment  DD Name  IMS Database Dataset Name
-------- -------- -------- ---------------------------------------------
********************************** TOP ***************************************
ITEMS    ITEMSDBD ITEMDD
******************************** BOTTOM ***************************************
```

*Figure 20. Associate IMS Segments With IMS Database Datasets*

## Panel

The Associate IMS Segments With IMS Database Datasets panel includes:

**Overriding**
**Dataset Prefix**
An optional default prefix for IMS Database Dataset Names listed on the panel. Specify 1 to 8 characters.

**DBD** Read-only name of the DBD associated with each IMS Legacy Table.

**Segment**
Read-only name of the segment within the DBD.

**DD Name**
Read-only name of the Data Definition (DD) or the physical dataset associated with each segment. **DD Name** cannot be modified. If IMS logging was requested, the DD Name IEFRDER is displayed (along with the pseudo-Segment IMSLOG) to identify the Log dataset.

**IMS Database**
**Dataset Name**
The name of the IMS Database Dataset to be processed when the Access Definition is used. **IMS Database Dataset Name** is populated with the Associated IMS Database Name from the Retrieval Definition, if any, or with any previously entered dataset name; otherwise, it is blank.

A Site Option (Require IMS Data Set Names) determines whether you can omit the data set name to allow IMS to dynamically allocate the data set. All users can specify '$MDA' as the data set name to choose dynamic allocation, regardless of this Site Option.

If you are including a data set from an IMS database with multiple data set groups, for each database you must either:

- specify all the data set names, or
- enable dynamic allocation by leaving the data set names blank (if your site option `Require IMS Data Set Names` is set to `N`), or
- enable dynamic allocation by specifying `'$MDA'` for all the data set names

You must use the same option for all the data set groups in a database.

To enter or replace the name, enter the location of the IMS Database Dataset associated with the DD name in the DBD. This data is then associated with the named Legacy Table during processing.

If IMS logging was requested, you must specify the Log dataset for DD Name IEFRDER. If a default name is displayed from the Provide Retrieval Information panel, you can override that name, if needed.

After you specify the IEFRDER dataset name and exit the Associate IMS Segments with IMS Database Datasets panel, the Allocate Dataset panel automatically displays. You must provide

sufficient Primary and Secondary space units on that panel to allocate the IEFRDER dataset. Failing to do so will cause IMS to abort processing and lock the database from further updates until a recover/rollback is done.

**Note:** You do not have to specify a dataset name for a HALDB because the appropriate dataset name will already be known to the IMS subsystem, but you do have to specify the IEFRDER dataset if it was not defaulted from the retrieval information specified on the Provide Retrieval Information panel.

Use the END command to return to the Choose an Access Definition panel from the display, or use the CANCEL command to abandon any changes made on this panel and return to the **Select Tables/Views for AD** panel.

---

# Selection Criteria

By default, all rows in each table are processed, subject to user and site maximums. Optim uses criteria to limit processing to specific rows from the Start Table and other tables in the Table List.

Types of selection criteria are:

- Archive criteria (date and indexing) for individual columns in a table.
- Selection criteria for individual columns in a table.
- An SQL WHERE clause for a table.
- Rows in the Start Table, selected with the Point-and-Shoot facility. (The Point-and-Shoot facility is available from the Select Tables/Views for AD panel and is discussed in "The Point-and-Shoot Facility" on page 72.)

## Combining Criteria

When more than one form of criteria is specified, the criteria are combined. However, if both Point-and-Shoot values and selection criteria are specified for a Start Table, an option on the panel for the process (e.g., Specify EXTRACT Parameters and Execute panel) allows you to use Point-and-Shoot values only or combine Point-and-Shoot values and selection criteria.

## Remove Criteria Specifications

To remove all archive criteria, selection criteria, and SQL WHERE clause specifications for a table, use the ALL primary or line command from the Select Tables/Views for AD panel. For example, to remove criteria for the CUSTOMERS table, specify the primary command as:

```
ALL CUSTOMERS
```

# Archive Criteria

Archive criteria determine the archived columns that are indexed and define the set of rows archived from each table. Archive criteria can be applied to one or more columns in a table and are combined with selection and other criteria to select rows during an Archive Process.

## ARC Command

To specify archive criteria, use the ARC primary or line command on the Select Tables/Views for AD panel. For example, to specify archive criteria for the ORDERS table, type the ARC **Cmd** (command) on the ORDERS line and press ENTER to display the Specify Archive Criteria for AD panel. Use this panel to specify the archive criteria for the selected table. (You can use the SEL, SQL, or COL command from the Specify Archive Criteria for AD panel to view or edit other criteria or data display settings.)

```
-- Specify Archive Criteria for AD: GRP.FOPDEMO.AD ----------------------------
Command ===>                                                  Scroll ===> PAGE

Table Name: FOPDEMO.ORDERS                                 Col 1 of 8     <<MORE
Combine All Column Criteria by ==> A   (A-AND, O-OR)

Cmd     Column Name     IDX              Criteria
---  ------------------ --- ------------------------------------------------
***  ******************************** TOP ************************************
___  ORDER_ID            D
___  CUST_ID             D
___  ORDER_DATE          N
___  ORDER_TIME          N
___  FREIGHT_CHARGES     N
___  ORDER_SALESMAN      N
___  ORDER_POSTED_DATE   N
___  ORDER_SHIP_DATE     N
***  ******************************** BOTTOM *********************************
```

*Figure 21. Specify Archive Criteria for AD*

## Panel

The Specify Archive Criteria for AD panel includes:

**Table Name**
The name of the current table or view.

**Combine All Column Criteria by**
The logical operator to apply to all criteria for the column. Specify:

**A** Combine all criteria with AND (default).

**O** Combine all criteria with OR.

**Cmd** The line command entry area. Valid commands are:

**DC** Specify date criteria for the column using the Define Date Criteria panel (see "Specify Date Criteria" on page 48).

**Z** Zoom to display all information for the column on a single panel.

**Note:** Zoom is a useful way to review characteristics of the column, such as data type or null eligibility before defining date criteria. From the Zoom display, you can then use the DC command to display the Define Date Criteria panel.

**Column Name**
The name of the column. Archive lists all columns in the table, in the order defined to DB2. You cannot edit, rearrange, insert or delete column names on this display. If the complete list cannot be displayed, you can scroll it. Notation next to **Table Name** indicates the relative position of the first visible column name. This notation is in the form, Col *x* of *y*, where *x* is the relative position of the first visible name and *y* the total number of names in the list.

**IDX** Indicator for indexing the column to facilitate searches of the archived data.

**N** Values in the column are not indexed. N is the default setting.

**D** Values in the column are indexed. The length of data in an indexed column must not exceed 254 bytes, and no more than 16 columns in a table can have a dense index.

**S** Values in the column are sparsely indexed (that is, high and low values indicate the possibility that a value occurs in the column.) Use the IDX command to designate N, D, or S for all columns.

**Archive Criteria**
Date criteria defined for a column.

## Specify Date Criteria

You can use the DC line command to display the Define Date Criteria panel. This panel allows you to define a specific date or a relative date to be used as criteria to select rows.

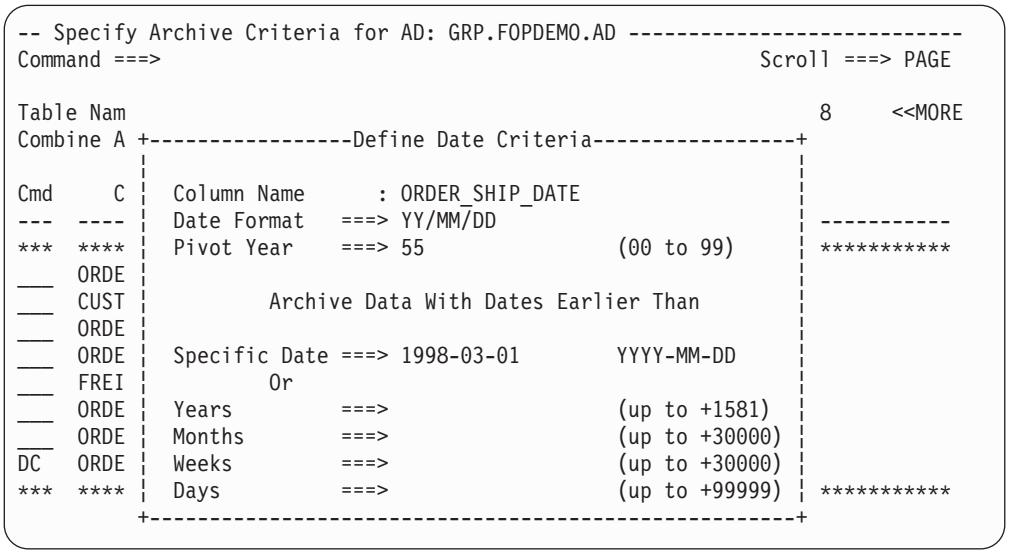



*Figure 22. Define Date Criteria*

## Panel

The Define Date Criteria panel includes:

**Column Name**
The name of the column for date criteria.

**Date Format**
The format of dates in the column. Specify the format or enter an asterisk for a list of date types that are appropriate for the data type and precision of the column.

If the column is a DB2 DATE column, Archive lists the format as DB2DATE.

**Pivot Year**
The year used to determine whether a two-digit year value is treated as occurring in the 20th century (1900s) or the 21st century (2000s). Specify a 2-digit numeric value. If the year is greater than or equal to this value, the 20th century is assumed.

**Pivot Year** is not displayed if the format is DB2DATE.

**Specific Date**
The cut-off date for archived data, specified in ISO, European, or USA format. Archive converts the date to your DB2 default format. Rows with a date earlier than the specified date are selected for archiving.

**Years** A number of years. This value is combined with any values in **Months**, **Weeks**, and **Days**, and offset from the current date to determine the cut-off date for archived data.

**Months**
A number of months. This value is combined with any values in **Years**, **Weeks**, and **Days**, and offset from the current date to determine the cut-off date for archived data.

**Weeks** A number of weeks. This value is combined with any values in **Years**, **Months**, and **Days**, and offset from the current date to determine the cut-off date for archived data.

**Days** A number of days. This value is combined with any values in **Years**, **Months**, and **Weeks**, and offset from the current date to determine the cut-off date for archived data.

**Note:** You can provide a **Specific Date** or a relative date, using values in **Years**, **Months**, **Weeks**, and **Days**, but not both.

Enter the desired date information and use END to return to the Specify Archive Criteria for AD panel. The term "DATE_CRITERIA" and the specified criteria will appear in **Criteria** on that panel.

### Available Commands

The primary commands available on the Specify Archive Criteria for AD panel include:

- BOTTOM
- BROWSE
- CANCEL
- CAPS
- CLEAR
- COLUMNS
- DC
- DOWN
- EDIT
- END
- EXPAND
- IDX
- LEFT
- OPTIONS
- RESET
- SAVE
- SEL CRIT
- SQL
- TOP
- UP
- ZOOM

**Note:** The EDIT command is available if Access is installed with Archive.

## Selection Criteria

Like archive criteria, selection criteria limit the set of rows extracted from each table in an Archive or Extract Process. You can also use selection criteria to manage the display of rows when browsing or editing data, or during a Point-and-Shoot session.

The Specify Selection Criteria for AD panel is used to specify the selection criteria for each table. (You can use the ARC, SQL, or COL commands to view or edit other criteria or data display settings from this panel.)

## SEL Command

Use the Specify Selection Criteria for AD panel to provide selection criteria for a table. To invoke this panel, use the SEL primary or line command from the Select Tables/Views for AD panel. For example, to specify selection criteria for the ITEMS table, enter the primary command:

```
SEL ITEMS
```

The prompts for selection criteria are displayed.

```
-- Specify Archive Criteria for AD: GRP.FOPDEMO.AD ----------------------------
Command ===>                                                  Scroll ===> PAGE

Table Name: FOPDEMO.ITEMS                                Col 1 of 6     <<MORE
Combine All Column Criteria by ==> A   (A-AND, O-OR)

Cmd     Column Name                        Selection Criteria
---  ------------------  ------------------------------------------------------
***  ******************************** TOP ************************************
___  ITEM_ID
___  ITEM_DESCRIPTION
___  CATEGORY
___  RATING              = :SUB_RATING
___  UNIT_PRICE          > 100
___  ON_HAND_INVENTORY
***  ***************************** BOTTOM ***********************************
```

*Figure 23. Specify Selection Criteria for AD*

## Panel

The Specify Selection Criteria for AD panel includes:

**Table Name**
The name of the current table or view.

**Combine All Column Criteria by**
The logical operator to apply to all archive, selection, and SQL WHERE criteria for the table. Specify:

**A** Combine all criteria with AND (default).

**O** Combine all criteria with OR.

**Cmd** The line command entry area. Valid commands are:

**M** *or* **MM**
Move a line to rearrange the order of columns for an edit, browse, or Point-and-Shoot data display. (This rearrangement has no effect on data selection or sort sequence.)

**A** *or* **B** Destination for a Move line command:
**A** = After
**B** = Before

**Z** Zoom to display all information for the column on a single panel.

**Column Name**
The name of the column. Initially, all columns are listed in the order defined to DB2. The names cannot be edited and columns cannot be inserted or deleted on this panel.
If the complete list cannot be displayed, you can scroll it. Notation next to **Table Name** indicates the relative position of the first visible column name. This notation is in the form, Col *x* of *y*, where *x* is the relative position of the first visible name and *y* the total number of names in the list.

**Selection Criteria**

Selection criteria for the column. Any valid SQL predicate is allowed:

**Valid Operators**

= <> ¬=
> >= ¬>
< <= ¬<
IN (a,b,c,d...) NOT IN (a,b,c,d...)
IS NULL IS NOT NULL
LIKE NOT LIKE
BETWEEN x AND y NOT BETWEEN x AND y

## Specify the Criteria

Specify selection criteria by supplying an appropriate operator and a corresponding value or a list of values. For example, enter > **100100** in **Selection Criteria** for the UNIT_PRICE column to select all items with a unit price greater than $100.00.

You can also specify a column name as the operand for selection criteria, using the LIST COL command. Enter the LIST COL command at the Command prompt, specify an operator for the column, and with the cursor positioned on that column, press ENTER to display a selection list of columns in the current table. The selected column name is added after the operator.

You can also use a substitution variable as the operand for selection criteria, preceding the substitution variable name with a colon (:). You can use the LIST SUBS command to display a list of available substitution variables. For example, to use an Access Definition to select ITEMS rows according to rating, you can define a substitution variable named SUB_RATING and enter **= :SUB_RATING** in **Selection Criteria** for the RATING column. At the time the process is executed, you are prompted for a rating value used to select rows for processing.

**Note:**

- Zoom is a useful way to review characteristics of a column, such as data type or null eligibility, needed to define selection criteria. From the Zoom display, you can enter any desired selection criteria. Use the Zoom primary command to return to the Specify Selection Criteria for AD panel.
- A substitution variable must be defined on the Substitution Variable Display panel. (For more information, see “Substitution Variables” on page 70.)
- Selection criteria applied to Legacy data described in a Legacy Table must use an internal SQL described in the *MOVE User Manual*, SQL Grammar for Legacy.

See Figure 23 on page 50 for an example of selection criteria specifications.

## Available Commands

The primary commands available on the Specify Selection Criteria for AD panel include:

- ALL
- ARC
- BOTTOM
- BROWSE
- CANCEL
- CAPS
- COLUMNS
- DOWN
- EDIT

- END
- EXPAND
- LEFT
- LIST COLUMNS
- LIST SUBS
- OPTIONS
- RESET
- SAVE
- SQL
- TOP
- UP
- ZOOM

**Note:** The EDIT command is available only if ACCESS is licensed. The ARC command is available for ARCHIVE only.

### Criteria Complete

When criteria for the current table are completed, use END to return to the Select Tables/Views for AD panel.

## SQL WHERE Clause Specifications

Sometimes selection criteria for one or more columns in a table cannot adequately meet your requirements. For example, you may need a combination of OR and AND logical operators, rather than combining all criteria with one. For those situations, use the SQL WHERE Clause panel to specify the WHERE clause for an SQL SELECT statement.

### SQL Commands

From the Select Tables/Views for AD panel, use the SQL primary or line command to invoke the SQL WHERE Clause panel. To indicate the desired table for the primary command, use the name as an operand or position the cursor on the name in the Table List. From the Describe Columns for AD, Specify Selection Criteria for AD, or Specify Archive Criteria panel, use the SQL primary command with no operands.

```
--------------- Enter an SQL WHERE Clause for a Table or View -----------------
Command ===>                                                 Scroll ===> PAGE

       SELECT ... FROM FOPDEMO.ORDERS
Cmd       Correlation Name ===> A                  WHERE             1 of 8
--- ------------------------------------------------------------------------
*** ******************************** TOP ***********************************
___ ORDER_SALESMAN LIKE 'SW%' and CUST_ID <'10000'
___ or ORDER_DATE <'2005-01-31'
___
___
___
___
___
___

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
An optional correlation name can be entered to refer to the base table
```

*Figure 24. Specify SQL WHERE Clause*

For reference, the name of the table is provided in the SELECT...FROM heading near the beginning of the panel. This heading includes an ellipsis (...) following SELECT to indicate the column list, which is generally too long for the available space.

## Correlation Name

To simplify typing the table name in the SQL WHERE clause, you can enter a 1- to 128-character Correlation Name or a substitute for the fully qualified table name in the FROM clause. A correlation name is especially useful as a prefix for names of columns when the SQL references columns of the same name from different tables.

## WHERE area

Type the desired SQL in the **WHERE** area of the panel. The scrollable area displays the SQL in segments of 8 lines of 72 positions each. A maximum of 425 lines is available for the SQL and documenting comments. Notation before the WHERE area indicates the number of lines and the relative position of the first displayed line. Standard DB2 conventions apply to comments; each line must begin with two hyphens (--).

## Column Names

You can specify an SQL WHERE clause using a column name. Use the LIST COLUMNS command to display a selection list of columns in the table. You can scroll this list and use the S line command to select a column name. The selected column name is added to the end of any text in the SQL WHERE clause.

## Substitution Variables

You can specify an SQL WHERE clause using a substitution variable that has been defined on the Substitution Variable Display panel. The name of the substitution variable must be preceded by a colon (:). (For more information, see “Substitution Variables” on page 70.) Use the LIST SUBS command to display a selection list of substitution variables for the table. You can scroll this list and use the S line command to select a substitution variable. The selected substitution variable is added to the end of any text in the SQL WHERE clause.

## Edit SQL WHERE Clause

Unless the SQL is applied to a Legacy Table, use standard DB2 SQL format for the SQL WHERE clause. When you type, any leading or trailing spaces on each line are maintained: only the spaces at the end of the last line of the SQL WHERE clause are deleted. To facilitate editing, a **Cmd** area is provided for each line. The functions and corresponding line commands are:

**Copy** C*n*, CC

**Insert** I*n*

**Repeat**
R*n*, RR

**Delete** D*n*, DD

**Move** M*n*, MM

Use A or B to indicate the destination for a Copy or Move line command.

Use the D line command to delete lines in an SQL WHERE clause. You can also replace the clause with blanks or press the Erase EOF key for each line of text.

**Note:** An SQL WHERE clause applied to Legacy data described in a Legacy Table must use an internal SQL described in the *Move User Manual*, section on SQL Grammar for Legacy Tables.

## Available Commands

The following primary commands are available:

- ARC
- BROWSE
- CANCEL
- COLUMNS
- EDIT
- END
- LIST COLUMNS
- LIST SUBS
- OPTIONS
- SAVE
- SEL CRIT
- SQLEDIT

**Note:** The EDIT command is available if Access is licensed. ARC is available for Archive only.

You can use the SQLEDIT command to invoke an ISPF edit session for the displayed SQL WHERE clause. All standard ISPF facilities are available. Use the ISPF COPY command to insert data from a file and edit as desired.

Use END to terminate the ISPF session and redisplay the SQL WHERE Clause panel. The edited data is inserted into the panel.

## Specification Complete

After you complete the SQL WHERE clause, use END to save the current specifications and terminate editing.

Use the SAVE command to save the SQL WHERE clause and continue editing. You may want to use SAVE periodically to validate the SQL WHERE clause.

END or SAVE validates the SQL WHERE clause before redisplaying the previous panel. The clause must be 425 lines or fewer and must be acceptable to DB2. You must resolve any errors before you can save the SQL WHERE clause and exit this panel.

When the Select Tables/Views for AD panel is redisplayed, "SQL" appears under **Status** to indicate that an SQL WHERE clause is specified for the table.

## Manage Data Displays

Use the Describe Columns for AD panel to review column characteristics and format the display of data in a browse, edit, or Point-and-Shoot session. The column settings do not affect any Archive, Extract, or Compare Process that uses the Access Definition.

Scroll LEFT from the Specify Archive Criteria for AD panel or the Specify Selection Criteria for AD panel to view the Describe Columns for AD panel. Alternatively, you can use the COL command from the Select Tables/Views for AD panel, the SQL WHERE Clause panel, the Specify Archive Criteria for AD panel, or the Specify Selection Criteria for AD panel.

The column specification prompts are shown in the following figure:

```
-- Describe Columns for AD: GRP.FOPDEMO.AD -----------------------------------
Command ===>                                                Scroll ===> PAGE

Table Name: FOPDEMO.CUSTOMERS                             Col 1 of 9     MORE>>

                         Disp  Acc  -Sort--  -Heading-
Cmd     Column Name      Y/N   S/U  Lvl A/D  N/L L/R/C   Data Type    Null Crit
---  ------------------  ----  ---  --- ---  --- ----- -------------- ---- ----
***  ******************************** TOP ************************************
___  CUST_ID              Y     U    __   _    N    C   CHAR(5)        NO   NO
___  CUSTNAME             Y     U    __   _    N    C   CHAR(20)       NO   YES
___  ADDRESS              Y     U    __   _    N    C   VARCHR(50)     NO   NO
___  CITY                 Y     U    __   _    N    C   VARCHR(15)     NO   NO
___  STATE                Y     U    __   _    N    C   CHAR(2)        NO   YES
___  ZIP                  Y     U    __   _    N    C   CHAR(5)        YES  NO
___  YTD_SALES            N     U    __   _    N    C   DECIMAL(7,2)   NO   NO
___  SALESMAN_ID          Y     U    __   _    N    C   CHAR(6)        YES  NO
___  PHONE_NUMBER         Y     U    __   _    N    C   CHAR(10)       YES  NO
***  ***************************** BOTTOM **********************************
```

*Figure 25. Describe Columns for AD*

### Panel

The Describe Columns for AD panel includes:

**Cmd** The line command entry area. Valid commands are:

- **M** Move a column in the list to change the display order.
- **Z** Toggle between the list of all columns and a single column display. (Not available for Compare browse.)

**Column Name**
The name of each column that appears in the table.

**Disp** Select the columns to appear in the data display. Specify:

- **Y** Display the column (default).

**N** Do not display the column.

**Note:**

- You must display at least one column and, to insert a row during an Access edit session, you must display any columns that are part of the primary key or a unique index.
- The initial setting for all columns is Y. You can use the DISPLAY N command to set all columns to N, or use DISPLAY Y to set all columns to Y.
- When a new column is added to a table, the default Disp value is based on settings for other columns. If all columns are set to Y, the default setting for a new column is also Y. If any column is set to N, the default for a new column is N.

**Acc** Establish edit privileges for each column. (If Access is not installed, this prompt is not displayed.)

**S** Values are display only.

**U** Values can be updated.
This setting can differ from the privilege defined in the DB2 Catalog. In any attempt to edit values using this Access Definition, the more restrictive setting is enforced.

**Sort** The display order for rows, based on the values specified for one or more columns. If you do not provide values, the rows are displayed in an unpredictable order. (You can use sort criteria with a column for which **Disp** is N.) Specify sort criteria as two values:

**Lvl** A numeric value indicating the order of priority of this column in arranging the rows. Specify unique and consecutive values from 1 - 64, with 1 indicating the highest priority.

**A/D** Sort direction as (A)scending or (D)escending. Ascending is the default.

**Heading**
The source of a column heading and its position in the data display. Specify the source as:

**N** The column name (default).

**L** The label defined in the DB2 Catalog.

Specify the position as:

**L** Left justified.

**R** Right justified.

**C** Centered (default).

**Data Type**
The data type and dimension of each column (for display only). Generally, the DB2 or Legacy data type is displayed. For example, CHAR(15) and DECIMAL (10,2) are displayed as such. However, due to space limitations, the following data type specifications are abbreviated. For each data type, the DB2 format is shown in bold, followed by the abbreviated format:

**VARCHAR**
VARCHR

**LONG VARCHAR**
LVARCHR

**FLOAT 4-byte length**
SINGLE FLOAT

**FLOAT 8-byte length**
DOUBLE FLOAT

**GRAPHIC**
GR

**VARGRAPHIC**
VGR

**LONG VARGRAPHIC**
VGR

**CHAR FOR MIXED DATA**
CH

**CLOB FOR MIXED DATA**
CLOB

**BINARY**
BIN

**VARBINARY**
VARBIN

**Note:** See "Editing and Browsing with Unsupported Data Types" on page 58

**Null** Null eligibility indicator (for display only). Possible values:

**YES** The column is NULL eligible.

**NO** The column is not NULL eligible.

**Crit** Selection criteria indicator (for display only). Possible values:

**YES** Criteria are specified.

**NO** Criteria are not specified.

## Available Commands

The primary commands available on the Describe Columns for AD panel include:

- ARC[1]
- BOTTOM
- BROWSE[2]
- CANCEL
- DISPLAY
- DOWN
- EDIT[2]
- END
- EXPAND
- FIND
- OPTIONS
- RESET
- RIGHT[3]
- SAVE[3]
- SEL CRIT[3]
- SQL[3]
- TOP
- UP
- ZOOM[3]

**Note:**

- [1]Archive only
- [2]Access only
- [3]Not available for Compare browse

### Editing and Browsing with Unsupported Data Types

During an Edit or Browse session, DB2 tables that contain columns with unsupported data types are managed as follows.

#### Edit

When using the Edit function:

- For tables containing columns with any unsupported data types, the Edit function is disabled and the message "UNSUPPORTED DATA TYPE" appears.
- For tables containing no columns with unsupported data types, the Edit function continues processing.

#### Browse

When using the Browse function:

- For tables containing some columns with supported data types and some columns with unsupported data types, the data type is displayed on columns with supported data types. The columns with unsupported data types are deselected from view and the data type is listed as "UNSUPPORTED."
- For tables containing columns with unsupported data types only, the Browse function is disabled and the message "UNSUPPORTED DATA TYPE" appears.

## Data Types Supported

Data types supported by Optim are listed as follows.

For detailed information on limitations or special considerations for a data type, see the documentation for the specific function or request. The following data types are supported for Optim processes with exceptions as noted:

- BINARY
- VARBINARY
- CHAR
  - When used with SEQ function, values for start and step are limited to the range -9223372036854775808 and 9223372036854775807.
  - When used with the RAND function, low and high values range from -9223372036854775808 to 9223372036854775807.
- VARCHAR
- LONG VARCHAR
- DATE
- TIME
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
  - Cannot be used in an Archive Action variable
  - Not supported by ODM Optim Connect
- GRAPHIC
- VARGRAPHIC
- LONG VARGRAPHIC
- INTEGER
- SMALL INT

- BIGINT
  - Cannot be used with: AGE function, substitution variables, TRANS CCN function
  - When used with RAND_LOOKUP function, maximum value of limit is 2,000,000,000.
  - When used with HASH_LOOKUP function, maximum value of seed is 2,000,000,000.
  - When used with SEQ function, values for start and step are limited to the range -9223372036854775808 and 9223372036854775807.
  - When used with the RAND function, low and high values range from -9223372036854775808 to 9223372036854775807.
  - In TRANS SSN processing, only 32 bits of the SSN value will be used for the source and destination values.
- DECIMAL
  - When used with SEQ function, values for start and step are limited to the range -9223372036854775808 and 9223372036854775807.
  - When used with the RAND function, low and high values range from -9223372036854775808 to 9223372036854775807.
- FLOAT
  - When used with SEQ function, values for start and step are limited to the range -9223372036854775808 and 9223372036854775807.
  - When used with the RAND function, low and high values range from -9223372036854775808 to 9223372036854775807.
- DECFLOAT
  - Cannot be used with: AGE, RAND, RAND_LOOKUP, SEQ, TRANS CCN, TRANS SSN functions; with substitution variables; in relationships
  - Values for this data type are displayed as: "***...***"
  - Literal values for this data type cannot be used as panel input, in control statements, selection criteria, or with FIND, CHANGE, or ONLY commands.
  - Cannot create an Archive Index from columns containing this data type.
  - No support for Open Data Manager (ODM)
  - When used in Column Maps, column data type and precision must match.
- BLOB, CLOB, DBCLOB
  - Cannot be used as row selection criteria or with data transformation library functions.
  - Access editor and Browse utility display only the first 32K bytes of data. Data cannot be modified.

## Zoom Column Information

Use a ZOOM command to switch between displaying information for a table on the Describe Columns for AD, Specify Selection Criteria for AD, or Specify Archive Criteria panel to displaying information for a single column in a given table. From these panels, you can also use the Z line command to zoom the display for a specific column. The ZOOM primary command also can be assigned to a function key.

The zoomed display is shown in the following figure:

```
-- Describe Columns for AD: GRP.FOPDEMO.AD ----------------------------------
Command ===>                                                Scroll ===> PAGE

Table Name: FOPDEMO.ORDERS                                        Col 8 of 8
Combine All Column Criteria by ==> A   (A-AND, O-OR)

Column Name                     : ORDER_SHIP_DATE
Data Type                       : CHAR(8)  NOT NULL
Updatable by DB2                : YES

Display This Column          ===> Y             (Y-Yes, N-No)
Sort Level                   ===> __            (1-64 or blank)
Ascending or Descending      ===> _             (A, D or blank)
Column Heading Position      ===> C             (L-Left, R-Right, C-Center)
Column Heading               ===> N             (L-Use Label, N-Use Name)
Access Rights                ===> U             (S-Select, U-Update)
Index Column                 ===> N             (D-Dense, S-Sparse, N-No)
Criteria ===>  = '2005-03-01'
```

*Figure 26. Zoomed Column Information*

This display includes information from the Specify Archive Criteria for AD panel, the Specify Selection Criteria for AD panel, or the Describe Columns for AD panel. Notation next tothe **Table Name** indicates the relative position of the currently displayed column and the total number of columns in the table.

The prompt for **Access Rights** is displayed only if Access is available and you are defining the Access Definition by means of Option 5 ADS or Option 6 DEFINITIONS from the **Main Menu**.

Although **Combine All Column Criteria** is displayed with each column in zoom mode, it applies to the entire table. Modifying the value in a zoomed column display modifies the value for the table.

Use the ZOOM primary command to return to the column list display.

### Available Commands

Use the scroll commands (UP, DOWN, TOP, and BOTTOM) to obtain a zoomed display of the previous, next, first and last columns in the table.

## Archive Actions

Archive provides a facility that allows you to define actions to be executed at one or more predefined points (action events) during an Archive, Delete, or Restore Process. Each action event can have only one SQL statement associated with it, although action events can share an SQL statement association.

**Note:** Any Restore Actions in the Access Definition can be overridden, disarmed, or augmented in the Table Map for the Restore Process. For details, see "Archive Actions" on page 271.

Action SQL statements are different from and should not be considered interchangeable with SQL WHERE clauses created to select data for processing. (See "SQL WHERE Clause Specifications" on page 52 for information about this feature.)

### ACTIONS Primary Command

To define an Archive Action, invoke the Select an Action to Be Defined panel from the Select Tables/Views for AD panel using the ACTIONS primary command with the desired table name as an operand. You can also enter the ACTIONS primary command and position the cursor on the desired table name in the Table List before pressing ENTER.

## ACT Line Command

Alternatively, use the ACT line command to display the scrollable Select an Action to Be Defined panel.

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD ---------------------------------
Command ===>                                                 Scroll ===> PAGE

 Pr +--------- Select an Action To Be Defined for FOPDEMO.CUSTOMERS -----+ UBS
    |                                                                    |
    |                                                          1 of 12   |
Def | Cmd              Action Description            Active  Shr         | E>>
Sta | --- ----------------------------------------- ------  ---         |  >>
Sta | *** ********************** TOP ************************          |  >>
    | ___ Before Extract of the First Row From a Table  N     N          |
Cmd | ___ Before Extract of a Row From a Table          N     N          |
--- | ___ After Extract of a Row From a Table           N     N          |
*** | ___ After Extract of Last Row From a Table        N     N          |
ACT | ___ Before Delete of the First Row From a Table   N     N          |
___ | ___ Before Delete of a Row From a Table           N     N          |
___ | ___ After Delete of a Row From a Table            N     N          |
___ | ___ After Delete of the Last Row From a Table     N     N          |
*** | ___ Before Restore of the First Row To a Table    N     N          |
    |                                                                    |
    | Line Commands: (S)elect, (INF)ormation, (CLR)Clear                 |
    |                                                                    |
    +--------------------------------------------------------------------+
```

*Figure 27. Select an Action To Be Defined*

## Panel

The Select an Action To Be Defined panel includes:

**Cmd** The line command entry area. Valid commands are:

- **S** Select the action event.
- **INF** Display a list of action events that share the action statement.
- **CLR** Clear the action statement association for the action event.

**Action Description**

Event for the action. Select:

- Before Extract of the First Row From a Table
- Before Extract of a Row From a Table
- After Extract of a Row From a Table
- After Extract of Last Row From a Table
- Before Delete of the First Row From a Table
- Before Delete of a Row From a Table
- After Delete of a Row From a Table
- After Delete of the Last Row From a Table
- Before Restore of the First Row To a Table
- Before Restore of a Row To a Table
- After Restore of a Row To a Table
- After Restore of the Last Row To a Table

**Active Indicator for association with action statement.**

- **N** No action statement is associated with the event.

**Y** An action statement is associated with the event.

**Shr** **Indicator for shared status of an action statement.**

**N** No associated action statement or action statement is not shared.

**Y** Associated action statement is shared with another action.

**Y*** Associated action statement is defined for this action and shared with another action.

## Shared SQL Statements

When an action statement is shared by action events, an asterisk after the Y under **Shr** indicates the primary action event, or the event that owns the shared action statement. Use the INF line command, as shown in the figure, to list action events that share the statement associated with the primary action.

```
+--------- Select an Action To Be Defined for FOPDEMO.CUSTOMERS -----+
|                                                                    |
|                                                        1 of 12     |
| Cmd              Action Description            Active  Shr        |
| --- -----                                                          |
| *** ***** +-----Before Extract of the First Row From a Table-----+ |
| INF Befor |                SQL Shared With                1 OF 2 |  |
| ___ Befor | ---------------------------------------------       |  |
| ___ After | ******************** TOP ********************       |  |
| ___ After | After Extract of a Row From a Table                 |  |
| ___ Befor | Before Delete of the First Row From a Table         |  |
| ___ Befor | ****************** BOTTOM ******************       |  |
| ___ After +-----------------------------------------------------+ |
| ___ After                                                          |
| ___ Before Restore of the First Row To a Table    N     N        |
| ___ Before Restore of a Row To a Table            N     N        |
| ___ After Restore of the First Row To a Table     N     N        |
| ___ After Restore of a Row To a Table             N     N        |
|                                                                    |
| Line Commands: (S)elect, (INF)ormation, (CLR)Clear                 |
|                                                                    |
+--------------------------------------------------------------------+
```

*Figure 28. Shared Actions List at INF Command*

## Define the Action

Use the S or SEL line command to select one or more action events. Archive displays the Enter an SQL Statement panel for each selection, in sequence.

```
------- Enter an SQL Statement to be used by the Action Being Defined ---------
Command ===>                                                 Scroll ===> PAGE

              Before Extract of the First Row From a Table
Cmd                                                                    1 of 5
--- ------------------------------------------------------------------------
*** ******************************** TOP ********************************
___
___
___
___
___
*** ***************************** BOTTOM ********************************

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the LIST VARIABLES command to add Action Variables, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
Use the SHARESQL command to select an Action whose SQL you wish to use
Use the COPYSQL command to select an Action whose SQL you wish to copy
```

*Figure 29. SQL Statement*

Use this panel to enter the complete action statement or call to a stored procedure that is executed when the action event occurs during processing. For example, you might use an action statement to record deletions by inserting the primary key value into a table before deleting the row. As another example, you can place an indicator in a row that has been restored so that it will not be archived a second time or can be deleted in normal database maintenance procedures.

Type the desired SQL in the panel. The scrollable area displays the SQL in segments of 8 lines of 72 positions each. A maximum of 425 lines is available for the SQL and documenting comments. Notation next to it indicates the number of lines and the relative position of the first visible line.

Standard DB2 conventions apply to comments; each line must begin with two hyphens (--).

**Note:** You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

## Line Commands

You can perform several editing functions using line commands. The functions and corresponding line commands are:

**Copy** C*n*, CC

**Insert** I*n*

**Repeat**
R*n*, RR

**Delete** D*n*, DD

**Move** M*n*, MM

Use A (After) or B (Before) to indicate the destination for a Copy or Move line command.

## Column Variables

A column variable can be used in the statement for any action event except one executed before processing the first row or after processing the last row. The column variable must be preceded by a colon (:) in the statement.

## Substitution Variables

You can use a substitution variable that has been defined on the Substitution Variable Display panel. The name of the substitution variable must be preceded by a colon (:). (For more information, see "Substitution Variables" on page 70.)

**Note:** If you created a Substitution Variable with the same name as a Column Variable (e.g., :ZIP), you may want to edit the substitution variable name if you want to use it in an SQL Statement because Optim will assume the variable is a Column Variable, not a Substitution Variable, as intended.

## Action Variables

Action Variables are built-in functions that can be used to return information about the specific Archive, Delete, or Restore Action. An Action Variable must be preceded by a colon (:) in the statement.

**For users of action variables with releases of Optim prior to 7.2:** Optim 7.2 increased processing limits and changed the precision for some action variables. In Optim 6.1 and earlier, precision was **integer** for the following variables. With Optim 7.2 and later, precision for the listed variables is **bigint**. Depending on the way in which you use these variables, you may need to make changes to accommodate the difference in data type. For example, if a variable inserts a value into a DB2 table, ensure that the destination column is defined to allow a bigint value.

FOP_TOTAL_ROWS_EXTRACTED

FOP_TBL_ROWS_EXTRACTED

FOP_TOTAL_ROWS_PROCESSED

FOP_TOTAL_ROWS_REMAINING_TO_BE_PROCESSED

FOP_TOTAL_ROWS_IN_ERROR

FOP_TBL_ROWS_PROCESSED

FOP_TBL_ROWS_REMAINING_TO_BE_PROCESSED

FOP_TBL_ROWS_IN_ERROR

The following General Variables are available to all action statements.

**FOP_ACTION**
A code that identifies the event for the action.
Return Value Type: integer

**FOP_ACTION_TEXT**
A text string that identifies the event for the action.
Return Value Type: char (20)

**FOP_ARCHIVE_FILE_NAME**
Name of the Archive File being processed.
Return Value Type: char (44)

**FOP_INDEX_FILE_NAME**
Name of any index file for the Archive File.
Return Value Type: char (44)

**FOP_GROUP_NAME**
Group name, if any, for the Archive File.
Return Value Type: char (8)

**FOP_USER_ID**
TSO ID for the user that created the Archive File.
Return Value Type: char (8)

**FOP_SERVER_NAME**
DB2 subsystem on which the operation is being executed.
Return Value Type: char (4)

**FOP_ARCHIVE_DESC**
Description, if any, for the Archive File.
Return Value Type: char (40)

**FOP_START_DATETIME**
Date and time the current operation began.
Return Value Type: timestamp

**FOP_CURRENT_DATETIME**
Current date and time.
Return Value Type: timestamp

**FOP_START_TABLE_ID**
Creator ID of the Start Table.
Return Value Type: char (128)

**FOP_START_TABLE_NAME**
Name of the Start Table.
Return Value Type: char (128)

**FOP_ACCESS_DEF_NAME**
Three-part name of the Access Definition, separated by periods.
Return Value Type: char (30)

**FOP_BATCH_ONLINE_FLAG**
B for batch execution; O for online execution.
Return Value Type: char (1)

**FOP_BATCH_JOBNAME**
Job name if batch execution; TSO User ID if online execution.
Return Value Type: char (8)

**FOP_PROCESS_TYPE**
A if Archive operation; B if Archive operation with delete (whether or not deferred); R if Restore operation; D if Delete operation.
Return Value Type: char (1)

**FOP_RETURN_CODE**
For CALL statements only. Return code supplied by the called stored procedure.
Return Value Type: integer

**FOP_RETURN_MESSAGE**
For CALL statements only. Optional text message supplied by the called stored procedure if FOP_RETURN_CODE is 2.
Return Value Type: char (80)

FOP_ACTION and FOP_ACTION_TEXT return the following values:

**FOP_ACTION 0**
FOP_ACTION_TEXT BefExtFirstRow

**FOP_ACTION 2**
FOP_ACTION_TEXT BefExtRow

**FOP_ACTION 3**
FOP_ACTION_TEXT AftExtRow

**FOP_ACTION 4**
FOP_ACTION_TEXT AftExtLastRow

**FOP_ACTION 5**
FOP_ACTION_TEXT BefDelFirstRow

**FOP_ACTION 7**
FOP_ACTION_TEXT BefDelRow

**FOP_ACTION 8**
FOP_ACTION_TEXT AftDelRow

**FOP_ACTION 9**
FOP_ACTION_TEXT AftDelLastRow

**FOP_ACTION 11**
FOP_ACTION_TEXT BefRestFirstRow

**FOP_ACTION 13**
FOP_ACTION_TEXT BefRestRow

**FOP_ACTION 14**
FOP_ACTION_TEXT AftRestRow

**FOP_ACTION 15**
FOP_ACTION_TEXT AftRestLastRow

The following Extract Variables are available to Extract action statements.

**FOP_TOTAL_ROWS_EXTRACTED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows processed at this point. (See Note.)
Return Value Type: bigint

**FOP_ARCHIVE_CRITERIA**
First 220 bytes of criteria used in the Archive operation. If no criteria were specified, returns a zero-length string.
Return Value Type: varchar (220)

**FOP_SRCTBL_CREATOR_ID**
Creator ID for the source table being processed.
Return Value Type: char (128)

**FOP_SRCTBL_NAME**
Name of the source table being processed.
Return Value Type: char (128)

**FOP_TBL_ROWS_EXTRACTED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the table being processed. (See Note.)
Return Value Type: bigint

**FOP_REFERENCE_TABLE**
Y if table is a reference table, N if not.
Return Value Type: char (1)

**FOP_ARCHIVE_DELETE_DATA**
Y if delete after archive was specified for the table, N if not.
Return Value Type: char (1)

The following Restore/Delete Variables are available to Restore and Delete action statements.

**FOP_ARCHIVE_ID**
A unique and arbitrary number of no logical significance that is assigned to the Archive File

being processed. Since this value is unique for each Archive File, it can serve as a DB2 index, if needed.
Return Value Type: integer

**FOP_TOTAL_ROWS_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows processed. (See Note.)
Return Value Type: bigint

**FOP_TOTAL_ROWS_REMAINING_TO_BE_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows remaining to be processed. (See Note.)
Return Value Type: bigint

**FOP_TOTAL_ROWS_IN_ERROR**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows in error.
Return Value Type: bigint

**FOP_SRCTBL_CREATOR_ID**
Creator ID of the source table being processed.
Return Value Type: char (128)

**FOP_SRCTBL_NAME**
Name of the source table being processed.
Return Value Type: char (128)

**FOP_DSTTBL_CREATOR_ID**
(For Restore only.) Creator ID for the destination table being processed.
Return Value Type: char (128)

**FOP_DSTTBL_NAME**
(For Restore only.) Name of the destination table being processed.
Return Value Type: char (128)

**FOP_TBL_ROWS_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the table that are processed. (See Note.)
Return Value Type: bigint

**FOP_TBL_ROWS_REMAINING_TO_BE_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the table that remain to be processed. (See Note.)
Return Value Type: bigint

**FOP_TBL_ROWS_IN_ERROR**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the table that are in error at this point.
Return Value Type: bigint

**FOP_AFTER_ROW_STATUS**
Status code returned by DB2 after processing a row.
Return Value Type: integer

**Note:** Any row count value (e.g., FOP_TBL_ROWS_PROCESSED) that is returned does not include the current row for **Before Row** events, but does include the current row for **After Row** events. For example, FOP_TBL_ROWS_PROCESSED for the first row of a table returns 0 for **Before Row** actions, and 1 for **After Row** actions.

An example of Action Variable usage is shown in the following figure:

```
------- Enter an SQL Statement to be used by the Action Being Defined ---------
Command ===>                                                Scroll ===> PAGE

              Before Extract of the First Row From a Table
Cmd                                                                    1 of 3
--- ------------------------------------------------------------------------
*** ******************************** TOP ***********************************
___ INSERT INTO FOPDEMO.ARCHIVE_ACTION
___ (ARCHIVE_CUSTID, ARCHIVE_FILE)
___ VALUES (CUST_ID, :FOP_ARCHIVE_FILE_NAME);
*** ***************************** BOTTOM *********************************

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the LIST VARIABLES command to add Action Variables, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
Use the SHARESQL command to select an Action whose SQL you wish to use
Use the COPYSQL command to select an Action whose SQL you wish to copy
```

*Figure 30. SQL Statement Using Action Variable*

Before each row is extracted, this action statement inserts CUST_ID and ARCHIVE_FILE name values (using the Action Variable FOP_ARCHIVE_FILE_NAME) into table FOPDEMO.ARCHIVE_ACTION.

## Calls to Stored Procedures

An Action statement can call a stored procedure using one or both parts of the name (i.e., *userid.procname*). If one part is used, the current User ID is assumed as the prefix.

Parameters passed with the CALL statement must be enclosed in parentheses. The number of parameters must be the same as the number in the procedure and must be passed in the same order. If the stored procedure does not require parameters, the CALL statement can pass an empty parameter list ( ) or omit the parentheses.

The CALL statement parameter list may include column variables, substitution variables, Action Variables, and constants.

- Type a column variable as a column name preceded by a colon (for example, :STOCK_NUMBER). The data type of the column or substitution variable must be the same as that for the parameter in the stored procedure and the variable length must equal or exceed that for the parameter.
- The column names and Optim variables entered in the CALL statement on the Enter an SQL Statement panel are issued by Archive in an SQLDA (e.g., CALL *procname* USING DESCRIPTOR *sqlda*).
- An error occurs if a column referenced by a column variable is null eligible unless, when defining the stored procedure to DB2, you specify PARAMETER STYLE as DB2SQL or GENERAL WITH NULLS.
- For null eligible fields, Archive automatically sets the parameters for a null indicator. When PARAMETER STYLE GENERAL WITH NULLS is specified, the null indicator is an array passed as a parameter in the stored procedure.
- The length of a parameter for an Action Variable must accommodate the Action Variable.
- Constants for CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, VARBINARY, DATE, TIME, and TIMESTAMP columns must be in single quotes. Constants for INTEGER, SMALLINT, and

DECIMAL cannot be in quotes. The length of a parameter for a character, graphic, or binary constant must accommodate the constant. The parameter lengths for DATE, TIME, and TIMESTAMP constants are 10, 8, and 26 bytes, respectively.

A stored procedure called by an action statement can cause archive processing to skip a row or to terminate the process by passing a value in the Action Variable, FOP_RETURN_CODE. Valid FOP_RETURN_CODE values are:

**0** Normal return (default).

**1** Skip the row. This code can be returned by a stored procedure called by a **Before Row** statement. An error occurs if this code is returned by a stored procedure called by a **Before First Row** or **After** Action statement.

**2** Terminate the process. An optional, 80-character message can be passed in the FOP_RETURN_MESSAGE Action variable.

**3** Normal return and Archive performs a DB2 commit at that point.

**Note:** If a stored procedure called by an Extract Action is intended to return a 3 in FOP_RETURN_CODE, the parameter list must be followed by **WITHHOLD**, outside the parentheses, such as:
CALL *procname*(*parm1*,*parm2*... ) WITHHOLD

**4** Return the table to a known state (i.e., rollback) and skip the row. (After Extract of Last Row, Delete or Restore Action events only.)

## LIST COLUMNS

Use the LIST COLUMNS primary command to display a list of column variables for the table. You can scroll this list and use the S line command to select a column variable from it. The selected column variable is added to the end of the action statement.

## LIST SUBS

Use the LIST SUBS primary command to display a list of substitution variables. You can scroll this list and use the S line command to select a substitution variable from it. The selected substitution variable is added to the end of the action statement.

## LIST VARIABLES

Use the LIST VARIABLES command to display a list of Action Variables when formulating an action statement. Use the S line command to select a variable from the list. The selected variable is added at the end of the action statement.

## SQLEDIT

Use the SQLEDIT command to invoke an ISPF edit session for the displayed action statement. All standard ISPF facilities are available. Use the ISPF COPY command to insert data from a file and edit as desired.

Use END to terminate the ISPF session and redisplay the Enter an SQL Statement panel. The edited data is inserted into the panel.

## SHARESQL

Use the SHARESQL command to select an action event that owns the action statement you wish to associate with the current action. A list of available action events is displayed. You can scroll this list and use the S line command to select a desired event. The statement associated with the selected event is

inserted into the action statement, overwriting anything already entered. You can edit the shared statement. If so, it is changed for all actions for which it is active.

### COPYSQL

Use the COPYSQL command to select an event that owns the action statement you wish to copy to the current event. A list of available events is displayed. You can scroll this list and use the S line command to select a desired event. The action statement for the selected event is inserted at the end of your entry. Editing the action statement does not affect the statement from which it is copied.

### Specification Complete

Use END to save the contents of the SQL Editor and return to the Select an Action To Be Defined panel. Use END again to return to the Select Tables/Views for AD panel. When the Select Tables/Views for AD panel is redisplayed, **Status** contains **ACT** to indicate that an Archive Action has been specified for the selected table.

## Substitution Variables

You can use a substitution variable in selection criteria, an SQL WHERE clause, or an Archive Action statement, but you must first define that variable on the Substitution Variable Display panel.

Use the LIST SUBS command on the Select Tables/Views for AD panel to display the Substitution Variable Display panel. You can provide a default value and a prompt string for each substitution variable you define.

```
                        Substitution Variable Display
Command ===>                                                Scroll ===> PAGE

 Line : EXP, I, D

                                                          Variable 1 of 2


Cmd     Name          Default Value                   Prompt String
--- ------------ ------------------------ ----------------------------------
*** ******************************** TOP ***********************************
___ :SUB_RATING  'PG'                     ENTER THE RATING
___ :TITLE                                ENTER THE MOVIE TITLE
*** ***************************** BOTTOM ***********************************
```

*Figure 31. Substitution Variable Display*

### Panel

The Substitution Variable Display panel includes:

**Cmd** The line command entry area. Valid commands are:

- **D** Delete substitution variable.
- **EXP** Display an expanded pop-up window for the default value of the substitution variable. This display provides two 50-character lines to enter the default value.
- **I** Insert line for new substitution variable.

**Name** Specify the name of the substitution variable, preceded by a colon (:). If you do not prefix the name with a colon, it is inserted automatically.

**Note:** If you use a substitution variable that has the same name as a column variable in an action statement, unpredictable results will occur. Therefore, it is recommended that you do not use a column name as the substitution variable name.

**Default Value**
Specify the default value of the substitution variable.

Values for CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, VARBINARY, DATE, TIME, and TIMESTAMP variables must be enclosed in single quotes. Values for INTEGER, SMALLINT, and DECIMAL variables cannot be enclosed in quotes. You can also specify a column name for the default value; do not enclose the column name in quotes.

Values for DECFLOAT or BIGINT cannot be defined as default values.

To enter or modify a default value that is more than 25 characters, use the EXP line command to display a 100 character area.

Default values are not validated until run time. If a value is the incorrect data type or size for the column, or does not conform to SQL syntax, processing errors may result.

**Note:**

- If you do not provide a default value, you must provide a value at run time, even if the substitution variable is not used in the process.
- If you provide a default value, you will not be prompted for a value at run time and the value will be used for online processing. For batch processing, you can provide the default value or override any default value, using the VAR keyword.

**Prompt String**
Enter any desired text to prompt for the substitution variable value at run time. Type the prompt string exactly as you want it to appear (a maximum of 35 characters). If a default value is not specified, this prompt is displayed on the Default Value panel prior to extracting or archiving data.

You can define any number of substitution variables for an Access Definition but are not required to use the substitution variables in selection criteria, SQL WHERE Clauses, or Archive Actions for the Access Definition. In an online process, you will be prompted for values if you do not provide a default. An error condition will occur in a batch process if no value is provided, either as a default or by using the VAR override.

## Default Value

If you do not provide a default value for a substitution variable on the Substitution Variable Display, the Default Value panel prompts you for a value prior to executing any process that uses it (for example, Extract, Archive, or Point-and-Shoot).

```
------------------------------ Default Value ---------------------------------
Command ===>                                                 Scroll ===> PAGE


    Before the process can continue, you must supply a value
    for the substitution variable listed.

    Variable Name: TITLE

    ENTER THE MOVIE TITLE:
```

*Figure 32. Default Value*

This panel displays the name of the substitution variable in **Variable Name**, and a prompt for entering a value for the substitution variable. (The text for the prompt is taken from **Prompt String**, if specified, on the Substitution Variable Display panel.) You must enter a value for the substitution variable to proceed with the process.

**Note:** Specifying a value on the Default Value panel modifies the Access Definition, adding the new default value to the substitution variable. During an Extract or Archive Process, you are prompted to save the modified Access Definition, if it is permanent, when you exit the Extract or Archive Process menu.

## The Point-and-Shoot Facility

The Point-and-Shoot facility is useful for selecting specific rows from the Start Table, which are used to start an Archive, Compare, Extract, or Restore Process. If you have also defined selection criteria for the Start Table, you can optionally override or supplement the selection criteria with the Point-and-Shoot list.

### Primary Key

Selected Start Table rows are identified by the primary key values; thus, the Start Table must have a primary key.

Optim stores the primary key values in a sequential or partitioned data set that you specify. This data set is sometimes referred to as a Row List. In later Point-and-Shoot sessions, you can add to this Row List or replace it. If the data set does not exist, Optim prompts for the required information and allocates the data set for you. (See Appendix B, "Allocating External Files," on page 417 for more information.) Once saved, the Row List can be used and modified as needed.

### Legacy Tables

Although you can use an external Row List file with a Legacy Table, you cannot use Point-and-Shoot to interactively select legacy data. (Refer to Appendix C, "Create a Row List File," on page 425 for more information.)

### Permanent or Temporary

You can initiate Point-and-Shoot from the Table List when creating or editing an Access Definition or before executing an Archive, Extract, or Compare Process. If you are defining an Access Definition to be used in a subsequent process, it is saved in the Optim Directory along with the Row List.
Point-and-Shoot specifications made when initiating an online Extract or Archive Process can be saved or recorded in a temporary Row List, available for the current process only. You can use a temporary Row List only when you invoke the Select Tables/Views for AD panel from the **EXTRACT Process** or **ARCHIVE Process** menu.

### POINT Command

Use the POINT command on the Select Tables/Views for AD panel to invoke the Point-and-Shoot facility. Optim prompts for the name of an input data set to allow you to edit an existing Row List.

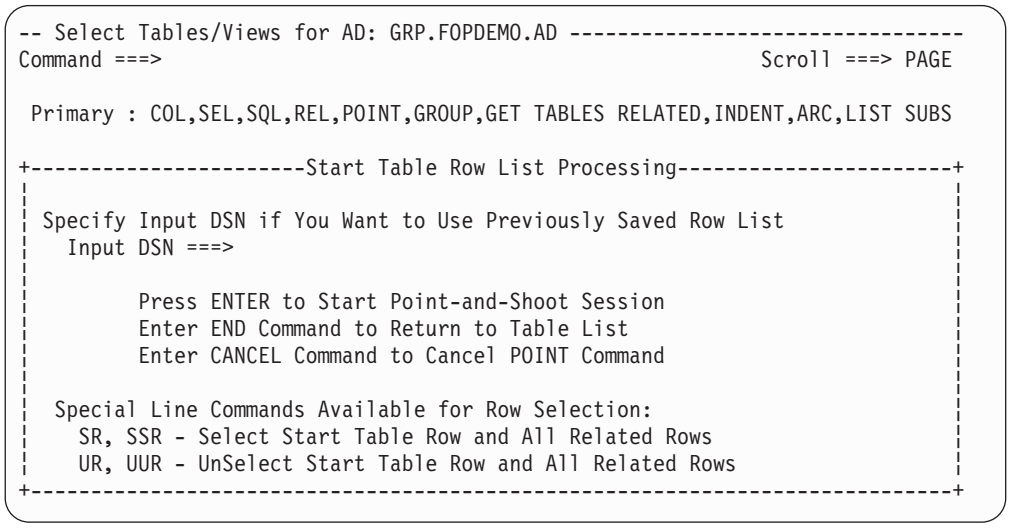



*Figure 33. Start Table Row List Selection Prompt*

**Note:** Prior to displaying this prompt, the Default Value panel may be displayed if you defined a substitution variable in the Access Definition, but did not specify a default value for the variable. You must specify a value to proceed. See "Substitution Variables" on page 70.

## Panel

The Start Table Row List Processing pop-up prompts for:

**Input DSN**
The name of a data set containing a Row List. Optim displays any previously specified **Input DSN** for the Access Definition. If none, the prompt is blank.

To edit an existing Row List, enter the data set name. You can type over any displayed name to change the entry or clear a name to drop a previously specified Row List.

After any changes are made, press ENTER, or use END or CANCEL as indicated on the panel and described as follows.

**ENTER**
Begin the Point-and-Shoot session.

Optim displays data that matches any selection or other criteria. If you are editing an existing Row List, the previously identified rows are selected on the display. If **Input DSN** is blank, the Point-and-Shoot session begins with no rows selected.

**END** Return to the Select Tables/Views for AD panel.

If an input data set name is supplied, the primary key values in the Row List remain in effect. Point-and-Shoot is not invoked. If the **Input DSN** is blank, no primary key values are used. (You can erase the **Input DSN** and use END to drop a previously specified Row List.)

**CANCEL**
Cancel the display and POINT command and return to the Table List. Any modifications are disregarded.

## Intervening Changes

If database rows are deleted or changed in the interval between the earlier selection of primary key values and the current Point-and-Shoot session, the rows that match primary key values in the Point-and-Shoot file may be missing. If so, a prompt is displayed to notify you, indicating the number of rows referenced in the Point-and-Shoot file.

- Press ENTER to proceed without modifying the file.
- Use END to remove unmatched primary key values from the Point-and-Shoot file.
- Use the CANCEL command to terminate the Point-and-Shoot session without modifying the file.

The following sections discuss the facilities that are available during a Point-and-Shoot session. For more information about the primary and line commands discussed in these sections, see the *Command Reference Manual*, Primary Commands and Line Commands.

## Initial Display for Point-and-Shoot

The initial Point-and-Shoot display is determined by the Start Table, any selection criteria (but not Archive Criteria) for the Start Table, and the **Begin Table Display With** setting.

For more details, see "Access Definition Parameters" on page 112.

### SELECTION CRITERIA Command

Use the SELECTION CRITERIA command to display the Specify Selection Criteria for Table panel at any time during a Point-and-Shoot session. You can use this panel to change the selection criteria for any table in the display. Any criteria are used for the Point-and-Shoot session only and are not saved in the Access Definition.

The Specify Selection Criteria for Table panel prompts for selection criteria. All facilities for the Specify Selection Criteria for AD panel are available. (For details on this panel, see "Selection Criteria" on page 49.) After specifying the desired selection criteria, press ENTER or use END to display the data.

If you change the selection criteria for the Start Table so that the set of rows selected in the Point-and-Shoot file are not appropriate, you are prompted to specify whether to continue with the session or to retain or discard the previously selected rows.

## Point-and-Shoot Basics

The Point-and-Shoot display is very similar to an ISPF/PDF display. However, to allow you to select specific rows (and, therefore, the primary key values) while browsing data, Point-and-Shoot provides screen elements and facilities not available in ISPF/PDF. This section discusses the Point-and-Shoot screen elements, commands, and display modes.

### Display Format

There are two basic display formats: columnar (default) and sidelabels. In columnar format, the headings are displayed across the beginning of the panel and the data is displayed in columns after the headings. More than one row from a table can be displayed.

```
ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
-------- ------- ---------- ---------- --------------- --------------
     7    00721  2005-04-26  14.22.31        9.22          NE005
     8    00021  2005-04-26  14.22.31        9.22          SC005
     9    02123  2005-04-26  14.22.31        9.22          SW005
```

In sidelabels format, the column headings are listed vertically on the panel with the data next to the headings. Though the display is limited to a single row of data, data from a greater number of columns is displayed for the row.

```
ORDER_ID          :      7
CUST_ID           : 00721
ORDER_DATE        : 2005-04-26
ORDER_TIME        : 14.22.31
FREIGHT_CHARGES   :   9.22
ORDER_SALESMAN    : NE005
ORDER_POSTED_DATE : 2005-04-27-16.59.00.000000
ORDER_SHIP_DATE   : 05/04/29
```

You can switch between these two display formats by using the SIDELABELS primary command or the SID line command. For more information about the sidelabels format, see "Sidelabels Format" on page 356.

**Note:** Most screen examples in this manual are in columnar format.

## Screen Elements

The screen elements in the columnar and sidelabels display formats are the same, only the positioning is different.

The following figure shows key screen elements in columnar format.

```
--------------------------- Optim: Point-and-Shoot ---------- 18 ROWS SELECTED
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================= 1 OF 1000 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___ S       7   00721  2005-04-26  14.22.31        9.22          NE005
___         8   00021  2005-04-26  14.22.31        9.22          SC005
___         9   02123  2005-04-26  14.22.31        9.22          SW005
___        10   00049  2005-04-26  14.22.31        9.22          WE005
___        11   00026  2005-04-26  14.22.31        9.22          SE005
___        12   00393  2005-04-26  14.22.31        9.22          NE005
___        13   03214  2005-04-26  14.22.31        9.22          NW005
___ S      14   00309  2005-04-26  14.22.31        9.22          SC005
___ S      15   00352  2005-04-26  14.22.31        9.22          SC005
___ S      16   00264  2005-04-26  14.22.31        9.22          NW005
___ S      17   00168  2005-04-26  14.22.31        9.22          SW005
___ S      18   01021  2005-04-26  14.22.31        9.22          SE005
___ S      19   00454  2005-04-26  14.22.31        9.22          NC005
___        20   00001  2005-04-26  08.16.09       14.80          NE005
___        21   05075  2005-04-26  08.16.09       14.80          SC005
___        22   00077  2005-04-26  08.16.09       14.80          SW005
___        23   00068  2005-04-26  08.16.09       14.80          WE005
```

*Figure 34. Screen Elements*

### The elements are:

**Command**

Prompt for primary commands. Refer to "Available Commands" on page 76 for a summary of available primary commands.

**Line Command**

Area for line commands. Refer to "Available Commands" on page 76 for a summary of available line commands.

In columnar format, the label is **Cmd**. In sidelabels format, the label is **LineCmd**.

**Status Flag**
Identifier for selected rows. The letter "S" is displayed for rows selected in the Point-and-Shoot session.

In columnar format, the label is **F**. In sidelabels format, the label is **Row Status**.

**Table Name**
The fully qualified name of the table or view. The name is truncated if the Creator ID and Table Name are more than 22 characters. This value cannot be edited; it is for display only.

**Short Name**
An identifier showing the level for each table (T*n*) or view (V*n*). In a multi-table display, T1 identifies the first level, T2, the second, etc. Several tables can be joined at any level except T1. Tables joined at one level (i.e., stacked tables) have the same level identifier. (See "Multiple Table Display" on page 341 for additional information about multi-way joins and stacked tables.) You can use the level identifier in place of the table name as a command operand.

**x OF y**
If all rows cannot be displayed, you can scroll the display. Notation next to **Table Name** indicates the relative position of the first visible row. In this notation, *x* is the relative position of the first visible row and *y* the total number of rows.

**Horizontal Scroll Indicator**
Indicator (in the form **MORE** or **<<MORE**) that additional data can be displayed by scrolling to the LEFT or RIGHT (refer to "Horizontal Scroll" on page 79).

**Column Headings**
Column names are used as headings for the rows from each table.

**Column x of y**
If the complete list of columns cannot be displayed, you can scroll it. This notation is in the form, Column *x* of *y*, where *x* is the relative position of the first visible column and *y* the total number of columns in the table. (Displayed in sidelabels format only.)

The number of Start Table rows you select is always displayed on the screen. In Figure 34 on page 75, 18 rows are selected.

## Available Commands

Both primary and line commands are available. Many of these commands are introduced in the following sections. Detailed information about each command is provided in the *Command Reference Manual*, Primary Commands and Line Commands.

### Primary Commands

The following is a list of primary commands available during a Point-and-Shoot session.

- ANCHOR
- ATTRIBUTES
- BOTTOM
- CANCEL
- CAPS
- COLUMNS
- COUNT
- DOWN
- END
- EXCLUDE
- EXPAND

- FIND
- HEX
- INDENT
- JOIN
- LEFT
- LIST
- LOCK
- MAX FETCH ROWS
- ONLY
- OPTIONS
- REPORT
- RESET
- RESORT
- RFIND
- RIGHT
- SEL CRITERIA
- SELECT RELATED
- SHOW
- SHOW SQL
- SIDELABELS
- SORT
- SQL
- START
- SWITCH
- TOP
- UNJOIN
- UNLOCK
- UNSELECT RELATED
- UP
- ZOOM

**Note:** The INDENT, JOIN, UNJOIN, START, and SWITCH commands are not available in sidelabels format.

### Line Commands

The following line commands are available during a Point-and-Shoot session.

**X, X***n*, **XX**
: Exclude
Temporarily remove a row, a number of rows, or a block of rows from the display.

**F, F***n* First
: Redisplay the first row or the first *n* rows in a block of excluded rows.

**J** Join
: Join to related tables.

**L, L***n* Last
: Redisplay the last row or the last *n* rows in a block of excluded rows.

**SR, SSR**
Select Related
Specify a row or block of rows to be selected in a Point-and-Shoot session.

**S, S*n*, SS**
Show
Redisplay one row, *n* rows, or all rows in a block of excluded rows.

**SID** Sidelabels
Display the row in sidelabels format, or return the display to columnar format.

**UNJ** Unjoin
Remove joined tables from the display.

**UR, UUR**
Unselect Related
Specify a row or block of rows to be unselected in a Point-and-Shoot display.

**Z** Zoom
Display all rows in the table or return to the multi-table display.

**Note:**

- The SR, SSR, UR, and UUR line commands are available only for the Start Table.
- The J and UNJ line commands are not available in sidelabels format.

## Select Rows

Select rows from the Start Table using the SR line command or the SELECT RELATED primary command. For example, in the following figure, a single row is selected by specifying the SR line command, and a block of rows is selected by specifying two SSR line commands; the first SSR line command identifies the first row in the block of rows, and the second SSR identifies the last row in the block.

```
--------------------------- Optim: Point-and-Shoot ---------- 18 ROWS SELECTED
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 1000 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
   _ S       7   00721  2005-04-26  14.22.31        9.22          NE005
SR_          8   00021  2005-04-26  14.22.31        9.22          SC005
             9   02123  2005-04-26  14.22.31        9.22          SW005
SSR         10   00049  2005-04-26  14.22.31        9.22          WE005
___         11   00026  2005-04-26  14.22.31        9.22          SE005
___         12   00393  2005-04-26  14.22.31        9.22          NE005
___         13   03214  2005-04-26  14.22.31        9.22          NW005
___         14   00309  2005-04-26  14.22.31        9.22          SC005
___         15   00352  2005-04-26  14.22.31        9.22          SC005
___         16   00264  2005-04-26  14.22.31        9.22          NW005
___         17   00168  2005-04-26  14.22.31        9.22          SW005
___         18   01021  2005-04-26  14.22.31        9.22          SE005
___         19   00454  2005-04-26  14.22.31        9.22          NC005
            20   00001  2005-04-26  08.16.09       14.80          NE005
SSR         21   05075  2005-04-26  08.16.09       14.80          SC005
___         22   00077  2005-04-26  08.16.09       14.80          SW005
___         23   00068  2005-04-26  08.16.09       14.80          WE005
```

*Figure 35. Select Rows Using Line Commands*

As an example of the SELECT RELATED primary command, enter the following to select all orders for salesman SC005:

```
SELECT RELATED IN ORDER_SALESMAN ALL SC005
```

Reverse a selection by using the UR line command or the UNSELECT RELATED primary command. In the following figure, the last selected row, Order 21, is unselected.

```
--------------------------- Optim: Point-and-Shoot ---------- 25 ROWS SELECTED
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 1000 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___ S       7   00721  2005-04-26  14.22.31        9.22          NE005
___ S       8   00021  2005-04-26  14.22.31        9.22          SC005
___         9   02123  2005-04-26  14.22.31        9.22          SW005
___ S      10   00049  2005-04-26  14.22.31        9.22          WE005
___ S      11   00026  2005-04-26  14.22.31        9.22          SE005
___ S      12   00393  2005-04-26  14.22.31        9.22          NE005
___ S      13   03214  2005-04-26  14.22.31        9.22          NW005
___ S      14   00309  2005-04-26  14.22.31        9.22          SC005
___ S      15   00352  2005-04-26  14.22.31        9.22          SC005
___ S      16   00264  2005-04-26  14.22.31        9.22          NW005
___ S      17   00168  2005-04-26  14.22.31        9.22          SW005
___ S      18   01021  2005-04-26  14.22.31        9.22          SE005
___ S      19   00454  2005-04-26  14.22.31        9.22          NC005
    S      20   00001  2005-04-26  08.16.09       14.80          NE005
UR_ S      21   05075  2005-04-26  08.16.09       14.80          SC005
___        22   00077  2005-04-26  08.16.09       14.80          SW005
___        23   00068  2005-04-26  08.16.09       14.80          WE005
```

*Figure 36. Unselect Rows Using Line Commands*

Only Start Table rows are selected or unselected directly. However, rows from other tables in the Access Definition are also selected if related to a selected Start Table row and, in a joined display, show the "S" status flag.

## Scroll Data

Several commands are available to scroll the data vertically and horizontally. Scroll vertically to view additional rows when the number of rows retrieved exceeds the number of lines available on the screen. Scroll horizontally to view additional columns when the width of the data display exceeds the width of the screen.

### Vertical Scroll

You can enter the scrolling commands UP, DOWN, TOP, and BOTTOM on the command line or assign them to program function keys. The ISPF SCROLL values, CSR, PAGE, DATA, HALF, MAX, and *n* (where *n* is the number of lines to scroll) are supported.

When a single table is displayed, vertical scrolling is identical to vertical scrolling in ISPF. Scrolling a multi-table display is discussed in "Coordinated Scroll" on page 348.

### Horizontal Scroll

During a Point-and-Shoot session, all columns may not be displayed on the screen. When the combined width of the columns exceeds the width of the screen, the scroll indicator, MORE with an appropriate direction arrow, is displayed on the information line for the table. For example, if the display width of a column is 20 characters but only 10 spaces remain on the screen, the column is omitted from the display until you scroll horizontally.

You can scroll horizontally using LEFT and RIGHT (usually assigned to PF10 and PF11) to display the remaining columns. Operands for these commands allow you to scroll by column name, numeric value, or cursor position. These commands perform the same functions in columnar or sidelabels format. In sidelabels format, if insufficient space is available, partial data from columns may be displayed.

When multiple tables are displayed, indicate the table to scroll by specifying a table name or identifier with the column name (for example, T2.CUST_ID) or by cursor position. If a table is not indicated, the lowest-level table is scrolled.

### Lock Columns for Horizontal Scroll

You can use the LOCK and UNLOCK commands to retain specified columns on the screen when scrolling horizontally. The LOCK command repositions the named column first on the screen and retains the column in that position when a horizontal scroll is performed. A locked column is identified by a series of plus signs, +, after the column heading. Multiple columns may be locked. The UNLOCK command unlocks the named locked column and returns it to the original position.

For more information about the LOCK and UNLOCK commands, see "Lock Columns" on page 363.

## Join Tables

A Point-and-Shoot session begins with a display of rows from the Start Table. You can add related rows from other tables listed in the Access Definition to the display using a Join command. The table for which the join is executed, or the *anchor table*, is joined to a related table from the Access Definition, and the related rows from that table are displayed.

For more information about joining tables, refer to "Join Command" on page 336.

## Manage the Display

Data for a Point-and-Shoot session may involve extensive data from hundreds of tables, each with dozens of columns and thousands of rows. Navigating this data to identify a specific subset of related rows can be daunting.

Optim provides many techniques and features to help you:

- Navigate large lists of data easily and efficiently.
- Find specific data and manage the display.
- Review information about columns or data.
- Arrange and format displays of wide data to accommodate browsing.

For more information about these features, see "Manage Data Displays" on page 55.

## Display SQL

During a Point-and-Shoot session, use the SHOW SQL command to display a generalized form of the SQL used to fetch the rows from any displayed table. By default, the SQL for the lowest-level table is displayed. To display the SQL for another table, supply either the table name or identifier with the command or position the cursor to the desired table. For example, to display the SQL for the highest-level table, enter:

```
SHOW SQL T1
```

The SQL may be in three parts.

- A declaration of host variables, if the corresponding columns have different data types or dimensions.
- Pseudo code to populate any host variables if declared.
- The "generalized" SELECT statement, which incorporates the host variables.

For example, if you enter the SHOW SQL command for the ORDERS table, a generalized SQL statement is displayed as:

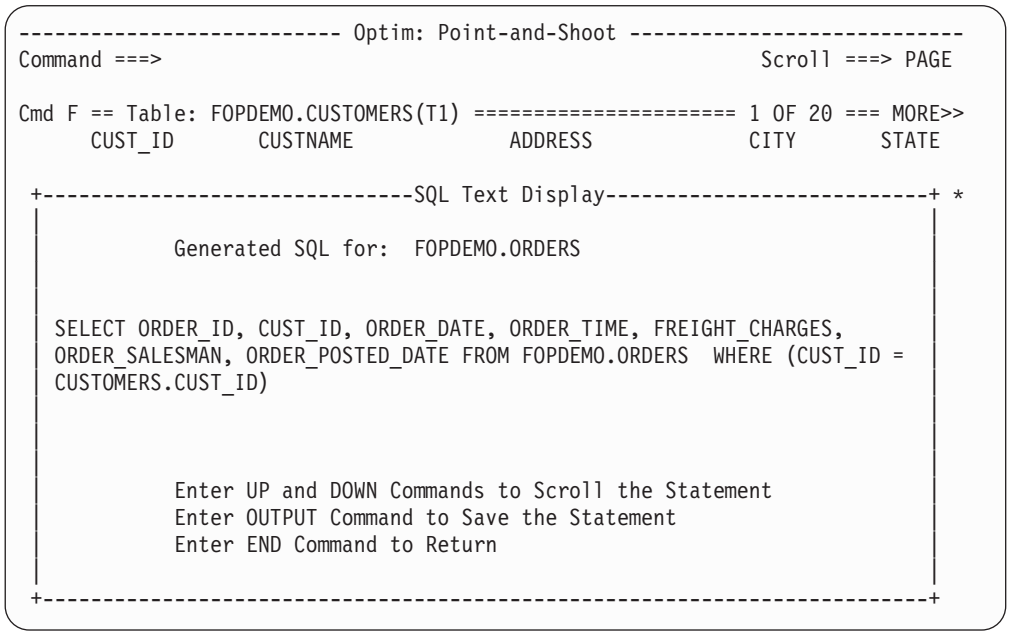



*Figure 37. Text Display of Generated SQL*

## Generalized WHERE Clause

This generalized form is probably more useful than the actual SQL generated by Optim, which includes the specific data values for columns defined in the relationship. For example, the generated SQL represented by the generalized SQL in the previous figure includes a value for CUST_ID, as in:

```
SELECT ...
FROM FOPDEMO.ORDERS WHERE CUST_ID = '17053'
```

In the generalized form, the second part of the predicate in the SQL statement is a meaningful name that represents the processing.

## Save the SHOW SQL Output

You can save or print the SQL Text Display using the OUTPUT command. When you enter the OUTPUT command, the Output Data Options panel is displayed.

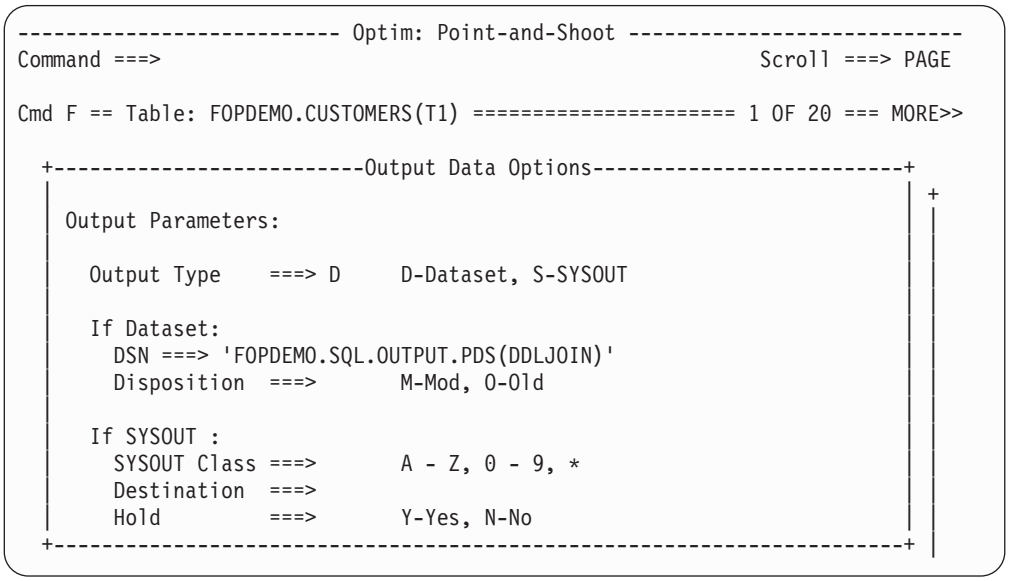



*Figure 38. Output Data Options*

## Panel

The Output Data Options panel includes:

**Output Type**
Option to save the SQL in a dataset or as a SYSOUT class for printing.

**If Dataset:**
DSN

The name of the dataset. Specify the name of a new or existing sequential file, with a record format of fixed or fixed block, or Partitioned Data Set (PDS) with member name enclosed in parentheses.

The dataset name is automatically prefixed with the default prefix if you do not enclose it in single quotes. (See "Editor and Display Options" on page 382 for information about the default prefix.)

If the specified data set does not exist, the Allocate Dataset panel is displayed (see Appendix B, "Allocating External Files," on page 417 for details).

To generate a selection list of datasets, specify an asterisk (*) or the DB2 LIKE character % at the end of the DSN specification. For example: 'FOPDEMO.DDL*' or 'FOPDEMO.DDL%'

Each generates a selection list of all data sets with a valid format and a DSN beginning with FOPDEMO.DDL.

**Disposition**
For an existing, sequential data set only, specify:

**M** MOD. The SQL data is appended to the data set.

**O** OLD. The contents of the file are replaced with the SQL data.

**If SYSOUT:**
You can direct the output to a SYSOUT class and use an output processor, such as SDSF, to print it.

**SYSOUT Class**
The output class for the printed output. Specify **SYSOUT Class** as an alphabetic or numeric character or an asterisk (*).

**Destination**
The SYSOUT destination. Specify a valid local or remote terminal, a node in the JES network, a local or remote printer or workstation, or a TSO User ID.

**Hold** Disposition of the output. Specify:

**Y** Output is held until released or deleted by an operator.

**N** Output is not held.

Press ENTER to continue processing. To return to the previous panel without specifying output data options, use END or CANCEL.

## Reports

At any time during a Point-and-Shoot session, you can produce a report about the contents of the session. Use the reporting facility to generate a report that includes some or all rows in a variety of formats. To invoke the report facility, use the REPORT command.

For more information about generating reports, refer to "Reports" on page 364.

## Terminate a Point-and-Shoot Session

Use END to terminate a Point-and-Shoot session. Before returning to the Select Tables/Views for AD panel, the Confirm Row List Processing prompt is displayed.

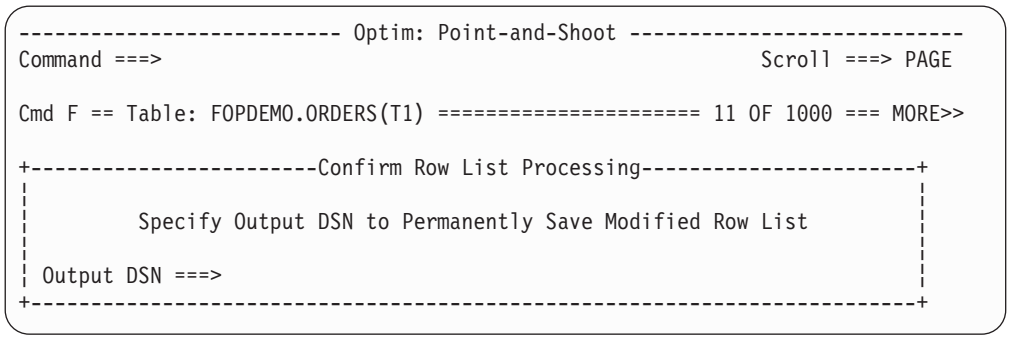



*Figure 39. Confirm Row List Processing*

### Output DSN Name

You can save the Row List for later use. The name of any Input DSN you supplied to begin the Point-and-Shoot session is shown in **Output DSN**. You can simply press ENTER to update the named data set. You can also overtype the name to create a new data set or to update a different data set. If you did not begin the session with an Input DSN, the area is blank and you must enter a name to save the Row List. The dataset name is saved with the Access Definition and replaces any previously specified data set name.

### Extract/Archive Process Note

If you began the Point-and-Shoot session from within an Extract or Archive Process request, you can leave **Output DSN** blank to use the Point-and-Shoot Row List for only the current Extract or Archive Process.

### Subsequent Requests

You can re-enter a Point-and-Shoot session to modify a saved Row List. If the changes are temporary and the POINT command is entered for a subsequent process during the same session, the following prompt

confirms that temporary selection is still in effect.

```
+--------------------- Confirm Row List Processing --------------------+
|                                                                       |
|           Currently Using a Temporary Start Table Row List            |
|                                                                       |
| Press ENTER to Continue Using List and Start Point-and-Shoot Session  |
| Enter END Command to Drop Temporary List and Return to Table List     |
| Enter CANCEL Command to Cancel POINT Command                          |
+-----------------------------------------------------------------------+
```

*Figure 40. Reconfirm Temporary Point-and-Shoot Values*

### Change the Start Table

If you change the Start Table and rows had been selected, you are notified that the Start Table specifications, including Point-and-Shoot selections, may be dropped since they may no longer be appropriate for the new Start Table.

```
+-----------------Drop Start Table Options Confirmation-----------------+
|                                                                       |
| Start Table Options will be Dropped due to Change of Start Table Name |
|                                                                       |
| Press ENTER Key to Confirm Drop of Start Table Options                |
| Enter END Command to Keep Start Table Options                         |
+-----------------------------------------------------------------------+
```

*Figure 41. Confirm Drop Start Table Options*

## Row List Status Indicated

After you terminate the Point-and-Shoot session, the Select Tables/Views for AD panel is redisplayed. **Start Table Options** indicates whether a Row List has been defined for the current Access Definition. For example, if a permanent Row List has been defined for the Access Definition, the Select Tables/Views for AD panel is displayed as:

```
-- Select Tables/Views for AD: GRP.FOPDEMO.AD ---------------------------------
Command ===>                                                Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                  Table 1 of 4  <<MORE
Default Creator ID ===> FOPDEMO                                       >>
Start Table        ===> ORDERS                                        >>
Start Table Options   : Row List (POINT)
                                             R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ------------------------------>> - - -------- -------- -------
*** ********************************* TOP ************************************
___             ORDERS                         N   ____   ________ TABLE
___             CUSTOMERS                    N N   ____   ________ TABLE
___             SALES                        N N   ____   ________ TABLE
___             SHIP_TO                      N N   ____   ________ TABLE
*** ******************************** BOTTOM ***********************************
```

*Figure 42. Start Table Options – Row List Indicated*

## Group Selection Processing

Group selection processing allows you to extract a sampling of Start Table rows and is especially useful when you want to sample a representative set of related data for specific data groups.

Although you can use other options on the Select Tables/Views for AD panel to define the data to be extracted (i.e., Point-and-Shoot, selection criteria, Every Nth row, etc.), group selection processing allows you to limit the selected rows to a number of different values in a specific column with a maximum number of rows to be extracted for each value. For example, you can specify that five rows from each of ten states are to be extracted from the Start Table.

**Note:** Group selection processing is available for Extract and Compare Processes, but not Archive Processes.

Group selection can be used in combination with other selection criteria and parameters, except the **EveryNth** specification. In general, the criteria are applied first and group selection processing is applied to the result.

To initiate group selection processing, enter the GROUP command on the Select Tables/Views for AD panel to display the **Start Table Group Selection Processing** pop-up window.

```
-- Select Tables/Views for AD: GRP.USER.ADSAMPLE ------------------------------
Command ===>                                                  Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS

+--------------------Start Table Group Selection Processing ------------------+
|                                                                              |
| To select a set of up to 'N' rows from each of up to 'M' distinct values     |
| for a specific column, provide the following information:                    |
|                                                                              |
| Column Name to be used for Group     ===> STATE                              |
| Number of distinct values to use (M) ===>          (1-4294967295,*=NO LIM)   |
| Number of rows with each value   (N) ===>          (1-4294967295,*=NO LIM)   |
|                                                                              |
| Note: To eliminate Group processing, clear the column name and press ENTER   |
|       Use the LIST command to get a selection list of column names           |
|       A limit must be specified for 'M', 'N', or both                        |
+------------------------------------------------------------------------------+
```

*Figure 43. Start Table Group Selection Processing*

### Column Name

To use group selection processing, you must provide the name of a column in the Start Table to be used for forming groups. You can use the LIST command to display a selection list of column names for **Column Name to be used for Group**. On the selection list, use the S line command to select a name. To eliminate group selection processing, clear **Column Name to be used for Group**.

As an example of the use of prompts for **Number of distinct values to use** and **Number of rows with each value**, assume the CUSTOMERS table is the Start Table and the STATE column is the **Column Name to be used for Group**.

### Distinct Values Limit and Number of Rows for Each Value

The criteria for selecting the sampling of rows can be based on distinct values in the STATE column. That is, you can limit the selection to a group of rows for some number of different states in the STATE column. The states are selected randomly. However, if you want to explicitly define the states to be used,

do this on the Specify Selection Criteria panel or on the Specify SQL WHERE Clause panel. Since the selection criteria is processed first, you can be sure to select rows from only the desired states. For example:

- To extract rows for customers in ten different states, specify "10" for the **Number of distinct values to use**.

  On the Specify Selection Criteria or Specify SQL WHERE Clause panel, you can provide criteria for individual state names (IN), a range of names (BETWEEN), and so forth. If the selection criteria apply to more than ten states, only ten are used, but the specific ten states to be selected is unpredictable.
- To extract five rows for each of the ten states, specify "5" as the **Number of rows for each value**.

You can use an asterisk (*) to select all rows found. For example, enter an asterisk for **Number of rows for each value** to obtain all rows for each included state. Alternatively, enter an asterisk for **Number of distinct values** to obtain a specific number of rows from all included states. You cannot enter an asterisk for both limits.

Use ENTER or END to return to the Select Tables/Views for AD panel, where group selection processing is indicated in **Start Table Options**. In the following figure, both group selection processing and a Row List apply.

```
-- Select Tables/Views for AD: GRP.USER.ADSAMPLE ------------------------------
Command ===>                                                 Scroll ===> PAGE

 Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,ARC,LIST SUBS
    Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,ARC,ACT,STA
                                                      Table 1 of 4 <<MORE
Default Creator ID ===> FOPDEMO                                        >>
Start Table        ===> CUSTOMERS                                      >>
Start Table Options   : Row List (POINT), Group Selection (GROUP)
                                                R D --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name    F A EveryNth RowLimit  Type
--- ----------- ----------------------------->> - - -------- -------- -------
*** ********************************* TOP ************************************
___ COL/SEL     CUSTOMERS                        N   ____   ________ TABLE
___ SQL         ORDERS                         N N   ____   ________ TABLE
___             ITEMS                          N N   ____   ________ TABLE
___             DETAILS                        N N   ____   ________ TABLE
*** ******************************** BOTTOM ***********************************
```

*Figure 44. Group Selection Processing Indicated*

## Select Relationships

Optim uses relationships between pairs of tables on the Table List to determine the paths used to process data. By default, all relationships are used. You can select the relationships used to process the data on the Specify Relationship Usage panel. You also can use that panel to indicate how RI rules determine the rows that are processed according to the parent-to-child and child-to-parent processing sequence.

### Data only

These specifications only affect the data that is processed. All relationships, including unselected relationships and relationships for reference tables, are copied to the file and can be used to recreate the original data model.

### Specify Relationship Usage Panel

The Specify Relationship Usage panel lists all relationships between the pairs of tables on the Table List, except tables marked as reference tables. Display this panel by entering the REL primary command on

the Select Tables/Views for AD panel or selecting the PATHS option from a Process menu (e.g., the **EXTRACT Process** menu). Relationships defined to the Optim Directory are displayed, as well as DB2, IMS, and IML relationships.

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 5

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************** TOP ***********************************
___ NEW    Y N       ORDERS                 DETAILS                ROD      DB2
___ NEW    Y N       CUSTOMERS              ORDERS                 RCO      DB2
___ NEW    Y N       *CUSTOMERS             ORDERS                 RCO1     OPT
___ NEW    Y N       ITEMS                  DETAILS                RID      DB2
*** ******************************** BOTTOM **********************************
```

*Figure 45. Specify Relationship Usage*

## Panel

The Specify Relationship Usage panel includes:

**Cmd** The line command entry area. Valid commands are:

- **S** Select a relationship.
- **U** Unselect a relationship.
- **D** Delete an unknown relationship.
- **I** Display information about a relationship.

**Status** The status of the relationship. The relationships are listed by status, in the order NEW, SELECT, UNSEL, and UNKNWN. Within each status, relationships are listed in the order in which the parent table is listed on the Select Tables/Views for AD panel. Possible values are:

- **NEW** The relationship is new to the list. The Specify Relationship Usage panel for the Access Definition is displayed for the first time, a new table has been added to the Table List, or a new relationship has been defined since the Access Definition was last edited.
- **SELECT** The relationship is used to process data. You can select any number of relationships.
- **UNSEL** The relationship is not used to process data.
- **UNKNWN** The relationship does not apply to the listed tables. This status occurs when a relationship is deleted from the Directory or Catalog or the default Creator ID for the Access Definition is changed and, as it affects one or more names on the Table List, a relationship no longer applies.

**Q1** Option for processing parent rows of a selected child row. In a Restore Process, settings used to create the Archive File are the default; otherwise, the default setting is Y. Specify:

- **Y** Satisfy RI rules by selecting all parent rows of a selected child row. Selection criteria are not applied when a relationship is traversed from child to parent.
- **N** Do not select parent rows.

**Q2** Option for processing child rows of a parent row that was selected to satisfy RI rules. This option is relevant when the **Q1** setting is Y for any relationship in which the parent table in this relationship is the parent. In a Restore Process, settings used to create the Archive File are the default; otherwise, the default setting is N. Specify:

**Y** Process all children.

**N** Do not process all children. Default.

Examples to clarify **Q1** and **Q2** specifications are provided in "Sample Scenarios" on page 97.

**Child Limit**
The maximum number of child rows to process for each parent using a specific relationship (e.g., process no more than five ORDERS for any CUSTOMERS row, using the relationship RCO).

Specify 0 to prevent use of the relationship to process any child rows for a specific parent. A **Child Limit** of 0 is useful when you include a relationship to prevent orphans (process parents for children) but do not want to process additional children.

**Note:** Do not confuse this setting with the **Row Limit** on the Select Tables/Views for AD panel. The **Row Limit** parameter limits the total number of rows processed from the table regardless of relationship traversal. Neither **Row Limit** nor **Child Limit** is typically used in an Archive or Restore Process.

**Parent** Name of the parent table in the relationship, listed as entered on the Select Tables/Views for AD panel (i.e., with or without Creator ID prefix).

If more than one relationship is defined for a pair of tables with the same parent and child designations, an asterisk is displayed before the name of the parent table for each occurrence.

**Child** The name of the child table in the relationship, listed as entered on the Select Tables/Views for AD panel, i.e., with or without Creator ID prefix.

**Relation Name**
The name of the relationship.

**Relation Type**
The source of the relationship.

**DB2** Relationship between DB2 tables, stored in the DB2 catalog.

**IMS** Logical or referential relationship between IMS Legacy tables in which the IMS data exists in the same database. (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**IML** Logical relationship, based on columns, between IMS Legacy tables where the IMS data is stored in separate databases. (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**OPT** Relationship between both DB2 and Legacy Tables stored in the Optim Directory.

**Note:** The relationship type may be shown as 'OPT', 'FOP', or 'PST' in a batch process report or ISPF panel.

## QUES1/QUES2

You can use the QUES1 and QUES2 commands to change the **Q1** and **Q2** settings to YES or NO for all listed relationships. These commands are useful, for example, when many relationships are listed and you want to override the default setting for most of them or when, after making several changes, you want to reset the settings to the default.

## CREATE RELATIONSHIP

If necessary, you can create Optim relationships using the CREATE RELATIONSHIP command on the **Specify Relationship Usage** panel. This command invokes the panels used to define relationships stored

in the Optim Directory. (Option 6 DEFINITIONS on the **Main Menu** also displays these panels. Information about defining relationships is provided in Chapter 4, "Relationships," on page 125.)

**Note:** A newly created relationship can be used in any process, but will not affect a Restore Process. Only relationships in the Archive File are used to restore data.

Use END or CANCEL to return to the Specify Relationship Usage panel. Newly defined relationships for tables on the Table List are included in the list of relationships and assigned a NEW Status.

## ACM

If necessary, you can use the ACM command to open the Choose Access Method / Key Lookup Limit pop-up dialog.

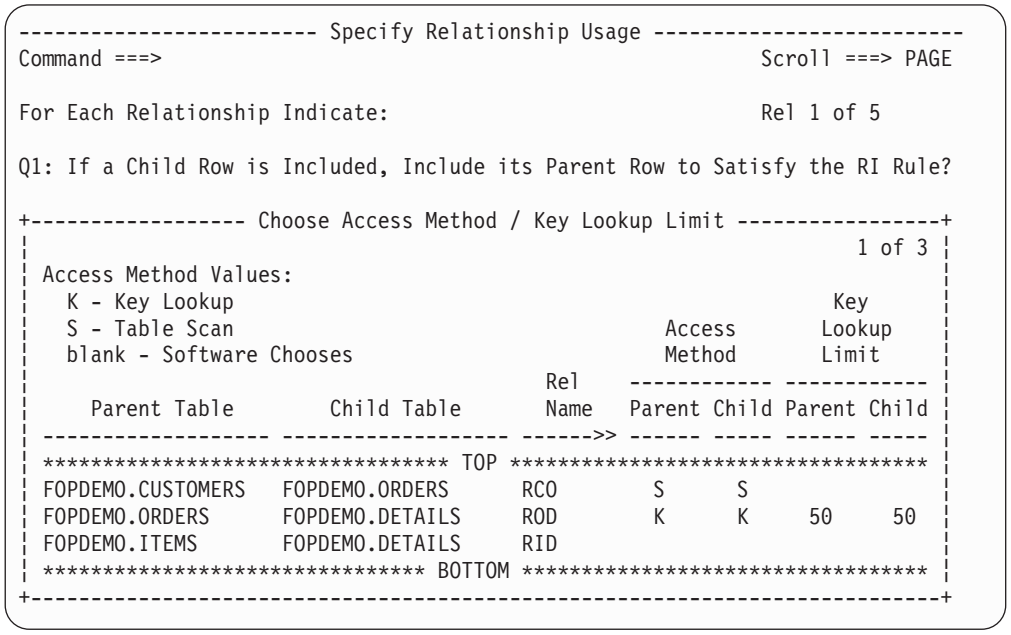



*Figure 46. Choose Access Method / Key Lookup Limit*

This panel allows you to override the default method (scan or key lookup) for accessing the parent or child table for each relationship. A scan reads every row in a table at one time, whereas a key lookup locates rows using a WHERE clause to search for primary or foreign key values. Additionally, you can change the maximum number of key lookups performed at one time for a table. Valid values are 1 through 1000. If a value is not specified, the default value specified via a user option is used. See "User Options" on page 372 for more information.

For more information, refer to "Table Access Strategy" on page 394.

## Available Commands

The following commands are available when the Specify Relationship Usage panel is displayed.

- ACM
- BOTTOM
- BROWSE
- CANCEL
- CREATE PRIMARY KEY
- CREATE RELATIONSHIP

- DELETE
- DOWN
- END
- EXPAND
- INDENT
- OPTIONS
- OUTPUT
- QUES1
- QUES2
- RESET
- SAVE
- SHOW INDEXES
- SHOW STEPS
- TOP
- UNSELECT
- UP
- USE

## Display Information

You can display information about a relationship to help decide whether to select it. Use the Information line command, I, to browse a relationship, as shown in the following figure in which information about the relationship RCO is displayed in the pop-up window.

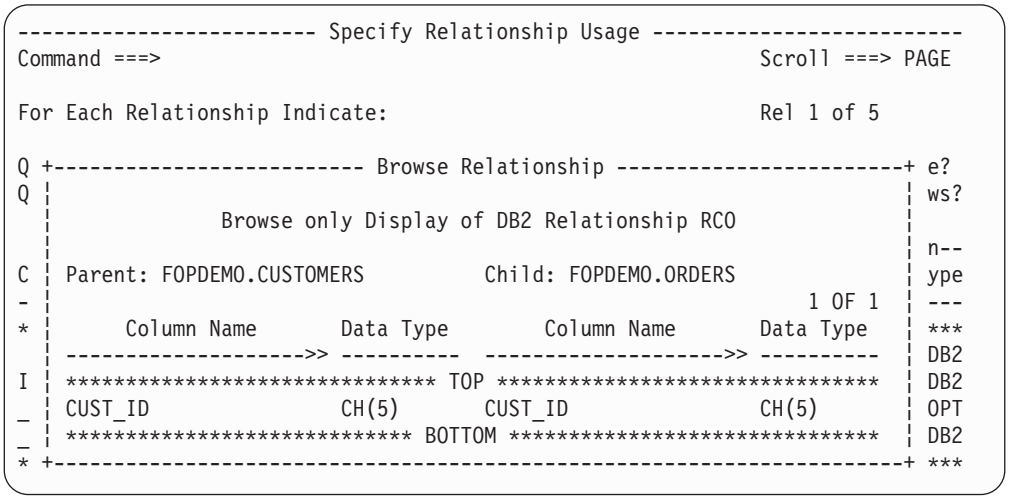



*Figure 47. Browse Relationship - Access Definition*

### Panel

The Browse Relationship panel includes:

**Parent** The fully qualified name of the parent table.

**Child** The fully qualified name of the child table.

**Column Name**
The expressions and names of columns for each table used in the relationship.

**Data Type**
The data type of the matched columns. In addition to the standard DB2 data types, **Data Type** may be EXPR to indicate that the relationship is based on an expression for which the data type is determined when the relationship is used.

Use END or press ENTER to redisplay the Specify Relationship Usage panel.

## Relationship Status

When you first display the Specify Relationship Usage panel, the order of the relationships matches the order of parent tables on the Table List and the names of the tables are shown as entered on the Select Tables/Views for AD panel.

Also, the first time a relationship is listed, it is assigned a NEW Status, as shown in Figure 48.

### Select and Unselect

On the relationship list, you can select and unselect relationships. Use the S line command to select a relationship and the U line command to unselect a relationship. In Figure 48, the first and second relationships are selected, the third is unselected, and the last is not explicitly selected or unselected.

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                 Scroll ===> PAGE

For Each Relationship Indicate:                              Rel 1 of 5

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************* TOP ************************************
S _ NEW    Y N       ORDERS                 DETAILS                ROD      DB2
S _ NEW    Y N       *CUSTOMERS             ORDERS                 RCO      DB2
U _ NEW    Y N       *CUSTOMERS             ORDERS                 RCO1     OPT
___ NEW    Y N       ITEMS                  DETAILS                RID      DB2
*** ******************************** BOTTOM ***********************************
```

*Figure 48. Select Relationships for Functions*

### Revised Status

The second and succeeding times that you use the REL command from the Select Tables/Views for AD panel, the status on the Specify Relationship Usage panel is revised according to the **Use NEW Relationships** setting on the Access Definition Parameters panel. If that setting is Yes, NEW relationships are selected by default and participate in the process; if that setting is No, NEW relationships are not selected. (See "Access Definition Parameters" on page 112 for more information about the Access Definition Parameters panel.) If the response to the prompt **Use NEW Relationships** is Yes, the Status is displayed as shown in the following figure.

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 4

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************* TOP ************************************
___ SELECT Y N       ORDERS                DETAILS               ROD      DB2
___ SELECT Y N       *CUSTOMERS            ORDERS                RCO      DB2
___ SELECT Y N       ITEMS                 DETAILS               RID      DB2
___ UNSEL  Y N       *CUSTOMERS            ORDERS                RCO1     OPT
*** ******************************** BOTTOM ***********************************
```

*Figure 49. Status Revised*

The revised status is listed under **Status** and in the position of the relationships on the list. New relationships are listed first, then selected relationships, unselected relationships, and unknown relationships.

If tables are added, new relationships are created, or if the Creator ID is modified, the changes can result in NEW and UNKNWN designations for relationships. If they are not handled explicitly on the Specify Relationship Usage panel, warning messages are issued when the process is performed. (For more information about performing a given process, see the appropriate *User Manual*.)

Since the default action for new relationships may not be desired, you should display the Specify Relationship Usage panel to handle changes to the relationship list. In fact, to ensure that the desired relationships and traversal paths are used, you should review the Specify Relationship Usage panel whenever you modify the names or Creator ID for one or more tables on the Select Tables/Views for AD panel.

## Add a Table: An Example

Assume the SALES table is added to the Table List. When you enter the REL command, the revised relationship list includes any additional relationships for the additional table. As in the following figure, any relationships with a status of NEW are listed first, those with SELECT status next, followed by those with an UNSEL status.

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 5

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************* TOP ************************************
___ NEW    Y N       SALES                  CUSTOMERS              RSC      DB2
___ SELECT Y N       ORDERS                 DETAILS                ROD      DB2
___ SELECT Y N       *CUSTOMERS             ORDERS                 RCO      DB2
___ SELECT Y N       ITEMS                  DETAILS                RID      DB2
___ UNSEL  Y N       *CUSTOMERS             ORDERS                 RCO1     OPT
*** ******************************** BOTTOM ***********************************
```

*Figure 50. Specify Relationship Usage Panel Redisplayed*

If you do not explicitly select or unselect the NEW relationship, it is automatically assigned a status based on the response to the Access Definition Parameters prompt for **Use NEW Relationships**.

## USE Command

You can enter the USE NEW command to change the status of all new relationships to the SELECT status. You can enter USE ALL to change the status of all new or unselected relationships to the SELECT status.

## UNKNWN Status

In a test environment, duplicate sets of tables that differ only by Creator ID may use comparable relationships with the same names. In those instances, Optim maintains the assigned status. However, when a same-name relationship is not defined for both sets, any unused, previously listed relationships are assigned an UNKNWN status, and any new relationships are assigned a NEW status.

**Note:** Optim treats a change in the Creator ID for individual tables as if the original name was deleted and a new table name was supplied.

## Change Default Creator ID: An Example

As an example, assume the default Creator ID has been changed, resulting in two new relationships being listed and the change in **Status** for an existing relationship to UNKNWN. In the following figure, the names of the two new relationships are added at the beginning of the list, and the name of the unknown relationship is placed at the end of the list. The other relationships retain their previous status.

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 8

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************** TOP ************************************
___ NEW    Y N       *ORDERS               DETAILS               RTOD     OPT
___ NEW    Y N       *ORDERS               SALES                 RTOSI    OPT
___ SELECT Y N       *ORDERS               DETAILS               ROD      DB2
___ SELECT Y N       *CUSTOMERS            ORDERS                RCO      DB2
___ SELECT Y N       SALES                 CUSTOMERS             RSC      DB2
___ SELECT Y N       ITEMS                 DETAILS               RID      DB2
___ UNSEL  Y N       *CUSTOMERS            ORDERS                RCO1     OPT
___ UNKNWN Y N       *ORDERS               SALES                 RTOS     OPT
*** ******************************** BOTTOM ***********************************
```

*Figure 51. Unknown Relationship*

### Delete UNKNWN

You can use the Delete line command, D, to remove any relationship from the list that is UNKNWN. However, since you can use the same Access Definition for similar sets of tables with different Creator IDs, you may retain the unknown relationships on the list to minimize your efforts if you use the same Access Definition multiple times, changing only the Creator ID. Optim retains the original status of all unknown relationships and reapplies that status when the Creator ID is modified so that the relationship is applicable.

**Note:** Unknown relationships do not affect a process; they are ignored.

### Important Note

The Start Table is always included in a process. To include other tables, a relationship is required. A table referenced in the Access Definition is not processed if it is not part of a selected relationship. In fact, if you unselect all relationships, only rows from the Start Table are processed. Any number of relationships can be selected or unselected.

### OUTPUT command

You can use the OUTPUT command to print the relationship list to an output file or SYSOUT.

## Display Traversal Information

After you have completed the Specify Relationship Usage panel, you can use the SHOW STEPS command on that panel to ensure that the data processed and the traversal paths used to process the data are as you expect.

### SHOW STEPS Command

SHOW STEPS evaluates the specified tables and relationships and displays a description of the steps performed to select the data.

## SHOW STEPS Example

As an example, assume data is processed from three tables: ORDERS, CUSTOMERS, and DETAILS. The ORDERS table is the Start Table. The ORDERS table is a child of CUSTOMERS, and DETAILS is a child of ORDERS. The SHOW STEPS text is displayed as:

1. Step 1: Extract Rows from Start Table FOPDEMO.ORDERS. No Row List, Selection Criteria or Statistical Controls used, therefore Start Table does not need to be Revisited, even if part of a Cycle.
2. Step 2: Extract Rows from FOPDEMO.CUSTOMERS which are Parents of Rows Previously Extracted from FOPDEMO.ORDERS in Step 1 to satisfy an RI rule using Relationship RCO.
3. Step 3: Extract Rows from FOPDEMO.DETAILS which are Children of Rows Previously Extracted from FOPDEMO.ORDERS in Step 1 using Relationship ROD.

   The text for the first step, the Start Table, varies depending on the specifications. In addition, the first step may include:

   - When a Point-and-Shoot list is used:

     Row List is used and Determines the Rows Selected.
   - When Selection Criteria and/or Statistical Controls are used without a Point-and-Shoot list:

     Selection Criteria and/or Statistical Controls are used, these Determine the Rows Selected.

   This simple example does not reflect the potential complexity of processing. Steps to select the data can be repeated any number of times based on the relationships and specifications on the Specify Relationship Usage panel. A single step can process rows from more than one table or rows from one table to satisfy more than one relationship. For example, if the DETAILS table is related to both the CUSTOMERS table and the ORDERS table, Step 3 is:
4. Step 3: Extract Rows from FOPDEMO.DETAILS which are Children of Rows Previously Archived from FOPDEMO.ORDERS in Step 1 using Relationship ROD.
   --AND--
   Extract Rows from FOPDEMO.DETAILS which are Children of Rows Previously Archived from FOPDEMO.CUSTOMERS in Step 2 using Relationship RCD.

Any table may be revisited multiple times in successive steps. Cycles may be involved causing a set of tables to be traversed repeatedly until a complete pass through the cycle does not result in processing any additional rows.

Based on the information provided by SHOW STEPS, you can change the relationship usage and execute the SHOW STEPS command again.

## IMS Steps

For IMS, each hierarchy or logical relationship is represented by a step.

1. Step 1: Extract Rows from Start Table FOPDEMO.IMSCUST. No Row List, Selection Criteria or Statistical Controls used, therefore Start Table does not need to be Revisited, even if part of a Cycle.
2. Table FOPDEMO.IMSCUST is a segment type in an IMS hierarchy. For each segment extracted from FOPDEMO.IMSCUST, the following processing will be done:

   + Extract Segments from FOPDEMO.IMSORDERS which are children of the FOPDEMO.IMSCUST segment.

   + Extract Segments from FOPDEMO.IMSDETAILS which are children of the FOPDEMO.IMSORDERS segments.

## Handling the Text

All standard ISPF scrolling commands are available in the SHOW STEPS display.

You also can use the OUTPUT command to direct the contents of the SHOW STEPS display and the Specify Relationship Usage panel to an output file or SYSOUT. You are prompted for the output parameters and, based on your responses, the output is written to the destination.

Use END to return to the Specify Relationship Usage panel.

### END and CANCEL Commands

From the Specify Relationship Usage panel, use END to return to the Select Tables/Views for AD panel, or use CANCEL to remove all changes made on the current panel and return to the Select Tables/Views for AD panel.

## Relationship Index Analysis

One or more missing indexes may cause performance problems in an Archive or Extract Process. On the Specify Relationship Usage panel, use the SHOW INDEXES command to analyze indexes for selected relationships in the Access Definition. The **Relationship Index Analysis** feature is a diagnostic tool for determining whether indexes are needed. If the status of an index is Partial or None, creating the necessary index may enhance processing performance.

```
------------------------ Relationship Index Analysis --------------------------
Command ===>                                                 Scroll ===> PAGE

                                                                    1 of 3
AD: GRP.FOPDEMO.AD

Parent Ix    Child Ix
Stat/Needed  Stat/Needed  Relationship                  Parent Table
-----------  -----------  ----------------------------  ----------------------
********************************** TOP ***************************************
NotAnlyzed   None       Y FOPDEMO.ORDERS.RCO            FOPDEMO.CUSTOMERS
Full       Y NotAnlyzed   FOPDEMO.DETAILS.RID           FOPDEMO.ITEMS
NotAnlyzed   Full       Y FOPDEMO.DETAILS.ROD           FOPDEMO.ORDERS
******************************** BOTTOM **************************************
```

*Figure 52. Relationship Index Analysis*

### Panel

The Relationship Index Analysis panel lists each selected relationship referenced in the Access Definition, with an analysis of indexes for the corresponding parent and child tables, including:

**AD** The name of the Access Definition.

**Stat** The status of the indexes for each relationship.

**Full** One or more indexes exist for the complete set of required columns.

**Partial** One or more indexes exist for a partial set of the required columns.

**None** No index exists for the necessary columns.

**NotAnlyzed**
An index is not needed.

**Needed**
When appropriate, a '**Y**' indicator will appear next to each status to indicate when an index is needed. If the relationship's Status is Full, the index exists. If the relationship's Status is Partial or None, an index may enhance performance.

This area is blank if an index is not needed or was not analyzed.

**Relationship**
The fully qualified name of the child table in the relationship, followed by the relationship name.

**Parent Table**
The fully qualified name of the parent table in the relationship.

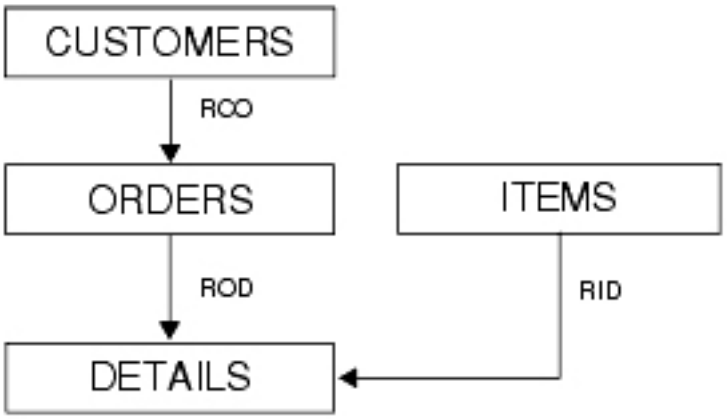
## Use Relationships

Optim uses relationships between tables to determine the traversal path for processing data. Use the Specify Relationship Usage panel to indicate the relationships that are traversed from table to table and how those relationships are used.

## Sample Scenarios

The examples in this section use a simple database structure to demonstrate functions of specifications on the Specify Relationship Usage panel. This structure includes four tables: CUSTOMERS, ORDERS, DETAILS, and ITEMS. CUSTOMERS is the parent of ORDERS, and ORDERS and ITEMS are parents of DETAILS.

### Basic Traversal Path




The most common traversal begins at the Start Table and proceeds through the data model, traversing all relationships from parent to child. For example, if the Start Table is CUSTOMERS, Optim automatically traverses downward to the ORDERS and DETAILS tables.

### Alter Traversal Path

You can alter the traversal path by selecting and unselecting relationships on the Specify Relationship Usage panel, as shown in Figure 53 on page 98. In addition, you can extend the traversal path using the **Q1** and **Q2** specifications.

The **Q1** and **Q2** specifications are relevant when:

- The Start Table is a child table. For example, if the ORDERS table is the Start Table, the **Q1** setting determines whether the related CUSTOMERS rows are extracted and, if so, the **Q2** setting determines if additional ORDERS rows, related to the CUSTOMERS rows, are selected.
- A table has more than one parent table. For example, if the ITEMS table is the Start Table, related rows are extracted from the DETAILS table because it is a child of the ITEMS table. However, the ORDERS table is also a parent of the DETAILS table. The **Q1** setting determines whether to traverse from child to parent to extract the ORDERS rows related to the DETAILS rows and, if so, the **Q2** setting determines if additional DETAILS rows, related to the ORDERS rows, are selected.

## Q1

The first setting, **Q1**, determines whether Optim follows a relationship from child to parent to extract data. By default, **Q1** is Y for Yes. The default setting ensures the relational integrity of the extracted data: that is, a parent row is extracted for every child row.

In the following figure, the Yes default for **Q1,** is shown for the three relationships.

```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 3

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************** TOP ************************************
___ SELECT Y N       CUSTOMERS              ORDERS                 RCO      DB2
___ SELECT Y N       ORDERS                 DETAILS                ROD      DB2
___ SELECT Y N       ITEMS                  DETAILS                RID      DB2
*** ******************************** BOTTOM ***********************************
```

*Figure 53. Default Specifications for Q1 and Q2*

### Multiple Parent Tables

Assume you want to extract the related ORDERS, DETAILS, and ITEMS rows for a specific set of customers.

1. Designate CUSTOMERS as the Start Table.
2. Specify selection criteria for the desired set of CUSTOMERS.
3. Use the default **Q1** setting (Yes) for RID, the relationship between ITEMS and DETAILS.

Optim extracts the CUSTOMERS rows that match the selection criteria and follows the relationship RCO to extract the child rows in the ORDERS table. Next, Optim follows the relationship ROD to extract the child rows in the DETAILS table. However, the ITEMS table is a parent to the DETAILS table. The **Q1** setting (Yes) directs processing to follow the relationship RID from child to parent and obtain the ITEMS rows related to the extracted DETAILS rows.

## Q2

The second setting, **Q2**, determines whether additional child rows are extracted when a parent row is extracted because of **Q1**. In other words, if Optim follows a relationship from child to parent and extracts a parent row, additional children of that parent are extracted when **Q2** is Yes.

### Child as Start Table

Assume you want to extract a set of CUSTOMERS and all related ORDERS beginning with specific criteria for ORDERS. You can:

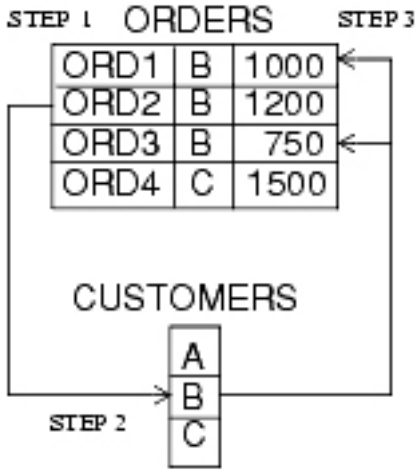
1. Specify ORDERS as the Start Table.
2. Specify selection criteria for the desired set of ORDERS.
3. Set **Q1** and **Q2** for relationship RCO to Yes, as shown in the following figure.

```
------------------------- Specify Relationship Usage --------------------------
Command ===>                                                 Scroll ===> PAGE

For Each Relationship Indicate:                              Rel 1 of 1

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************** TOP ***********************************
___ SELECT Y Y       CUSTOMERS              ORDERS                 RCO      DB2
*** ********************************* BOTTOM **********************************
```

*Figure 54. Both Q1 and Q2 Set to Yes*

Consider the following example. Customer B has three orders: 1, 2, and 3. Using Point-and-Shoot with the ORDERS table as the Start Table, only the row for order 2 is selected. **Q1** and **Q2** are Yes for the relationship RCO between CUSTOMERS and ORDERS.

The rows are extracted in the following sequence.




**Steps**

1. Based on the Point-and-Shoot selection, row **ORD2** of ORDERS is extracted.
2. The CUSTOMERS row **B** is extracted because of **Q1**.
3. ORDERS rows **ORD1** and **ORD3** are extracted because of **Q2**.

Since **Q2** is Yes, Optim also extracts the remaining ORDERS rows that are children of the extracted CUSTOMERS rows.

Note that rows extracted from a table because of **Q2** must meet any selection criteria specified for that table. In addition, they are subject to any statistical controls.

## Child to Parent for Multiple Relationships

Consider another example. Assume you need to extract orders that are outstanding because a specific item is discontinued. The set of extracted data should provide full information about the orders. To obtain complete information, data must be extracted from the CUSTOMERS, ORDERS, DETAILS, and ITEMS tables. Specify:

1. The ITEMS table as the Start Table.
2. Selection criteria for the discontinued ITEMS row.
3. Y for **Q1** for all relationships and for **Q2** for the relationship ROD, as shown in the following figure.

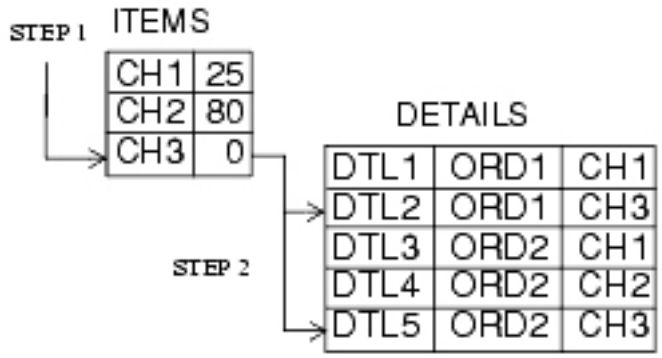
```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 3

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************* TOP ************************************
___ SELECT Y N       ITEMS                 DETAILS               RID      DB2
___ SELECT Y Y       ORDERS                DETAILS               ROD      DB2
___ SELECT Y N       CUSTOMERS             ORDERS                RCO      DB2
*** ******************************** BOTTOM ***********************************
```

*Figure 55. Set Q2 to Yes for One Relationship*

The rows are extracted as follows:

## ITEMS to DETAILS

After the specified ITEMS row is selected, the related DETAILS are extracted by traversing the relationship RID from parent to child.

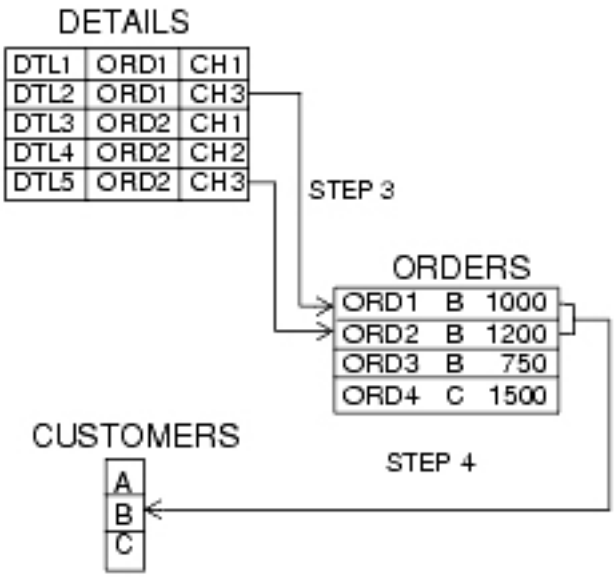



**Steps**

1. On the basis of the selection criteria, the single ITEMS row with a quantity of zero is extracted.
2. The DETAILS rows related to the item are extracted because of the parent to child traversal of relationship RID.

## DETAILS to ORDERS to CUSTOMERS

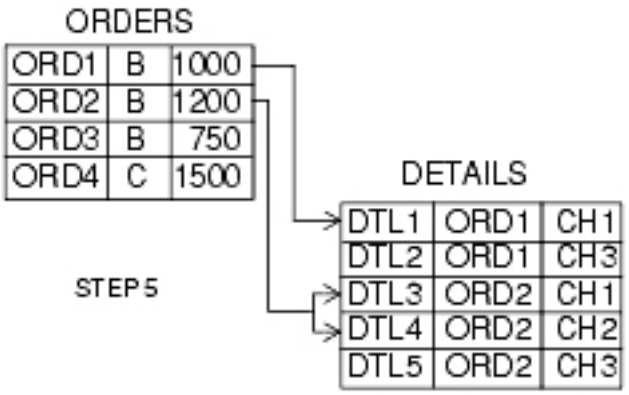
The ORDERS related to the DETAILS are extracted, followed by the CUSTOMERS related to the ORDERS.




**Steps**

1. The ORDERS that are related to these DETAILS are extracted by traversing the relationship ROD from child to parent because **Q1** is Yes.
2. The CUSTOMERS that placed these ORDERS are extracted by traversing the relationship RCO from child to parent, because **Q1** is Yes.

## ORDERS to DETAILS



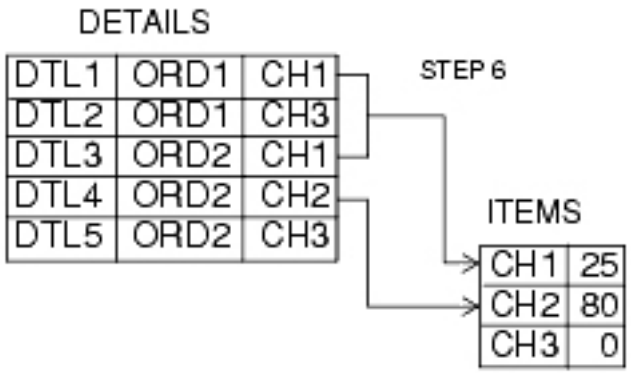

**Steps**

1. The additional DETAILS that are part of the ORDERS are extracted because **Q2** is Yes for the relationship between ORDERS and DETAILS.

The additional DETAILS related to the ORDERS are extracted.

## DETAILS to ITEMS

The ITEMS for the additional extracted DETAILS are extracted.




**Steps**

1. The ITEMS for the additional DETAILS are extracted because **Q1** is Yes for the relationship between ITEMS and DETAILS.

## No Additional ORDERS

Only the ORDERS for the selected ITEMS are extracted. Additional ORDERS for the extracted CUSTOMERS are not extracted because **Q2** is No for the relationship RCO between CUSTOMERS and ORDERS.

## Summary

**Q2** allows you to extract additional children. That is, if parent rows are extracted to satisfy **Q1** (making the set of extracted data referentially intact), the additional children for those parents are also extracted.

## Multiple Children

Often a table is parent to more than one table. The specifications for **Q1** and **Q2** apply only to the pair of tables in the relationship. For example, if a row in the CUSTOMERS table is extracted because **Q1** is Yes for the relationship between CUSTOMERS and ORDERS, then **Q2** for that relationship applies to related ORDERS rows only. To extract data from other child tables, **Q2** must be set to Yes for those relationships.

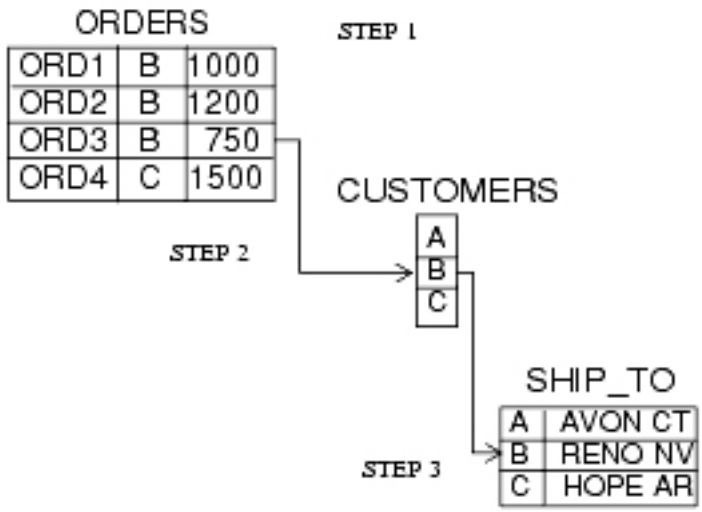
Assume the SHIP_TO and the ORDERS tables are children of the CUSTOMERS table. To obtain the shipping information for specific orders, specify:

1. The ORDERS table as the Start Table.
2. The selection criteria for the desired set of ORDERS.
3. Y for **Q1** for both relationships and for **Q2** for the relationship RCST. These settings are shown in the following figure.

*Figure 56. Set Q1 and Q2 for Multiple Children*

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                 Scroll ===> PAGE

For Each Relationship Indicate:                              Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************** TOP ***********************************
___ SELECT Y N       CUSTOMERS             ORDERS                RCO      DB2
___ SELECT Y Y       CUSTOMERS             SHIP_TO               RCST     OPT
*** ******************************** BOTTOM *********************************
```

The rows are extracted in the sequence shown in the following figure:




**STEPS**

1. A single ORDERS row is extracted, based on the selection criteria.
2. The CUSTOMERS row **B** is extracted because **Q1** is Yes for the relationship RCO.
3. The SHIP_TO row **B** is extracted because **Q1** and **Q2** are Yes for the relationship RCST.

## Traversal Cycles

Optim can cycle through tables to process data. These “traversal cycles” depend on the specifications for each relationship.

The traversal cycle in this example results from extracting the item with an ITEM_ID of AD005 and traversing as described in the following steps to obtain a complete set of orders. ITEMS is the Start Table. (Note that the CUSTOMERS table is not used in this example; the relationship between CUSTOMERS and ORDERS, RCO, is unselected.) The Specify Relationship Usage panel is defined as shown in the following figure:

```
------------------------ Specify Relationship Usage --------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 3

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************* TOP ***********************************
___ SELECT Y N       ITEMS                 DETAILS               RID      DB2
___ SELECT Y Y       ORDERS                DETAILS               ROD      DB2
___ UNSEL  Y N       CUSTOMERS             ORDERS                RCO      DB2
*** ******************************** BOTTOM *********************************
```

*Figure 57. Set Q1 and Q2 for Multiple Relationships*

The following steps are performed for this traversal cycle:

1. Extract the ITEMS with ITEM_ID AD005.
2. Extract the DETAILS related to the ITEMS row by traversing the relationship RID from parent to child.
3. Extract the ORDERS related to the DETAILS by traversing the relationship ROD from child to parent. (**Q1** is Yes.)
4. Extract the additional DETAILS for those ORDERS by traversing the relationship ROD from parent to child. (**Q2** is Yes.)
5. Extract the ITEMS related to the additional DETAILS by traversing the relationship RID from child to parent. (**Q1** is Yes.)

A complete cycle is made, beginning with the ITEMS table and ending at the ITEMS table.

### Multiple Traversal Cycles

Optim also can process multiple traversal cycles. Although unlikely, the previous example can be extended to show multiple traversal cycles.

After extracting the ITEMS rows for the additional items in Step 5, you may want to extract the ORDERS and DETAILS for these ITEMS. To accomplish this, **Q2** for the relationship RID is set to Yes. After Step 5 in the previous example, Optim traverses the relationship RID from parent to child to extract the remaining DETAILS related to these additional ITEMS. Then Steps 3, 4, and 5 are repeated to extract additional ORDERS, the DETAILS related to those ORDERS, the ITEMS related to those DETAILS, and so on.

## Referential Cycles

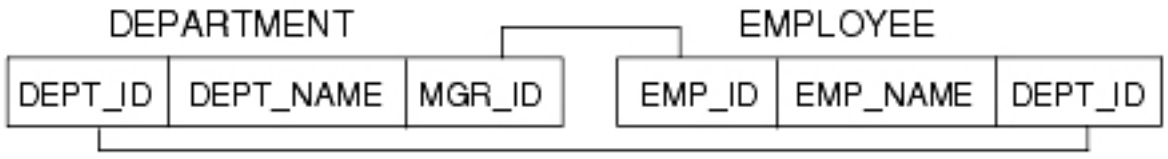
Optim can extract data from tables that are related to each other by referential cycles. With a referential cycle, you start at one table and return to it after traversing a series of one or more relationships. (In contrast, many DB2 databases are defined as hierarchies or networks. To process data, you proceed up or down through the hierarchy or the network and never return to the Start Table.)

### Two Tables

For this discussion, assume there are two tables: DEPARTMENT and EMPLOYEE. DEPT_ID is the primary key of the DEPARTMENT table; EMP_ID is the primary key of the EMPLOYEE table.

## Relationship

The relationships are shown in the following figure:




The data in each table is as follows:

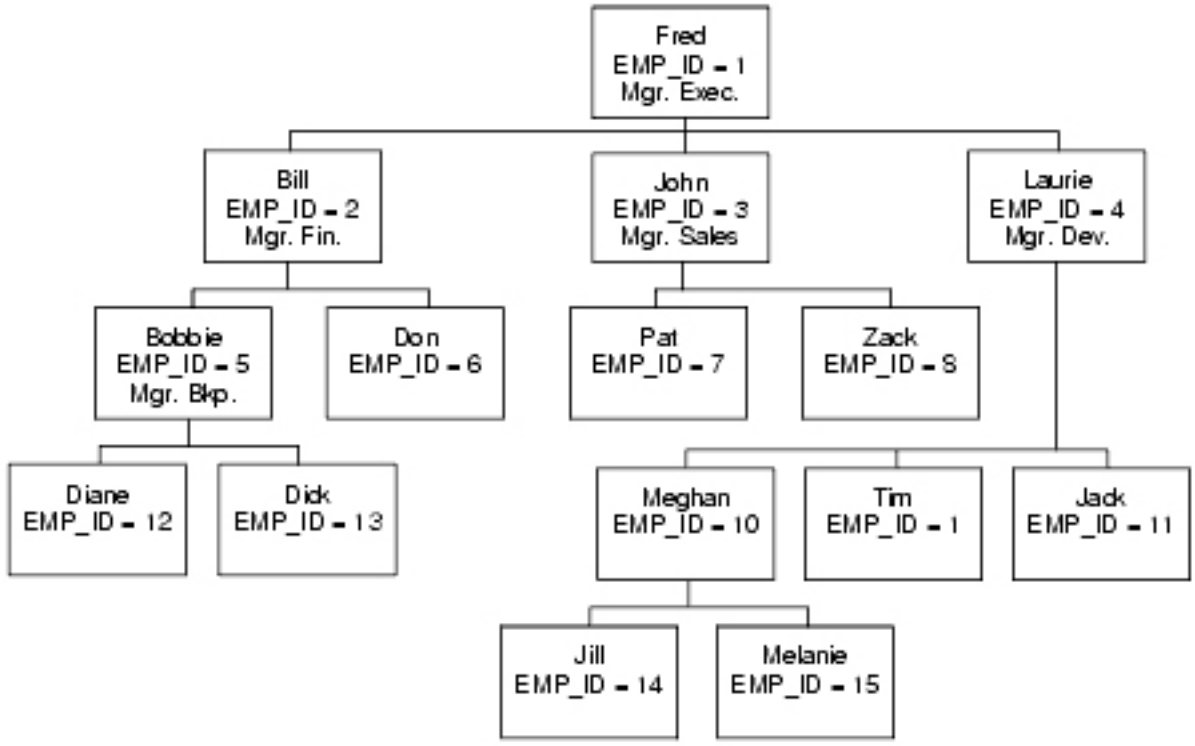
*Table 1. DEPARTMENT*

| DEPT_ID | DEPT_NAME | MGR_ID |
|---|---|---|
| A | Executive | 1 |
| B | Finance | 2 |
| C | Sales | 3 |
| D | Development | 4 |
| E | Bookkeeping | 5 |
| F | Support | 9 |

*Table 2. EMPLOYEE*

| EMP_ID | EMP_NAME | DEPT_ID |
|---|---|---|
| 1 | Fred | A |
| 2 | Bill | A |
| 3 | John | A |
| 4 | Laurie | A |
| 5 | Bobbie | B |
| 6 | Don | B |
| 7 | Pat | C |
| 8 | Zack | C |
| 9 | Meghan | D |
| 10 | Tim | D |
| 11 | Jack | D |
| 12 | Diane | E |
| 13 | Dick | E |
| 14 | Jill | F |
| 15 | Melanie | F |

The following is a sample company organization chart with pertinent values in the tables.




In the following pages, a sample query is used to demonstrate how to handle cyclic relationships through the specifications for **Q1** and **Q2**. Changing these specifications changes the set of data extracted from these tables.

## Example 1

Extract all employees in the Development Department, including employees who are members of subordinate departments. The DEPARTMENT table is the Start Table. The Point-and-Shoot facility is used to select the appropriate row in the DEPARTMENT table. Since the existing relationships only need to be traversed from parent to child, specifications for **Q1** and **Q2** are set to No, as shown in the following figure:

```
------------------------- Specify Relationship Usage --------------------------
Command ===>                                                 Scroll ===> PAGE

For Each Relationship Indicate:                              Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table           Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************** TOP ***********************************
___ SELECT N N       DEPARTMENT            EMPLOYEE              MEMBER   DB2
___ SELECT N N       EMPLOYEE              DEPARTMENT            MANAGER  DB2
*** ******************************** BOTTOM *********************************
```

*Figure 58. Set Q1 and Q2 to NO*

The following steps are performed to extract the data:

1. Extract the row containing the Development Department, Dept_ID = D, from the DEPARTMENT table.
2. Using the relationship MEMBER, search the EMPLOYEES table for all employees in the Development Department with EMPLOYEE.DEPT_ID = D. Three EMPLOYEE rows are extracted:

   EMP_ID = 9
   EMP_ID = 10
   EMP_ID = 11
3. The relationship MANAGER is used to search the DEPARTMENT table to determine whether any extracted employees are managers. The employee with EMP_ID = 9, Meghan, is a manager of the Support Department, DEPT_ID = F. This row is extracted from the DEPARTMENT table.
4. The relationship MEMBER is used to search the EMPLOYEE table to extract all employees that belong to the Support Department, DEPT_ID = F.

   Two EMPLOYEE rows are extracted:

   EMP_ID = 14
   EMP_ID = 15
5. The relationship MANAGER is used to search the DEPARTMENT table to determine if either of the two extracted rows in step 4 are managers. They are not. No more rows are extracted.

## Summary

Two rows are extracted from the DEPARTMENT table:

DEPT_ID = D
DEPT_ID = F

Five rows are extracted from the EMPLOYEE table:

EMP_ID = 9
EMP_ID = 10
EMP_ID = 11
EMP_ID = 14
EMP_ID = 15

## Example 2

In addition to all employees in the Development Department and subordinate departments, extract the manager of the Development Department.

As in Example 1, the DEPARTMENT table is the Start Table and the row in the DEPARTMENT table where DEPT_ID = D is selected during Point-and-Shoot. To extract the manager of the Development Department, the relationship MANAGER must be traversed from child to parent. **Q1** for the relationship MANAGER is changed to Yes, as shown in the following figure.

```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************* TOP ***********************************
___ SELECT N N       DEPARTMENT            EMPLOYEE              MEMBER   DB2
___ SELECT Y N       EMPLOYEE              DEPARTMENT            MANAGER  DB2
*** ******************************** BOTTOM *********************************
```

*Figure 59. Set Q1 to Extract Parent*

The steps shown in Example 1 are performed to extract the data. In addition, Example 2 also traverses the relationship MANAGER from child, DEPARTMENT, to parent, EMPLOYEE. Thus, Optim extracts the manager of the Development Department, Laurie. An additional row from the EMPLOYEE table is extracted, EMP_ID = 4.

## Example 3

In addition to all employees in the Development Department and the manager of the Development Department, extract any other departments this manager manages.

To obtain the other departments, both Q1 and Q2 for the relationship MANAGER are set to Yes, as shown in the following figure.

```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- --------------------- --------------------- ------>> ---
*** ********************************* TOP ***********************************
___ SELECT N N       DEPARTMENT            EMPLOYEE              MEMBER   DB2
___ SELECT Y Y       EMPLOYEE              DEPARTMENT            MANAGER  DB2
*** ******************************** BOTTOM *********************************
```

*Figure 60. Set Q1 and Q2 to YES*

In this example, after extracting Laurie from the EMPLOYEE table, Optim follows the MANAGER relationship from parent, EMPLOYEE, to child, DEPARTMENT. Optim checks the DEPARTMENT table to determine whether Laurie is manager of any other department. Based on the sample employee hierarchy used for these examples, no additional rows are extracted. Laurie manages only one department.

## Example 4

In addition to all employees in the Development Department, the manager of the Development Department and any other departments the extracted managers manage, extract any other departments where the manager is a member.

To obtain the other departments where the manager is a member, Q1 for the relationship MEMBER is set to Yes, as shown in the following figure.

```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************* TOP ***********************************
___ SELECT Y N       DEPARTMENT             EMPLOYEE               MEMBER   DB2
___ SELECT Y Y       EMPLOYEE               DEPARTMENT             MANAGER  DB2
*** ******************************** BOTTOM ********************************
```

*Figure 61. Set Q1 to YES*

In this example, after extracting Laurie from the EMPLOYEE table, Optim follows the MEMBER relationship from child to parent, from EMPLOYEE to DEPARTMENT. Optim checks the DEPARTMENT table to determine if Laurie is a member of any other department. Laurie is a member of the Executive Department. An additional row from the DEPARTMENT table is extracted, DEPT_ID = A.

Since **Q1** is Yes for the MANAGER relationship, Optim follows this relationship from child to parent, from DEPARTMENT to EMPLOYEE, and extracts the manager of the Executive Department, Fred. An additional row from the EMPLOYEE table is extracted, EMP_ID = 1.

## Example 5 Disable RI Cycles

Extract only the employees that are members of the Development Department. Unlike Example 1, do not extract employees in subordinate departments.

To obtain only the employees in the Development Department, **Q1** and **Q2** are set to No for the relationship MEMBER. In addition, the relationship MANAGER is unselected. This relationship is not traversed, as shown in the following figure.

```
------------------------ Specify Relationship Usage -------------------------
Command ===>                                                Scroll ===> PAGE

For Each Relationship Indicate:                             Rel 1 of 2

Q1: If a Child Row is Included, Include its Parent Row to Satisfy the RI Rule?
Q2: If a Parent Row is Included to Satisfy any RI Rule, Include All Child Rows?

           Q Q Child                                                --Relation--
Cmd Status 1 2 Limit      Parent Table          Child Table          Name  Type
--- ------ - - ----- ---------------------- ---------------------- ------>> ---
*** ********************************* TOP ***********************************
___ SELECT N N       DEPARTMENT             EMPLOYEE               MEMBER   DB2
___ UNSEL  N N       EMPLOYEE               DEPARTMENT             MANAGER  DB2
*** ******************************** BOTTOM ********************************
```

*Figure 62. Unselect to Disable RI Cycle*

Setting **Q1** and **Q2** to N for No disables the RI cycle. Optim extracts the DEPARTMENT row for the Development Department and, using the relationship MEMBER, traverses from parent to child to extract

the three employees in the department. Since the MANAGER relationship is unselected, it is not traversed. The extract is complete.

## Dynamically Define the Access Definition

You can dynamically add tables and relationships to an Access Definition and specify sort, SQL, or selection criteria.

The ability to alter an Access Definition while using it to browse or edit data is controlled by parameters specified on the Access Definition Parameters panel (see "Access Definition Parameters" on page 112).

This dynamic facility is available only when the Access Definition editor is invoked from Option 5 ADS on the **Main Menu**, or from the **Choose a Definition Option** menu. Use the BROWSE or EDIT command to invoke a browse or edit session for the set of data defined by the Access Definition. During this session you can "join" to tables not included in the Access Definition and specify sort, SQL, and selection criteria for any table as you display the data.

For the most part, the facilities available when browsing or editing data are the same as those provided during a Point-and-Shoot session, and are not discussed here. Based on the Access Definition Parameters, the following facilities are also available during a browse or edit session for an Access Definition invoked from the **Main Menu** or the **Choose a Definition Option** menu.

- Use the NEW operand with the JOIN command to specify the name of a table not listed in the Access Definition. The newly joined table and any pertinent relationships are automatically included in the Access Definition.
- All specifications for sort, SQL, and selection criteria are included in the Access Definition.

This dynamic capability provides another means of defining an Access Definition, and it allows you to view the set of data you want to ensure the correct Access Definition.

**Note:** Any dynamic specifications you make during a Point-and-Shoot session, or when you use the BROWSE command from the Access Definition editor during an Extract, Compare, or Archive Process, are temporary and are not saved in the Access Definition.

### Browse or Edit Tables (Access only)

You also can invoke a browse or edit session for a table, rather than using an Access Definition. Access creates a temporary Access Definition that references the selected table and includes any criteria specifications. A prompt allows you to save and name this temporary Access Definition at the end of the session. The saved Access Definition reflects all dynamic specifications made during the browse or edit session, including all joins, sort, SQL, and selection criteria. For details about defining an Access Definition while browsing or editing a table, see the *Access User Manual*, Dynamic Access.

## Database Changes

An Access Definition can be affected by changes to the database. The extent of the effect varies, depending upon what is changed and whether or not changed data is directly referenced in the Access Definition. The following lists the possible changes and their potential effect.

### Column Added

A column that is added to a table referenced in an Access Definition is automatically processed. Any columns added to a table referred to as the source table in a Column Map are included in the list of unused columns. If a column is added to the destination table, Optim attempts to match a source column to the new column. If a source column is not available, Optim applies the defined column default. (For information on Column Maps, see Chapter 5, "Column Maps," on page 155.)

For Access, a new column is displayed if all columns in the table are displayed. If selected columns in the table are displayed, a new column is not displayed. You can also use the Describe Columns for AD panel to specify whether to display the new column.

## Column Deleted

When a column is deleted from a table referenced in an Access Definition, you should be aware of the impact. If you process the affected table, you receive a message:

**Error** You receive an error message if the column is referenced in the process. A column is referenced when it is:

- Included in an SQL predicate.
- Part of the primary key.

**Note:** For Access, a column is also referenced when explicitly designated for display or referenced in the sort criteria. You can use the Access Definition to browse or edit data, but the affected table cannot be the Start Table, and you cannot join to the affected table.

You must remove any explicit references to the column in the Access Definition.

**Warning**
You receive a warning message if the deleted column is not explicitly referenced. Processing can continue.

**Note:** For Access, you can use the Access Definition to browse or edit the table. A column is implicitly selected for display if all columns are displayed by default.

Any column that is deleted from a table cannot be explicitly designated in a Column Map.

## Column Modified

If the characteristics of a column have changed since the Access Definition was last modified, a warning message is issued when a process is attempted or you attempt to display data from the table.

For example, a change in the data length or data type affects the data handling by an Insert, Restore, or Delete Process, if the column participates in the process. An appropriate warning message is issued. You should verify that the desired data will be selected.

## Table Added

Adding a table to the database has no direct effect on the Access Definition or any processes. In a browse or edit session, you can join to the new table and thereby reference it in the Access Definition.

## Table Deleted

If a table in an Access Definition is deleted from the database, the table is labeled as UNKNOWN in the Access Definition. The Access Definition can be used, but the deleted table cannot be the Start Table, and you cannot join the UNKNOWN table.

Optim issues a warning message for this table when a process is initiated.

## Relationships Added, Deleted, or Modified

When relationships are added or deleted, they directly affect the specifications on the Specify Relationship Usage panel. Deleted relationships are placed at the end of the list and assigned the UNKNWN status. New relationships are included at the beginning of the list and assigned the NEW status. If the Specify Relationship Usage panel is not displayed, all NEW relationships are included in the

process by default and a warning message is displayed to inform the user of this situation. Set **Use NEW Relationships** on the Access Definition Parameters panel to No to prevent these NEW relationships from being traversed.

Modified relationships are handled in the same manner as newly specified relationships.

**Note:** Changes to relationships may affect the Join facility during a browse or edit session.

## Access Definition Parameters

You can provide a variety of parameters for the Access Definition using the Access Definition Parameters panel.

Display this panel by using the PARAMETERS command on the Select Tables/Views for AD panel.

```
----Access Definition Parameters: GRP.USER.ADSAMPLE --------------------------
Command ===>

Dynamically Add Tables    ===> Y          (Y-Yes, N-No)

Modify Selection Criteria ===> Y          (Y-Yes, N-No)

Begin Table Display with  ===> D          (D-Data,
                                           S-Criteria for Start Table only,
                                           Q-SQL for Start Table only,
                                           A-Criteria for All Tables)

Changes to AD During Edit ===> P          (P-Permanent, T-Temporary)

Use NEW Relationships     ===> Y          (Y-Yes, N-No)

Apply Criteria Self Ref   ===> Y          (Y-Yes, N-No)

Archive Expiration Value  ===>            (YYYY-MM-DD or DDDDD)
```

*Figure 63. Access Definition Parameters*

### Panel

The Access Definition Parameters panel includes:

**Dynamically Add Tables**
Indicator for joins with tables that are not referenced in the Access Definition. This setting applies when browsing data while defining an Access Definition, or when an Access browse or edit session is invoked from a menu option. Specify:

**Y** New tables can be added. Default.

**N** New tables cannot be added.

**Modify Selection Criteria**
Indicator for defining or editing selection criteria when browsing data while defining an Access Definition (for all Optim components), when editing data for Access, or during a Point-and-Shoot session (for Archive, Compare, and Move). Specify:

**Y** Criteria can be specified or modified. Default.

**N** Criteria cannot be specified or modified.

**Begin Table Display with**
Indicator for the beginning display when browsing data while defining an Access Definition (for

all Optim components), when browsing or editing data for Access, or during a Point-and-Shoot session (for Archive, Compare, and Move). Specify:

**D** Data
Display begins with data from the named Start Table. Default.

**S** **Criteria for Start Table Only**
Display begins with a prompt for selection criteria for the named Start Table.

**Q** **SQL for Start Table Only**
Display begins with the SQL WHERE panel for the named Start Table.

**A** **Criteria for All Tables**

Display begins with a prompt for selection criteria for the named Start Table and all subsequently joined tables.

**Changes to AD During Edit**
Indicator for the life of changes to an Access Definition during an Access edit or browse session. Specify:

**P** Changes are saved for future use.

**T** Changes apply to the current session only.

**Use NEW Relationships**
Indicator for traversing NEW relationships that have not been explicitly selected or unselected. Specify:

**Y** Traverse NEW relationships. Default

**N** Do not traverse NEW relationships.

**Apply Criteria Self Ref**
Indicator for the application of selection criteria when a table is self-referenced. This setting applies only when browsing or editing data using Access. Specify:

**Y** Apply any selection criteria for a table each time the table is referenced. Default.

**N** Apply selection criteria to the Start Table the first time it is accessed, but not when the table is self-referenced.

**Archive Expiration Value**
Expiration parameters for Archive Files generated in Archive Processes that use the Access Definition. (Archive only) Specify:

*blank* No expiration date.

***YYYY-MM-DD***
An explicit date, after which the data in the Archive File is considered to be expired. An error occurs if an Archive Process uses the Archive File created with this Access Definition after the specified date.

***DDDDD***
The number of days until expiration.

You can use the batch DIRECTORY utility to list expired Archive Files and the DELETE utility to delete them. (See the *Batch Utilities Guide*, Maintenance Utilities for detailed information on these utilities.)

## Available Commands

The following primary commands are available when the Access Definition Parameters panel is displayed.

- BROWSE

- CANCEL
- EDIT
- END
- OPTIONS
- SAVE

**Note:** The EDIT command is available for Access only.

Use END to return to the Select Tables/Views for AD panel.

# Chapter 3. Primary Keys

A primary key is the column or columns that contain values that uniquely identify each row in a table.

A primary key is needed:

- To extract or archive data from a table that is visited more than once during a process (for example, a child table that has two or more parent tables referenced in the Access Definition).
- To insert, restore, or delete data from a table. The primary key is used to determine whether a row exists.
- To enable a Point-and-Shoot session for a Start Table.

Optim uses primary keys that are defined to the DB2 Catalog. You can, however, define Optim primary keys to supplement those in the DB2 Catalog. If a primary key is not defined and is required to perform a specific operation, Optim prompts you to create an Optim primary key.

You can define two types of primary keys that are stored in the Optim Directory:

- An Explicit primary key applies to a single table.
- A Generic primary key applies to any tables that have the same base name, column names, and attribute specifications, but different Creator IDs.

There is no difference in function or appearance between generic and explicit primary keys. However, if a table has keys of both types, Optim uses the explicit primary key.

## Legacy Tables

For *Move* or *Compare for IMS, VSAM, and Sequential Files* only, you can create an Optim primary key for a VSAM or sequential file Legacy Table as though the Legacy Table were a DB2 table. Fields in a Legacy record are treated as columns for the purpose of creating a primary key.

You can select and browse IMS primary keys, but IMS primary keys cannot be modified. Refer to *Move User Manual* or *Compare for IMS, VSAM, and Sequential Files*.

## Match Keys

When a primary key has not been defined for either table in a pair of tables to be compared, you are prompted to define a "match key." (Compare only)

Match keys are used to "match" rows from one source with rows from the other source for the comparison. When available, Compare uses a primary key from a source table for the Compare Process. However, when a usable primary key is not defined in the DB2 Catalog or in the Optim Directory for either table, you are prompted to create a match key.

For the most part, the prompts to define match keys are the same as those documented in this section for defining primary keys. When defining a match key, the column names in both source tables must match — either directly, having the same base name and attributes, or indirectly, using a Column Map to match unlike names.

Also note that, unlike the DB2 primary keys, match keys need not be based on a unique index and, unlike OPT primary keys, match keys are not available to other users or processes. Match keys reside in the Compare Definition only.

### For Relationships

If present, primary keys assist in defining relationships. Specifically, when you create a relationship and the parent table has a primary key, the names of the primary key columns are automatically inserted into the Define Relationship panel. You may then modify the column information as desired.

## Select a Primary Key

To edit or create a primary key, select Option 6.1 Primary Keys from the **Main Menu**. You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu and select Option 1 PRIMARY KEYS. In either case, the Choose a Primary Key panel is displayed.

```
----------------------------- Choose a Primary Key  ------------------------
COMMAND ===>

Primary Key:
  Creator ID ===> FOPDEMO                                          >
  Table Name ===> CUSTOMERS                                        >

Primary Key Type               ===> OPT    (P|O-OPTIM, D-DB2, I-IMS, A-All)

Use '_' for DB2 LIKE character ===> NO     (Y-Yes, N-No)
```

*Figure 64. Choose a Primary Key*

Use this panel to name a new Optim primary key and select existing Optim primary keys to modify or delete. You can also browse DB2 and IMS primary keys.

### Panel Prompts

The Choose a Primary Key panel includes:

**Primary Key:**
The Creator ID and Table Name for the primary key. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Creator ID**
The Creator ID for the table for which the primary key is being created or modified. The default is the previously entered value or the TSO ID of the current user if a **Creator ID** has never been specified. Specify 1 to 128 characters. To create or edit a generic primary key, type an asterisk at this prompt. (Refer to "Edit a Generic Primary Key" on page 122 for details).

**Table Name**
The name of the table for which the primary key is defined or modified. Specify 1 to 128 characters.

**Note:** See "Long Object Names (LONs)" on page 8 for information on entering Creator IDs and Table Names of up to 128 characters.

**Primary Key Type**
Indicate the type of primary keys to include in the selection list. This value has no effect if you indicate an explicit **Creator ID** and **Table Name**. Specify:

**P or O**
Include primary keys defined in the Optim Directory.

**D** Include primary keys defined to DB2.

**I** Include primary keys defined to IMS. (Optim Move for Legacy only.)

**A** Include all types.

**Use '_' for DB2 LIKE character**
Use of the underscore ( _ ) character. Specify Y to use the underscore as a DB2 LIKE character, or specify N if it is used literally as part of the name.

**Note:** When the '_' character is used in conjunction with the '%' DB2 LIKE character, the '_' is treated as a 'DB2 like' character, and not as a literal.

For example, depending on the use of the underscore character, A_B is a three-character name containing the characters "A_B", as entered, or a three-character name that begins with "A" and ends with "B" with any valid character in the middle. The default is **No**, which means that "_" is not handled as a DB2 LIKE character.

### Explicit Table Name

When you supply an explicit **Creator ID** and **Table Name** and the table does not exist, an error message is displayed on the Choose a Primary Key panel. If the supplied name is for a DB2 table, Optim checks the DB2 Catalog to determine if a primary key is defined for the table. If the name references an IMS Legacy Table, Optim checks the IMS DBD to determine if a key is defined for the associated IMS segment.

- If a DB2 or z/OS primary key for the table exists, the primary key information is displayed. This information cannot be modified.
- If a DBMS primary key for the table does not exist, Optim checks the Optim Directory. If a primary key is defined in the Optim Directory, the information is displayed and can be modified.
- If the database or Legacy Table exists and no primary key information is available, you are prompted to define a new primary key for the table. The new primary key is stored in the Optim Directory.

### Selection List

A selection list is requested using DB2 LIKE syntax or by leaving one or both of the Primary Key prompts blank (i.e., Creator ID and Table Name).

- DB2 tables, IMS Legacy Tables, or all are listed, depending on the value specified for **Primary Key Type**.
- If no tables satisfy the selection list criteria, a message is displayed on the Choose a Primary Key panel.

## Primary Key Selection List

This topic discusses the primary key selection list.

An example of a primary key selection list is shown in the following figure.

```
---------------------------- Select Primary Keys -----------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 4

Cmd Creator        Table        Type    By           Date
--- -------- ------------------ ---- -------- ------------------
    ****************************** TOP ***********************************
___ *        ORDERS             OPT  DB2MGR   2005-09-03-08.45.50
___ FOPDEMO  CUSTOMERS          DB2  DB2MGR   2005-08-26-16.06.30
___ FOPDEMO  DETAILS            DB2  DB2MGR   2005-07-09-11.55.47
___ FOPDEMO  SHIP_INSTR         OPT  DB2MGR   2005-07-09-11.55.47
    ****************************** BOTTOM ********************************
```

*Figure 65. Select Primary Keys*

Tables for which a primary key is defined are listed. The Creator ID and table name are displayed. An asterisk (*) in **Creator** indicates a generic primary key. **Type** indicates the source of the primary key, as defined in the Optim Directory (OPT), the DB2 Catalog (DB2), or in an IMS DBD (IMS).

## Description

A User Option setting determines whether the description of each Optim primary key is displayed. See "User Options" on page 372 for information about the **Selection List Format** option.

## Line Commands

The **Cmd** area of the panel is used to enter line commands.

**Cmd** The line command entry area. Valid line commands are:

- **S** Select a primary key.
- **D** Delete an Optim primary key.
- **C** Copy an Optim primary key.
- **R** Rename an Optim primary key.
- **AT** Modify attributes of an Optim primary key.
- **I** Display information about an Optim primary key

See "Object Selection List Functions" on page 14 for more information about available commands.

The following messages may be displayed when you enter a line command for a primary key.

- The message "*BUSY" is displayed when a primary key is being modified by another user.
- The message "*NO COLUMN MATCH" is displayed when there are no Legacy Table columns that match an IMS primary key.

## Primary Key Attributes

To display the attributes of an Optim primary key, use the I line command next to the name of the primary key. The following figure shows the Primary Key Attributes panel.

```
--------------------------- Primary Key Attributes ----------------------------
Command ===>                                                 SCROLL ===> PAGE

Table ID           : FOPDEMO
Table Name         : CUSTOMERS
Description        :
Security Status    : PUBLIC

Last Modified By   : DB2MGR
Modified On        : 2005-08-26-16.06.30
```

*Figure 66. Primary Key Attributes*

This is a read-only display. Optim generates and maintains the information on this panel based on your specifications for the primary key.

## Edit or Browse a Primary Key

This topic discusses editing and browsing primary keys.

When you enter the fully qualified name of a table on the Choose a Primary Key panel and the table:

- Has a DB2 or IMS primary key, the Browse DB2 Primary Key or Browse IMS Primary Key panel is displayed to allow you to view information about the primary key.
- Has no primary key, the Define Optim Primary Key panel is displayed to allow you to create the primary key.
- Has only an Optim primary key, the Modify Optim Primary Key panel is displayed.

The appearance of these panels is identical. The operation of the Define Optim Primary Key and the Modify Optim Primary Key panels is identical and is described in this section. However, the Browse Primary Key panel is read-only.

```
--------------------------- Modify OPTIM Primary Key --------------------------
Command ===>                                                 Scroll ===> PAGE

Modify OPTIM Primary Key for FOPDEMO.SHIP_INSTR
Limit Key to 64 Columns and 2000 Bytes

Cmd          Column Name             Data Type                1 OF 1
--- ---------------------------- ---------------
*** ********************** TOP **********************
___ SHIP_INSTR_ID                DECIMAL(5,0)
*** ********************** BOTTOM **********************
```

*Figure 67. Modify Optim Primary Key*

### Panel

The Primary Key panel shows the name of the table for the primary key, as well as:

**Cmd** The line command entry area. Valid line commands on the Define/Modify Optim Primary Key panel are:

**D, Dn, or DD**
Delete one or more column names. DD is the block form of the command.

**I or In** Insert one or more blank lines.

**M, Mn, or MM**
Move one or more column names. MM is the block form of the command.

**A or B** Destination (After or Before) for the Move line command.

**Column Name**
The name of each column in the primary key. When editing an Optim primary key, you can specify a maximum of 64 columns with a total of 2000 bytes.

**Data Type**
The data type for each column in the primary key. Optim automatically displays the data type. This value cannot be modified.

## Selection List of Columns

Use the LIST COLUMNS command to display the columns available to be included in the primary key. Columns currently included in the primary key are not listed. For example, type the LIST COLUMNS command to display the Select One or More Columns pop-up for the SHIP_INSTR table.

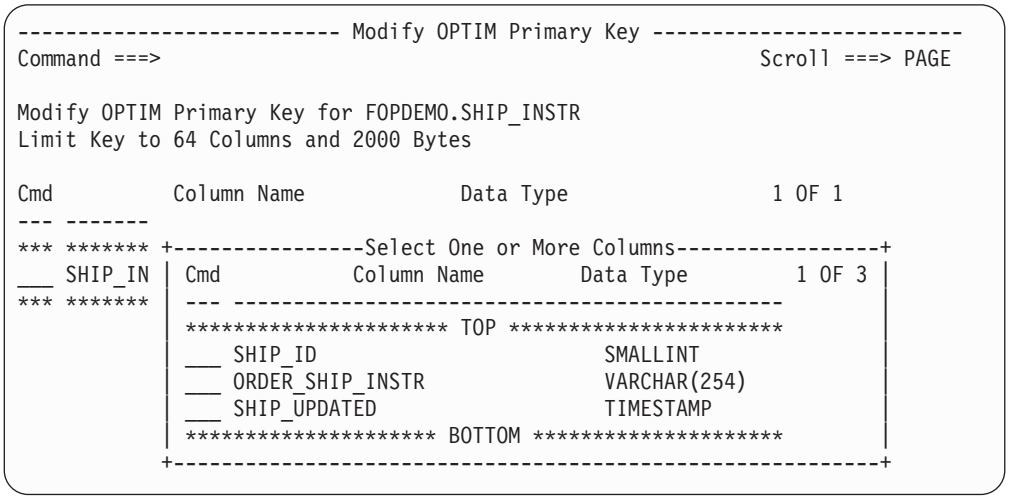



*Figure 68. Select One or More Columns*

For each available column, the name and data type are listed. Use the S line command to select columns for the primary key. When finished, press ENTER or use END to redisplay the Modify Optim Primary Key panel. The selected column names are inserted in the list of columns comprising the key.

## Selection List of Indexes

One or more unique indexes on a database table are often defined for various purposes. To define a primary key based on an existing index, you can use the LIST UNIQUE INDEX primary command to display a selection list of unique indices. Use the S line command to select the appropriate index or indices. When you select an index as the primary key, the names of columns in the unique index replace all column names previously specified on the Modify Optim Primary Key panel.

**Note:** LIST UNIQUE INDEX requires a tablespace scan of the catalog table, SYSIBM.SYSINDEX. This scan may be time consuming.

## Specification Complete

Press ENTER or use END to end the Modify Optim Primary Key panel processing.

Primary key information is stored in the Optim Directory and used the next time the table is accessed. Any change to the information affects all users.

## Available Primary Commands

The following primary commands are available from the Modify Optim Primary Key panel.

ATTRIBUTES
BOTTOM
CANCEL
DELETE
DOWN
END
GENERIC
LIST COLUMNS
LIST UNIQUE
RESET
TOP
UP

## Description

You can display or edit a description for an Optim primary key to be displayed in selection lists. Use the ATTRIBUTES command to display the Object Attributes panel, which provides a 40-character **Description** prompt. For additional information about the Object Attributes panel, see "Specify Description and Security Status" on page 380.

## Delete a Primary Key

In addition to using the DELETE command on the Modify Optim Primary Key panel to delete an Optim primary key, you can also remove all column names and use END.

## Convert to a Generic Primary Key

To convert an explicit primary key to a generic primary key, use the GENERIC command. Enter the GENERIC command on the Modify Optim Primary Key, the Define Optim Primary Key, or the Browse DB2 Primary Key panel to convert the displayed explicit primary key into a generic primary key.

If a generic primary key already exists, a prompt requests confirmation to override it. If you convert an Optim primary key for which a generic primary key does not exist, the following pop-up prompts you to delete or retain the original, explicit Optim primary key. Use END to retain the existing explicit primary key, or press ENTER to delete it.

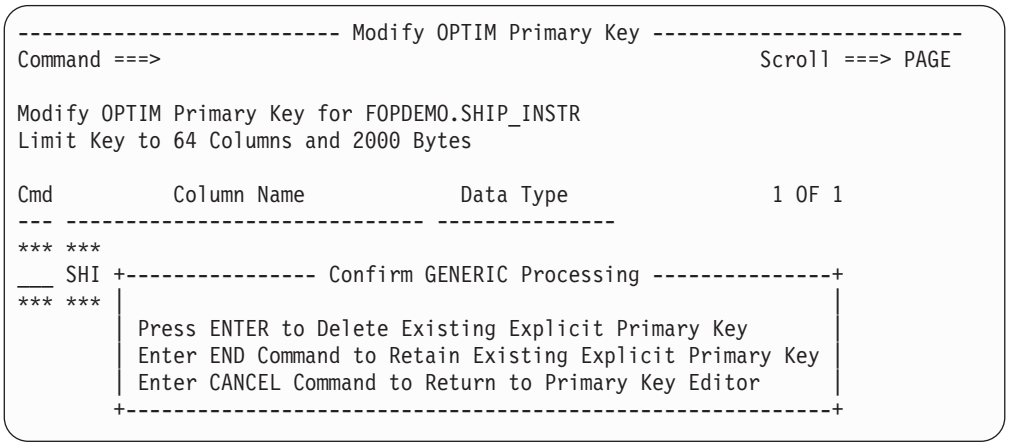



*Figure 69. Confirm GENERIC Processing Prompt*

Once converted, you can edit the generic primary key as you would an explicit primary key, using techniques described in this section.

## Edit a Generic Primary Key

For various site-specific reasons, a database may include several identical sets of tables, with the same base names, columns, and attribute specifications, but different Creator IDs. This arrangement might be useful in a test environment, for example, to provide a discrete set of tables and data for each software developer or group of developers. As another example, a provider of administration services might maintain a set of tables for each client's data. For your convenience in these and similar instances, you can define a generic primary key that applies to all tables with the same base name, regardless of Creator ID.

There is no difference in function or appearance between generic primary keys and explicit primary keys. Explicit primary keys, however, take precedence over generic primary keys, so if an explicit primary key exists for a table, it is used, instead of any generic primary key that might exist.

Generic primary keys are identified with an asterisk (*) in the Creator ID. You can convert an existing primary key to a generic primary key, as described in “Convert to a Generic Primary Key” on page 121, or you can create a generic primary key using the same methods used to create an explicit primary key.

### Create or Edit a Generic Primary Key

To edit or create a generic primary key, display the Choose a Primary Key panel by selecting Option 6.1 Primary Keys from the **Main Menu**, or Option 6 DEFINITIONS to display the **Choose a Definition Option** menu and then Option 1 PRIMARY KEYS. Type an asterisk in **Creator ID** and the desired table name or use DB2 LIKE syntax in **Table Name**.

Press ENTER to display the Define Optim Primary Key panel, overlaid with a pop-up, prompting for a Creator ID, as shown in the following figure.

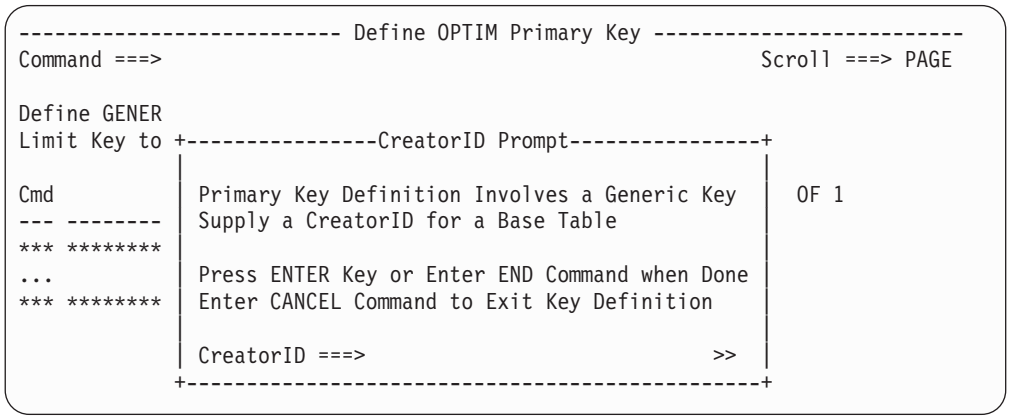



*Figure 70. Specify Creator ID for Base Table*

Before you can choose the columns for the generic primary key, you must supply a Creator ID for the base table, which is used to validate the generic primary key and provide column information. Specify a valid Creator ID of 1 to 128 characters.

You may then identify the columns for the generic primary key, as described for explicit primary keys. (See "Edit or Browse a Primary Key" on page 119 for additional information.) After the generic primary key is defined, any table that is named SHIP_INSTR and contains the key columns can use the generic primary key.

## Modify a Generic Primary Key

If the generic primary key specified on the Choose a Primary Key panel exists or you select a generic primary key from the Select Primary Keys panel, the Modify Optim Primary Key panel is displayed. To modify a generic primary key, you must provide a Creator ID to identify the base table. You are not required to provide the same Creator ID that was used to create the generic primary key, but all columns in the key must be included in the new base table. If you specify a different Creator ID, and the new base table does not include all the columns, the Confirm New Base CreatorID prompt is displayed.

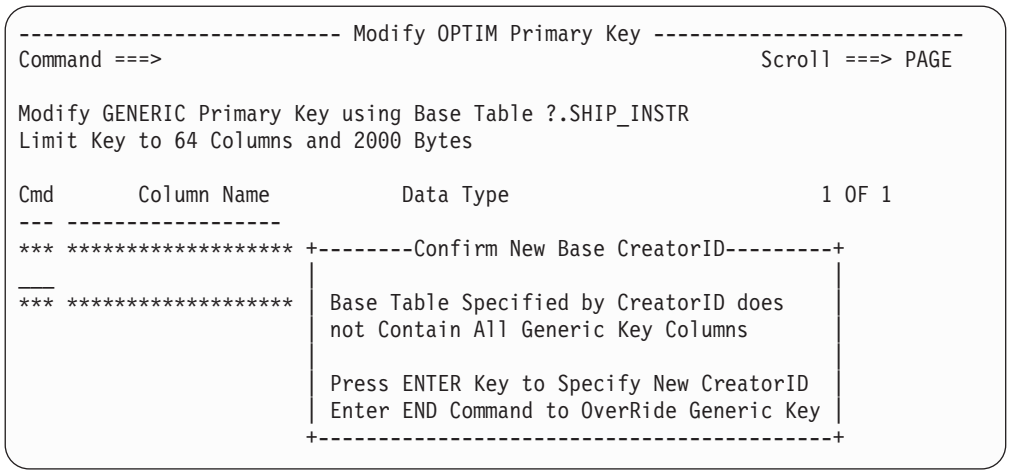



*Figure 71. Confirm New Base Creator ID*

You can change your specification or override the existing generic primary key. If you choose to override, the existing generic primary key is deleted and you can redefine the generic primary key. After establishing a base table, you can use the facilities described for modifying a primary key.

# Chapter 4. Relationships

A relationship is a defined connection between the rows of two tables. To support a relationship, DB2 requires a parent table primary key of 1 to 64 columns that guarantees uniqueness. DB2 also requires that each child table have a corresponding foreign key made up of the same number of columns. In addition, corresponding primary and foreign key columns must have identical data types and attributes.

### Extended Relationships

While you can use Optim to define relationships that conform to DB2 requirements, you can also define more flexible, "extended" relationships (referred to as Optim relationships) for which a number of the DB2 restrictions are relaxed. For these relationships:

- Primary keys and foreign keys are not required.
- Corresponding columns from the parent and child table need not be identical, but must be compatible. Compatible column types are described in Appendix D, "Compatibility Rules," on page 429.
- You can use an expression, which can include string/graphic/binary literals, numeric constants, concatenated column values, and substrings, to emulate application-managed relationships and allow you to manipulate sets of relational data in the same manner as in your production environment.
- You can also use a relationship exit to define the way in which columns with internally incompatible data are related. An example of paired columns for which a relationship exit might be useful is in defining a relationship between a numeric column, for which an Optim relationship adds leading zeroes and a char column with leading blanks.

In addition, an Optim relationship can be stored in the Optim directory as:

- An Explicit relationship, used for a single pair of tables.
- A Generic relationship, used for one or more pairs of tables that have the same base name, column names, and attributes, but different Creator IDs.

There is no difference in function or appearance between generic and explicit relationships. However, explicit relationships take precedence over generic relationships.

**Note:** On ISPF panels and batch process reports, an Optim relationship may be identified as a PST, OPT or FOP relationship. Those terms are used interchangeably when referencing Optim relationships.

### Legacy Tables

You can also create an Optim relationship for a Legacy Table as though the Legacy Table were a DB2 table. Fields in a Legacy record are treated as columns for the purpose of creating a relationship.

You can create an Optim relationship between DB2 tables, between Legacy Tables, or between a DB2 table and a Legacy Table, with either table as the parent and the other as the child.

### Materialized Query Tables

You cannot define an Optim relationship for a Materialized Query Table.

## Select a Relationship

To edit or create a relationship, select Option 6.2 Relationships from the Main Menu.

You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu (see "Choose a Definition Option" on page 11) and select Option 2 RELATIONSHIPS. In either case, the Choose a Relationship panel is displayed.

```
---------------------------- Choose a Relationship -----------------------------
OPTION ===>                                                 SCROLL ===> PAGE

  1  CREATE   - Create a Relationship for Specified Parent or Child Table
  2  MODIFY   - Modify a Relationship for Specified Child Table
  3  LIST     - List All Relationships for Specified Table

Specify Table Name (Child for OPTION 2, Parent or Child for OPTIONS 1 and 3)
  Creator ID ===> FOPDEMO                                                >>
  Table Name ===> CUSTOMERS                                              >>

Specify Relationship Name (OPTIONS 1 and 2)
  Relationship Name ===>                                                   >>

Specify Relationship Type (OPTIONS 2 and 3)
  Relationship Type ===>                     (P|0-OPTIM, D-DB2, I-IMS, A-All)

Use '_' for DB2 LIKE character ===> NO       (Y-Yes, N-No)
```

*Figure 72. Choose a Relationship*

**Note:** You can also display this panel by entering the CREATE RELATIONSHIP command from the Select Tables/Views for AD panel or the Specify Relationship Usage panel while editing an Access Definition.

The Choose a Relationship panel allows you to create, modify, and select relationships. To use this panel, select an option and provide the needed information. You can type the necessary information on the panel or use DB2 LIKE syntax.

## Panel Options

The options on the Choose a Relationship panel include:

**1 - CREATE**
Create an Optim relationship involving a specific table. For this option, you must provide the name of one table in the relationship and may also provide **Relationship Name**. The specified table can be either the parent or child in the relationship.

After you specify a table name or select one from a selection list, the Create a Relationship panel is displayed. That panel is described in "Create a Relationship" on page 129.

**2 - MODIFY**
Modify an Optim Relationship or browse a DB2 relationship. For this option, you must indicate the **Relationship Type** and **Relationship Name**.

After you indicate the Relationship Type and type a Relationship Name or select one from a selection list, the Modify Relationship panel (for an Optim relationship) or the read-only Browse Relationship panel (for a DB2 relationship) is displayed.

**3 - LIST**
Use this option to obtain a selection list of relationships for one or more specified tables. For this option, you must indicate the **Relationship Type** and **Table Name**.

Once you indicate the Relationship Type and supply a Table Name by typing it on the panel or selecting a table from a list, the Select Relationship panel is displayed (refer to "Relationship Selection List" on page 127). Select a table from the list to display the Modify Relationship panel (for an Optim relationship) or the read-only Browse Relationship panel (for a DB2 relationship).

### Panel Prompts

The prompts on the Choose a Relationship panel include:

**Specify Table Name**
Together, Creator ID and Table Name make up the fully qualified name of the database table. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Creator ID**
The Creator ID for the table for which the relationship is being created or modified. The default is the previously entered value or the TSO ID of the current user if a **Creator ID** has never been specified. Specify 1 to 128 characters. To create or edit a generic relationship, type an asterisk at this prompt. (Refer to "Edit a Generic Relationship" on page 152 for details).

**Table Name**
The name of the table for which the relationship is defined or modified. Specify 1 to 128 characters.

**Note:** See "Naming Conventions" on page 6 for information on entering Creator IDs, Table Names, and Relationship Names of up to 128 characters.

**Relationship Name**
The unqualified name of the relationship. Specify 1 to 128 characters. For Option 2, specify the **Relationship Type** and use DB2 LIKE syntax or leave **Relationship Name** blank to obtain a selection list.

**Relationship Type**
Indicate the type of relationship to include in the selection list. This prompt has no effect for Option 1 if you indicate an explicit **Creator ID**, **Table Name**, and **Relationship Name**. Specify:

**P** *or* **O** Include relationships defined to the Optim Directory.

**D** Include relationships defined to the DB2 Catalog.

**I** Include IMS relationships (*Move* or *Compare for IMS, VSAM, and Sequential Files* only).

**A** Include all types of relationships.

**Use '_' for DB2 LIKE character**
Indicate whether the underscore "_" is used as a DB2 LIKE character or literally as part of the name. The default is NO, which means "_" is not handled as a DB2 LIKE character.

**Note:** When the '_' character is used in conjunction with the '%' DB2 LIKE character, the '_' is treated as a 'DB2 like' character, and not as a literal.

For example, depending on the use of the underscore character, "A_B" can be a three-character name containing the characters "A_B", as entered, or a three-character name that begins with "A" and ends with "B" with any valid character in the middle.

## Relationship Selection List

When you request a selection list for Option 2 or select Option 3 on the Choose a Relationship panel, the Select Relationships panel lists all relationships for a specified table.

For example, specify the table FOPDEMO.CUSTOMERS and select Option 3 LIST on the Choose a Relationship panel to display a selection list similar to the following:

```
---------------------------- Select Relationships -----------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 5
               LR-List Rels for Other Table

    --------- Parent ----------  --------- Child -----------
Cmd Creator        Table         Creator        Table        Relation   Type
--- -------- ------------------  -------- ----------------- ------>> ---------
    ******************************** TOP ***************************************
___ FOPDEMO  CUSTOMERS           FOPDEMO  ORDERS            RCO        DB2
___ FOPDEMO  CUSTOMERS           FOPDEMO  ORDERS            TEST       OPT
___ FOPDEMO  CUSTOMERS           FOPDEMO  SALES             SALES      OPT
___ FOPDEMO  CUSTOMERS           FOPDEMO  SHIP_TO           RCST       OPT
___ FOPDEMO  SALES               FOPDEMO  CUSTOMERS         RSC        DB2
```

*Figure 73. Select Relationships*

The list is formatted so that:

- Each line describes a single relationship, including the Creator ID and table name for both the parent and the child tables, and the name of the relationship.
- **Type** indicates the source of the relationship, as defined in the Optim Directory (OPT), the DB2 Catalog (DB2), or in an IMS DBD (IMS).

  **Note:** The relationship type may be shown as 'OPT', 'FOP', or 'PST' in a batch process report or ISPF panel.

## Description

A User Option setting determines whether the description of each Optim relationship is displayed on this panel. See "User Options" on page 372 for information about the **Selection List Format** option.

## Line Commands

Use the **Cmd** area of the panel to enter line commands. The following line commands are available:

**S** Select a relationship.

- If you select an Optim relationship, the Modify Relationship panel is displayed. This panel is discussed in "Edit or Browse a Relationship" on page 130.
- If you select a DB2 relationship, the Browse Relationship panel is displayed. You cannot modify DB2 and IMS relationships or select an IMS relationship.
- If you select multiple relationships, each is displayed in turn as you use END. After all selected relationships are displayed, the Choose a Relationship panel is redisplayed.

**D** Delete an Optim relationship.

**C** Copy an Optim relationship to create a new one.

**R** Rename an Optim relationship.

**AT** Modify the attributes of an Optim relationship.

**I** Display information about an Optim relationship.

See "Object Selection List Functions" on page 14 for more information about these line commands and available primary commands.

**LR** List relationships for the table that is *not* specified as criteria for the selection list. For example, enter LR for the last relationship listed in Figure 73 to display a list of all relationships in which FOPDEMO.SALES is the parent or child.

### Relationship Attributes

To display the attributes of an Optim relationship, use the I line command next to the name of the primary key. The following figure shows the Relationship Attributes panel.

```
------------------------- Relationship Attributes ----------------------------
COMMAND ===>                                                 SCROLL ===> PAGE

Child Table ID     : FOPDEMO
Child Table Name   : CUSTOMERS
Relationship Name  : RCO

Description        :
Security Status    : PUBLIC
Last Modified By   : FOPABC
Modified On        : 2005-11-02-10.08.17

Parent Table ID    : FOPDEMO
Parent Table Name  : CUSTOMERS
```

*Figure 74. Relationship Attributes*

This is a read-only display. Optim generates and maintains the information on this panel according to your specifications for the relationship.

## Create a Relationship

To create a relationship, you must provide the fully qualified name of a table in the relationship. You can type the fully qualified table name or select a table from a selection list, before selecting Option 1 CREATE. The Create a New Relationship pop-up window is then displayed to prompt you for the required information.

```
--------------------------- Choose a Relationship -----------------------------
OPTION ===>                                                  SCROLL ===> PAGE

  1  CREATE   - Create a Relationship for Specified Parent or Child Table

+-------------------------Create a New Relationship-------------------------+
|                                                                           |
| Specified Table   : FOPDEMO.ORDERS                                        |
| Table Type     ===> C                  (P-PARENT, C-CHILD)                |
|                                                                           |
| Leave blank or include wild cards for Table Selection List                |
|                                                                           |
| Other Table:                                                              |
|   Creator ID ===> FOPDEMO                                              >> |
|   Table Name ===> CUSTOMERS                                            >> |
|                                                                           |
| Relationship Name ===> CUSTORD                                         >> |
+---------------------------------------------------------------------------+
```

*Figure 75. Create New Relationship*

### Panel

The Create a New Relationship pop-up displays the name of the table for which you want to create an Optim relationship. It allows you to indicate the role for this table in the relationship, and prompts for

the name of the second table in the relationship. A slightly different version of this pop-up is displayed during an Access edit or browse session if you issue a JOIN command and no relationship exists between the two tables.

**Specified Table**
The table name provided on the Choose a Relationship panel. This value cannot be modified.

**Table Type**
The role of the **Specified Table** in the relationship. This value is profiled. Specify:

**P** Parent

**C** Child

**Other Table:**
The name of the second table in the relationship. **Creator ID** and **Table Name** make up the fully qualified name of the database table. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Creator ID**
The Creator ID for the second table for which the relationship is being created. Specify 1 to 128 characters.

**Table Name**
The name of the second table for which the relationship is being created. Specify 1 to 128 characters.

**Relationship Name**
The name of the relationship. If you provided a name on the Choose a Relationship panel, it is displayed and can be edited. Specify 1 to 128 characters.

Press ENTER to display the Define Relationship panel. Refer to “Edit or Browse a Relationship” for details.

## Edit or Browse a Relationship

When you enter the required information on the Create a New Relationship panel, the Define Relationship panel is displayed. If using Option 2, the display varies according to the type of relationship.

If the relationship:

- Is a DB2 relationship, the Browse Relationship panel is displayed.
- Is an Optim relationship, the Modify Relationship panel is displayed.

The appearance of these panels is identical, but the Browse Relationship panel is read-only. The operation of the Define Relationship and the Modify Relationship panels (i.e., the Relationship editor) is identical and is described in this section.

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE

                      Define OPTIM Relationship CUSTORD
          Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS

    Exit Name ===>
                                                                      1 OF 1
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ CUST_ID                  CH(5)       CUST_ID                  CH(5)
*** ******************************** BOTTOM **********************************
```

*Figure 76. Define Relationship*

The first portion of the Relationship editor indicates the relationship name and lists commands that might be especially useful when defining or modifying a relationship. The next portion of the panel is divided into two parts. One portion shows the Parent table name and lists the parent table Column Names and Data Types. The portion next to it shows the Child table name and lists the child Column Names and Data Types.

Enter the exit name if using a relationship exit or leave Exit Name blank to create a relationship that does not use an exit. See "Exit Routines for Relationships" on page 132 for information about relationship exits.

At least one parent table column and one child table column must be referenced in a relationship, and at least one relationship entry must relate a column from one table to a column from the other. That is, the **Column Name** for each table must list an actual column name or column substring from that table. Additionally, you cannot relate a literal or constant to a literal or constant.

The combination of all entries in **Column Name** cannot exceed 2000 characters minus *n*, where *n* equals the number of nullable columns in the key. Also, no more than 64 columns from each table can be specified in a relationship. A concatenated entry in **Column Name** cannot include more than 64 columns and, if an entry is 2000 characters or includes 64 columns, no other **Column Name** entries for the table can be made.

## Line Commands

In order to define or edit an Optim Relationship, you can manipulate individual lines or parts of lines in the Relationship editor. The following line commands are available:

**C[n]** Copy one line, a number (*n*) of lines, or a block of lines (using the block form, CC).

**D[n]** Delete one line, a number (*n*) of lines, or a block of lines (using the block form, DD).

The entire line is deleted. To delete a column name, position the cursor at the beginning of the column name and press EOF or overtype the prompt with blank spaces.

**EXP** Column names and values are protected if they exceed the space available in Column Name. Use the EXP line command to display the Expanded Relationship Editor prompt to edit long column names or values.

**I[n]** Insert one blank line or a number (*n*) of blank lines.

**M[n]** Move one line, a number (*n*) of lines, or a block of lines (using the block form, MM).

**R[n]** Repeat one line, a number (*n*) of lines, or a block of lines (using the block form, RR).

**A** The "after" destination for copied or moved lines.

**B** The “before” destination for copied or moved lines.

**O** The “overlay” destination for copied or moved lines.

# Exit Routines for Relationships

You can use a site-defined exit routine to define the way in which two tables are related. Exits are useful for pairing columns with internally incompatible data that cannot be defined in other ways. For example, an exit can define a relationship using a numeric column and a char column by adding BLANKs to the numeric value. The resulting numeric value is equal in length to the value in the char column. As another example, an exit might establish a relationship using a binary column indicating the number of days since 1900 and a date column. A relationship exit can also be used to adjust column values according to specific needs before passing control for Optim relationship processing.

## Using an Exit Routine

A relationship exit is identified by the relationship exit module name. To use an exit, enter the **Exit Name** on the Define OPTIM Relationship panel. The exit name must be a valid MVS module name and, to use the relationship, the module must exist. If the module does not exist or cannot be found, you can save the relationship but the following message is displayed.

```
EXIT NOT FOUND
```

After entering the exit name, list all columns used in the relationship, using only valid column names or the FOP_IMSKEY keyword. You must list at least one column for the parent table and one for the child table. The list can include a different number of columns, of different data types, for each table. The columns must exist in the respective tables and the data types must be supported for Optim relationships. NULL values are supported. Use of literals, substrings, or other functions and expressions is not allowed with an exit.

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE

                      Define OPTIM Relationship REL1
         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.EMPLOYEE              Child: FOPDEMO.ORDERS

    Exit Name ===>EXIT1
                                                                      1 OF 1
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ************************************
___ .....                                .....
___ HIREDATE                 CH(8)       ORDER_DATE              TIMESTAMP
___ .....
*** ******************************** BOTTOM ***********************************
```

*Figure 77. Define Relationship*

Depending upon the direction of processing, the exit in this example might adjust HIRE_DATE values in the IMS table EMPLOYEE to conform to a TIMESTAMP data type so that Optim can match HIRE_DATE with ORDER_DATE in the DB2 table ORDERS.

## Writing a Relationship Exit Routine

All exits must conform to the following requirements:

- The exit can be written in assembly language, VS COBOL II, PL/I, or C.

- The exit is loaded and executed using standard z/OS rules for load module processing. It must be reentrant and must be available to all Optim online and batch processes using the relationship that calls the exit.
- Each exit must be compiled and linked as a separate load module. This load module is loaded dynamically at execution time.
- If the exit accesses DB2, the DBRM for that exit program must be bound into the Optim plan.
- The exit must be valid for both parent-to-child traversals and child-to-parent traversals.

Both a parent and a child table are needed for all relationships. A relationship might be traversed from parent to child or from child to parent, however. The exit is called for each row in the table from which the relationship is traversed (the "from" table), whether it is the child table or the parent table. The exit converts values in the table from which the relationship is traversed to make them compatible with values in the table to which the relationship is traversed (the "to" table). At the time the exit is called is called, the addresses of four structures are passed to the exit. These structures provide:

1. Parameters for the relationship exit
2. Statistics about the "from" table
3. Statistics about the "to" table
4. A work area

## Parameters

The relationship exit control block is used with every call to pass the following parameters to the exit using standard OS linkage conventions.

1. The address of a structure that contains the relationship exit parameters:

*Table 3. Exit parameter structure*

| Name | Offset | Length in Bytes | Data Type | Description |
|---|---|---|---|---|
| RelExitID | +0 | 8 | Character | "OPTRELX" |
| RelNameL | +8 | 2 | Binary | Length of exit name |
| RelName | +10 | 128 + 1 | Character | Name of exit with NULL termination |
| RelExitNameL | +139 | 2 | Binary | Length of exit module name |
| RelexitName | +141 | 8 + 1 | Character | Name of exit module with NULL termination |
| OptRelExitVer | +150 | 2 | Binary | Version of exit, initially 1 |
| OptimRel | +152 | 4 | Character | Release and version of Optim |
| SSN | +156 | 4 | Character | Subsystem name |
| UserIDL | +160 | 2 | Binary | Length of primary user ID |
| UserID | +162 | 8 + 1 | Character | Primary user ID with NULL termination |
| SQLIDL | +171 | 2 | Binary | Length of SQL ID |
| SQLID | +173 | 128 + 1 | Character | SQL ID with NULL termination |

*Table 3. Exit parameter structure (continued)*

| Name | Offset | Length in Bytes | Data Type | Description |
|---|---|---|---|---|
| SCreatorIDL | +302 | 2 | Binary | Length of Creator ID for the table from which the relationship is traversed |
| SCreatorID | +304 | 128 + 1 | Character | Creator ID, with NULL termination, for the table from which the relationship is traversed |
| STableNameL | +433 | 2 | Binary | Length of name for the table from which the relationship is traversed |
| STableName | +435 | 128 + 1 | Character | Name, with NULL termination, for the table from which the relationship is traversed |
| DCreatorIDL | +564 | 2 | Binary | Length of Creator ID for the table to which the relationship is traversed |
| DCreatorID | +566 | 128 + 1 | Character | Creator ID, with NULL termination, for the table to which the relationship is traversed |
| DTableNameL | +695 | 2 | Binary | Length of name for the table to which the relationship is traversed |
| DTableName | +697 | 128 + 1 | Character | Name, with NULL termination, for the table to which the relationship is traversed |
| GlobalWorkL | +826 | 2 | Binary | Length of shared work area |
| pGlobalwork | +828 | 4 | Address | Pointer to shared work area |
| TblWorkL | +832 | 2 | Binary | Length of table work area |
| pTblWork | +834 | 4 | Address | Pointer to table work area |
| ParentChildFlag | +838 | 1 | Character | "P" if DTableName is parent, "C" if DTableName is child |
| IMSKeyL | +839 | 2 | Binary | Length of concatenated IMS key |

*Table 3. Exit parameter structure (continued)*

| Name | Offset | Length in Bytes | Data Type | Description |
|---|---|---|---|---|
| pIMSKey | +841 | 4 | Address | Concatenated key |
| NumRows | +845 | 8 | binary | Number of STableName rows processed (including current row) |
| TotRowsS | +853 | 8 | binary | Total number of STableName rows or -1 to indicate unknown |
| TotRowsD | +861 | 8 | binary | Total number of DTableName rows or -1 to indicate unknown |
| MessageLen | +869 | 2 | Binary | Length of message returned by exit |
| Message | +871 | 256 | Character | Message returned by exit |
| Flag1 | +1127 | 1 | Character | OR representation of:<br>**00** No flags<br>**01** First call to exit<br>**02** Last call to exit for last table<br>**04** Processing last table<br>**08** Termination call to exit |
| Reserved | +1128 | 1 | Character | |

2. The address of an SQLDA for the table from which the relationship is traversed. The exit can determine which is the "from" table in many ways, including the ParentChildFlag setting, the table name, or the column name. The SQLDA for the table identifies the columns that participate in the relationship. An SQL variable for each participating column describes the data from the current row and column.

*Table 4. SQLDA for each table in a relationship*

| Name | Offset | Length in Bytes | Data Type | Description |
|---|---|---|---|---|
| ID | +0 | 8 | Character | Identifier |
| bc | +8 | 4 | Binary | Number of bytes |
| TotCols | +12 | 2 | Binary | Total number of columns or var occurrences |
| NumCols | +14 | 2 | Binary | Number of columns used in relationship |
| Col | +16 | 44 | SQLVAR | SQLDA for column |

Example of SQLDA for a DB2 table:

```
/*-------------------------------------------------------------------
 *       DB2 for z/OS SQLDA structure
 *-------------------------------------------------------------------*/
typedef struct sSQLDA {
  char  ID[8];                 // SQLDA identifier
  long  bc;                    // number of bytes in the table
  short TotCols;               // total number of columns or var occurrences
  short NumCols;               // number of columns used in relationship
  SQLVAR Col[1];               // SQL variable for the first column in relationship
  SQLVAR Col[2];               // SQL variable for the next column
  . . . .
  SQLVAR Col[n];               // SQL variable for the next column

} SQLDA;
```

For each column that participates in the relationship, the SQL variable structure is:

*Table 5. SQLVAR for each column in a relationship*

| Name | Offset | Length in Bytes | Data Type | Description |
|---|---|---|---|---|
| DataType | +0 | 2 | Binary | SQLTYPE for column, as a value defined in DB2 SQL Reference and reflecting the NULL eligibility for the column |
| Length | +2 | 2 | Binary | Length of column value in bytes |
| pData | +4 | 4 | Address | Pointer to column data |
| pInd | +8 | 4 | Address | NULL indicator variable |
| NameLen | +12 | 2 | Binary | Number of characters in column name |
| Name | +14 | 30 | Character | Name of column |

Example of an SQLDA variable for a DB2 column:

```
typedef struct sSQLVar {
  short DataType;                    // Timestamp, NULL ineligible
  short Length;                      // length
  char  *pData;                      // pointer to column data
  short *pInd;                       // pointer to null ind var
  short NameLen;                     // length of column name
  char  Name[ORDER_DATE];            // column name or label
} SQLVAR;
```

3. The address of the SQLDA for the table to which the relationship is traversed. The exit populate the SQLDA with values used to select rows from the "to" table.
4. The address of the relational exit work area.

## Return Codes

The exit sets one of the following return codes before passing control back to the Optim solution:

**Return Code**
Meaning

**0** Processing is successful.

**4** Row is skipped or does not participate in the relationship.

**8** Fatal error. Terminate execution. To return an error message, the exit must place the message in

the message area and set the message length in the parameter area. The error message is printed with other Optim diagnostic messages issued for an abnormal process termination. Note that the exit can place diagnostic messages and set the appropriate message length in the exit message area at any time and the messages are included in diagnostic output when the Optim TRACE facility is activated.

## Information Returned from the Exit

Most of the parameters provide information that is passed to the exit and the exit should not change them. In addition to return codes, the exit can return:

- Messages and message lengths
- SQLVAR for each column in the table to which the relationship is traversed
- pData (address of column data)
- pInd (address of NULL indicator variable)

## Performance

Because the exit might be called frequently, avoid unnecessary processing. For example, the exit can perform initialization processing in the first call and save information for subsequent calls in the work area.

## Termination Call

A termination call is made to the exit after all rows in DTableName are processed. The call is identified by a value of 08 in Flag1. The exit can use this call to clean up any dynamically acquired storage. Note that a termination call can occur without a preceding first call as when a path is not followed for a relationship that potentially participates in processing.

## Samples

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE



         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.TABLE1                  Child: FOPDEMO.TABLE2

    Exit Name ===>EXIT1
                                                                        1 OF 1
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ******************************** TOP ***********************************
___ CHARCOLUMN                 CH()      INTEGERCOLUMN             SMALLINT
*** ******************************* BOTTOM **********************************
```

*Figure 78. Sample 1: Define relationship using DB2 CHAR and DB2 SMALLINT columns*

A C relationship exit module uses the following sample logic to establish a relationship between TABLE1 and TABLE2 that works in either direction:

```
for (i = 0; i < pDstSQLDA->NumCols; i++) { /* only 1 column so i==0  */
if(pExitParms->ParentChildFlag == 'C') { /* Parent to Child conversion */
}
if(pExitParms->ParentChildFlag == 'P') { /* Child to Parent */
```

```
  sprintf(pDstSQLDA->Col[i].pData, "%5d",*(int*)pSrcSQLDA->Col[i].pData);
  pDstSQLDA->Col[i].Length = 5;
} }
return(RC_EXIT_SUCCESS);
```

```
------------------------------ Modify Relationship -------------------------
Command ===>                                                 Scroll ===> CSR

         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.EMPLOYEE_IMS         Child: FOPDEMO.ORDERS

    Exit Name ===> EXIT2
                                                                    1 OF 2
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ HIREDATE                 CH(8)       ORDER_DATE               DATE
*** ******************************** BOTTOM*********************************
```

*Figure 79. Sample 2: Define relationship using IMS date field and DB2 DATE column*

A C relationship exit module uses the following sample logic to establish a relationship between EMPLOYEE_IMS and ORDERS that works in either direction:

```
if(pExitParms->ParentChildFlag == 'C')
*(short*)pDstSQLDA->Col[0].pData = (short)atoi(pSrcSQLDA->Col[0].pData);
if(pExitParms->ParentChildFlag == 'P') {
strncpy(pDstSQLDA->Col[i].pData, pSrcSQLDA->Col[i].pData + 5, 2);  //Month
strncpy(pDstSQLDA->Col[i].pData + 3, pSrcSQLDA->Col[i].pData + 8, 2); //Day
strncpy(pDstSQLDA->Col[i].pData + 6, pSrcSQLDA->Col[i].pData + 2, 2); //2 digit year
pDstSQLDA->Col[i].pData[2] = '/';
pDstSQLDA->Col[i].pData[5] = '/';
}
```

## Sample Exits in C and COBOL

Sample exits are included with Optim. The samples illustrate how to use the data areas available to the exit and the type of processing that can be accomplished using an exit. The source for sample exits is distributed with Optim in the FOPSAMP library. FOPSREL2 is a sample C exit that demonstrates basic data conversion between various DB2 data types, for example, from CHARACTER to DECIMAL PACKED and to CHARACTER from DECIMAL PACKED. FOPSRXS1 is a COBOL sample that demonstrates conversion between character and numeric data.

## IMS Concatenated Key

If data from an IMS concatenated key is needed to create a relationship between an IMS table and another table, use the special FOP_IMSKEY keyword. When using the FOP_IMSKEY keyword with a relationship exit, the parent table must be an IMS start table and FOP_IMSKEY must be the only keyword under the column heading. The IMS concatenated key and length will be passed to the exit in the first parameter, the Exit parameter structure. The IMS concatenated key information can then be used to fill in the child destination columns.

When extracting an IMS table with the FOP_IMSKEY keyword, the selected rows from the IMS start table are extracted first. A list of unique concatenated keys from these extracted rows is then sorted and used to select rows from the child destination table.

The default is to use the full IMS concatenated key from the root segment to the current segment. If only a subset of levels from the IMS concatenated key is needed to create the relationship, use FOP_IMSKEY parameters to specify the needed levels.

The format of FOP_IMSKEY is:

- FOP_IMSKEY with no parameters or FOP_IMSKEY(*) is the default. This default creates a concatenated key for all levels including the current segment.
- FOP_IMSKEY(first level to use, next level to use, ...) creates a concatenated key for the levels that are specified. The level parameters must be less than or equal to the level of the IMS segment used to create the relationship. For example: FOP_IMSKEY(1,3,5) where IMS segment is >= 5. The root segment is level '1'.

**Example**

An IMS database, SALES has a root segment key (level 1) that is 5 characters long and a customer segment key (level 2) that is 6 characters long. A DB2 table has an 11 character column containing IMS concatenated key information from levels 1 and 2 of the SALES database. To create a relationship between the level 2 customer segment and the DB2 table, use the FOP_IMSKEY as shown.

```
------------------------ Modify Relationship --------------------
Command ===>                                      Scroll ===> CSR

         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: PSIMS.CUSTOMER_IMS           Child: FOPDEMO3.SALES

    Exit Name ===> IMSKEY1
                                                          1 OF 1
Cmd    Column Name    Data Type         Column Name    Data Type
--- ---------------->> ----------  ------------------>> ----------
*** *************************** TOP ***************************
___ FOP_IMSKEY(*)     *IMSKEY*    IMS_CONCATKEY_DATA   CH(11)
*** ************************ BOTTOM****************************
```

*Figure 80. Define relationship using FOP_IMSKEY*

When exit IMSKEY1 is called, the IMS concatenated key is copied to the DB2 column. The following code shows only that portion of the exit. If needed, the key can be split and copied to several columns. See the FOPSAMP library for all code needed to create an exit.

```
/*
    Check key length and abort if key is not 11 chars
  */
  if(pExitParms->IMSKeyL != 11) {
     printf("\n***Error*** IMSKeyL != 11 (%hd)\n", pExitParms->IMSKeyL);
     return(RC_EXIT_ABORT);
  }

  /*
    Move the IMS concatenated key to DB2 destination col
  */
  memcpy(pDstSQLDA->Col[0].pData, pExitParms->pIMSKey, 11);
  return(RC_EXIT_SUCCESS);
```

## Populate Column Names

Most often, a relationship is composed of corresponding columns from the parent and child tables. There are several ways to populate the Define Relationship panel with column names.

You can:

- Use the column names automatically inserted by Optim.
- Insert the column names manually.
- Select column names from a selection list.

## Automatically Inserted Column Names

When you create a new relationship, Optim attempts to populate the Define Relationship panel with as much meaningful relationship information as possible.

### With Primary Key

If the parent table has a primary key, one area of the panel is populated with the names of primary key columns and their associated data types. If the child table includes any columns with names that match the names of the primary key columns and the columns have compatible data types, another area of the panel is populated with those child column names. (Note that an Optim relationship is unlike a DB2 relationship in that it does not require identical attributes in corresponding parent/child columns, but the corresponding columns must be compatible, as defined in Appendix D, "Compatibility Rules," on page 429.)

If the child table has no columns with matching names, Optim checks for a column with identical attributes and, if only one column matches, populates the panel with that column name. If a matching child table column cannot be selected, the **Column Name** on the panel remains blank and the **Data Type** is identified as REQUIRED.

### Without Primary Key

If the parent table does not have a primary key, **Column Name** and **Data Type** are not populated for the parent or child table. You must enter the appropriate column names in the relationship for both tables.

## Manually Insert Column Names

If column names are automatically listed by Optim, you can still manually insert additional column names. Use the Insert line command, I, to insert a blank line under **Column Name**, and then type the appropriate column name or use the LIST COLUMNS command to display a list from which you can select the desired columns. (The **Data Type** for each column is automatically inserted.)

### Move or Copy Lines with Overlay

Use the O (Overlay) line command to copy or move a **Column Name** value to a blank **Column Name** in a target line. Consider the following example in which the M (Move) line command is used with the O line command to move a parent column name to a line with a child column name.

```
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ******************************** TOP ***********************************
M__ CUST_ID                  CH(5)                                REQUIRED
O__                          REQUIRED    CUST_ID                  CH(5)
*** ******************************* BOTTOM **********************************
```

Press ENTER to move the CUST_ID column name and data type to the target line and delete the moved line.

```
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ CUST_ID                  CH(5)       CUST_ID                  CH(5)
*** ******************************** BOTTOM ***********************************
```

In contrast, consider the copy operation, which does not remove the original line. In this example, the C (Copy) line command is used with the O line command to copy a parent column name to a line with a child column name.

```
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
C__ CUST_ID                  CH(5)                                REQUIRED
O__                          REQUIRED    CUST_ID                  CH(5)
*** ******************************** BOTTOM ***********************************
```

The target line is updated and the copied line is retained.

```
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ CUST_ID                  CH(5)                                REQUIRED
___ CUST_ID                  CH(5)       CUST_ID                  CH(5)
*** ******************************** BOTTOM ***********************************
```

If both **Column Names** in the target line have entries, the copy or move operation has no effect.

## Literals and Constants

You can enter string/graphic/binary literals and numeric constants, as well as column names, in place of a child or parent **Column Name**. The **Data Type** for a literal or a constant in **Column Name** reflects the attributes of the literal or numeric constant.

The column corresponding to a literal must have a data type of CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, or VARBINARY. You cannot specify NULL as a literal value. The literal is preceded by an identifier (except for the string literal, which does not require an identifier) and surrounded by delimiters. The following table shows the identifier for each type of literal and an example of how to specify each:

**String Literal**
'SW012'

**String Hexadecimal Literal**
**X**'05B3'

**Graphic Literal**
**G**'间月日消息爵'

**Graphic Hexadecimal Literal**
**GX**'05B305B3'

**Binary Hexadecimal Literal**
**BX**'C141C242'

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE

                     Define OPTIM Relationship CUSTORD
         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS
                                                                      1 OF 2
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ CUST_ID                  CH(5)       CUST_ID                  CH(5)
___ 'SW012'                  CH(5)       ORDER_SALESMAN           CH(6)
*** ******************************** BOTTOM **********************************
```

*Figure 81. String Literal*

The column corresponding to a numeric constant must have a data type of DECIMAL, INTEGER, BIGINT, or SMALLINT. Numeric constants cannot include floating point or hexadecimal values.

## Data-Driven Relationships

A table is sometimes related to one of several child tables according to the values in a particular parent column—in what is referred to as a data-driven relationship. As demonstrated in the preceding example, you can define such a relationship with literals. As another example, consider the following example in which tables are used to determine employee insurance rates for males and females.

The ID, age, and sex for each employee are in the EMPLOYEE table.

*Table 6. EMPLOYEE*

| EMPLOYEE_ID | AGE | GENDER |
|---|---|---|
| 058-44-2244 | 38 | F |
| 106-46-0619 | 40 | M |
| 248-91-2890 | 27 | M |

The FEMALE_RATES table provides age-based insurance rates for female employees. The MALE_RATES table provides the insurance rates for male employees.

*Table 7.* **FEMALE_RATES**

| AGE | RATE/$1000 |
|---|---|
| 38 | .25 |
| 39 | .33 |
| 40 | .43 |

*Table 8.* **MALE_RATES**

| AGE | RATE/$1000 |
|---|---|
| 38 | .31 |
| 39 | .38 |
| 40 | .47 |

The EMPLOYEE table is related to the FEMALE_RATES table when the employee is female, and to the MALE_RATES table when the employee is male. To emulate this structure, you must define two relationships.

The first relationship, between the EMPLOYEE table and the FEMALE_RATES table, is active when the value in the GENDER column corresponds to the string literal 'F'. AGE, in the EMPLOYEE table, corresponds to the AGE column in the FEMALE_RATES table.

**EMPLOYEE**
FEMALE_RATES

**GENDER**
'F'

**AGE** AGE

The second relationship, between the EMPLOYEE table and the MALE_RATES table, is identical to the first relationship, except the GENDER column corresponds to the string literal 'M'.

**EMPLOYEE**
MALE_RATES

**GENDER**
'M'

**AGE** AGE

For any row in the EMPLOYEE table, only one relationship can be satisfied because the EMPLOYEE.GENDER value can only be 'M' or 'F'. Once the appropriate relationship is chosen, the related rows are identified by the values in the AGE columns.

## Select Columns from a List

Selecting column names from a selection list is faster than typing them and prevents errors. Use the LIST COLUMNS command to list columns in the parent or child table.

Use the PARENT or CHILD operands to indicate the appropriate table when requesting a column selection list, or you can type the LIST COLUMNS command without an operand and position the cursor on the desired table name or list of columns and press ENTER to display the column selection list. The selection list includes the data type for each column to assist you in selecting compatible columns.

For example, enter LIST COLUMNS CHILD to redisplay the Relationship editor with a selection list of child table columns.

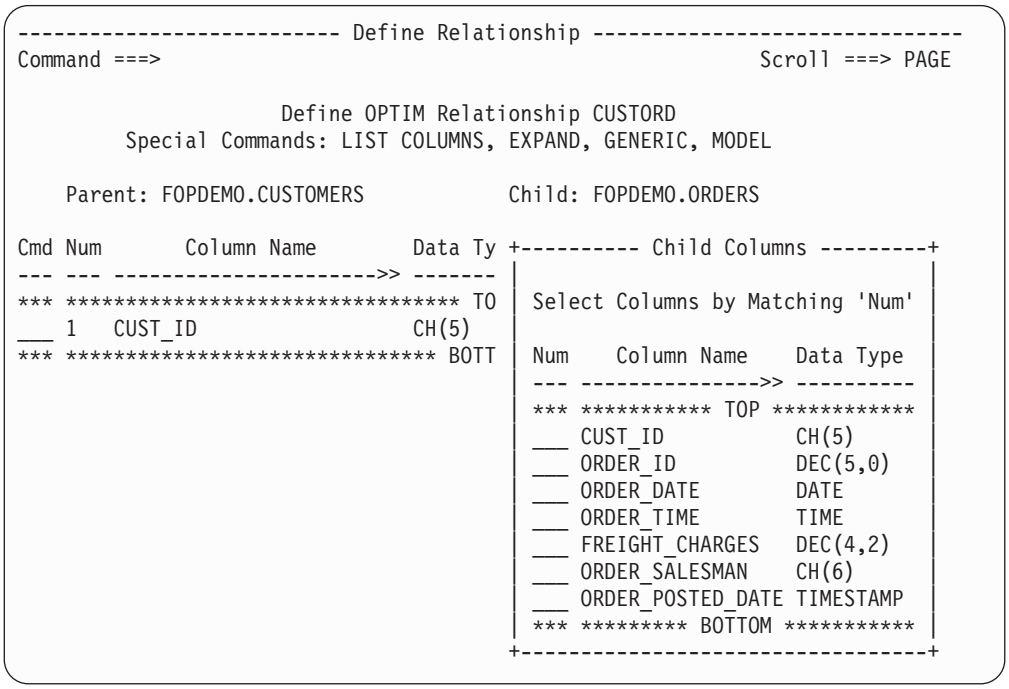



*Figure 82. Child Columns*

A new column, **Num**, is displayed next to **Column Name** for the parent table and the parent column names are numbered, while the selection list overlays the child table area of the Relationship editor. To use the selection list, type the number of the corresponding parent column next to the child column name in the selection list, as shown in Figure 83.

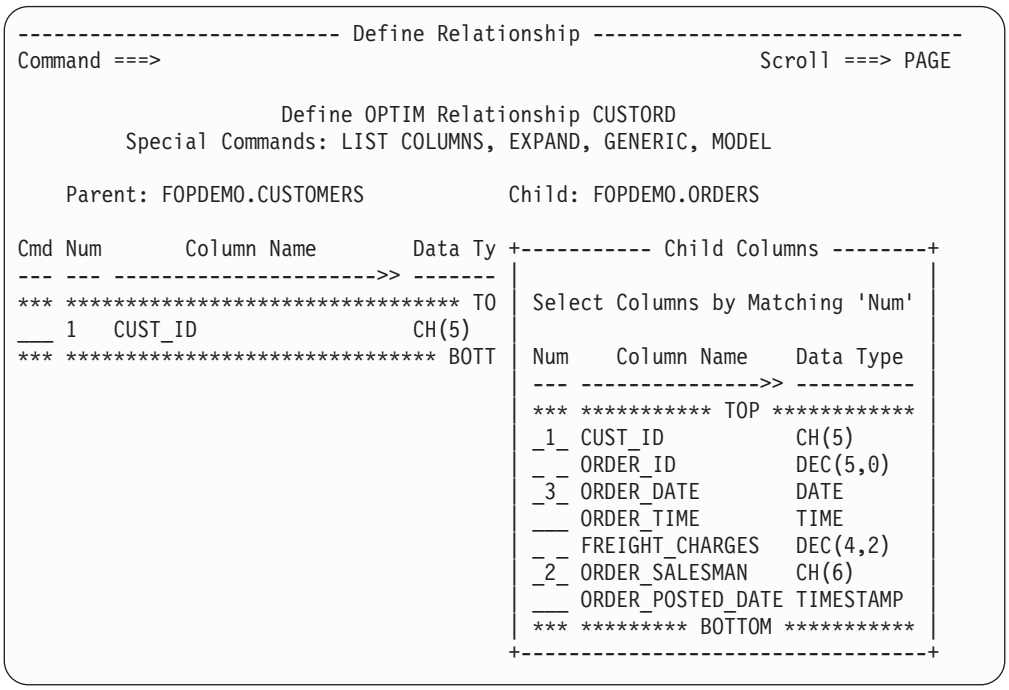



*Figure 83. Select Child Columns*

You can match as many as 64 pairs of columns by selecting from the list. If you want to select a column or columns for which a corresponding column is not listed in the Relationship editor, type the number for the position in order to insert a new entry in the relationship. Additional new entries are inserted if

you specify a series of sequential numbers that begin with a number that is one greater than the number of the last entry, as in the example, in which the user typed 1, 2, and 3.

Use END to add the selected column names to the child table, as shown in the following figure. Notice that the **Num** column is no longer displayed on the panel.

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE

                      Define OPTIM Relationship CUSTORD
         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS
                                                                    1 OF 3
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ CUST_ID                  CH(5)       CUST_ID                  CH(5)
___                          REQUIRED    ORDER_SALESMAN           CH(6)
___                          REQUIRED    ORDER_DATE               DATE
*** ****************************** BOTTOM ********************************
```

*Figure 84. Selected Columns Inserted*

Alternatively, you can use the Select line commands, S or SS, to select columns from the selection list. These names are added, in the order shown on the selection list, at the end of the **Column Name** list.

## Expressions

Powerful flexibility allows you to define an Optim relationship on the basis of columns that have a different data structure—you can combine columns with concatenation or use the substring function to obtain partial column values.

### Concatenate Columns

At times, two or more columns in a parent or child table match a single, composite column in a table related by an application. To define an Optim relationship for this type of data structure, use the concatenation operator (CONCAT or `||`) in **Column Name**. Character and graphic data columns can be concatenated.

For example, if a parent table includes a ZIP1 column for the first 5 digits of the ZIP code and a ZIP2 column for the additional 4 digits, but the child table has only one ZIP column for the 10-character (including hyphen) ZIP code, you can define this relationship by entering one of the following in **Column Name** for the parent table:

`ZIP1 || ZIP2` *or* `ZIP1 CONCAT ZIP2`

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                 Scroll ===> PAGE

                      Define OPTIM Relationship CUSTORD
         Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS
                                                                      1 OF 1
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ ZIP1 || ZIP2             CH(9)       ZIP                       CH(9)
*** ******************************** BOTTOM ***********************************
```

*Figure 85. Concatenated Columns*

When you use a concatenation operator, the **Data Type** reflects the attributes of the expression formed by the concatenation. For example, if the data type for ZIP1 is CHAR(5) and ZIP2 is CHAR(4), the data type of the concatenated expression is CHAR(9).

If you need more space to enter a concatenated expression, use the EXPAND command or EXP line command to display an expanded window to enter the appropriate column names (see page "EXPAND Column Name Values" on page 147 for details). This method is particularly useful for specifying column names that are Long Object Names (LONs). Column Name LONs can be a maximum of 30 characters.

Note the following when defining a concatenated expression:

- CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, and VARBINARY columns, string literals, and substring functions can be concatenated.
- Special Registers are not supported.
- A zero-length string literal, usually indicated as ' ', is not valid in a concatenated expression.

## Substring Columns

When one column in a table contains a value that is in two or more columns in another table, you may find it useful to use a portion of the composite column to define a relationship. Use the SUBSTR function to select a substring of the value in a character or graphic column.

**Note:** You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

`SUBSTR ( column-name, start [, length] )`

**column-name**
The name of a non-numeric column with a character or graphic data type.

**start** The starting position of the substring as an integer greater than zero. The *start* value is required.

**length** The optional length of the substring as an integer greater than zero. If *length* is omitted, the substring consists of the value beginning at the *start* position and ending with the last character. The value of *start* plus *length* cannot exceed the data length for the column.

**Note:** The SUBSTR function is identical to the DB2 SUBSTR function except when handling a combination of character and graphic data. Since the SUBSTR parameters are character-based and the DB2 SUBSTR parameters are byte-based, when you use the functions with mixed character and graphic data, Optim and DB2 will produce different results.

In the earlier example, the parent table includes a ZIP1 column for the first 5 digits of the ZIP Code and a ZIP2 column for the additional 4 digits, while the child table has only one ZIP column for the

9-character (excluding hyphen) ZIP code. As an alternative to concatenating ZIP1 and ZIP2, you can use the substring function to obtain values from ZIP by entering the following in **Column Name** for the child table:

```
SUBSTR(ZIP,1,5) and
SUBSTR(ZIP,6)
```

When you specify a substring function, the **Data Type** reflects the attributes of the substring.

```
--------------------------- Define Relationship -------------------------------
Command ===>                                                  Scroll ===> PAGE

                      Define OPTIM Relationship CUSTORD
          Special Commands: LIST COLUMNS, EXPAND, GENERIC, MODEL

    Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS
                                                                    1 OF 2
Cmd       Column Name        Data Type         Column Name        Data Type
--- ---------------------->> ----------  ---------------------->> ----------
*** ********************************* TOP ***********************************
___ ZIP1                     CH(5)       SUBSTR(ZIP,1,5)          CH(5)
___ ZIP2                     CH(4)       SUBSTR(ZIP,6)            CH(4)
*** ******************************** BOTTOM **********************************
```

*Figure 86. Substring of Column*

If you need more space to enter a substring function, use the EXPAND command or EXP line command to display the specified column in an expand window (see "EXPAND Column Name Values" for details).

## Concatenate or Substring—Performance

Although these two variations are functionally equivalent and retrieve the same set of related rows, the concatenation is more efficient than the series of substrings. In general, concatenation results in better performance because it directs DB2 to use an index if one is available.

The following SQL is generated for the first relationship:

```
SELECT * FROM TABLE2
  WHERE TABLE2.ZIP = 'composite value from both
      parent columns'
```

The following SQL is generated for the second relationship:

```
SELECT * FROM TABLE2
  WHERE SUBSTR(TABLE2.ZIP, 1, 5) = 'value from
      parent ZIP1 column'
  AND   SUBSTR(TABLE2.ZIP, 6, 4) = 'value from
       parent ZIP2 column'
```

The second predicate in the second example references the low-order portion of a column; thus, DB2 does not use an index on the ZIP column, even if one exists, and must scan all rows in the table.

## EXPAND Column Name Values

The space available on the Define Relationship panel may be insufficient to enter certain values in **Column Name**, such as Column Name LONs and expressions using concatenation and substrings. If an entry exceeds the space available on the panel, the display is truncated and you cannot edit it directly. In such instances, type the EXPAND primary command, then position the cursor on the line for the desired parent/child column pair, and press ENTER to display the Expanded Relationship Editor. Alternatively, you can use the EXP line command.

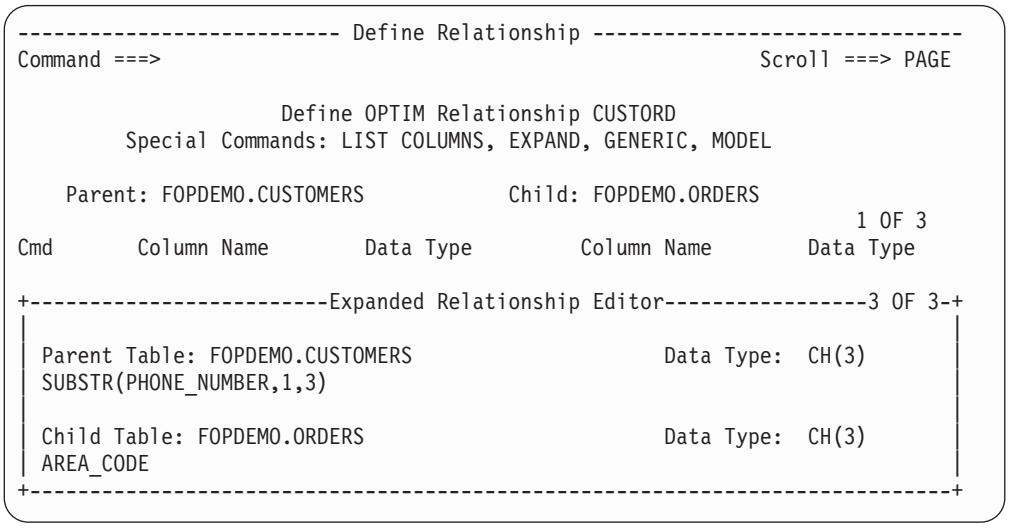



*Figure 87. Expanded Relationship Editor*

The Expanded Relationship Editor provides a 75-character area for both parent and child **Column Name** values. While it is displayed, you can scroll to the previous or next pair of columns using the UP, DOWN, TOP, or BOTTOM commands.

You can enter column names, string literals, numeric constants, substring functions, or concatenated values. Also, the LIST COLUMNS command (described on page "Select Columns from a List" on page 143) is available while the **Expanded Relationship Editor** is displayed. Only one column name, however, may be selected using the S line command. The selected name is added to the end of the entry, preceded by the concatenation operator, ||. You can remove the concatenation operator, if desired.

Use CANCEL to abandon any changes to the current pair of columns and return to the Relationship editor. Any changes made prior to scrolling to the current pair of columns are retained.

Use END to return to the Relationship editor.

## Model a Relationship

You can use a DB2 or Optim relationship as a model for a new, similar relationship. On the **Define Relationship**, **Modify Relationship**, or Browse Relationship panel, enter the MODEL command to display the **Model New Relationship** pop-up window.

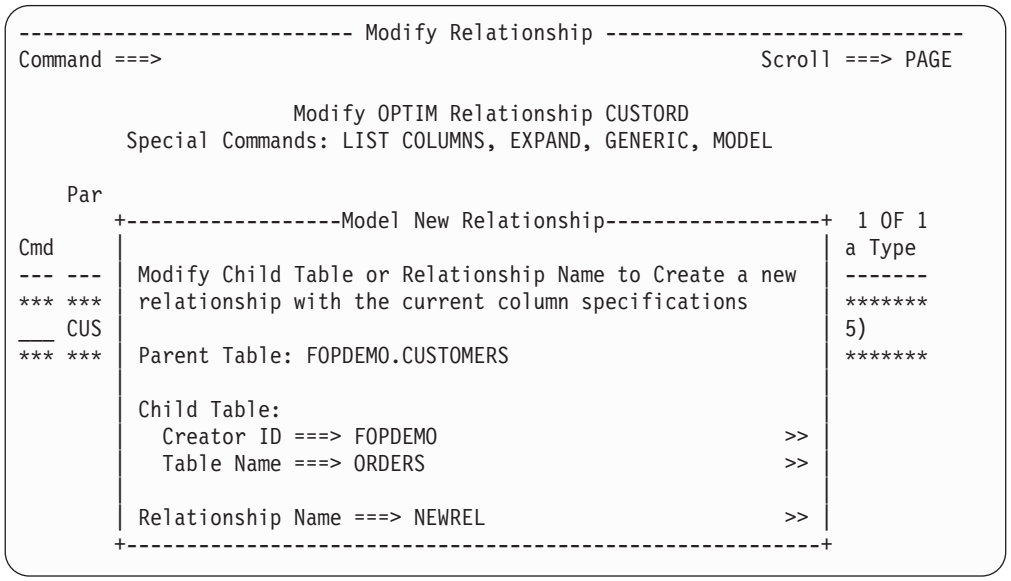



*Figure 88. Model New Relationship*

The displayed information reflects the values for the current relationship. The name of the **Parent Table** cannot be modified, but you can change the child table **Creator ID** and **Table Name**, and the **Relationship Name**.

## If Relationship Exists

The full name of a relationship is the fully qualified child table name with the relationship name. If your entries result in the name of a relationship already in the Directory, the following prompt is displayed. (If you duplicate the name of a DB2 relationship, an error occurs.)

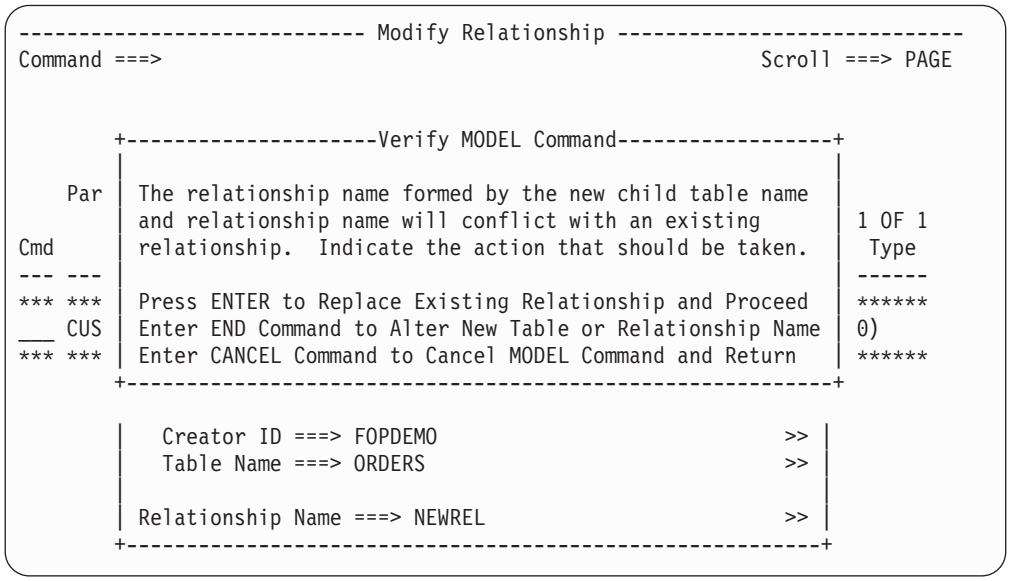



*Figure 89. Verify MODEL Command*

Press ENTER to display the Define Relationship panel for the new relationship. Use END to return to the Model New Relationship prompt to alter the table name or the relationship name. Use CANCEL to cancel the MODEL command and return to the previously displayed panel.

## Save and Use a Relationship

Only a valid relationship can be saved. Use the SAVE command to save and continue editing a relationship. Use END (usually assigned to PF3) to save the relationship and return to the Choose a Relationship panel. Use CANCEL to abandon changes and return to the Choose a Relationship panel.

Optim verifies the compatibility of corresponding columns each time you press ENTER on the Relationship editor. If the columns are not compatible, an error message is displayed to identify the nature of the problem, and the cursor is positioned on the erroneous entry.

To save the relationship, you must correct any incompatibility. Information about column compatibility is provided in Appendix D, "Compatibility Rules," on page 429.

### Effect of Database Changes

A relationship is not affected if new columns are added to a table or if columns outside the relationship are dropped or altered. If a table referenced in a relationship is dropped, however, the relationship is invalid and is no longer available to satisfy a JOIN request or to be used in an Extract or Archive Process. Moreover, the invalid relationship cannot be updated, but it can be deleted. Use the D line command on the Select Relationships panel to delete the relationship.

If a column is dropped from a table or is altered so that it is incompatible with the corresponding column in the relationship, the relationship is invalid and is not available to satisfy a JOIN request or to be used in an Extract or Archive Process. If you attempt to update such a relationship, it is displayed in the Relationship editor with an error message indicating that it includes invalid entries. You may then correct the column specifications.

### Primary Commands

The following primary commands are available on the Define Relationship and Modify Relationship panels. Detailed information about each command is provided in the *Command Reference Manual*, Primary Commands.

ATTRIBUTES
BOTTOM
CANCEL
CAPS
CLEAR
DELETE
DOWN
END
EXPAND
GENERIC
LIST COLUMNS
MODEL
RESET
SAVE
TOP
UP

## Description

You can edit the description that appears in selection lists for an Optim relationship. To do that, use the ATTRIBUTES command to display the Object Attributes panel, and then type an up to 40-character **Description** at the prompt. For additional information about the Object Attributes panel, see "Specify Description and Security Status" on page 380.

## CLEAR

The CLEAR command removes all column specifications from the Define Relationship or Modify Relationship panel. This command is useful when you want to completely redefine the relationship.

## Delete a Relationship

The DELETE command deletes the currently displayed relationship and redisplays the previous panel. You can also delete the displayed relationship with the END command after you delete or blank-out all column entries.

## Convert to a Generic Relationship

To convert an explicit relationship to a generic relationship, use the GENERIC command. Enter the GENERIC command on the Modify Relationship, the Define Relationship, or the Browse Relationship panel to convert the displayed explicit relationship into a generic relationship. The GENERIC command converts only relationships in which both tables have the same Creator ID. If the tables do not have the same Creator ID, an error message is displayed.

If a generic relationship already exists, a prompt requests confirmation to override it. If you are converting a relationship for which a generic relationship does not exist, the following pop-up prompts you to delete or retain the original, explicit relationship.

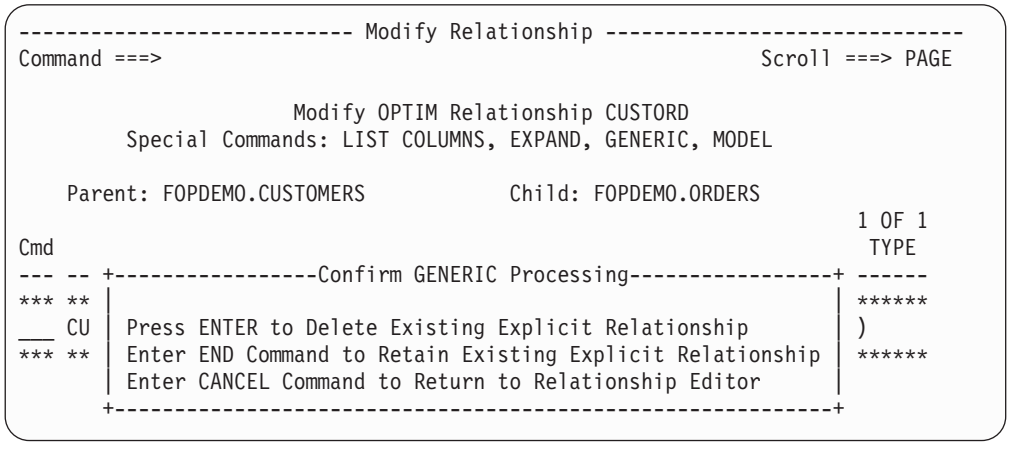



*Figure 90. Confirm GENERIC Processing*

If the name matches that of an existing generic relationship, a confirmation prompt is displayed. You may then decide whether to replace the existing generic relationship.

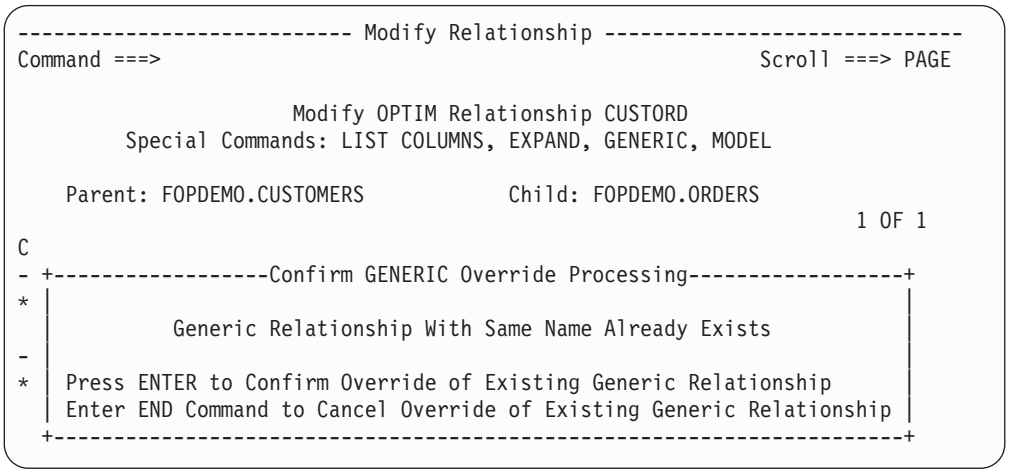



*Figure 91. Generic Relationship Override Confirmation*

Once converted, you can edit the generic relationship as you would an explicit relationship, using techniques described in this section.

## Edit a Generic Relationship

For various site-specific reasons, a database may include several identical sets of tables, with the same base names, columns, and attribute specifications, but different Creator IDs. This arrangement might be useful in a test environment, for example, to provide a discrete set of tables and data for each software developer or group of developers. As another example, a provider of administration services might maintain a set of tables for each client's data. For your convenience in these and similar instances, you can define a generic relationship that applies to all tables with the same base name, regardless of Creator ID.

There is no difference in function or appearance between generic relationships and explicit relationships. Explicit relationships, however, take precedence over generic relationships, so if an explicit relationship exists for a specific pair of tables, it is used, instead of any generic relationship that might exist.

Generic relationships are identified with an asterisk (*) in the Creator ID. You can convert an existing relationship to a generic relationship, or you can create a generic relationship using the same methods used to create an explicit relationship, as described on page “Convert to a Generic Relationship” on page 151. The tables used to define or modify a generic relationship must have the same Creator ID. When a generic relationship is used to join tables or extract or archive data, however, the tables are not required to have the same Creator ID.

### Create or Modify a Generic Relationship

To edit or create a generic relationship, select Option 6.2 Relationships from the Main Menu, or select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu and then Option 2 RELATIONSHIPS to display the Choose a Relationship panel. Type an asterisk in **Creator ID** and the desired table name or use DB2 LIKE syntax in **Table Name**. After choosing the appropriate option (that is, 1 to Create or 2 to Modify), press ENTER to display a pop-up, prompting for Creator ID, as shown in the following figure.

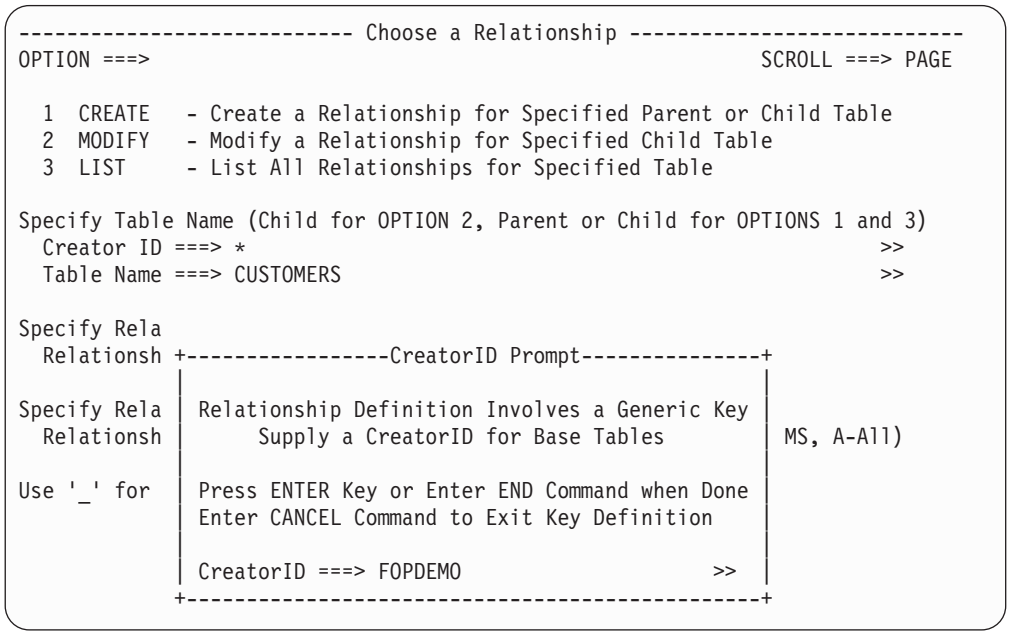



*Figure 92. Specify Creator ID for Base Tables*

Type a Creator ID and press ENTER to display the Create a New Relationship pop-up.

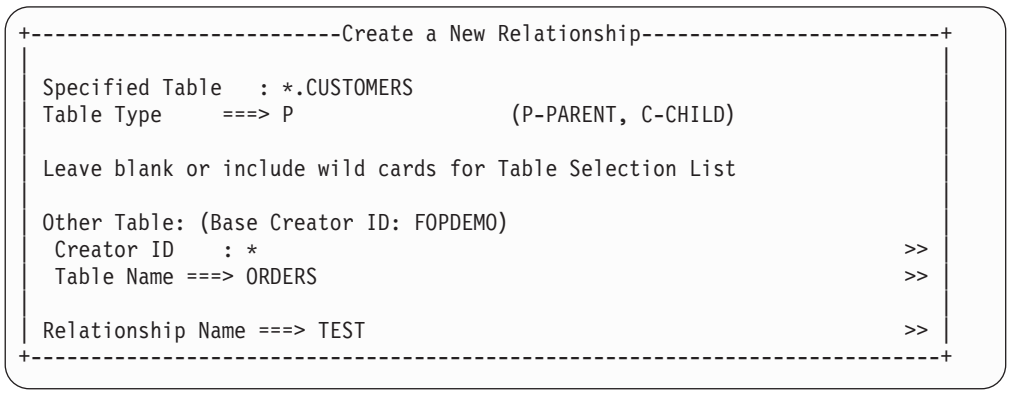



*Figure 93. Create New Generic Relationship*

The table name entered on the Choose a Relationship panel is displayed as the **Specified Table**, and an asterisk is in **Other Table: Creator ID**. These values cannot be modified. Provide the **Table Name** for the second table. When the panel is displayed for the first time, the **Table Type** for the **Specified Table** is shown as C for child, with the **Other Table** as the parent. If you want the **Specified Table** to be the parent table and the **Other Table** to be the child, you can change the **Table Type** to P. The value you specify for **Table Type** is profiled.

Press ENTER to display the Define Relationship panel (refer to "Edit or Browse a Relationship" on page 130).

## Modify a Generic Relationship

If modifying a generic relationship, you are prompted for a Creator ID to identify the base tables. You are not required to specify the same Creator ID that was used to create the generic relationship, however, all columns referenced in the relationship must be included in the tables. If the columns are not contained in the tables indicated by the specified Creator ID, you are prompted to specify a different Creator ID or

override the existing generic relationship. If you elect to override, the existing generic relationship is deleted and you can redefine the generic relationship.

For example, when you select Option 2 MODIFY for a generic relationship named *.ORDERS.CUSTORD, with the CUSTOMERS table as the parent and the ORDERS table as the child, the CreatorID prompt is displayed.

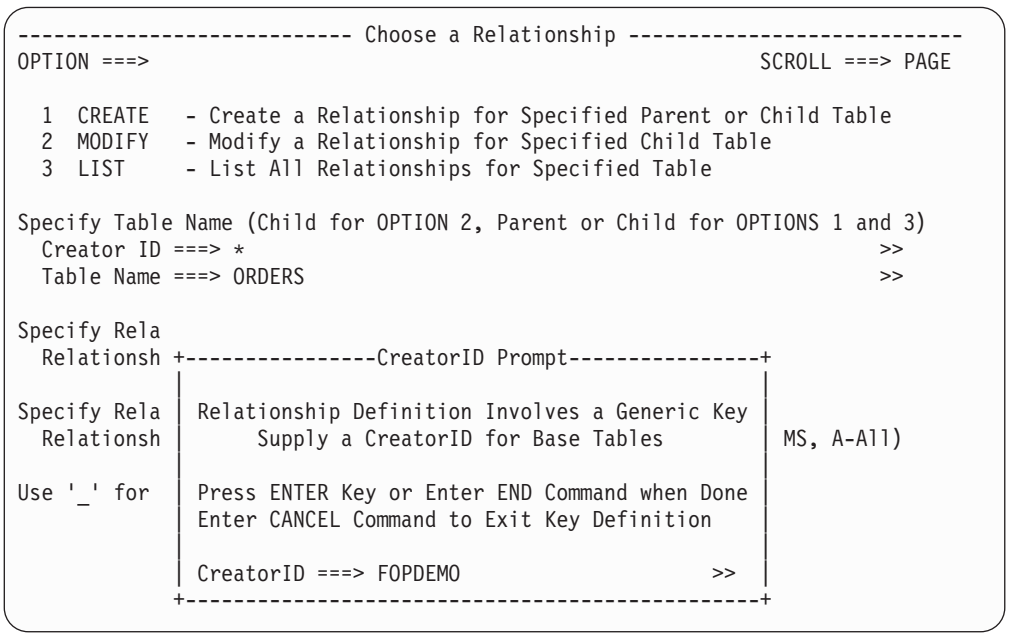



*Figure 94. Prompt for Creator ID*

Once a Creator ID is specified, the Modify Relationship panel is displayed, as shown in the following figure.

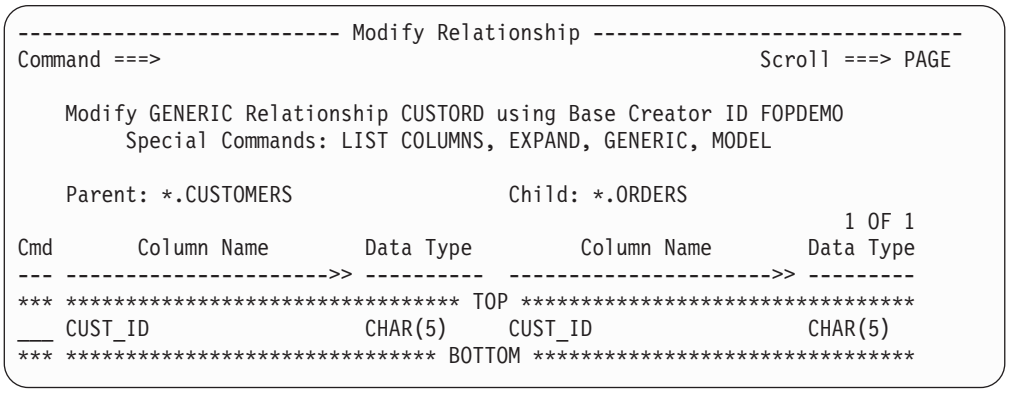



*Figure 95. Modify Generic Relationship*

You can modify a generic relationship using the same functions used to modify an explicit relationship.

# Chapter 5. Column Maps

A Column Map provides specifications needed to direct data from source columns to destination columns, match pairs of columns from separate tables for a Compare Process, or exclude columns from processing. You can use a Column Map for a pair of tables regardless of Creator ID or table name.

For example, a Column Map defined to map columns in a table named FOPDEMO.CUSTOMERS to those in a table named TEST.CUSTOMERS can also be used to map the identical tables PROD.CUSTOMER1 and QA.CUSTOMERS.

A Convert, Insert, Load, Restore, or multi-table Compare Process requires a Table Map, which may reference one or more Column Maps. A Compare Process for a single table comparison may reference a Column Map directly. Column Maps stored in the Optim Directory are available for reuse or sharing with other users. A local Column Map is stored as part of a Table Map or Compare Request and is otherwise not available.

A Column Map must be used when column names or attributes do not match, when data transformations are needed, or when one or more columns are excluded from processing. A Column Map used in a Convert, Insert, Load, or Restore Process can modify data. In a Convert, Insert or Load Process a Column Map can age dates. Advanced methods allow you to split data from one table into several tables or copy one source column to several destinations. Column Maps also allow you to create a single destination table from the data in a joined view. Be aware of this restriction: A table can be referenced only once as the destination; thus, you cannot enter the names of both a table and one or more synonyms, views, or aliases for the table as a destination table. Also, a Column Map cannot transform data during a Compare Process.

For data transformations that are beyond the capacity of a Column Map, you can specify an exit routine, column map expression, or local column map procedure. (Exit routines, programs written in Assembler, VS COBOL II, PL/I, or C programming language, are discussed in “Exit Routines for Source Column Values - Optim v7.2 and earlier” on page 174.) Column map expressions and procedures using Lua functions are discussed in “Working with column map expressions and procedures using Lua functions” on page 184.

Column Maps provide great control and flexibility for Archive, Move, and Compare Processes. For example, you can use a Column Map to bypass selected source columns for one process and a different Column Map to incorporate those columns in a different process.

Compare Column Maps can be used with Archive and Move Processes; however, Column Maps that modify data cannot be used with Compare.

## Move and Archive Column Maps

Typically, in Archive and Move processing, source values are derived from the source columns. A Column Map, however, allows you to modify data and specify a variety of values for the destination. You can:

- Mask sensitive data.
- Insert literals, default values, or DB2 special registers into columns.
- Propagate changes in primary key or foreign key values to all related tables in the process.
- Derive values, using expressions, custom exit routines, or local column map procedures to insert into destination columns.
- Adjust date values in numeric, DATE, and TIMESTAMP columns. (MOVE only)

A Legacy Table, used with *Move* or *Compare for IMS, VSAM, and Sequential Files*, describes legacy data for use with Optim. You can apply a Column Map to columns (fields) in a Legacy Table as though it were a DB2 table.

### Compare Column Maps

For Compare processing, Column Maps allow you to pair Source 1 and Source 2 columns with dissimilar names or compatible data types and exclude specific columns from the Compare Process. Also, you can match dissimilarly named columns from two source tables for a Match Key definition. A Column Map used in a Compare Process cannot modify data.

**Note:** In this chapter, references to the source table also apply to the Compare Source 1 table and references to the destination table apply to the Source 2 table.

## Select a Column Map

You can enter the name of a new Column Map on a Table Map panel or the **Specify COMPARE Sources** panel to display the Column Map editor or select Option 6.3 Column Maps from the **Main Menu**.

You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu (see "Choose a Definition Option" on page 11 ) and select Option 3 COLUMN MAPS. If using Option 6, the Choose a Column Map panel is displayed. Use this panel to name a new Column Map or to select an existing Column Map to modify or delete.

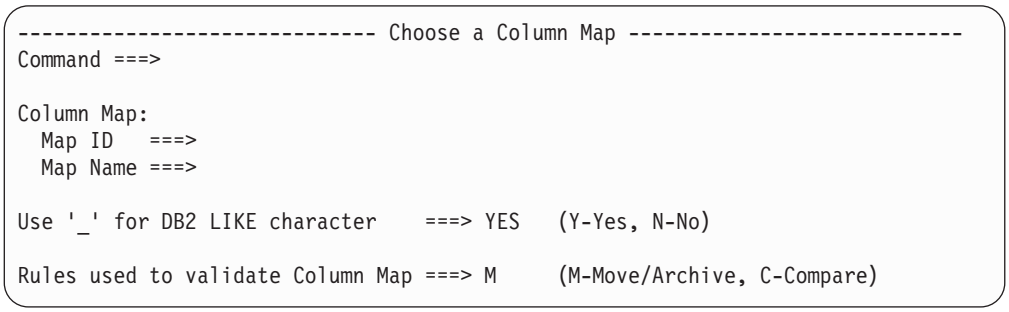



*Figure 96. Choose a Column Map*

### Panel

The Choose a Column Map panel includes:

**Column Map:**
The Map ID and Map Name for the Column Map. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Map ID**
The Map ID for the Column Map that is being created or modified. The default is the previously entered value. Specify 1 to 8 characters.

**Map Name**
The name of the map that is being defined or modified. Specify 1 to 12 characters.

**Note:** You can enter the name of an existing Column Map to use as a model for the Column Map you wish to create. After editing the Column Map, save it under a different name.

**Use '_' for DB2 LIKE character**
Use of the underscore ( _ ) character. Specify Y if the underscore is used as a DB2 LIKE character or N if it is used literally as part of the name.

For example, depending upon the use of the underscore character, A_B is a three-character name containing the characters 'A_B', as entered, or a three-character name that begins with “A” and ends with “B” with any valid character in the middle. The default is N, which means that “_” is not recognized as a DB2 LIKE character.

**Rules used to validate Column Map**
Setting to apply Archive or Move validation rules or Compare validation rules. This prompt is displayed if Compare is installed. Specify:

**M** Comply with rules for Archive or Move.

**C** Comply with rules for Compare.

Archive and Move support mapping columns with compatible attributes, eliminating columns from processing, and modifying data. Compare only allows mapping of columns with compatible attributes and elimination of columns from processing.

## Naming Conventions

Logical naming conventions for Column Maps help you organize the maps and identify their use. For example, all maps created by a specific user could share a common Map ID, with the Map Name reflecting specific projects. Thus, FOPUSER.INVCUST, FOPUSER.ACCTCUST, and FOPUSER.TSTCUST might be names for Column Maps created by the user FOPUSER. Each Column Map might apply to the same destination CUSTOMERS table, with the source mapping varied according to the application to be tested. The Column Maps of another user who tests different processes from the same applications against different data in the same table might be named FOPUSER2.INVCUST, FOPUSER2.ACCTCUST, and FOPUSER2.TSTCUST.

In other circumstances, one set of maps might be used to mask information in certain columns, while another set, available only for use by authorized personnel, might be used for processes that include sensitive data.

Also, logical naming conventions enhance the usefulness of the POPULATE command, which automatically inserts the names of Column Maps into a Table Map panel.

## Explicit Names

When you supply an explicit **Map ID** and **Map Name** and the Column Map does not exist, the Specify Column Map Tables panel is displayed, allowing you to specify the names of tables with columns that will be mapped. If the Column Map exists, the Modify Column Map panel is displayed.

## Selection List

A selection list is requested by using DB2 LIKE syntax or leaving one or both prompts blank as criteria. Column Maps that match the criteria are listed on the Select Column Maps panel. If no Column Maps satisfy the criteria, a message is displayed.

## Validation Rules

When creating a Column Map, you must consider the purpose of the Column Map and the Optim component with which it is used. A single Column Map can be used with Archive to restore data, with Compare to select data for comparison, or with Move to migrate extracted data to other tables or platforms.

A Column Map defined for a Compare Process can be used in Archive or Move processing; however, the Archive and Move functions are not available to Compare, and a Column Map defined for Compare processing may not provide functions needed for Archive and Move. Therefore, you may want to use

naming conventions or other means to segregate Column Maps defined for Compare processing.

## Column Map Selection List

The setting for **Rules used to validate Column Map** on the Choose a Column Map panel determines the format of the selection list and the rules applied to a selected Column Map. If you select Archive and Move rules, the selection list identifies the source and destination tables and Archive and Move rules apply to the selected Column Map. If you select Compare rules, the selection list identifies the Source 1 and Source 2 tables and Compare rules apply to the selected Column Map.

An example of a Column Map Selection List when Archive and Move rules are selected is shown in the following figure.

```
--------------------------- Select Column Maps --------------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AR-Attr, I-Info 1 of 8

    -------- Map -------- -------- Source --------- ------ Destination ------
Cmd    ID        Name     Creator       Table       Creator     Table
--- -------- ----------—— ------>> --------------->> ------>> -------------->>
     ****************************** TOP ***********************************
___  ARC1997  ORDERS      FOPDEMO  ORDERS           RESTORED ORDERS
___  ARC1997  FOPCUST     FOPDEMO  CUSTOMERS        RESTORED CUSTOMERS
___  ARCHIVE  TEST        OPTIM    SHIP_TO          RESTORED SHIP_TO
___  ARCHIVE  TEST2       OPTIM    CUSTOMERS        FOPDEMO  CUSTOMERS
```

*Figure 97. Select Column Maps*

Column Maps that match the criteria are listed. The **Map ID** and **Name** are displayed, as well as Creator ID and Table Name for source and destination tables.

### Description

A User Option determines whether the description of each Column Map is displayed on this panel. See "User Options" on page 372 for information about the **Selection List Format** option.

### Line Commands

The **Cmd** area of the panel is used to enter line commands. The following line commands are available:

**Cmd** The line command entry area. Valid line commands are:

- **S** Select a Column Map.
- **D** Delete a Column Map.
- **C** Copy a Column Map.
- **R** Rename a Column Map.
- **AT** Modify attributes of a Column Map.
- **I** Display information about a Column Map.

See "Object Selection List Functions" on page 14 for more information about available commands.

### Select a Column Map

Use the S line command or the SELECT primary command to select a Column Map to display in the Modify Column Map panel. If the source table or file for the Column Map does not exist, the Specify Column Map Tables panel is displayed, prompting for source and destination table names. After you

provide the table names, the Define Column Map panel is displayed, allowing you to define a Column Map. (The Modify Column Map panel is the same as the Define Column Map panel, and is described in "Column Map Editor" on page 162.)

You can also use the SELECT primary command to display a Column Map that is not included on the selection list. For example, to select the Column Map named FOPDEMO.TESTMAP, enter:

SELECT FOPDEMO.TESTMAP

If the Column Map named on the SELECT primary command does not exist, the Specify Column Map Tables panel is displayed to allow you to create a new Column Map.

### Column Map Attributes

To display the attributes of a Column Map, type I in **Cmd** next to the name of the Column Map. The following figure shows the read-only Column Map Attributes panel for an Archive or Move Column Map, which includes **Source Table** and **Destination Table**. Compare Column Maps will include **Source 1 Table** and **Source 2 Table** information.

```
----------------------------- Column Map Attributes --------------------------
Command ===>

  ID                : ARC1997
  Name              : ORDERS

  Description       :
  Security Status   : PRIVATE
  Last Modified By  : FOPDEMO
  Modified On       : 2010-08-02 10.36.05

  Number of Columns : 8
  Destination Table : RESTORED.ORDERS

  Source Type       : FILE

  Extract File Name : FOPDEMO.ARCHIVE.FILE
  Source Table      : FOPDEMO.ORDERS
  Validations       : YES
```

*Figure 98. Column Map Attributes*

This is a read-only display. Optim generates and maintains the information on this panel based on your specifications for the Column Map. **Number of Columns** is the number of columns in the destination table, while **Source Type** indicates whether the source is a database table (TABLE), a table in an Extract or Archive File (FILE) or no table was provided (NONE). Note that **Extract File Name** is blank unless **Source Type** is FILE. **Validations** indicates whether the Column Map was validated when created.

## Tables for a Column Map

Creating a Column Map requires information about the tables that are mapped. The Specify Column Map Tables panel prompts for the names of source and destination tables or, for a Compare Process, the names of Source 1 and Source 2 tables. These tables provide information (column names and data types) used to populate the Column Map editor.

You must provide information about the destination, but can create a Column Map without identifying a Source or Source 1 table, manually entering the source information instead. The **Specify Column Map Tables** panel is displayed when you provide the name of a new Column Map or specify "LOCAL" while editing a Table Map or if you provide the name of a new Column Map on the Choose a Column Map

panel. This panel is also displayed if, from the Select Column Maps panel, you select a Column Map for which a source file or table does not exist.

```
--------------------------- Specify Column Map Tables ------------------------
Command ===>                                                 SCROLL ===> PAGE

Specify Destination Table (Required) and Source Table (Optional)
for Column Map ARCDEMO.ORDERS

Use Destination Columns From an Existing Table:
  Creator ID ===> RESTORED                                    >>
  Table Name ===> ORDERS                                      >>

Use Source Columns From:
  An Extract File:
    DSN of Extract File ===> 'FOPDEMO.ARCHIVE.FILE'
    Extract Table       ===> FOPDEMO.ORDERS                        >>

  Or, an Existing Table (Use LIKE Syntax for List):
    Source Creator ID   ===>                                       >>
    Source Table Name   ===>                                       >>
```

*Figure 99. Specify Column Map Tables*

The prompts for Destination or Source 2 table include:

**Use Destination Columns From an Existing Table:**
The Creator ID and Table Name for the Destination or Source 2 table. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Creator ID**
The Creator ID for the Destination or Source 2 table. The default is the SQL ID for the current user. Specify 1 to 128 characters.

**Table Name**
The name of the Destination or Source 2 table. Specify 1 to 128 characters.

**Note:** See "Long Object Names (LONs)" on page 8 for information on entering Creator IDs and Table Names of up to 128 characters.

The prompts for Source or Source 1 table include the following:

**Use Source Columns From:**
The Source or Source 1 table for Column Name and Data Type information used to populate the Column Map editor. Optionally specify a table in an Extract or Archive File, a DB2 table, or a Legacy Table.

**Note:** Only one type of source entry is allowed. You cannot specify both file and table entries.

**An Extract File:**

**DSN of Extract File**
The name of an Extract or Archive File for the Source or Source 1 table. Specify the name of a file (on disk, not tape) explicitly by enclosing it in quotes; otherwise the default **Data Set Prefix**, specified on the User Options panel, is prefixed to the name. You can obtain a selection list of data sets by using a wildcard character in the last position.

**Extract Table**
The fully qualified name of the Source or Source 1 table in the file (e.g., FOPDEMO.ORDERS). Use DB2 LIKE syntax or leave blank to select from a list of tables in the file.

**Or, An Existing Table:**

**Source Creator ID**
The Creator ID for the Source or Source 1 table. Specify 1 to 128 characters.

**Source Table Name**
The name of the Source or Source 1 table. Specify 1 to 128 characters.

You can enter an explicit value or DB2 LIKE syntax for both prompts in any combination, or for a single prompt with blanks for the other.

## Source or Source 1 Table

The Column Map must reside in the Optim Directory for the Destination or Source 2 subsystem. In most cases, you are connected to the Destination or Source 2 subsystem when defining a Column Map and would typically select a Source or Source 1 table from an Archive or Extract File used in the process for which the Column Map is defined. If a file is not available, you can select a database or Legacy Table as the source for column names. If you are processing data across DB2 subsystems, however, only a source table in an Archive or Extract File is available.

### No Source Table

In certain circumstances, you may not have access to either the source table or the Archive or Extract File. If so, you can create a Column Map without identifying a source table and manually enter the column names that map to the Destination or Source 2 columns. Since Optim cannot reference definitions for the manually entered source columns, validation is performed during processing.

Leave the Use Source Columns From: prompts on the Specify Column Map Tables panel blank to indicate there is no source table. A prompt is displayed to confirm that a source table is not desired. Press Enter to proceed with the Column Map definition. Use END to return to the Specify Column Map Tables panel, where you can identify a source table.

### Multiple Source Tables

When creating a Column Map manually, you can sometimes avoid errors and labor from typing source column names by using the CHANGE TABLES command to return to the Specify Column Map Tables panel. Enter the name of a table that has needed source column names and press Enter to return to the Column Map editor. Any previous Source Column entries are retained and you can use the LIST UNUSED or LIST ALL commands to select names of columns. You can use CHANGE TABLE and LIST several times to collect source column names from accessible tables to avoid typing each source column name.

For example, assume you do not have access to the source table and the destination table contains 100 columns. One method of inserting the source column names in the Column Map is to type the 100 entries. However, if an accessible table contains columns with 50 of the needed names, you can use CHANGE TABLE and specify this table as the source for the Column Map. Then, you can use LIST to insert the 50 names rather than type them. Also, if a second table includes another 25 columns with needed names, you can reuse CHANGE TABLE and LIST to include those names. In fact, since the destination table may include many of the needed source column names, you can specify it as the source in order to list those names in the Column Map.

Using CHANGE TABLE is a convenient way to avoid typing in names of source columns that are present in other tables. Note that you can turn validation off in order to save a Column Map created from several sources. See “Validate Column Map” on page 167 for additional information.

## Column Map Editor

After you identify the tables on the Specify Column Map Tables panel and press Enter, the Column Map editor is displayed, listing the Destination or Source 2 columns and data types with any matching source columns. The Column Map editor is also displayed from the Table Map editor or from the Specify COMPARE Sources panel, when you enter the name of a new Column Map or use the MAP command.

The Column Map editor is labelled as the Define Column Map panel when you provide the name of a new Column Map, and the Modify Column Map panel when displayed from the Table Map editor or you enter the name of an existing Column Map. In the initial display for a new Column Map, destination columns are automatically paired with any source columns with compatible data types that have the same name. Source 1 and Source 2 columns are paired automatically if both name and attributes (data type and precision) match exactly.

```
-- Define Column Map: ARC1997.ORDERS ------------------------------------------
Command ===>                                                Scroll ===> PAGE

Corresponding Columns MUST Have Compatible Data Types
Use LIST UNUSED Command for List of Unused Source Columns
Use LIST ALL Command for List of All Source Columns                    VAL ON
                                                                       MOVE
                                                                       1 OF 8
    --------FOPDEMO.ORDERS----------  -------------RESTORED.ORDERS-------------
Cmd     Source Column     Data Type   Num Destination Column Data Type  Status
--- -------------------->> ----------  --- ------------------ ---------- -------
*** ********************************** TOP ************************************
___ ORD_ID                CH(5)         1 ORD_ID             CH(5)      EQUAL
___                                     2 CUST_ID            CH(5)      NOTUSED
___                                     3 ORDER_DATE         DATE       NOTUSED
___                                     4 ORDER_TIME         TIME       NOTUSED
___                                     5 FREIGHT_CHARGES    DEC(4,2)   NOTUSED
___ ORDER_SALESMAN        CH(6)         6 ORDER_SALESMAN     CH(6)      EQUAL
___                                     7 ORDER_POSTED_DATE  TIMESTAMP  NOTUSED
___ ORDER_SHIP_DATE       CH(8)         8 ORDER_SHIP_DATE    CH(8)      EQUAL
*** ********************************** BOTTOM *********************************
```

*Figure 100. Define Column Map - Initial Column List*

The Column Map editor includes:

**Cmd** The line command area. Valid line commands on the Define/Modify Column Map panel are:

- **CLR** Clear **Source** or **Source 1 Column** entry.
- **D** Delete entries with UNKNOWN status.
- **EXP** Display **Source** or **Source 1 Column** entry in a pop-up window to specify a column value or column map expression.
- **I** Display status information about a source and destination pairing. Useful for *ERROR* status.
- **SRC** Replace **Source** or **Source 1 Column** entry with name of Source Column.
- **AGE** Display AGE function prompts (Move only).

### Source or Source 1

The fully qualified name of the Source or Source 1 table is displayed before the headings:

**Column**
The value that is mapped to the **Destination** or **Source 2 Column** listed on the same line. For a Compare Process, specify a source column name or blanks. For processes other than Compare,

you can also enter a literal, an expression, a local column map procedure, a function, an exit routine, a special register, or NULL. Valid entries vary according to the destination **Data Type**.

**Note:** If you eliminate a foreign key column from a Compare Process by eliminating the Source 1 value, the relationship cannot be used to determine whether related changes have occurred. Also, to use a Match Key in a Compare Process, all Match Key columns must be processed. If you omit a primary key column from the Compare Process, you are prompted to specify a Match Key from columns in the Column Map.

**Data Type**

The data type for the source. Generally, **Data Type** is blank if the **Source Column** value is not the name of a column in the current source table. When the **Source Column** entry is an AGE function specification, however, the data type is either blank or PROP (if the aged value is to be propagated).

The data types for source and destination must be compatible. See Appendix D, "Compatibility Rules," on page 429 for more information.

The data type is automatically supplied and cannot be modified. Generally, the DB2 or Legacy data type is displayed; however, due to space limitations, the following data types are abbreviated. Data types are shown, followed by the abbreviations:

**CHAR**
**CHAR FOR MIXED DATA**
CH

**VARCHAR**
VCH

**LONG VARCHAR**
LVR

**DECIMAL**
DEC

**FLOAT 4-byte length**
SNGL FLOAT

**FLOAT 8-byte length**
DBL FLOAT

**BLOB** BL

**CLOB** CL

**CLOB FOR MIXED DATA**
CLOB

**GRAPHIC**
GR

**VARGRAPHIC**
**LONG VARGRAPHIC**
VGR

**BINARY**
BIN

**VARBINARY**
VARBIN

## Destination or Source 2

The fully qualified name of the Destination or Source 2 table is displayed before the headings:

**Num** A sequential number assigned, in ascending order, to each Destination or Source 2 column. Use these values with source column selection lists or the EXPAND command.

**Num** is populated for columns in the current destination table only.

**Column**
The read-only names of the Destination or Source 2 columns listed in the order defined in the table.

**Data Type**
The data type for the Destination or Source 2 column. **Data Type** is blank if the column is not in the current destination or source table. Due to space limitations, some data types are abbreviated. For more information, see the Source or Source 1 **Data Type** description.

**Status** The status of each column pairing. The matched pairs are assigned the status EQUAL or MAPPED. Any pair that does not match is assigned NOTUSED or REQUIRD status.

**EQUAL**
The columns have the same name, data type, and length.

***ERROR***
The specified mapping is invalid. A Column Map cannot be saved with columns in *ERROR* status. Use the I line command to determine the reason for the error.

**EXIT** The source column value is determined by a site-provided exit routine.

**EXPR** The source column value is an expression or a function.

**LITERAL**
The source column value is a literal.

**LuaPROC**
The source column value is a column map expression or local column map procedure using the Lua scripting language.

**MAPPED**
The columns do not have identical data types or length but are compatible. This status also applies to mapped columns that are not validated.

**NOT_INS**
A value cannot be inserted into the column. For example, the column may be an expression in a view.

**NOTUSED**
No source entry is specified. For load or insert processing, the default value for the destination column is inserted. For update processing, the value in the destination column is not updated. The column does not participate in a Compare Process.

**NULL** The source column value is NULL.

**REQUIRD**
No source column value is provided and the destination column is defined as NOT NULL (with no default column value). For update processing, the value in the destination column is not updated. For insert processing, a value is required.

When you save a Column Map with columns in REQUIRD status, a confirmation prompt reminds you that the Column Map can be used only to update rows. Press Enter to save the Column Map or use END to return to the editor from this pop-up.

**Note:**

- You cannot use a Column Map with columns in REQUIRD status for insert or load processing.
- You cannot use a Column Map with a primary key column in REQUIRD.

**SPC_REG**
The source column value is a special register.

**TRANSFM**
The source column value is a data privacy transformation library function.

**UNKNOWN**
The column does not exist in the current Destination or Source 2 table. This status may occur if the column is dropped from the destination table after the map is created or if the destination table specification is changed.

Use the D line command or the DELETE UNKNOWN primary command to delete entries in UNKNOWN status and adjust the column count.

## Available Commands

The following primary commands are available:

- ATTRIBUTES
- BOTTOM
- CANCEL
- CAPS
- CHANGE TABLE
- CLEAR
- DELETE UNKNOWN
- DOWN
- END
- EXPAND
- LIST ALL/UNUSED
- RESET
- SAVE
- SHOW
- TOP
- UP
- VALIDATION

## Column Map Description

Use the ATTRIBUTES command to display the Object Attributes panel, which allows you to specify a description for the Column Map. This panel provides a 40-character area to display and edit the description. (Site management determines whether this panel also displays a prompt for Security Status.) For additional information about the Object Attributes panel, see “Specify Description and Security Status” on page 380.

## SAVE

After editing a Column Map, you can save it under a different name, using the SAVE command with the fully qualified name as an operand. If you invoked the Column Map editor from the Table Map editor or the Specify Compare Sources panel, you can also use SAVE LOCAL to save the Column Map as part of the Table Map or Compare Definition.

## Manage Display

By default, all Destination or Source 2 columns are displayed in the Column Map editor. When the list of columns is extensive, you can use the SHOW command to limit the display to columns having a specific status. This command is useful, for example, to focus on columns that have not been mapped.

To display columns that have NOTUSED status, enter:

```
SHOW NOTUSED
```

For example, after entering the command on the panel in Figure 100 on page 162, the following is displayed.

```
-- Define Column Map: ARC1997.ORDERS ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Corresponding Columns MUST Have Compatible Data Types
Use LIST UNUSED Command for List of Unused Source Columns
Use LIST ALL Command for List of All Source Columns                    VAL ON
                                                                       MOVE
                                                                       1 OF 5
    --------FOPDEMO.ORDERS----------  -------------RESTORED.ORDERS-------------
Cmd     Source Column     Data Type   Num Destination Column Data Type  Status
--- -------------------->> ----------  --- ------------------ ---------- -------
*** ********************************** TOP ************************************
___                                     2 CUST_ID            CH(5)      NOTUSED
___                                     3 ORDER_DATE         DATE       NOTUSED
___                                     4 ORDER_TIME         TIME       NOTUSED
___                                     5 FREIGHT_CHARGES    DEC(4,2)   NOTUSED
___                                     7 ORDER_POSTED_DATE  TIMESTAMP  NOTUSED
*** ********************************* BOTTOM **********************************
```

*Figure 101. Display NOTUSED Columns*

Use SHOW or SHOW ALL to redisplay all destination columns, regardless of status.

### Add Destination Columns

To add Destination or Source 2 columns to the Column Map, use the CHANGE TABLE command to redisplay the Specify Column Map Tables panel and change the destination table name. When you redisplay the Column Map editor, the names of columns from the newly specified destination table are listed before those from any previous destination tables.

**Note:** The CHANGE TABLE command is not available when the Column Map editor is invoked from the Table Map editor. However, you can change the name of the destination table from the Table Maps.

### Clearing Source Columns

While editing the Column Map, you may want to remove one or more source column specifications. You can use the CLEAR primary command to remove all source column specifications or the CLR line command to remove the specifications for a single column.

The CLR line command is especially useful for clearing values that have been entered on the expanded source column display. You can remove these specifications without invoking the expanded display. Using CLR or CLEAR is the only way to remove the AGE function.

## Edit Source Column Values

In addition to automatic mapping of source to Destination or Source 2 columns, you can explicitly enter a value using one of the following methods:

- Selecting from a list.
- Typing a value such as a column name, literal, constant, or expression.
- Specifying a site-defined exit routine.
- Specifying a column map expression or local column map procedure using the Lua scripting language.

### Validate Column Map

When you specify a source column value and press Enter, Optim confirms, to the extent possible, that the entry is compatible with the Destination or Source 2 column. Optim also validates that any special registers, functions, expressions, and local column map procedures use the proper syntax and are appropriate for destination columns.

By default, Optim validates the data types of paired columns to ensure that they are compatible. Any validation errors are indicated by *ERROR* status. You can obtain an explanation of the error by using the I line command. You cannot save a Column Map with an *ERROR* status.

When a source table is specified in a Column Map, validation confirms that the Insert, Restore, Convert, or Load Process will operate properly. However, if you use multiple tables to obtain source column names, you can disable validation to allow you to save the Column Map. For example, you should disable validation if you choose column names from one table and then select additional column names from another table that does not include the columns from the first table. You should also disable validation when the data types of the source columns used in a process are different from the data types of the columns used to provide the source column names.

Note that Optim always validates the compatibility of source and destination columns during a process that uses a Column Map.

#### Disable Validation

Use the VALIDATION command to enable or disable validation. To disable automatic validation, enter:

```
VAL OFF
```

To enable validation, enter:

```
VAL ON
```

If validation is ON, a pairing of a column in the source table that matches exactly the destination column is identified by the EQUAL status. If validation is OFF, MAPPED status is displayed for the same pair of columns.

The validation status, VAL ON or VAL OFF, is displayed on the side of the Define Column Map panel before the column count.

## LIST

The LIST command displays a list of columns in the current Source or Source 1 table. Use LIST ALL to display a list of all columns in the table. Use LIST UNUSED to display a list of columns that have not been paired. With either list, you can map columns by entering a Destination or Source 2 **Num** value for the listed column.

### Example

The following figure shows the list produced from using the LIST UNUSED command on the panel shown in Figure 101 on page 166. The data type specifications in the list assist in selecting compatible columns. For more information about column compatibility, see Appendix D, “Compatibility Rules,” on page 429.

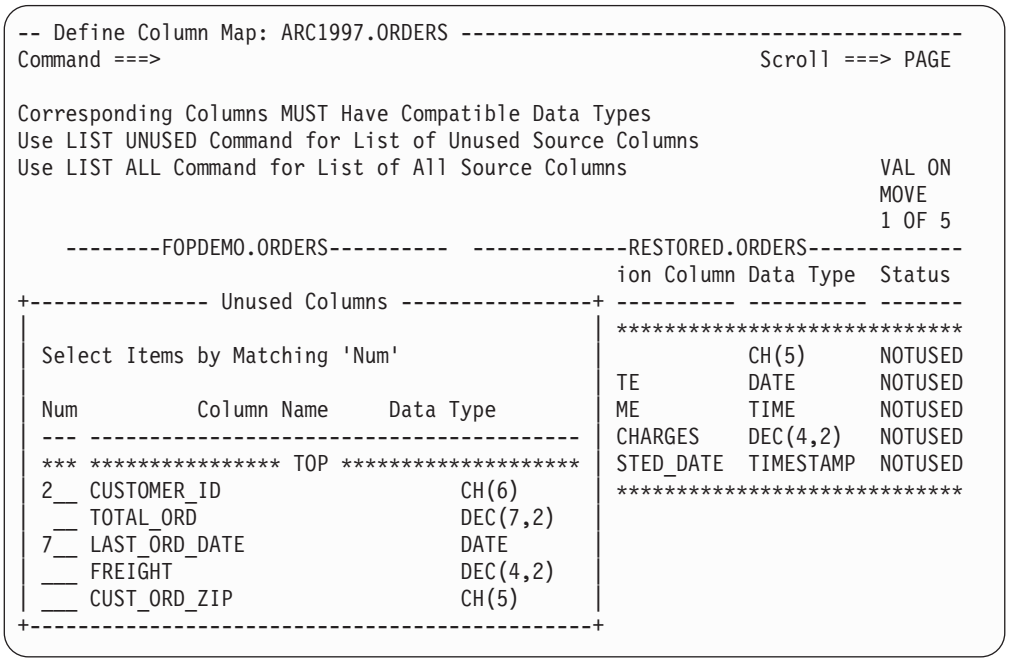



*Figure 102. Match Unused Source Columns*

In the figure, the source column CUSTOMER_ID is mapped to the destination column CUST_ID, and the source column LAST_ORD_DATE is mapped to the destination column, ORDER_POSTED_DATE. The remaining source columns are not mapped.

To scroll the list of source columns, use the UP, DOWN, TOP, and BOTTOM commands or the PF keys assigned to these functions.

## END List

Use END to return to the Define Column Map panel from the list. If validation is ON, the panel displays EQUAL status for all mappings with matching data types, *ERROR* status for any invalid mappings, and MAPPED status for all other mappings. If validation is off, MAPPED status is displayed for all valid mappings. (For additional information on validation, see “Validate Column Map” on page 167.)

The following figure shows the Define Column Map panel after executing the SHOW ALL command to redisplay all destination columns and the specified source columns.

```
-- Define Column Map: ARC1997.ORDERS ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Corresponding Columns MUST Have Compatible Data Types
Use LIST UNUSED Command for List of Unused Source Columns
Use LIST ALL Command for List of All Source Columns                    VAL ON
                                                                       MOVE
                                                                       1 OF 8
    --------FOPDEMO.ORDERS----------  -------------RESTORED.ORDERS-------------
Cmd     Source Column     Data Type   Num Destination Column Data Type  Status
--- ------------------>> ----------  --- ------------------ ---------- -------
*** ********************************** TOP ************************************
___ ORD_ID                CH(5)         1 ORD_ID             CH(5)      EQUAL
___ CUSTOMER_ID           CH(6)         2 CUST_ID            CH(5)      MAPPED
___                                     3 ORDER_DATE         DATE       NOTUSED
___                                     4 ORDER_TIME         TIME       NOTUSED
___ EXIT CHEKFRGT                       5 FREIGHT_CHARGES    DEC(4,2)   EXIT
___ ORDER_SALESMAN        CH(6)         6 ORDER_SALESMAN     CH(6)      EQUAL
___ LAST_ORD_DATE         DATE          7 ORDER_POSTED_DATE  TIMESTAMP  MAPPED
___ ORDER_SHIP_DATE       CH(8)         8 ORDER_SHIP_DATE    CH(8)      EQUAL
*** ********************************* BOTTOM **********************************
```

*Figure 103. Define a Column Map - Return from LIST*

## Type a Source Column Value

On the Column Map editor, you can type any of the following in **Source Column**:

- A column name. The source column need not have the same name or data type as the destination column. Any value bounded by the escape character designated in the DB2 installation options is assumed to be a column name. If the column is in the current source table, the source column data type and the appropriate status are displayed automatically. If not, Data Type is blank and the status is MAPPED.
- A literal value. A character literal must be bound by the SQL string delimiter character designated in the DB2 installation options. The data type for a literal is blank and the status is LITERAL.
- NULL or a special register. Special registers are: CURRENT TIME or CURRENT_TIME, CURRENT DATE or CURRENT_DATE, CURRENT TIMESTAMP or CURRENT_TIMESTAMP, CURRENT SQLID or CURRENT_SQLID, CURRENT TSOID or CURRENT_TSOID, and USER.
- A function.
- A concatenated or arithmetic expression.
- An exit routine.
- A column map expression or local column map procedure using the Lua scripting language.

### EXPAND Commands

If the value you want to enter exceeds the space available in **Source Column**, use the EXP line command or the EXPAND primary command to display an ISPF panel on which you can enter an expanded entry of up to 2000 characters. Type EXPAND at the command prompt, place the cursor in the desired **Source Column** position, and press Enter. You can also specify the destination column name or number with the EXPAND command (such as EXPAND ORDER_SALESMAN or EXPAND 6).

The following figure shows an expression entered on the ISPF panel for the column ORDER_SALESMAN. The expression creates a value by concatenating 'FOP' with a random number greater than or equal to 100 and less than or equal to 999.

```
EDIT       COLUMN-MAP-EXPRESSION-TEXT                    Columns 00001 00072
Command ===>                                               Scroll ===> PAGE
****** ***************************** Top of Data ****************************
000001 'FOP' || RAND(100,999)
****** **************************** Bottom of Data ***************************
```

*Figure 104. Expanded Source Column*

You can scroll the expanded source column display to the previous or next column using UP, DOWN, TOP, and BOTTOM.

Use END to return to the Define Column Map panel. If the length of the source specification exceeds 21 characters, the data is truncated and the source column value is protected. Expand the source column again to change the value.

### Completed Column Map

Use END to save the Column Map and redisplay the panel from which the Define/Modify Column Map panel was invoked. You cannot save a Column Map with a column pairing in *ERROR* status.

Use the SAVE command to save the modified Column Map under a new name. You can use this command to model a new Column Map on an existing one.

## Functions for Source Column Values

Several functions allow you to manipulate the source data prior to inserting or restoring it. These functions and their syntax are explained in the following discussion.

### Propagating Primary and Foreign Key Values

The PROP function propagates values in the primary key or foreign key to all related tables. This function is especially useful for generating multiple sets of related data. Provide the value for the source column in the Column Map and the value is automatically propagated during processing.

***valuecolumn-name***

Assigns a value to a column and propagates that value to all related tables.

***value*** is one of the following:

- A literal.
- An expression such as a function or function concatenated with a literal, etc.
- A special register.
- An exit routine.

***column-name*** is the source column containing the value that is the subject of the function. The resulting value is inserted into the destination column of the mapped table and the appropriate destination column in the participating related tables.

The ***column-name*** parameter is required if a source column does not match the destination column name or if ***value*** is a concatenated expression.

The following examples illustrate this function.

**PROP('A' CONCAT SEQ(1, 1))**

Propagate a literal value that is concatenated with a numerical value, starting with 1 and incrementing it by 1, in the destination column and the destination columns of the related tables.

**PROP(SEQ(10000, 5))**

Generate a sequential number, starting with 10000 and incrementing it by 5, and propagate that number in the destination column and the destination columns of the related tables.

**PROP(SEQ(10000, 5),CUST_NUMBER)**

Generate a sequential number, starting with 10000 and incrementing it by 5, and propagate that number in the destination column and the destination columns of the related tables.

**PROP(EXIT CHEKCUST)**

Call an exit routine to establish the value for the destination column and propagate that number in the destination columns of the related tables.

Note the following when using PROP:

- You can propagate a primary or foreign key, but not both in one operation.
- If propagating a value from a foreign key column using a value in a named column, only one column name can be used in the PROP statement. The column must participate in the relationship and be common to both parent and child tables.
- If using an exit routine, the PROP function and the exit routine must apply to the same column. Optim does not call the exit before propagating the column to another table.
- Extended relationships are supported.
- Conflicting PROP specifications cannot be identified until the process is executed. (The Control File identifies any errors.)
- The PROP function is not supported for conversion in self-referencing relationships.
- In insert or load processing, rows are discarded if propagated primary key values duplicate an existing value. Rows from related tables with the propagated value in a foreign key are also discarded to ensure that "foster" child rows are not inserted.
- The PROP function is not allowed with update processing.
- Move allows you to propagate values from a column with an AGE function. Select a setting on the Aging Specifications panel to propagate the aged value. See the "Move Age function" on page 202. The data type for the source column is shown as PROP.
- In a Convert Process, tables are processed in the order in which they were extracted. If using Convert to propagate a value from parent to child, the Access Definition for the Extract Process must list the parent table before the child.

Before executing a process, you can display propagation specifications for the process and print a report using the OUTPUT command.

## Functions to Manipulate Values

Other functions used to manipulate values are SUBSTR, RAND, and SEQ.

**Note:** You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

**SUBSTR (*column-name*, *start*[, *length*])**

Inserts a substring of the contents of the named source column.

*column-name*
is the name of a non-numeric column with a character or graphic data type.

*start* and *length* are integers with a value greater than or equal to 1, where *start* indicates a position within the string. *start* plus *length* cannot exceed the length attribute for the column.

Only *column-name* and *start* are required. If one integer is specified, it is considered the *start* value for a substring that ends with and includes the last character in the column.

**RAND(*low, high*)**

Inserts a random number. The number is bounded by the values for **low** and **high**. Valid **low** and **high** values are listed in **Values for Parameters**,.

**SEQ(start, step)**

Inserts a sequential number. The number is based on the **start** value and incremented by the **step** value. A list of valid **start** and **step** values follows. If the SEQ value exceeds 2147483648, it is reset to the start value.

### Values for Parameters

The range of valid values for **low**, **high**, **start**, and **step** varies according to the data type and function.

| Data Type | Function | Low Value | High Value |
|---|---|---|---|
| **Character** | **SUBSTR** | -2147483648 | 2147483647 |
| **Character** | **RAND, SEQ** | -9223372036854775808 | 9223372036854775807 |
| **Integer** | **SUBSTR, RAND, SEQ** | -2147483648 | 2147483647 |
| **Small integer** | **SUBSTR, RAND, SEQ** | -32768 | 32767 |
| **BIGINT** | **SUBSTR** | N/A | N/A |
| **BIGINT** | **RAND, SEQ** | -9223372036854775808 | 9223372036854775807 |
| **Floating point** | **SUBSTR** | -2147483648 | 2147483647 |
| **Floating point** | **RAND, SEQ** | -9223372036854775808 | 9223372036854775807 |
| **Decimal** | **SUBSTR** | maximum number of significant digits before the decimal | maximum number of significant digits before the decimal |
| **Decimal** | **RAND, SEQ** | -9223372036854775808 | 9223372036854775807 |

For example, if the data type for a column is defined as DEC(6,2), the maximum number of significant digits before the decimal is 4 because two digits are specified after the decimal. Therefore, the range for **low** and **high** or **start** and **step** for this column is -9999 through 9999 inclusive.

The special registers CURRENT DATE or CURRENT_DATE, CURRENT TIME or CURRENT_TIME, and CURRENT TIMESTAMP or CURRENT_TIMESTAMP cannot be expressed by functions.

## Expressions for Source Column Values

A concatenation of arithmetic expressions, exit routines, values, or functions, other than DB2 special registers or LOOKUP, can be inserted in the destination column. Use the CONCAT keyword or two vertical bars, `||`, to indicate concatenation.

For example, you can concatenate a literal value, 'OPT', with a function, SUBSTR, using either operator:

```
'OPT' CONCAT SUBSTR(ORD_ID,1,3)
'OPT' || SUBSTR(ORD_ID,1,3)
```

Since the expression is validated at run time, the source data type is blank and the status is EXPR. An invalid expression causes a process to discard any affected row and mark it as a conversion error.

The specifications in the Column Map are combined with the specifications for an individual request on the Aging Parameters panel, which is used while defining an Insert, Convert, or Load Process.

When RAND or SEQ are in a concatenated expression, a VARCHAR string is returned.

### Arithmetic Expressions

Specify an arithmetic expression in **Source Column** to insert a value in any numeric destination column, including decimal, integer, small integer, and float data types. To specify an arithmetic expression, use the following syntax:

```
value operand value
```

Note that you cannot combine a RAND, SEQ, or SUBSTR function and an arithmetic expression. For example, given an integer column named AGE, the expression AGE + 7 is valid but the expression AGE + RAND(1,99) is not valid.

***value* can be a source column name or a numeric constant.**

- One *value* must be a column name.
- Zero (0) cannot be specified as a *value*.
- If NULL is specified as a *value*, NULL is inserted in the destination column.

***operand* can be:**

| | |
|---|---|
| **+** | Add |
| **-** | Subtract |
| * | Multiply |
| **/** | Divide |

The following are examples of arithmetic expressions that can be specified as the source value in a Column Map.

FREIGHT_CHARGES + 2.00
ON_HAND_INVENTORY - 15
UNIT_PRICE * 1.1
TOTAL_COST / UNIT_PRICE

The following error conditions are detected only at run time and are reported in the Control File.

- Overflow.
- Null value in a non-nullable column.
- Field too large to participate in computation.

## Special Register - for an IMS Concatenated Key

Optim includes a special register for obtaining the value of an IMS Concatenated Key. You can use this special register in an INSERT or CONVERT process to substitute the value of the concatenated key for an actual column value.

For an IMS Legacy Table stored in an Extract File, you can create a Column Map that includes the special register, FOP_IMSKEY. In the example shown, FOP_IMSKEY is used to replace the SALESMAN_NAME from the source column with the IMS Concatenated Key in the destination column:

```
    ---------FOPQAHID.SALES---------  -------------FOPQAHID.SALES--------------
Cmd     Source Column     Data Type   Num Destination Column Data Type  Status
--- -------------------->> ----------  --- ------------------ ---------- -------
*** ********************************** TOP ***********************************
___ SALES                 CH(54)        1 SALES              CH(54)     EQUAL
___ SALESMAN_ID           CH(6)         2 SALESMAN_ID        CH(6)      EQUAL
___ FOP_IMSKEY                          3 SALESMAN_NAME      CH(20)     SPC_REG
___ AGE                   INTEGER       4 AGE                INTEGER    EQUAL
```

In the following example, FOP_IMSKEY is used in an expression. The SALESMAN_NAME will be replaced by the first 5 bytes of IMS Concatenated Key and the literal 'A'.

Use the EXP line command in the Column Map editor and enter the following expression.

```
------------------------------------------------------------------------------
EDIT       COLUMN-MAP-EXPRESSION-TEXT                     Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 SUBSTR(FOP_IMSKEY,1,5)||'A'
 ---------FOPQAHID.SALES---------  -------------FOPQAHID.SALES--------------
Cmd     Source Column     Data Type   Num Destination Column Data Type  Status
--- -------------------->> ----------  --- ------------------ ---------- -------
*** ********************************** TOP ***********************************
___ SALES                 CH(54)        1 SALES              CH(54)     EQUAL
___ SALESMAN_ID           CH(6)         2 SALESMAN_ID        CH(6)      EQUAL
___ SUBSTR(FOP_IMSKEY,1,5)              3 SALESMAN_NAME      CH(20)     EXPR
```

**Note:** FOP_IMSKEY is valid only for a Legacy IMS source table.

## Exit Routines for Source Column Values - Optim v7.2 and earlier

You can define an exit routine to derive a value for an individual destination column. Exits are useful for special processing, data manipulation, and literals that exceed the capacity of built-in **Source Column** functions.

Use an exit to:

- Perform data transformations that are beyond the scope of a column map. For example, an exit can be used to change an employee department number for selected rows based on a complex algorithm.
- Mask confidential information. For example, an exit can be used to change an employee name or salary information on selected rows.
- Select specific rows to process and discard all others.

**Note:** Column map exit routines created with releases of Optim for z/OS solutions prior to v7.1 must be recompiled.

### Using an Exit Routine

Enter the keyword EXIT followed by a space and the 1- to 8-character name of the exit routine in **Source Column**. When you press Enter, the status is displayed. The existence of the exit routine is not verified until executing a process that uses the column map.

For example, to specify the exit routine CHEKFRGT as the source of data to insert in the FREIGHT_CHARGES column, type the following in the corresponding **Source Column**:

```
EXIT CHEKFRGT
```

See Figure 103 on page 169 for an example of this exit specification.

You can also provide a string of parameters to be queried by the exit routine. The combined exit name and parameter string must be no more than 75 characters, including the required single quotes around the parameter string. The exit must provide any validation and error messages. For example, to pass parameters to the CHEKFRGT exit, use the EXP line command or EXPAND command to expand the entry area, and type:

```
EXIT CHEKFRGT 'parm1, parm2, parm3'
```

**Note:** The opening and closing quotes are passed to the exit routine. You cannot include quotes inside the parameter string.

## Requirements

All exits must conform to the following requirements:

- The exit must be written in Assembly Language, VS COBOL II, PL/I, or C.
- The exit must be reentrant and must be available to all Optim online and batch processes that use the column map.
- Each exit must be compiled and linked as a separate load module. This load module is loaded dynamically at execution time.
- The exit must set the value in the destination column.
- If the exit accesses DB2, the DBRM for the exit program must be bound into the Optim plan.
- The exit must be built as an LE subroutine executing in the existing LE environment when compiled with an IBM LE-enabled compiler.

## Default Parameters

During processing, the exit is called for each row in the mapped table. With every call, four parameters are passed to the exit using standard OS linkage conventions. The parameters are:

1. The address of a structure that contains:

   **DB2 subsystem name**
   Name of the DB2 subsystem for the table (4 characters, right-filled with blanks).

   **Optim ID**
   Version of the Optim release (4 characters, right-filled with blanks).

   **Primary SQLID**
   Primary authorization ID for the signed-on user account (8 characters, right-filled with blanks).

   **Current SQLID**
   Current authorization ID (128 characters, right-filled with blanks). This is the primary SQLID or, if the user changes SQLID within Optim, a secondary authorization.

   **Source Table**

   **Creator ID**
   Creator ID for the source table (1 to 128 characters, right-filled with blanks).

   **Length**
   Length of source table name (2 bytes, binary).

   **Table Name**
   Name of the source table (1 to 128 characters).

   **Destination Table**

   **Creator ID**
   Creator ID for the destination table (1 to 128 characters, right-filled with blanks).

   **Length**
   Length of destination table name (2 bytes, binary).

   **Table Name**
   Name of the destination table (1 to 128 characters).

   **Column Identifier**
   Relative position of the column (beginning with 1) in the table, equal to the subscript of the SQLVAR entry for the column in the SQLDA describing the destination row (2 bytes, binary).

**Address**
Address of a 256-byte global work area (that is, an area accessible to exits for columns in all destination tables). Use this work area to share information among exits (4 bytes, pointer).

**Address**
Address of a 256-byte table work area that is unique to the table and accessible to exits for all columns in the table (4 bytes, pointer).

**Address**
Address of the user parameters passed with the EXIT call (4 bytes, pointer).

2. Address of an SQLDA describing the source row. (The SQLDA described in the appropriate DB2 documentation.)
3. Address of an SQLDA describing the destination row.
4. Address of a 256-byte work area that is unique to the column for which the exit is called, even if the exit is used for more than one column. Use this work area to pass information from one exit call to another for a column, serve as a save area or for other reasons. This area is initialized to binary zeros on the first call.

## Performance

Because the exit might be called frequently, avoid unnecessary overhead. For example, write the exit to perform initialization processing in the first call and save information for subsequent calls in the work area. Also, if you write the exit in COBOL or PL/I, be sure to maintain the COBOL or PL/I run time environment across calls to the exit program to prevent terminating and reestablishing this environment with each call. The specific technique used depends on the compiler.

Note that the run time environment is created only once for an exit written in COBOL II. For example, when using LE/370, a reusable run time environment is established with the option RTEREUS, either within the site default options module, CEEDOPT, or in the application specific options module, CEEUOPT. The delivered setting is OFF. In order to change this setting for your exit, assemble CEEUOPT, changing the statement "RTEREUS=(OFF)" to "RTEREUS=(ON)." The member CEEUOPT is generally found in the LE SCEESAMP data set. You should then compile your exit program, and during the linkedit step, provide an INCLUDE statement for CEEUOPT. (For more information, consult the appropriate IBM documentation.)

## Setting Destination Values

When the exit is called, values in all destination and source columns are available except in destination columns that are yet to be assigned a value by an exit. To set the value in the destination column, the exit can examine data in the source row or data in other columns in the destination row. The order of destination columns need not match that of source columns because a destination column for which an exit is specified is always processed last. To locate the column in the source SQLVAR that corresponds to the column indicated by the destination column identifier, you can search the source SQLVAR one time and save the index value for later reference (as shown in the sample exits).

The exit is passed a copy of the destination row and can set a value only in the column for which the exit is specified. The destination value supplied by the exit is passed back into the actual row. The exit must adhere to the following requirements:

- The exit must not change any other storage areas.
- The exit must return a value appropriate for the column.
- The exit must clear or set the null indicator if the column allows nulls.
- Primary key information is used to build internal work areas. For this reason, an exit that assigns a value to a column in the primary key must generate the same output value if called multiple times for the same source row.

## Special Registers

The destination columns that use the special registers CURRENT DATE or CURRENT_DATE, CURRENT TIME or CURRENT_TIME, and CURRENT TIMESTAMP or CURRENT_TIMESTAMP are assigned values by DB2. These columns are marked in the SQLDATA field as:

**-2** CURRENT DATE
CURRENT_DATE

**-1** CURRENT TIME
CURRENT_TIME

**-3** CURRENT TIMESTAMP
CURRENT_TIMESTAMP

When the mapped source column is NULL, special processing is provided for DATE, TIME, and TIMESTAMP destination columns that are marked as NOT NULL WITH DEFAULT. For these columns, SQLDATA is set to -4. These special values affect only destination columns assigned values by the Optim solution, not those assigned values by an exit.

## Return Codes

The exit must set one of the following return codes:

**0** Destination column has been assigned a value.

**4** Destination column cannot be assigned a value. Discard the row.

**8** Terminate execution. To return an error message, place the message in the work area and set the unused space to blanks or null.

## Termination Call

A termination call is made to the exit when all destination table processing is complete. The call is identified by a value of 0 in the destination column identifier. The exit can use this call to clean up any dynamically acquired storage.

## Sample Exits

Sample exits are included with the Optim solutions. The samples illustrate how to use the data areas available to the exit and the type of processing that can be accomplished using an exit. These exits are distributed on the installation media. The source is available in the installation library. FOPPEXA is available as a load module in the product load library.

**FOPPEXA**
An Assembler exit that assigns a sequential value to a column for each row in the destination table. This can be used to mask sensitive information. The results are similar to those achieved by using the SEQ function in a Column Map, but the exit can be used as an example of exit processing in Assembler.

**FOPPEXC**
A COBOL exit that performs two operations on the CUSTOMERS table distributed with Optim. First, it filters rows to discard any row with a value of 0 in the YTD_SALES column. Second, it changes the SALESMAN_ID column for customers in Florida.

**FOPPEXLE**
A column map exit written in C. This sample also requires member FOPPSMEX from the samplib dataset.

For more details on these sample exits, consult the program source.

## Exit Routines for Source Column Values - Optim v11.3

You can define an exit routine to derive a value for an individual destination column. Exits are useful for special processing, data manipulation, and literals that exceed the capacity of built-in **Source Column** functions. Beginning with Optim version 11.3, column maps support variable array processing.

Column map user exits created with versions of Optim prior to 11.3 continue to be supported. Refer to "Exit Routines for Source Column Values - Optim v7.2 and earlier" on page 174 for details.

You can use a column map exit to:

- Perform data transformations that are beyond the scope of a column map. For example, an exit can be used to change an employee department number for selected rows based on a complex algorithm.
- Mask confidential information. For example, an exit can be used to change an employee name or salary information on selected rows.
- Select specific rows to process and discard all others.

**Note:** Column map exit routines created with releases of Optim for z/OS solutions prior to v7.1 must be recompiled.

### Using an Exit Routine

Enter the keyword EXIT followed by a space and the 1- to 8-character name of the exit routine in **Source Column**. When you press Enter, the status is displayed. The existence of the exit routine is not verified until executing a process that uses the column map.

For example, to specify the exit routine CHEKFRGT as the source of data to insert in the FREIGHT_CHARGES column, type the following in the corresponding Source Column: `EXIT CHEKFRGT`

You can also provide a string of parameters to be queried by the exit routine. The combined exit name and parameter string must be no more than 75 characters, including the required single quotes around the parameter string. The exit must provide any validation and error messages. For example, to pass parameters to the CHEKFRGT exit, use the **EXP** line command or **EXPAND** command to expand the entry area, and type: `EXIT CHEKFRGT 'parm1, parm2, parm3'`

**Note:**

The opening and closing quotes are passed to the exit routine. You cannot include quotes inside the parameter string.

When you specify an exit for an arrayed column, each instance of the column is processed by a separate instance of the user exit.

### Requirements

All exits must conform to the following requirements:

- The exit must be written in Assembly Language, VS COBOL II, PL/I, or C.
- The exit must be reentrant and must be available to all Optim online and batch processes that use the column map.
- Each exit must be compiled and linked as a separate load module. This load module is loaded dynamically at execution time.
- The exit must set the value in the destination column.
- If the exit accesses DB2, the DBRM for the exit program must be bound into the Optim plan.
- The exit must be built as an LE subroutine executing in the existing LE environment when compiled with an IBM LE-enabled compiler.

### Default Parameters

During processing, the exit is called for each row in the mapped table. With every call, seven parameters are passed to the exit using standard OS linkage conventions. The parameters are:

1. The address of a structure that contains:

   **DB2 subsystem name**
   Name of the DB2 subsystem for the table (4 characters, right-filled with blanks).

   **Optim ID**
   Version of the Optim release (4 characters, right-filled with blanks).

   **Primary SQLID**
   Primary authorization ID for the signed-on user account (8 characters, right-filled with blanks).

   **Current SQLID**
   Current authorization ID (128 characters, right-filled with blanks). This is the primary SQLID or, if the user changes SQLID within Optim, a secondary authorization.

   **Source Table**

   **Creator ID**
   Creator ID for the source table (1 to 128 characters, right-filled with blanks).

   **Length**
   Length of source table name (2 bytes, binary).

   **Table Name**
   Name of the source table (1 to 128 characters).

   **Destination Table**

   **Creator ID**
   Creator ID for the destination table (1 to 128 characters, right-filled with blanks).

   **Length**
   Length of destination table name (2 bytes, binary).

   **Table Name**
   Name of the destination table (1 to 128 characters).

   **Column Identifier**
   This value is provided for compatibility. Relative position of the column (beginning with 1) in the table, equal to the subscript of the SQLVAR entry for the column in the SQLDA describing the destination row (2 bytes, binary).

   **Address**
   Address of a 256-byte global work area (that is, an area accessible to exits for columns in all destination tables). Use this work area to share information among exits (4 bytes, pointer).

   **Address**
   Address of a 256-byte table work area that is unique to the table and accessible to exits for all columns in the table (4 bytes, pointer).

   **Address**
   Address of the user parameters passed with the EXIT call (4 bytes, pointer).

   **Row number**
   Current row number in table (4 bytes, binary)

   **Total rows**
   Total number of rows in table (4 bytes, binary)

**Length of the Source Row Data**
Length of the source row data (4 bytes, binary).

**CMDA Column Identifier**
Relative position of column (beginning with 1) in table. Subtract 1 to use as the subscript of the CMVAR entry for the column in the CMDA describing the destination row (2 bytes, binary).

2. This parameter is provided for compatibility. Address of an SQLDA describing the source row. (The SQLDA described in the appropriate DB2 documentation.)
3. This parameter is provided for compatibility. Address of an SQLDA describing the destination row.
4. Address of a 256-byte work area that is unique to the column for which the exit is called, even if the exit is used for more than one column. Use this work area to pass information from one exit call to another for a column, serve as a save area or for other reasons. This area is initialized to binary zeros on the first call.
5. Address of the API level value desired by the exit. Should be set upon first entry and not changed. (4 bytes, binary).
   0 – not selected (functions as 1)
   1 – pre release 11.3.0
   2 - Enhanced Legacy functionality (support for variable array columns, enhanced support for fixed arrays). Can be used for all data sources.
6. Address of a CMDA describing the source row.
7. Address of a CMDA describing the destination row.

The source and destination CMDA structures are similar to the SQLDA but have additional information that is useful for array processing.

**CMDA**

| Field | Offset | Length, Type | Description |
|---|---|---|---|
| ID | 0 | 8, char | "CMDA " |
| bc | 8 | 4, binary | Length of structure |
| Level | 12 | 2, binary | Structure level, intially 1 |
| TotCols | 14 | 2, binary | Total columns |
| NumCols | 16 | 2, binary | Used columns |
| CMVAR | 18 | Varies, CMVAR | Column descriptor |

**CMVAR**

| Field | Offset | Length, Type | Description |
|---|---|---|---|
| Type | 0 | 2, binary | DB2 column type |
| Len | 2 | 2, binary | Length of column value |
| pData | 4 | 4, binary | Address of column value |
| pInd | 8 | 4, binary | Address of null indicator |
| Instance | 12 | 2, binary | Instance number if in array relative to 1. 0 if not in array. |
| Name Length | 14 | 2, binary | Length of column name value |
| Name | 16 | 30, char | Column name value |

| Field | Offset | Length, Type | Description |
|---|---|---|---|
| Label Length | 46 | 2, binary | Length of column label value |
| Label | 48 | 90, char | Column label value |
| Flag1 | 138 | 1, (8) bits | 0x80 NO_VALUE indicates that this column has no value in this row instance |
| Unused | 139 | 61, char | unused |

## Performance

Because the exit might be called frequently, avoid unnecessary overhead. For example, write the exit to perform initialization processing in the first call and save information for subsequent calls in the work area. Also, if you write the exit in COBOL or PL/I, be sure to maintain the COBOL or PL/I run time environment across calls to the exit program to prevent terminating and reestablishing this environment with each call. The specific technique used depends on the compiler.

Note that the run time environment is created only once for an exit written in COBOL II. For example, when using LE/370, a reusable run time environment is established with the option RTEREUS, either within the site default options module, CEEDOPT, or in the application specific options module, CEEUOPT. The delivered setting is OFF. In order to change this setting for your exit, assemble CEEUOPT, changing the statement "RTEREUS=(OFF)" to "RTEREUS=(ON)." The member CEEUOPT is generally found in the LE SCEESAMP data set. You should then compile your exit program, and during the linkedit step, provide an INCLUDE statement for CEEUOPT. (For more information, consult the appropriate IBM documentation.)

## Processing

The exit is entered once for each row. After all rows are processed it is entered for termination allowing the exit to clean up and release any resource it acquired. When entered for a row, CMDA Column Identifier has a positive value indicating which destination column is being processed. When entered for termination, CMDA Column Identifier contains a value of zero.

The exit should set the integer value pointed to by the 5th parameter (API Level) to the value 2, indicating it was written for the level 2 interface. If it does not, level 1 will be assumed and the exit will not be able to process arrays and should not refer to the new parameters documented in this section.

## Arrays

The source CMDA contains a column descriptor for every column in the source row. There is a separate descriptor for every instance of an arrayed column. Columns for variable arrays also appear as well as the columns that follow the variable array. The Array column instance number contains a value of 0 when the column is not arrayed and a value of 1 – *n* indicating the occurrence of the column within the row. When a column does not have a value for the row (either the variable array does not have the maximum values or the row length is insufficient) the NO_VALUE flag is set. The data pointer may vary from row to row due to different column composition because of variation in the variable array occurrences.

## Variable Arrays

A variable array is an array whose elements occur a variable number of times. The number of times the elements occur in a particular row is dependent upon a numeric value in the row.

The columns in a variable array appear in the column mapping dialog. If you specify the exit for one of these columns, the exit will be called for every instance of the column in the array (from 0 to *n* times) for each row. Each column instance is processed by the exit as a distinct column. This is similar to specifying the exit on multiple non-arrayed columns.

The columns that follow the variable array have offsets that vary depending on the size of the variable array. These columns are in the "variable area". Since the data location varies for the columns in the variable area it is important for the exit to refer to the CDMA for the data pointer for every row instance.

When the exit is specified for a column in a variable array or in the variable area it must declare itself an API Level 2 exit by setting the API Level value to '2' upon the first call to the exit. If it does not this will be considered an error and the row will be discarded.

### Setting Destination Values

When the exit is called, values in all destination and source columns are available except in destination columns that are yet to be assigned a value by an exit. To set the value in the destination column, the exit can examine data in the source row or data in other columns in the destination row. The order of destination columns need not match that of source columns because a destination column for which an exit is specified is always processed last. To locate the column in the source SQLVAR that corresponds to the column indicated by the destination column identifier, you can search the source SQLVAR one time and save the index value for later reference (as shown in the sample exits).

The exit is passed a copy of the destination row and can set a value only in the column for which the exit is specified. The destination value supplied by the exit is passed back into the actual row. The exit must adhere to the following requirements:

- The exit must not change any other storage areas.
- The exit must return a value appropriate for the column.
- The exit must clear or set the null indicator if the column allows nulls.
- Primary key information is used to build internal work areas. For this reason, an exit that assigns a value to a column in the primary key must generate the same output value if called multiple times for the same source row.

### Special Registers

The destination columns that use the special registers CURRENT DATE or CURRENT_DATE, CURRENT TIME or CURRENT_TIME, and CURRENT TIMESTAMP or CURRENT_TIMESTAMP are assigned values by DB2. These columns are marked in the SQLDATA field as:

**-2** CURRENT DATE
CURRENT_DATE

**-1** CURRENT TIME
CURRENT_TIME

**-3** CURRENT TIMESTAMP
CURRENT_TIMESTAMP

When the mapped source column is NULL, special processing is provided for DATE, TIME, and TIMESTAMP destination columns that are marked as NOT NULL WITH DEFAULT. For these columns, SQLDATA is set to -4. These special values affect only destination columns assigned values by the Optim solution, not those assigned values by an exit.

### Return Codes

The exit must set one of the following return codes:

**0** Destination column has been assigned a value.

**4** Destination column cannot be assigned a value. Discard the row.

**8** Terminate execution. To return an error message, place the message in the work area and set the unused space to blanks or null.

### Termination Call

A termination call is made to the exit when all destination table processing is complete. The call is identified by a value of 0 in the destination column identifier. The exit can use this call to clean up any dynamically acquired storage.

### Sample Exits

Sample exits are included with the Optim solutions. The samples illustrate how to use the data areas available to the exit and the type of processing that can be accomplished using an exit. These exits are distributed on the installation media. The source is available in the installation library. FOPPEXA is available as a load module in the product load library.

**FOPPEXA**
An Assembler exit that assigns a sequential value to a column for each row in the destination table. This can be used to mask sensitive information. The results are similar to those achieved by using the SEQ function in a Column Map, but the exit can be used as an example of exit processing in Assembler.

**FOPPEXC**
A COBOL exit that performs two operations on the CUSTOMERS table distributed with Optim. First, it filters rows to discard any row with a value of 0 in the YTD_SALES column. Second, it changes the SALESMAN_ID column for customers in Florida.

**FOPPEXLE**
A column map exit written in C. This sample also requires member FOPPSMEX from the samplib dataset.

For more details on these sample exits, consult the program source.

## Column map procedures

A column map procedure is a procedure that is used to mask or transform the data in a source column. You can specify a column map procedure for data transformations in a convert, insert, load, or restore process that are beyond the scope of native column map functions. You define column map procedures in a column map using the Lua scripting language.

For details on defining column maps, see Chapter 5, “Column Maps,” on page 155.

One function of a column map procedure is to generate values that could not otherwise be defined for the destination column. This function is useful for handling special processing and data manipulation according to site-defined rules. Column map procedures are not limited to data transformation, however. You can also use a column map procedure to reject rows on the basis of custom processing, to create a report tailored to the needs of your site, or to implement conditional data transformations.

For instructions on defining a local column map procedure using Lua functions, see “Defining a local column map procedure using Lua functions” on page 193.

# Working with column map expressions and procedures using Lua functions

You can define a column map expression or column map procedure to mask or transform the data in a column using the Lua scripting language.

You can use most standard Lua functions in a column map expression or column map procedure. In addition, you can use functions that are specific to Optim. For more information about the Lua scripting language, see the Lua web site at http://www.lua.org. For the Optim-specific functions, see "Lua functions for column map expressions and procedures."

## Lua functions for column map expressions and procedures

A column map expression or column map procedure can contain either a complex expression (unstructured set of Lua statements) or a set of standard functions (structured set of Lua statements).

### Complex expression

A *complex expression* is an unstructured block of statements. When a column map expression or procedure contains a complex expression, the complex expression is called for every row processed. A complex expression is functionally the same as a `cm_transform()` function, but a complex expression cannot contain a formal `function` statement.

The following example complex expression replaces `nil` values in a column with `1` and leaves all other values unchanged.

```
srcvalue = optim.source.getcolumnvalue()
if srcvalue == nil then
  optim.target.setcolumnvalue('1')
else
  optim.target.setcolumnvalue(srcvalue)
end
```

### Structured Lua expression

The structured Lua expression must contain the `cm_transform()` function. It also may contain any valid combination of the other standard Optim Lua functions. The standard Optim Lua functions may not call each other, but may call other user-defined functions. User-defined functions may not call the standard Optim Lua functions.

For an example of a structured Lua expression, see "Sample structured Lua expression" on page 187.

### Standard Optim Lua function names

Use the following function names in your Lua column map expressions and procedures. Optim Lua processing calls each of these functions automatically at the point indicated.

| Name | Description | Required |
|---|---|---|
| `cm_load()` | This function is called before any tables are processed. | No |
| `cm_unload()` | This function is called after all tables are processed. | No |
| `cm_starttable()` | This function is called at the start of processing for each table. | No |
| `cm_endtable()` | This function is called at the end of processing for each table. | No |

| Name | Description | Required |
|---|---|---|
| cm_transform() | This function is called for every row processed. | Yes |

## Global functions

The following functions are available in all column map expression and procedure execution contexts.

| Name | Description |
|---|---|
| optim.print() | Print messages to process report. |
| optim.date() | Get current system date formatted in DB2 ISO character format (different from os.date()). |
| optim.time() | Get current system time formatted in DB2 ISO character format (different from os.clock()). |
| optim.timestamp() | Get current timestamp formatted in DB2 ISO character format (different from os.date("*t")). |
| optim.userid() | Get current user's TSOID or BATCH JOB owner. |
| optim.sqlid() | Get current user's SQLID. |
| optim.imskey() | Get value of the concatenated IMS key for the record being processed, when available (a nil, otherwise). |

## Extract File/Archive File functions

Use the following functions to get information about the source and target Extract Files/Archive Files.

| Name | Description |
|---|---|
| optim.source.getdbalias() | Get the source location/server name if a location has been defined while connecting to DB2. Or get the string SUBSYS: *ssss*, where *ssss* is the source local DB2 subsystem to which Optim connected. |
| optim.source.getcreatorid() | Get the creator ID of the source Extract File/Archive File. |
| optim.target.getdbalias() | Get the target location/server name if a location has been defined while connecting to DB2. Or get the string SUBSYS: *ssss*, where *ssss* is the target local DB2 subsystem to which Optim connected. |
| optim.target.getcreatorid() | Get the creator ID of the target Extract File/Archive File. |

## Table functions

Use the following functions to get information about the source and target tables. Table functions can be used only within a complex expression or within the cm_starttable(), cm_endtable(), and cm_transform() functions.

| Name | Description |
|---|---|
| optim.source.gettablename() | Get the name of the source table. |
| optim.target.gettablename() | Get the name of the target table. |

## Column functions

Use the following functions to get information about the source and target columns, transform column data, and write the result to the target column.

| Name | Description |
|---|---|
| `optim.source.getcolumnvalue()`<br>`optim.source.getcolumnvalue("COLNAME")`<br>`optim.source.getcolumnvalue(n)` | Get a value from the current source column (), the named source column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function for nonnumeric columns and for numeric columns that have data types other than the following data types:<br>• `BIGINT`<br>• `DECIMAL`<br>• `DOUBLE`<br>• `FLOAT`<br>• `DECIMAL FLOAT`<br>• `NUMBER`<br>Use this function only with the `cm_transform()` function or in complex expressions. |
| `optim.source.getcolumnasdouble()`<br>`optim.source.getcolumnasdouble("COLNAME")`<br>`optim.source.getcolumnasdouble(n)` | Get a value in double-precision format from the current source column (), the named source column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function to get data from columns that have the following data types:<br>• `BIGINT`<br>• `DECIMAL`<br>• `DOUBLE`<br>• `FLOAT`<br>• `DECIMAL FLOAT`<br>• `NUMBER`<br>Use this function only with the `cm_transform()` function or in complex expressions. |
| `optim.source.getcolumnlength()`<br>`optim.source.getcolumnlength("COLNAME")`<br>`optim.source.getcolumnlength(n)` | Get the length of the current source column (), the named source column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the `cm_starttable()`, `cm_endtable()`, and `cm_transform()` functions. |
| `optim.source.getcolumnname()`<br>`optim.source.getcolumnname("COLNAME")`<br>`optim.source.getcolumnname(n)` | Get the name of the current source column (), the named source column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the `cm_starttable()`, `cm_endtable()`, and `cm_transform()` functions. |
| `optim.source.getcolumntype()`<br>`optim.source.getcolumntype("COLNAME")`<br>`optim.source.getcolumntype(n)` | Get the DB2 data type of the current source column (), the named source column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the `cm_starttable()`, `cm_endtable()`, and `cm_transform()` functions. |

| Name | Description |
|---|---|
| optim.source.getnumcolumns() | Get the number of columns in the source table. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.target.setcolumnvalue()<br>optim.target.setcolumnvalue("*COLNAME*")<br>optim.target.setcolumnvalue(*n*) | Set the value of the current target column (), the named target column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function only with the cm_transform() function or in complex expressions. |
| optim.target.getcolumnlength()<br>optim.target.getcolumnlength("*COLNAME*")<br>optim.target.getcolumnlength(*n*) | Get the length of the current target column (), the named target column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.target.getcolumnname()<br>optim.target.getcolumnname("*COLNAME*")<br>optim.target.getcolumnname(*n*) | Get the name of the current target column (), the named target column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.target.getnumcolumns() | Get the number of columns in the target table. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.target.getcolumntype()<br>optim.target.getcolumntype("*COLNAME*")<br>optim.target.getcolumntype(*n*) | Get the data type of the current target column (), the named target column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.target.iscolumnnullable()<br>optim.target.iscolumnnullable("*COLNAME*")<br>optim.target.iscolumnnullable(*n*) | Determine whether the target column is nullable: the current target column (), the named target column ("*COLNAME*"), or column (*n*) in the column map, where *n* is a number representing the *n*th column. The function returns true if the column is nullable and returns false if the column is not nullable. Use this function in a complex expression or with the cm_starttable(), cm_endtable(), and cm_transform() functions. |
| optim.rejectrow() | Skip row and go to the next row. Use this function only with the cm_transform() function or in complex expressions. |

## Sample structured Lua expression

The Optim sample library, SFOPSAMP, includes the Lua sample files.

- FOPLUAT - demonstrates use of various functions to capture and write environment, table, and column metadata.
- FOPLUAF - demonstrates use of various file operations.
- FOPLUAG - illustrates the structure of a column map procedure with its standard functions: cm_load, cm_unload, cm_starttable, cm_endtable, and cm_transform.

The following sample structured Lua expression illustrates the structure of a column map expression or procedure using the standard functions: cm_starttable, cm_endtable, and cm_transform. It can be found in the FOPLUAT sample file.

```
--Lua
------------------------------------------------------------------------
--
--   Purpose:       Display metadata for tables, columns as well as
--                  execution environment.
--
--   Name:          FOPLUAT (OPTIM.TBLINFO)
--
--   Author:        IBM Corporation
--
--   Revision:      1.0
--
--   Description:   Captures and writes environment, table and column
--                  metadata.
--                  This data is written to the Optim process report
--
--   Input:         Parameters and values needed to process request.
--
--   Arguments:     None
--
--   Setup Parms:   'SOURCE' to limit output only to the source
--                   metadata.
--                  'TARGET' to limit output only to the target
--                   metadata.
--                  'ENVIRON' to limit output only to the environment
--                   data.
--                  'ALL' to produce source, target and environment
--                   data output.
--                  'NONE' which will not produce any environment output
--
--   Output:        Entries within Optim Process report showing
--                  specifics about the processing environment,
--                  tables and columns.
--
--   Returns:       0 or error which ends the process with no changes
--                  made.
--
------------------------------------------------------------------------
------------------------------------------------------------------------
--  cm_starttable function - Called at the start of processing for each
--  table
------------------------------------------------------------------------
function cm_starttable()
  optim.print("  *** Start of Process ***")
  -- Default scope is "ALL"
  ScopeLimit = "ALL"
  -- We have not built the message table yet
  MSGTableBuilt = false
  -- Table of valid scope values
  local ScopeTable = {["SOURCE"]=1,["TARGET"]=2,["ENVIRON"]=3,
    ["ALL"]=4,["NONE"]=5}
  if (ScopeTable[ScopeLimit] == nil) then
    Process_Msg("MSG001", true)
  elseif (ScopeLimit ~= "ALL") then
    Process_Msg("MSG009", false, ScopeLimit)
  else
    Process_Msg("MSG010", false)
  end
  Process_Msg("MSG003", false, optim.timestamp())
  if (ScopeLimit == "ENVIRON" or ScopeLimit == "ALL") then
    -- Capture environment attributes and options (when they
    -- become available)
    Process_Msg("MSG011", false, "Current Time         : ",
```

```
   optim.timestamp())
  Process_Msg("MSG011", false, "Active SQL ID        : ",
   optim.sqlid())
  Process_Msg("MSG011", false, "Active User ID       : ",
  optim.userid())
  Process_Msg("MSG011", false, "Source DB2 Subsystem : ",
  optim.source.getdbalias())
  Process_Msg("MSG011", false, "Target DB2 Subsystem : ",
  optim.target.getdbalias())
  Process_Msg("MSG011", false, "Optim Temp Directory : ",
  optim.app.TempDir())
  Process_Msg("MSG011", false, "Optim Data Directory : ",
  optim.app.DataDir())
  Process_Msg("MSG011", false, "Optim Script Name    : ",
  optim.app.Script())
  Process_Msg("MSG011", false, "Optim CompanyName    : ",
  optim.app.CompanyName())
  Process_Msg("MSG011", false, "Optim Release        : ",
  optim.app.OptimRelease())
  Process_Msg("MSG011", false, "Optim Build          : ",
  optim.app.OptimBuild())
  Process_Msg("MSG011", false, "Optim Error          : ",
  optim.app.Error())
  Process_Msg("MSG011", false, "Instance             : ",
  optim.app.Instance())
  Process_Msg("MSG011", false, "ThreadId             : ",
  optim.app.ThreadId())
  Process_Msg("MSG011", false, "ThreadHandle         : ",
  optim.app.ThreadHandle())
  Process_Msg("MSG011", false, "Operating Sys        : ",
  optim.app.Platform())
  Process_Msg("MSG011", false, "Operating SysRel     : ",
  optim.app.OpSysRelease())
  Process_Msg("MSG011", false, "Operating SysBuild   : ",
  optim.app.OpSysBuild())
  Process_Msg("MSG011", false, "Operating ServPak    : ",
  optim.app.OpSysCSD())
  Process_Msg("MSG011", false, "Server UserId        : ",
  optim.app.ServerUserId())
  Process_Msg("MSG011", false, "Computer Name        : ",
  optim.app.ComputerName())
end
-- If requested get source table metadata
if (ScopeLimit == "SOURCE" or ScopeLimit == "ALL")then
  --Identify source creator, table name
  Process_Msg("MSG005", false, optim.source.getcreatorid(),
           optim.source.gettablename())
  --Identify Source columns
  for i = 1, optim.source.getnumcolumns(), 1 do
    local CurSrcColName =optim.source.getcolumnname(i)
    Process_Msg("MSG007", false, i, CurSrcColName,
                optim.source.getcolumntype(CurSrcColName),
                optim.source.getcolumnlength(CurSrcColName))
  end
end
-- If requested get target table metadata
if (ScopeLimit == "TARGET" or ScopeLimit == "ALL")then
  --Identify target creator, table name
  Process_Msg("MSG006", false, optim.target.getcreatorid(),
           optim.target.gettablename())
  --Identify target columns
  for i = 1, optim.target.getnumcolumns(), 1 do
    local CurTgtColName =optim.source.getcolumnname(i)
    if (optim.target.iscolumnnullable(CurTgtColName)) then
      NullAble = "YES"
    else
      NullAble = "NO"
```

```
      end
      Process_Msg("MSG008", false, i, CurTgtColName,
                  optim.target.getcolumntype(CurTgtColName),
                  optim.target.getcolumnlength(CurTgtColName),
                  NullAble)
    end
  end
end
--------------------------------------------------------------------------
--  cm_transform function - Called for each row processed
--------------------------------------------------------------------------
function cm_transform()
  optim.target.setcolumnvalue(optim.source.getcolumnvalue())
end
--------------------------------------------------------------------------
--  cm_endtable function - Called at the end of processing for each
--  table
--------------------------------------------------------------------------
function cm_endtable()
  Process_Msg("MSG004", false, optim.timestamp())
end
--------------------------------------------------------------------------
--  Process_Msg function - Called when a message is requested to be
--  issued.
--  The message is written to the Optim process report.
--  1st argument is the message identifier indicating what message to
--  issue.
--  2nd argument is the boolean value 'TRUE' or 'FALSE' to indicate if
--  the process should be aborted.
--  3rd through nth arguments define any substitution values to be used
--  within the message text that require a value.
--  Example: Process_Msg("MSG001",true,"Optim","Distributed",11)
--  requests that message "MSG001" is searched for in MsgTable and
--  issued with the values "Optim", "Distributed" and 11 being
--  substituted for the '&S' literals respectively and the process is
--  to be aborted.
--------------------------------------------------------------------------
function Process_Msg(...)
  local IntEMsg
  -- If not done already, build the message table
  if (MSGTableBuilt == false) then
    MSGTableBuilt = true
    -- MsgTable contains the text of the messages, values '&S' will be
    -- replaced with a value passed by the caller
    MsgTable = {
      ["MSG001"] =
        "Scope Value Not ENVIRON, SOURCE, TARGET, ALL or NONE",
      ["MSG002"] =       "Only [SOURCE|TARGET|ENVIRON|ALL] Allowed",
      ["MSG003"] =
        "***  Start of Table Processing &S ***",
      ["MSG004"] =
        "***  End of Table Processing &S ***",
      ["MSG005"] =
        "* Source Table: &S.&S *",
      ["MSG006"] =
        "* Destination Table: &S.&S *",
      ["MSG007"] =
        "Source Column &S:      &S &S(&S)",
      ["MSG008"] =
        "Destination Column &S: &S &S(&S), NULLABLE(&S)",
      ["MSG009"] =
        "Scope Limited to: &S",
      ["MSG010"] =
        "Scope: Unlimited",
      ["MSG011"] =
        "&S &S"
    }
```

```
  end
  -- If called without minimum arguments, syntax error
  if (select("#",...) < 2) then
    IntEMsg = "  Minimum 'Process_Msg' Syntax is "..
              "'Process_Msg(MSGID,true|false)'\n"
    WriteRec(IntEmsg)
  -- If the requested message is not in the table, inform caller
  -- and return
  elseif (MsgTable[select(1,...)] == nil) then
    IntEMsg = "  Requested Message, " .. select(1,...) ..
              " Not Found In Message Table\n"
    WriteRec(IntEmsg)
  -- Issue the message using the text from the table and any
  -- substitution values supplied
  else
    if (select("#",...) == 2) then  -- No substitutions
      WriteRec("  " .. MsgTable[select(1,...)] .. "\n")
    else  -- Some substitutions
      -- Run through argument list to replace '&S' entries with
      -- requested value(s) passed
      -- Copy message text from the table
      local Msg = MsgTable[select(1,...)]
      for i = 1, select("#",...) - 2, 1 do
        -- Replace '&S' with passed argument
        Msg = string.gsub(Msg, "&S", select(i+2,...), 1)
      end
      -- Issue message with substitutions passed to this function
      WriteRec("  " .. Msg .. "\n")
    end
    -- ABORT process after issuing msg?
    if (select(2,...) == true) then
      error("  *** Aborting Process as Requested by User ***\n",2)
    end
  end
end
--------------------------------------------------------------------------
--  WriteRec function - Called to write output to the process report
--------------------------------------------------------------------------
function WriteRec(Record)
  optim.print(Record)
end
```

## Limitations of using Lua functions in column map expressions and procedures

Column map expressions and procedures do not support certain data types and Lua functions.

### Unsupported data types

The following data types are not supported in column map expressions and procedures:

- BINARY/VARBINARY
- XML
- CLOB/BLOB, DBCLOB
- ROWID

### Numeric data processed in double-precision format

Column map expressions and procedures process numeric data in double-precision format. Use the `optim.source.getcolumnasdouble()` function to get data from columns that have the following data types. A runtime error is generated if you use `optim.source.getcolumnvalue()` to get data from columns that have the following data types.

- BIGINT
- DECIMAL

- DOUBLE
- FLOAT
- DECIMAL FLOAT
- NUMBER

### Encoding

Optim for z/OS uses native data encoding in internal processing of column map expressions and procedures. This encoding reflects the CCSID of the Optim DB2 plan. Currently an Optim DB2 plan may only be bound with EBCDIC encoding.

### Unsupported functions

The following categories of functions are not supported in column map expressions and procedures.

- Functions for loading and building modules in the Lua `package` library:
  - `module`
  - `require`
  - `package.cpath`
  - `package.loaded`
  - `package.loaders`
  - `package.loadlib`
  - `package.path`
  - `package.preload`
  - `package.seeall`
- `string.dump()`

## Guidelines for using Lua functions in column map expressions and procedures

The following guidelines will help you use Lua functions to define a column map expression or procedure.

- A column map expression or procedure may include a single Lua statement setting the column value, or multiple Lua statements, grouped into a Lua chunk.
- Change the caps mode to CAPS OFF in the COLUMN-MAP-EXPRESSION-TEXT or COLUMN-MAP-PROCEDURE-TEXT ISPF edit session. The Lua scripting language is case-sensitive.
- If you are copying Lua statements from a dataset, note that the COLUMN-MAP-EXPRESSION-TEXT and COLUMN-MAP-PROCEDURE-TEXT panels enforce text margins of 1 and 72.
- To preserve the column map expression or procedure you entered, save the Column Map as a named object.

## Defining a column map expression using Lua functions

You can define a column map expression on the Define Column Map or Modify Column Map panel.

1. On the Define Column Map or Modify Column Map panel, enter **EXP** next to the **Source Column** for which you want to define the column map expression using Lua functions. Press **ENTER**.
2. In the COLUMN-MAP-EXPRESSION-TEXT ISPF edit session, enter the column map expression text, starting with `--Lua` on the first line. This convention differentiates the Lua expression from an Optim expression. See “Lua functions for column map expressions and procedures” on page 184 for a list of the Optim Lua functions you can use in the expression text. Also refer to “Guidelines for using Lua functions in column map expressions and procedures.”
3. Enter the **END** command to return to the Define Column Map or Modify Column Map panel. The **Source Column** value is replaced with the beginning of the column map expression text. The **Status** column displays `LuaPROC`.

### Defining a local column map procedure using Lua functions

You can define a local column map procedure on the Define Column Map or Modify Column Map panel.

1. On the Define Column Map or Modify Column Map panel, enter **PROC LOCAL** in place of the **Source Column** value for which you want to define the column map procedure. Use the following syntax for the appropriate source column: `PROC LOCAL [ ( ['parm1' [,'parmn'] ] ) ]`. You can specify up to eight optional parameter values. Each parameter must be enclosed in single quotes. The parameter values are case-sensitive. Enclose the parameters in parentheses. For example, enter `PROC LOCAL ('parm1','parm2')`. Press **ENTER**.
2. In the COLUMN-MAP-PROCEDURE-TEXT ISPF edit session, enter the column map procedure text, including any parameters. You can enter up to 2000 characters. See "Lua functions for column map expressions and procedures" on page 184 for a list of the Optim Lua functions you can use to define the column map procedure. Also refer to "Guidelines for using Lua functions in column map expressions and procedures" on page 192.
3. Enter the **END** command to return to the Define Column Map or Modify Column Map panel. The **Status** for the **Source Column** value displays `LuaPROC`.
4. To add column map procedure text or parameters that did not fit in the COLUMN-MAP-PROCEDURE-TEXT ISPF editor, enter **EXP** in the **Cmd** column next to the **Source Column** value for which you defined the local column map procedure.
5. In the COLUMN-MAP-EXPRESSION-TEXT ISPF edit session, enter the remaining column map procedure text and parameters.
6. Enter **END** to return to the Define Column Map or Modify Column Map panel.

## Data Privacy functions

Data Privacy functions provide various methods to transform or mask sensitive data. They require an Optim Data Privacy License.

### LOOKUP function

The LOOKUP function obtains the value for a destination column from a DB2 table, the lookup table, according to the value in the source column. Use the LOOKUP function to translate a source value to a corresponding look-up value, which is placed in the destination.

There are two forms of the LOOKUP function, single column and multiple column. The single column form inserts a value into a single destination column. The multiple column form inserts values from multiple lookup table columns into corresponding destination columns, which are based on a single source column value. You can use the LOOKUP function alone or in combination with other Column Map functions to achieve the appropriate data masking results.

You can enter the multiple column LOOKUP function for any source column that will be replaced by a lookup table value, but you must edit the Column Map to remove the names of remaining source columns that also will be replaced.

Use the IGNORE parameter to ignore the lookup table and use a source value when a row in a specified source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR].

Use the PRESERVE parameter to ignore the lookup table and use a source value when a source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR]. If the lookup table does not contain a value for a source column, PRESERVE=NOT_FOUND inserts the source column value at the destination.

The LOOKUP function uses the following syntax:

```
LOOKUP( [sourcecol,...] [dest=(col1,coln,...),]
        lktablename(search,{value|values=(col1,coln,...) } )
        [cache|nocache]
        [ignore=(colname(spaces,null,zero_len),...]
        [PRESERVE=([NOT_FOUND]|colname(spaces,null,zero_len),...)])
```

*sourcecol*
Name of the source table column that contains the search value (optional). If not specified, the name of the destination column is used.

**Note:** The *source* column must be compatible with the *search* column.

**dest=** Names of the destination table columns in which values from the lookup table are inserted. Required for multiple column lookup.

*col1*, *coln*, **...**
Destination table column names. The order of the column names must correspond to the lookup table columns in the **source=** parameter.

*lktablename*
Name of the lookup table. You can specify the DB2 table name as `creatorid.tablename` or `tablename`. If you do not fully qualify the table name, the qualifiers for the destination table are used.

*search* Name of the column in the lookup table that contains a value to match against the search value from the source column.

*value* Name of the column in the lookup table that contains the translated search value to be inserted at the destination. Required for single column lookup.

**Note:** The `value` column must be compatible with the destination column.

**values=**
Names of the columns in the lookup table that contain values to be inserted at the destination. Required for multiple column lookup.

*col1, coln*, **...**
Lookup table column names. The order of the column names must correspond to the destination table columns in the **dest=** parameter.

**cache | no_cache**
Specify CACHE (default) to maintain a table of found lookup values in memory or NO_CACHE to discard found values. Using CACHE is faster when you are retrieving a value many times, but requires extra memory.

**ignore=**
List of source columns with values that are inserted at the destination instead of the lookup value when the column has a row with a stated value (NULL, SPACES, or zero-length VARCHAR).

*col* The source column name. For single column lookup, enter one column name only. For multiple column lookup, the order of the column names must correspond to the destination table columns in the **dest=** parameter. The number of columns must equal the columns in the **dest=** parameter, and at least one column must include values. To not specify values for a column, do not enter a value. For example, `coln()`.

**null** Ignore the lookup table if the source column row has a NULL value.

**spaces** Ignore the lookup table if the source column row has a SPACES value. For CHAR columns only.

**zero_len**
Ignore the lookup table if the source column row has a zero-length VARCHAR value.

**PRESERVE=**

List of source column values (NULL, SPACES, or zero-length VARCHAR) to insert at the destination instead of the lookup value. Use PRESERVE=NOT_FOUND to insert the source column value at the destination if the lookup table does not contain a value.

**NOT_FOUND**

Insert the source column value at the destination, if no match is found in the lookup table.

*colname*

A column for which specified source values are inserted at the destination. The source values are as follows:

null

spaces

zero-length varchar

**Note:** `Preserve=` and `ignore=` are mutually exclusive. `Ignore=` will be deprecated in a future release. The col, null, spaces, and zero-length varchar operands have the same effect when used with either `preserve=` or `ignore=`.

## Single Column Example

Use the LOOKUP function to translate the source value in a lookup table to a corresponding value in another table.

For example, assume the source column, STATE, contains state abbreviations (such as `NJ`) and the destination column is to contain the complete state name (in this example, `New Jersey`). A lookup table that is named `STATE_LOOKUP` contains a column (`CODE`) for state abbreviations or codes and a column (`NAME`) for the corresponding names.

To obtain the value for the destination column by using the `STATE_LOOKUP` table, specify:

```
LOOKUP(STATE,STATE_LOOKUP(CODE,NAME))
```

The LOOKUP function searches for a value in the CODE column of the `STATE_LOOKUP` table that matches the value (`NJ`) in the source table STATE column. When a match is found, the function inserts the corresponding value from the NAME column (`New Jersey`) in the destination column.

## Multiple Column Example

Use the LOOKUP function to insert values from columns in a lookup table row into columns in a destination table row, which is based on a value in a source column.

Here is an example. Based on a source column (`SOC_SEC`) that contains social security numbers, you can replace values in destination columns (`FIRST_NAME` and `LAST_NAME`) with first and last names from a lookup table. A table that is named `NAME_LOOKUP` contains a column (`SSN`) with the social security numbers from the source table. That table also contains columns (`FIRST_MASK` and `LAST_MASK`) to mask corresponding names in the destination.

To replace names in the destination table that are based on a social security number, specify:

```
LOOKUP(SOC_SEC,DEST=(FIRST_NAME,LAST_NAME),NAME_LOOKUP(SSN,VALUES=(FIRST_MASK, LAST_MASK)))
```

The LOOKUP function searches for a value in the `SSN` column of the `NAME_LOOKUP` table that matches the value in the source table `SOC_SEC` column. When a match is found, the function inserts the corresponding

values from the lookup table FIRST_MASK and LAST_MASK columns into the corresponding destination columns.

### Ignore Example

Use the following statement to extend the Single Column Example, where you want to use the source NULL and SPACES values instead of lookup table values:

```
LOOKUP(STATE,STATE_LOOKUP(CODE,NAME),IGNORE=(STATE(NULL,SPACES)))
```

### No_Cache Example

Use the following statement to extend the Single Column Example, where you do not want to maintain a table of found lookup values in memory:

```
LOOKUP(STATE,STATE_LOOKUP(CODE,NAME),NO_CACHE)
```

## Random LOOKUP function

The Random LOOKUP function selects a value at random from a specified lookup table to insert in a destination column. The function generates a random number between 1 and the limit or number of rows in the lookup table to use as a subscript into the table. The column value or values from the row that correspond to the subscript are inserted in the destination column.

The Random LOOKUP function is not supported for DECFLOAT data type.

There are two forms of the Random LOOKUP function, single column and multiple column. The single column form inserts a value into a single destination column. The multiple column form inserts values from multiple lookup table columns into corresponding destination columns.

You can enter the multiple column Random LOOKUP function for any source column that will be replaced by a lookup table value, but you must edit the Column Map to remove the names of remaining source columns that also will be replaced.

use the IGNORE parameter to ignore the lookup table and use a source value when a row in a specified source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR].

Use the PRESERVE parameter to ignore the lookup table and use a source value when a source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR]. If the lookup table does not contain a value for a source column, PRESERVE=NOT_FOUND inserts the source column value at the destination.

The Random LOOKUP function uses the following syntax:

```
RAND_LOOKUP(lktablename,{columnname | dest=(col1,coln,...),values=(col1,coln,...)}
           [limit] [ignore=(colname(spaces,null,zero_len),...)]
           [PRESERVE=([NOT_FOUND]|colname(spaces,null,zero_len),...)])
```

*lktablename*
: Name of the lookup table. You can specify the lookup table name as creatorid.tablename or tablename. If the table name is not fully qualified, destination table qualifiers are used.

*columnname*
: Name of the column in the lookup table that contains the values to be randomly selected for insertion at the destination. Required for single column lookup.

**dest=** Names of the destination table columns in which values from the lookup table are inserted. Required for multiple column lookup.

*col1, coln, ...*
Destination table column names. The order of the column names must correspond to the lookup table columns in the **values=** parameter.

**values=**
Names of the columns in the lookup table that contain values to be inserted at the destination. Required for multiple column lookup.

*col1, coln, ...*
Lookup table column names. The order of the column names must correspond to the destination table columns in the **dest=** parameter.

**limit** Optional limit on number of rows from the lookup table that is used to select column values. Specify an integer, up to a maximum value of 2,000,000,000. If no limit is specified, all rows are used.

**Note:** You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

A table of column values is generated in memory. The size of this table might be limited by system resources.

**ignore=**
List of source columns with values that are inserted at the destination instead of the lookup value when the column has a row with a stated value (NULL, SPACES, or zero-length VARCHAR).

*col* The source column name. For single column lookup, enter one column name only. For multiple column lookup, the order of the column names must correspond to the destination table columns in the **dest=** parameter. The number of columns must equal the columns in the **dest=** parameter, and at least one column must include values. To not specify values for a column, do not enter a value. For example, `coln()`.

**null** Ignore the lookup table if the source column row has a NULL value.

**spaces** Ignore the lookup table if the source column row has a SPACES value. For CHAR columns only.

**zero_len**
Ignore the lookup table if the source column row has a zero-length VARCHAR value.

**PRESERVE=**
List of source column values (NULL, SPACES, or zero-length VARCHAR) to insert at the destination instead of the lookup value. Use PRESERVE=NOT_FOUND to insert the source column value at the destination if the lookup table does not contain a value.

**NOT_FOUND**
Insert the source column value at the destination, if no match is found in the lookup table.

*colname*
A column for which specified source values are inserted at the destination. The source values are as follows:

null

spaces

zero-length varchar

**Note:** `Preserve=` and `ignore=` are mutually exclusive. `Ignore=` will be deprecated in a future release. The col, null, spaces, and zero-length varchar operands have the same effect when used with either `preserve=` or `ignore=`.

### Single Column Example

To select a value at random from the STATE column in the first 50 rows of a table that is named STATE_LOOKUP and insert it in the destination column, specify:

```
RAND_LOOKUP(STATE_LOOKUP,STATE,50)
```

### Multiple Column Example

To select values from the CITY, STATE, and ZIP columns in a random row of a table that is named STATE_LOOKUP and insert them in the corresponding destination columns, specify:

```
RAND_LOOKUP(STATE_LOOKUP,DEST=(CITY,STATE,ZIP),VALUES=(CITY,STATE,ZIP))
```

### Ignore Example

Use the following statement to extend the Single Column Example, where the source column is named STATES and you want to use the source NULL and SPACES values instead of lookup table values:

```
RAND_LOOKUP(STATE_LOOKUP,STATE,50,IGNORE=(STATES(NULL,SPACES)))
```

## Hash Lookup function

The Hash LOOKUP function obtains the value for a destination column from a lookup table, according to a hashed value derived from a source column. Use the Hash LOOKUP function to consistently mask data when you use the same source and lookup tables in any environment. The source column that is hashed does not need to be a column that will be replaced by lookup table values.

There are two forms of the Hash LOOKUP function, single column and multiple column. The single column form inserts a value into a single destination column. The multiple column form inserts values from multiple lookup table columns into corresponding destination columns, based on a single hash value from a source column.

You can enter the multiple column Hash LOOKUP function for any source column that will be replaced by lookup table values, but you must edit the Column Map to remove the names of remaining source columns that also will be replaced.

The lookup table must include a key column that contains sequential number values without any gaps, and the remaining columns contain replacement values. The key column must be a numeric data type. The lookup table is typically indexed. The function hashes a source column to derive sequential numbers from 1 to the maximum value in the key column of the lookup table. The hashed value from the source table is matched with the sequential numbers in the lookup table, and values from the corresponding lookup table row are inserted at the destination.

The function assigns NULL, SPACES (for CHAR columns), and zero-length VARCHAR values to the numbers -1 (for NULL), -2 (for SPACES), and -3 (for zero-length VARCHAR). The lookup table should include a row for each of these numbers, which will allow you to insert a lookup value for each of these source values. If one of these source values is found and a corresponding number is not in the lookup table, a conversion error is reported.

Use the IGNORE parameter to ignore the lookup table and use a source value when a row in a specified source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR)].

Use the PRESERVE parameter to ignore the lookup table and use a source value when a source column contains a specified value [NULL, SPACES (for CHAR columns), or zero-length VARCHAR]. If the lookup table does not contain a value for a source column, PRESERVE=NOT_FOUND inserts the source column value at the destination.

You can use the TRIM parameter to specify characters that will be trimmed from the source value before it is hashed. For example, to trim commas from a source value, the values Smith, John, and Smith John will each be hashed to the same value. You can also use this parameter to convert the source value to uppercase before it is hashed. If the source value is converted to uppercase, the trim characters are also converted to uppercase.

**Note:** Before Optim 11.3, the HASH_LOOKUP function automatically trimmed trailing blanks from columns of fixed-length CHARACTER and GRAPHIC data when the length of the column exceeded 14 characters. Beginning in Optim 11.3, you must use the **\r** parameter to trim trailing blanks.

You can use the SEED parameter to vary the calculation that is performed by the hashing algorithm. The hashed value from the source column and the SEED value are matched with a sequential number from the lookup table to obtain the replacement value for the destination column.

Here is the syntax:

```
HASH_LOOKUP( [sourcecol,...]
             [trim=([char1char2...][\u][\r])]
             [dest=(col1,coln )]
             lktablename(search,{value |values=(col1,coln,...)})
             [cache|nocache]
             [ignore=(colname(spaces,null,zero_len),...)] |
             [PRESERVE=([NOT_FOUND]|colname(spaces,null,zero_len),...)]
             [seed=n])
```

*sourcecol*
: Name of the source table column from which hashed values are derived (optional). If not specified, the name of the destination column is used.

**dest=** Names of the destination table columns in which values from the lookup table are inserted. Required for multiple column lookup.

*col1, coln, ...*
: Destination table column names. The order of the column names must correspond to the lookup table columns in the **values=** parameter.

**trim=** List of characters that are trimmed from the source value before it is hashed. For Single-Byte (SBCS) characters, you can also use trim to convert the source value to uppercase before it is hashed.

*char1char2...*
: Characters that are trimmed from the source value before it is hashed. For example, if you are trimming commas from the source, the values Smith, John, and Smith John hash to the same value. The list is case sensitive. You can specify a space or comma as a character. After the initial occurrence of a character, any additional occurrences in the list are ignored. If the value is NULL or all spaces after characters are trimmed, the source value is not hashed and is assigned the appropriate reserved value (-1 or -2).

  To specify a backslash, \, or a right parentheses, ), you must precede the character with a backslash escape character. For example, to specify a right parenthesis, enter: trim=(\)). You can use the escape character only with a backslash, a right parenthesis, the uppercase indicator, or as part of the indicator to remove trailing blanks (\r).

  **Note:** Restriction on using backslash escape character and Japanese character sets: Some Japanese character sets use a code page with a hexadecimal value for the backslash escape character that is different from the hexadecimal value required by Optim. Optim

requires hexadecimal value E0 for the backslash escape character. If you are using a code page that represents the backslash escape character with a hexadecimal value other than E0, use the ISPF editor command HEX ON and modify the value to E0 for it to be processed correctly.

**\u** Converts the source value to uppercase before it is hashed. Any trimmed characters are also converted to uppercase. Upper and lowercase letters hash differently. For example, John and JOHN hash to different values. If you want upper and lowercase values to hash to the same value, use trim=(\u) to convert the source value to uppercase before it is hashed.

**\r** Trims all trailing blanks from the source value before it is hashed. This parameter is not supported for VARCHAR and VARGRAPHIC characters.

*lktablename*
Name of the lookup table. You can specify the lookup table name as `creatorid.tablename` or `tablename`. If you do not fully qualify the table name, the qualifiers for the destination table are used.

*search* Name of the column in the lookup table that contains sequential values to match against the hash values from the source column.

*value* Name of the column in the lookup table that contains the translated search value to be inserted at the destination. Required for single column lookup.

*values=*
Names of the columns in the lookup table that contain values to be inserted at the destination. Required for multiple column lookup.

*col1, coln, ...*
Lookup table column names. The order of the column names must correspond to the destination table columns in the **dest=** parameter.

**cache | no_cache**
Specify CACHE to maintain a table of found lookup values in memory or NOCACHE to discard found values. Using CACHE is faster when you are retrieving a value many times, but requires extra memory.

**ignore=**
List of source columns with values that are inserted at the destination instead of the lookup value when the column has a row with a stated value (NULL, SPACES, or zero-length VARCHAR).

*col* The source column name. For single column lookup, enter one column name only. For multiple column lookup, the order of the column names must correspond to the destination table columns in the **dest=** parameter. The number of columns must equal the columns in the **dest=** parameter, and at least one column must include values. To not specify values for a column, do not enter a value. For example, `coln()`.

**null** Ignore the lookup table if the source column row has a NULL value.

**spaces** Ignore the lookup table if the source column row has a SPACES value. For CHAR columns only.

**zero_len**
Ignore the lookup table if the source column row has a zero-length VARCHAR value.

**PRESERVE=**
List of source column values (NULL, SPACES, or zero-length VARCHAR) to insert at the destination instead of the lookup value. Use PRESERVE=NOT_FOUND to insert the source column value at the destination if the lookup table does not contain a value.

**NOT_FOUND**
Insert the source column value at the destination, if no match is found in the lookup table.

*colname*
A column for which specified source values are inserted at the destination. The source values are as follows:

null

spaces

zero-length varchar

**Note:** `Preserve=` and `ignore=` are mutually exclusive. `Ignore=` will be deprecated in a future release. The col, null, spaces, and zero-length varchar operands have the same effect when used with either `preserve=` or `ignore=`.

**seed=** Use **seed=** to vary the hashing algorithm calculation. Specify 1 - 2,000,000,000. If you use a value of 0, the seed parameter is ignored.

### Single Column example

Use the Hash LOOKUP function to insert values from a column in a lookup table into a destination table column, based on a value that is hashed from a source column.

For example, assume the source column, `FIRST_NAME`, contains first names and the destination column includes replacement first names from the lookup table. A lookup table, `NAME_LOOKUP`, contains a column (`FIRST`) with first names and a column (`SEQ`) containing sequential values.

To obtain values for the destination column by using the `NAME_LOOKUP` table, specify:

```
HASH_LOOKUP(FIRST_NAME,NAME_LOOKUP(SEQ,FIRST))
```

The Hash LOOKUP function matches the hash values from the source column with values in the `SEQ` column of the `NAME_LOOKUP` table. When a match is found, the function inserts the corresponding value from the `FIRST` column into the destination column.

### Multiple Column example

Use the Hash LOOKUP function to insert values from columns in a lookup table row into columns in a destination table row, based on a value that is hashed from a source column.

Here is an example. Based on values that are hashed from a source column (`FIRST_NAME`) that contains first names, you can replace values in destination columns (`FIRST` and `LAST`) with first and last names from a lookup table. A lookup table that is named `NAME_LOOKUP` contains a column (`SEQ`) with sequential values. That table also contains columns (`FIRST_MASK` and `LAST_MASK`) to mask values in the destination.

To replace names in the destination table based on a value hashed from a source column, specify:

```
HASH_LOOKUP(FIRST_NAME,DEST=(FIRST,LAST),NAME_LOOKUP(SEQ,VALUES=(FIRST_MASK,LAST_MASK)))
```

The Hash LOOKUP function matches the hash values from the source `FIRST_NAME` column with values in the `SEQ` column of the `NAME_LOOKUP` table. When a match is found, the function inserts the corresponding values from the lookup table `FIRST_MASK` and `LAST_MASK` columns into the corresponding destination columns.

### Ignore example

Use the following statement to extend the Single Column Example, where you want to use the source NULL and SPACES values instead of lookup table values:

```
HASH_LOOKUP(FIRST_NAME,NAME_LOOKUP(SEQ,FIRST),IGNORE=(FIRST_NAME(NULL,SPACES)))
```

### No_Cache example

Use the following statement to extend the Single Column Example, where you do not want to maintain a table of found lookup values in memory:

```
HASH_LOOKUP(FIRST_NAME,NAME_LOOKUP(SEQ,FIRST),NO_CACHE)
```

### Trim example

Use the following statement to trim spaces and commas from the source value and convert the source value to uppercase before it is hashed:

```
HASH_LOOKUP(FIRST_NAME,TRIM=( ,\u),NAME_LOOKUP(SEQ,FIRST))
```

This statement trims spaces from the source value before hashing:

```
HASH_LOOKUP(FULL_NAME,TRIM=( ),NAME_LOOKUP(SEQ,FULLNAME))
```

## Move Age function

The Move AGE function:

- Processes character, numeric, DATE, or TIMESTAMP columns.
- Is not supported for LOB, XML, BINARY, or DECFLOAT columns.
- Ages to an explicit date or by explicit or relative values.
- Increments or decrements by days, weeks, months, years, or a combination of these units. Also, you can specify aging by a number of business units (such as payday, quarters, and so on).
- Includes parameters to automatically adjust aged data to comply with business rules.

When the Column Map is used in an Insert, Load, or Convert Process, you can specify

- Default values for source columns defined with AGE for which you have not provided all required values. For example, you can use one Column Map for multiple processes and specify unique aging amounts or business rules for each process without modifying the AGE function parameters on the Column Map.
- Aging values for all DATE and TIMESTAMP columns not explicitly defined.
- Whether aging is performed for an individual process.

### AGE

AGE increments dates based on various parameters. To specify parameters to age dates before inserting them at the destination, use the AGE line command. This command invokes the Aging Specifications panel, which allows you to provide values to generate the AGE function and its parameters. Assume that AGE was specified for the ORDER_SHIP_DATE column. Some **Aging Specifications** values for the column are included in the following figure.

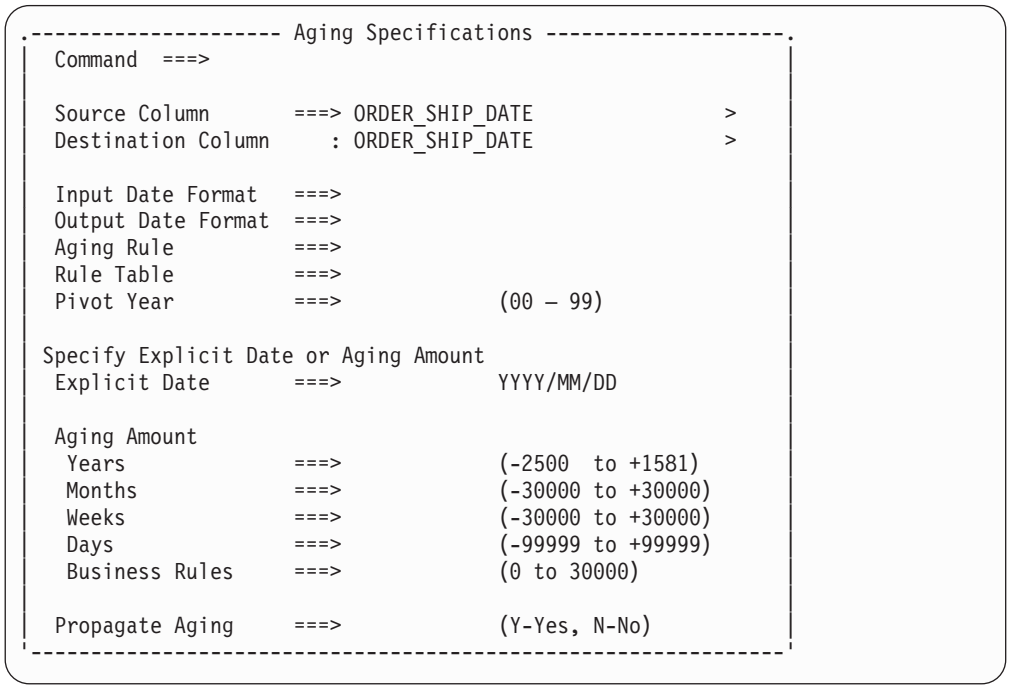



*Figure 105. Aging Specifications*

The `Input Date Format` defines a two-digit year, so the `Pivot Year` is included to determine the century. An `Output Date Format` is not specified in the figure, but you might specify a different format, such as MMDDYYYY, to adjust the input to a four-digit year output.

The other values, such as aging values and rules, are not specified. These values are to be supplied for each process when the Column Map is used.

## Aging Specifications panel

The Aging Specifications panel includes the following items:

**Source Column**
The name of the source column to age.

**Destination Column**
The name of the destination column, which you cannot modify.

**Input Date Format**
The date format of the source data. You can specify a distributed date format or a format that is defined by the site. The format must be valid for the column length and type.

Use an asterisk to display a selection list of formats valid for the column. This value is required. For more information about specifying the date format, see "Date Formats" on page 205.

**Output Date Format**
Specify the date format for the aged data. You can be specify a distributed date format or a format that is defined by the site. The format length must match the input format length.

Use an asterisk to display a selection list of formats valid for the column. If not specified, the `Input Data Format` is used. For more information about specifying the date format, see "Date Formats" on page 205.

**Aging Rule**
The name of the default aging rule to use. An aging rule is required only when you are aging by business units or if a rule table is specified. If an aging rule is not specified, the aging rule that is specified for the process is used.

Use an asterisk to display a selection list of rules in the current rule table. Specify NONE to do linear aging. An aging rule is not applied. For more information, see the *Customization Guide,* Customize Aging Rules for additional information about aging rules.

**Rule Table**
The name of the rule table that contains the specified aging rule. If not specified, the default rule table is used.

**Pivot Year**
The year that is used to determine whether a two-digit year value is handled as occurring in the 20th century (1900) or the 21st century (2000). If the year is greater than or equal to the pivot year, 1900 is used. If not specified, the pivot year that is defined for the process is used.

**Explicit Date**
An explicit date to use. The specified aging rule is applied to this date. The date must be in the form `YYYY/MM/DD` or `YYYY/DDD`.

## Aging Amount

**Years** The number of years to increment or decrement the date as a value in the range `-2500` to `+1581`.

**Months**
The number of months to increment or decrement the date as a value in the range `-30000` to `+30000`.

**Weeks** The number of weeks to increment or decrement the date as a value in the range `-30000` to `+30000`.

**Days** The number of days to increment or decrement the date as a value in the range `-99999` to `+99999`.

**Business Rules**
The number of occurrences of a business rule to adjust the date as a value in the range 0 - 30000. If you specify a value for `Business Rules`, you must specify an `Aging Rule`.

For example, if NEXTPAYDAY is the `Aging Rule`, the value in `Business Rules` is used to adjust the date by the specified number of paydays. Therefore, a 4 in `Business Rules` adjusts the date to the fourth payday after the date in the column. (The date to be adjusted is not included in the calculation.)

You can specify either a value for `Business Rules` or values for calendar units (`Years`, `Months`, `Weeks`, or `Days`), but not both.

**Propagate Aging**
An option to propagate the resulting age value. If you choose to propagate the value, `PROP` is displayed in `Data Type` for the source column; otherwise, this prompt is not displayed. For information about propagation, see "Propagating Primary and Foreign Key Values" on page 170.

The combined values for `Years`, `Months`, `Weeks`, and `Days`cannot result in an aging amount greater than 1581 years. If it does exceed that amount, an error results when the aging is attempted.

## DATE and TIMESTAMP Columns

You can direct Move to age values in DATE and TIMESTAMP columns by using the global specifications for the processor. However, if you want to age specific DATE and TIMESTAMP columns exclusively, age these columns to unique values, or use an exit, specify the AGE function for the column. These DB2 defined data types must have the format DB2DATE.

DATE and TIMESTAMP columns that are defined as nullable are skipped if they contain NULL. These columns are not aged and are included in the reported count of skipped columns. These skipped columns are handled the same as other skipped values according to the process specifications.

## Date Formats

Date formats specify the format of a date column. These are encoded internally as a list of possible data types in a single format table. New formats can be added easily to the table without requiring coding for each individual format.

Each date format is specified by its name. Specify these formats in the `Input Date Format` and `Output Date Format` prompts on the Aging Specifications panel. You can also display a selection list of values appropriate for the specific column data type by entering an asterisk in these prompts.

Valid names are determined by the data type of the column. Move can support a date format of MMDDYY as a character column and a decimal column, even though these are processed differently.

## Date Components

The formats that are distributed with Move are formatted with the following characters to represent the date component.

**CC** Two-digit century.

**YY** Two-digit year without century.

**YYY** Three-digit year relative to 1900.

**YYYY (or) CCYY**
Four-digit year.

**MM** Two-digit month.

**MMM** Three-character abbreviation for the month (such as Jan or JAN).

**DD** Two-digit day.

**DDD** Three-digit Julian day.

**DDDDDD**
Lilian date (that is, the number of days since Oct. 14, 1582, the date the Gregorian Calendar was adopted).

**/** Slash in date.

**-** Dash in date.

***** Any delimiter in date.

**U** Unsigned decimal. The letter U precedes the format.

## Date Format examples

Move supports a wide variety of internal date storage schemes. For instance, a `YYMMDD` date column might be stored in various ways:

- `CCYYMMDD`
- `CCYY*MM*DD`
- `CCYY/DDD`
- `DDMMYY`
- `DD*MM*YY`
- `DD/YY/MM`
- `DDMMMCCYY`
- `DDD-YY`
- `MMDDYY`

- MM*DD*YY
- MM/DD
- MMM*DD*CCYY
- MMM-YYYY
- YYDDD
- YYMM
- YY*DDD
- YY*MM
- YYMMDD
- YY*MM*DD
- YYYYDDMM

Character formats and numeric formats, as either packed decimal or binary columns, are allowed for any format without delimiters. The following are some examples of date formats.

### AGE specifications complete

After you define values for the AGE function on the Aging Specifications panel, use END to return to the Column Map editor. The AGE function is identified by the string AGE*(column)* where *column* is the name of the source column to be aged. You can modify these values by using the AGE line command again to redisplay the Aging Specifications panel. However, you must use the CLR line command to remove the entire AGE function specification or the SRC line command to replace the AGE function with the source column.

## Data Privacy Transformation Library Functions

The data privacy transformation library functions allow you to mask personal data such as social security numbers, credit card numbers, and e-mail addresses. A data privacy license, separate from the Optim product license, is needed to use these functions.

You can generate transformed data that is valid and unique. When a transformation library function is called, all destination and source columns are available except those destination columns that are yet to be assigned a value by an exit or transformation library function. The parameters in a transformation library function are validated at run time.

### TRANS SSN

Use the TRANS SSN function to generate a valid and unique U.S. Social Security Number (SSN). By default, TRANS SSN algorithmically generates a consistently altered destination SSN based on the source SSN. TRANS SSN can also generate a random SSN when the source data does not have an SSN value or when there is no need for transforming the source SSN in a consistent manner.

An SSN is made of 3 subfields. The first 3 digits (area) represent an area generally determined by the state in which the SSN is issued. The next 2 digits (group) define a group number corresponding to the area number. The last 4 digits (serial) are a sequential serial number. Regardless of the type of processing, default or random, TRANS SSN will generate an SSN with a group number appropriate to the area number.

The default processing method generates an SSN that includes the source area number as well as altered group and serial numbers based on the source SSN.

The random processing method generates an SSN that can include the source area number and uses a group number most recently issued by the Social Security Administration for the destination area number. Serial numbers begin with 0001 and are incremented by 1 for each additional SSN generated for

the area number. When the serial number exceeds 9999, the serial number will be reset to 0001 and the group number preceding the number most recently issued for the area number will be used.

The syntax of TRANS SSN is:

**TRANS SSN ['[=*flags*] [*sourcecol*]']**

**flags** You can specify one or more case-insensitive processing option flags.

**n** Generate a random SSN that is not based on a source value.

**i** Ignore invalid source values and copy them to the destination. Invalid values include the following:

- values less than 9 characters
- values with an area number not used by the Social Security Administration
- values containing non-numeric characters in CHAR or VARCHAR columns

For more information about invalid source values, see "Data Types Allowed" and "Skipped Rows" on page 208.

**m** Use the maximum group of all SSN area values, including values from 773 through 899, and excluding invalid area numbers.

**r** Generate a random SSN that includes the source area number.

**v** Validate the source group number by comparing it with numbers used by the Social Security Administration.

**-** The destination SSN should include dashes separating the fields (e.g., 123-45-6789). Requires a character-type destination column at least 11 characters long.

**sourcecol**
The source column name. If a source column name is not specified, the destination column name will be used. If a source column name is not specified and the destination column name does not match a column name in the source table, an error will occur during processing.

## Data Types Allowed

The following source and destination data types are permitted:

**BIGINT**
32 bits of the value used for the source and destination.

**CHAR**
The length of data in the column must be from 9 to 256 characters.

**DECIMAL**
The precision of the column must be 9 and the scale 0.

**INTEGER**
No restrictions.

**VARCHAR**
The length of data in the column must be from 9 to 254 characters.

If a source or destination column does not adhere to these restrictions, an error will occur during processing.

## Destination Processing Rules

The following rules apply to the destination SSN value, according to the destination data type or value:

**CHAR**

If the source value is 0, spaces, or a zero-length VARCHAR, the source value will be copied, unchanged, to the destination.

If a source value is 11 characters or more and includes embedded dashes (-), or if the '-' flag is specified, the destination value will include dashes if the destination column length is 11 characters or more.

**DECIMAL,**
**INTEGER**

If the source value is 0, spaces, or a zero-length VARCHAR, the destinations value will be 0.

**VARCHAR**

If the source value is 0, spaces, or a zero-length VARCHAR, the source value will be copied, unchanged, to the destination.

If a source value is 11 characters or more and includes embedded dashes (-), or if the '-' flag is specified, the destination value will include dashes if the destination column length is 11 characters or more.

**NULL** If the source value is NULL, the destination value will be NULL.

## Skipped Rows

The following conditions may cause a source row to be skipped and not written to the destination:

- The source value is NULL, and the destination column does not allow a NULL value.
- The source column is CHAR or VARCHAR, and the source value is less than 9 characters, contains a non-numeric character (other than dashes between the 3 subfields), or is too large.
- The source area number has not been used by the Social Security Administration.
- The source group number has not been used with the area number by the Social Security Administration (only if the 'v' flag has been specified).
- The source serial number is 0000, or the SSN is a reserved value not issued by the Social Security Administration (e.g., 078-05-1120).
- The source value cannot be converted to a format TRANS SSN supports.

## Error Messages

The following error messages may be issued:

**SSN01**

Parm on Col ccccc ("ppp") is invalid

The parameter *ppp*, specified on destination column *ccccc*, is not valid. Check the processing option flags specified.

**SSN02**

Col ccccc not on source

The column name entered as the sourcecol parameter or the destination column name (if sourcecol was omitted) was not found on the source table.

**SSN03**

Source Col ccccc-aaa invalid

The format of the source column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Type, Length, Precision, or Scale.

**SSN04**

Dest Col ccccc-aaa invalid

The format of the destination column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Type, Length, Precision, or Scale.

**SSN05**

Get col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS SSN tried to get the value of source column *ccccc*.

**SSN08**

Put col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS SSN tried to set the value of destination column *ccccc*.

If any other errors occur, contact Optim Technical Support.

### Example 1

The following example uses a source column name that matches the destination column and generates a random SSN that is not based on the source value:

**TRANS SSN '=n'**

### Example 2

The following example uses a source column name (SOCIAL) that differs from the destination column and generates an SSN using the default processing method and including dashes:

**TRANS SSN '=- SOCIAL'**

## TRANS CCN

Use the TRANS CCN function to generate a valid and unique credit card number (CCN). By default, TRANS CCN algorithmically generates a consistently altered CCN based on the source CCN. TRANS CCN can also generate a random value when the source data does not have a CCN value or when there is no need for transforming the source CCN in a consistent manner.

A CCN, as defined by ISO 7812, consists of a 6-digit issuer identifier followed by a variable length account number and a single check digit as the final number. The check digit verifies the accuracy of the CCN and is generated by passing the issuer identifier and account numbers through the Luhn algorithm. The maximum length of a CCN is 19 digits.

The default processing method generates a CCN by including the first 4 digits of the issuer identifier from the source CCN and altering the remaining 2 digits of the issuer identifier number and the account number based on the source CCN. A valid check digit is also assigned.

The random processing method generates a CCN that can include the first 4 digits of the source issuer identifier number or an issuer identifier number assigned to American Express®, Discover, MasterCard, or VISA. A valid check digit is also assigned. If the first four digits of a source issuer identifier number are included, the first account number based on those digits will begin with 1, and for each additional CCN that uses those digits, the account number will be incremented by 1.

The syntax of TRANS CCN is:

**TRANS CCN [ ('[=*flag*] [*sourcecol*] [ preserve=invalid ] ' ) ]**

*flag* Specify an option flag to generate a random CCN.

**n** Generate a random CCN that is not based on a source value and includes an issuer identifier number assigned to American Express, Discover, MasterCard, or VISA.

**r** Generate a random CCN that includes the first 4 digits of the source issuer identifier number.

*sourcecol*
The source column name. If a source column name is not specified, the destination column name is used.

If a source column name is not specified and the destination column name does not match a column name in the source table, an error will occur during processing.

**preserve=invalid**
If the source column contains an invalid CCN, do not replace it with a generated value. The source column value will be used in the destination column.

## Data Types Allowed

The following source and destination data types are permitted:

**CHAR**
The column length must be from 13 to 256 characters.

**VARCHAR**
The column length must be from 13 to 254 characters.

**PACKED DECIMAL**
The precision of the column must be from 13 to 254 and the scale must be zero.

If a source or destination column does not adhere to these restrictions, an error message occurs.

## Destination Processing Rules

The following rules apply to the destination CCN value, according to the destination data type or value:

**CHAR**
If the source value is spaces or a zero-length VARCHAR, the destination value will be set to spaces.

**VARCHAR**
If the source value is spaces or a zero-length VARCHAR, the destination length will be 0.

**PACKED DECIMAL**
If the source value is zero, the destination value will be zero.

**NULL** If the source value is NULL, the destination value will be NULL.

## Skipped Rows

The following conditions may cause a source row to be skipped and not written to the destination:

- The source value is NULL, and the destination column does not allow a NULL value.
- The source value is less than 13 characters, contains a non-numeric character, is too large, or has an incorrect check digit.
- The source value length is not valid for the credit card issuer.
- The source value cannot be converted to a format TRANS CCN supports.

## Error Messages

The following error messages may be issued:

**CCN01**

Parm on Col ccccc ("ppp") is invalid

The parameter *ppp*, specified on destination column *ccccc*, is not valid. Check the processing option flags specified.

**CCN02**

Col ccccc not on source

The column name entered as the sourcecol parameter or the destination column name (if sourcecol was omitted) was not found on the source table.

**CCN03**

Source Col ccccc-aaa invalid

The format of the source column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Type or Length.

**CCN04**

Dest Col ccccc-aaa invalid

The format of the destination column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Type or Length.

**CCN05**

Get col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS CCN tried to get the value of source column *ccccc*.

**CCN08**

Put col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS CCN tried to set the value of destination column *ccccc*.

If any other errors occur, contact Optim Technical Support.

### Example 1

The following example uses a source column name (CREDIT) that differs from the destination column and generates a random CCN not based on the source value:

**TRANS CCN '=n CREDIT'**

### Example 2

The following example uses a source column name (CREDIT) that differs from the destination column and generates a CCN using the default processing method:

**TRANS CCN 'CREDIT'**

## TRANS EML

Use the TRANS EML function to generate an e-mail address. An e-mail address consists of two parts, a user name followed by a domain name, separated by '@'. For example, user@domain.com.

TRANS EML generates an e-mail address with a user name based on either destination data or a literal concatenated with a sequential number. The domain name can be based on an e-mail address in the source data, a literal, or randomly selected from a list of large e-mail service providers. The e□mail address can also be converted to uppercase or lowercase.

TRANS EML can generate a user name based on the values in one or two destination table columns (usually containing the name(s) of a user). Processing options allow you to use only the first character of the value in the first column (e.g., the initial letter of a first name) and separate the values from both columns using either a period or an underscore.

If the user name is based on a single destination column value or a literal, the name will be concatenated with a sequential number. If a user name is based on values in two destination table columns and a separating period or underscore is not used, the values are concatenated. If a parameter is not provided for the user name, the name will be formed by the literal "email" concatenated with a sequential number. Sequential numbers for user names are suffixes that begin with 1 and are incremented by 1.

The syntax of TRANS EML is:

**TRANS EML ['[=*flags*] ,[{*sourcecol* | "domain" | , }**
**[{*name1col*[*name2col*] | *"userpfx"*}] ]']**

**flags** You can specify one or more case-insensitive processing option flags.

- **n** Generate a random domain name from a list of large e-mail service providers.
- **.** Separate the *name1col* and *name2col* values with a period.
- **_** Separate the *name1col* and *name2col* values with an underscore.
- **i** Use only the first character of the *name1col* value.
- **l** Convert the e-mail address to lowercase.
- **u** Convert the e-mail address to uppercase.

**sourcecol**
The source column name with e-mail addresses used to provide the domain name.

If neither the 'n' flag nor the *domain* parameter are defined, the domain name in the source column is used. (If *sourcecol* is not defined, the source column name is based on the destination column name.)

If a source column name is not specified and the destination column name does not match a column name in the source table, an error will occur during processing.

**domain**
A literal, up to 31 characters, that forms the domain name.

**,** A comma is required if neither a *sourcecol* nor a *domain* parameter are defined and you define either a literal or column name(s) for the domain name.

**name1col**
A destination table column name with values used to form the first (or only) part of the user name.

**name2col**
A destination table column name with values used to form the second part of the user name.

**userpfx**
A literal, up to 31 characters, that is concatenated with a sequential number to form the user name.

## Data Types Allowed

The following source and destination data types are permitted:

**CHAR**
The column length must be from 3 to 256 characters.

**VARCHAR**
The column length must be from 3 to 254 characters.

If a source or destination column does not adhere to these restrictions, an error message will be issued.

### Destination Processing Rules

The following rules apply to the destination e-mail value, according to the destination data type or value:

**CHAR**
If the source value is spaces or a zero-length VARCHAR, the destination value will be set to spaces.

**VARCHAR**
If the source value is spaces or a zero-length VARCHAR, the destination length will be 0.

**NULL** If the source value is NULL, the destination value will be NULL.

### Skipped Rows

The following conditions may cause a source row to be skipped and not written to the destination:

- The source value is NULL, and the destination column does not allow a NULL value.
- The source value is a VARCHAR less than 3 characters long.
- The source e-mail value does not contain a '@'.
- The source value cannot be converted to a format TRANS EML supports.

### Error Messages

The following error messages may be issued:

**EML01**
Parm on Col ccccc ("ppp") is invalid

The parameter *ppp*, specified on destination column *ccccc*, is not valid. Check the processing option flags specified.

**EML02**
Col ccccc not on source

The column name entered as the sourcecol parameter or the destination column name (if sourcecol was omitted) was not found on the source table.

**EML03**
Source Col ccccc-aaa invalid

The format of the source column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Availability, Type, or Length.

**EML04**
Dest Col ccccc-aaa invalid

The format of the destination column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Availability, Type, or Length.

**EML05**
Get col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS EML tried to get the value of source column *ccccc*.

**EML08**
Put col ccccc data-rc=nnn

A return code of *nnn* was returned when TRANS EML tried to set the value of destination column *ccccc*.

**EML09**

Domain literal sssss too long

The string *sssss*, specified as the domain name literal (*domain*), exceeds 31 characters.

**EML10**

User literal sssss too long

The string *sssss*, specified as the user name literal (userpfx), exceeds 31 characters.

**EML11**

Name1 Col ccccc not on dest

The *name1col* column name was not found on the destination table.

**EML12**

Name1 Col ccccc-aaa invalid

The format of the *name1col* column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Availability, Type, or Length.

**EML13**

Name2 Col ccccc not on dest

The *name1col* column name was not found on the destination table.

**EML14**

Name2 Col ccccc-aaa invalid

The format of the *name2col* column is not supported because the attribute *aaa* is invalid. The value of *aaa* will be Availability, Type, or Length.

If any other errors occur, contact Optim Technical Support.

## Example 1

The following example uses a literal (optim.com) to form the domain name and two destination table columns (FIRST_NAME and LAST_NAME) to form a user name that includes an underscore:

**TRANS EML '=_ "optim.com" FIRST_NAME LAST_NAME'**

## Example 2

The following example uses a domain name from the source column and a literal (OptimUser) to form a user name that will be suffixed with a sequential number:

**TRANS EML ', "OptimUser" '**

# Chapter 6. Data privacy providers

The Optim data privacy providers are a common code base of masking functions across multiple platforms and interfaces. Optim for z/OS allows you to call the Optim data privacy providers directly from a Column Map.

Refer to the *Customization Guide* section titled Install the Optim Data privacy providers for details on installing and configuring.

You can specify data privacy providers in a column map to mask:

- Credit Card (CCN) - credit card numbers
- National IDs (NID) - National ID numbers, enforcing country NID standards. National IDs can be masked for these countries: US, UK, Canada, Italy, France, and Spain.
- Email (EML) - Email addresses
- Affinity (AFF) (field transformation) - mask value of character or numeric data and retain data format
- Date (AGE) - add or subtract from an input date

## Field definition parameter

The field definition (FLDDEF1) parameter describes fields to use for processing and is required for each provider function.

The field definition (FLDDEF1) parameter uses the following syntax:

```
FLDDEF1 = ( NAME = colname,
  DATATYPE = datatype-value,
  [ PRECISION = field-precision-value ],
  [ SCALE = field-scale-value ],
  [ LENGTH = field-length-value ],
  [ CODEPAGE = codepage-value ],
  [ CPTYPE = { DB2ZOS | ODBC |ANY |NONE } ] ) ,
```

**FLDDEF1**

Required. Describes a source column to use for processing.

**NAME**

Required. The source column name.

**DATATYPE (or DT)**

Required. The data type of the source column.

Enter one of the following values:

**Note:** All data types are not valid for all data privacy providers. Refer to the section for the data privacy provider you plan to use. Information for each provider includes data types supported.

**CHAR**

Fixed size character data that is left-justified and space padded.

**DATE**

Date in ISO format: *%YYYY-%MM-%DD*

**DATETIME_CHAR**

Fixed size character data that contains a date-time value that is left-justified and space padded.

**DATETIME_VARCHAR**
Variable size character data that starts with a short integer value that indicates the length, in bytes, of the character date-time value that follows.

**DECIMAL_370**
Each byte of a packed decimal field represents two decimal numbers. The rightmost/last byte contains a single decimal number ending with the sign. The typical sign bits (last 4 bits or nibble of last byte) are 0xOC for positive signed number, 0x0F for unsigned (considered positive) number, and 0x0D would define the number as negative.

**DOUBLE (use for FLOAT if number is in the range 1.7E +/- 308 (15 digits))**
A double-precision floating point number. Range of values: 1.7E +/-308 (15 digits).

**FLOAT**
A floating point number. Range of values: 3.4E +/- 38 (7 digits).

**INTEGER**
A 4-byte signed integer. Range of values: -2,147,483,648 to 2,147,483,647.

**LONG_LONG**
An 8-byte signed numeric value. Range of values: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

**SMALLINT**
A 2-byte signed integer value. Range of values: -32,768 to 32,767.

**TIME** The time in ISO format: *%HH:%MM:%SS*

**TIMESTAMP**
The timestamp in ISO format: *%YYYY-%MM-%DD-%HH.%MI.%SS.%FFFFFF*

**U_LONG_LONG**
An 8-byte unsigned numeric value. Range of values: 0 - 18,446,744,073,709,551,615.

**VARCHAR**
Character data that starts with a short integer value that indicates the length, in bytes, of the character data to follow.

**LENGTH (or LEN)**
An integer value that specifies the length of the field. This parameter is used only with character data types.

**PRECISION (or PRE)**
An integer value that specifies the precision of a numeric field.

**SCALE (or SCA)**
A short integer value that specifies the scale of a numeric field.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the field. This parameter can override the CP value that is specified as part of the provider syntax when fields in the same syntax expression use different codepages.

This parameter is required when the following data types are used: CHAR, DATETIME_CHAR, VARCHAR, DATETIME_VARCHAR, and one of the following situations occur:

- The codepage information is not available within the data privacy application environment.
- The CP parameter was not specified as part of the provider syntax.

- The codepage for a field that is described by the FLDDEF parameter is different than the codepage specified as part of the provider syntax.

**CPTYPE (or CPT)**
The codepage type. This parameter can override the CPT value that is specified as part of the provider syntax when fields in the same syntax expression use different codepage types.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

This parameter is required when the following data types are used: CHAR, VARCHAR, DATETIME_CHAR, DATETIME_VARCHAR, and one of the following situations occur:

- The CP parameter was specified as part of the FLDDEF syntax.
- The CPT parameter was not specified as part of the provider syntax.
- The codepage type for a field that is described by the FLDDEF parameter is different from the codepage type specified as part of the provider syntax.

Enter one of the following values:

**DB2ZOS (or DBZ)**
DB2 z/OS

**ODBC**
ODBC

**ANY** Any DBMS

**NONE**
No DBMS

## Affinity privacy provider

Use the affinity privacy provider to mask data while maintaining the format and character types of the source values. For example, the provider can maintain the format of data such as account numbers or driver's license numbers while masking character data with character data and numeric data with numeric data.

If the source data is uppercase, alphabetic characters, the affinity privacy provider generates uppercase, alphabetic characters at the destination. Source data in lowercase alphabetic characters will produce lowercase alphabetic characters at the destination. Numeric data in the source generates numeric data at the destination. The provider masks alphabetic and numeric characters, but copies other characters in the source data to the destination. For example, a credit card number in the format *nnnn nnnn nnnn nnnn* is masked to a different number that includes spaces at the same intervals, while one formatted with dashes is masked as *nnnn nnnn nnnn nnnn*. Using this provider, you can generate unique values, a different value for each occurrence of the same source, and values with a length different from the source.

**Note:**

- Affinity currently does not support columns that contain multibyte data (MBCS and DBCS). Attempting to use Affinity with MBCS or DBCS character data produces unpredictable results.
- Affinity masks Latin English alphabetic characters only (A-Z, a-z). Alphabetic characters such as ñ ç or ü are copied to the destination without being changed.

The provider includes two algorithms for masking data, a default algorithm and a format preserving encryption (FPE) algorithm.

The FPE algorithm is encryption-based and offers stronger masking capability than the default. The FPE algorithm is based on the Advanced Encryption Standard 256-bit (AES-256) algorithm, can optionally use

an encryption key that is supplied by the user, and can produce masked values that are unique. The same user-supplied key produces repeatable masked values. For this reason, knowledge of the key should be secured from unauthorized users to prevent reverse engineering to discover the original values.

The FPE algorithm also produces outputs that are extremely varied and without a discernible pattern. For example, two close or similar source values (such as 001 and 002) are masked with values that are not similar (such as 196 and 837), thus masking any pattern these values may have had in the source data.

The TWEAK parameter, available with the FPE algorithm only, can mask identical strings within source values by using other strings within each source value to influence the masking process. For example, for two source values such as ABC-123 and DEF-123, the TWEAK parameter can mask the 123 string in each value by using the preceding string to produce the masked values ABC-981 and DEF-704.

While the FPE algorithm offers an industry standard level of encryption, consider using the default algorithm when your data does not require such encryption. The most significant differences between the two algorithms are the degree of varied masking patterns, the time that is required to complete the masking effort, and the strength of the masked results. The default algorithm usually outperforms the FPE algorithm in processing time, as the FPE algorithm in most cases takes more time to complete. The complexity of the input data can add more time to the masking operation of the FPE algorithm, depending on the amount of parallel processing and I/O being performed.

## Examples

**Account numbers**

Masking account numbers might require that the account number format is maintained while producing repeatable results for testing. To meet these requirements, you might use the following syntax:

```
trans pro=aff, algo=fpe, mtd=rep, key="Xyz123", rule=num, whenmatch=prefix("33"),
flddef1=(name=acctnbr,  datatype=varchar)
```

This example uses the following parameters:

**ALGO=FPE**
The FPE algorithm offers a high level of encryption and produces masked values that are much less likely to reveal the original source value than the default.

**MTD=REP**
This parameter produces consistent target values when the same data is processed multiple times.

**KEY="fpekey123"**
This parameter provides the case-sensitive encryption key for the FPE algorithm. To produce repeatable results, use the same key value.

**RULE=NUM**
This parameter ensures that numeric source values produce numeric target values.

**WHENMATCH=PREFIX**
This parameter prevents identical source and target values by adding a prefix to target values that match the source.

This syntax produces following results:

| Source values | Masked values |
|---|---|
| acct12345 | uaho84506 |
| acct12345 | uaho84506 |
| acct56789 | uaho66769 |

**Driver's license numbers**

Masking drivers license numbers might require that the format is maintained, as the format might be unique to the issuing agency. To meet these requirements, you might use the following syntax:

```
trans pro=aff, algo=FPE, mtd=rep, key="fpekey123", flddef1=(name=driver_number,dt=varchar)
```

This example uses the following parameters:

**ALGO=FPE**
The FPE algorithm offers a high level of encryption and produces masked values that are much less likely to reveal the original source value than the default.

**MTD=REP**
This parameter produces consistent target values when the same data is processed multiple times.

**KEY="fpekey123"**
This parameter provides the case-sensitive encryption key for the FPE algorithm. To produce repeatable results, use the same key value.

This syntax produces following results:

| Source values | Masked values |
|---|---|
| M2267 89890 34567 | O0280 50902 54225 |
| M2267 89890 34567 | O0280 50902 54225 |
| P2267 89890 34567 | D9254 86713 09303 |

**Passport numbers**

Passport number data might contain identical strings that occur among multiple values. To produce masked data that does not include such strings, use the TWEAKS parameter. This data might also contain hyphens that can be removed from masked output by using the REMOVE parameter. To meet these requirements, you might use the following syntax:

```
trans pro=aff, algo=fpe, mtd=rep, key="fpekey123", iterations=20, tweaks=yes,
flddef1=(name=passport_nbr,  dt=varchar)
```

This example uses the following parameters:

**ALGO=FPE**
The FPE algorithm offers a high level of encryption and produces masked values that are much less likely to reveal the original source value than the default.

**MTD=REP**
This parameter produces consistent target values when the same data is processed multiple times.

**KEY="fpekey123"**
This parameter provides the case-sensitive encryption key for the FPE algorithm. To produce repeatable results, use the same key value.

**ITERATIONS=20**
This parameter determines the number of times the FPE algorithm processes each value to ensure uniqueness.

**TWEAKS=YES**
This parameter ensures that identical strings are masked with unique strings.

This syntax produces following results:

| Source values | Masked values |
|---|---|
| US20139999 | YT03385676 |
| US20139999 | YT03385676 |
| CA11233388 | AU96923919 |

Some types of data you can mask using the Affinity provider are:

- Account numbers - Masking account numbers might require that the account number format is maintained while producing repeatable results for testing.
- Driver's license numbers - Masking drivers license numbers might require that the format is maintained, as the format might be unique to the issuing agency.
- Passport numbers - Passport number data might contain identical strings that occur among multiple values.

## Syntax

The affinity privacy provider uses the following syntax:

```
TRANS PROVIDER = AFFINITY,
[ ALGO = {DEFAULT | FPE } ] ,
 METHOD = {REPEATABLE | HASH },
```

These parameters are valid when ALGO=FPE is specified.

```
 [ KEY = { "literal" | @environment-variable-name } ]  ,
 [ ITERATIONS = iterations-value ]  ,
 [ TWEAKS = { YES | NO } ]  ,
```

This parameter can be used when ALGO=DEFis specified.

```
 [ SEED = { "literal" | @variable | RANDOM } ]  ,
```

The following are Data definition parameters:

**FLDDEF1** = ( **NAME** = *colname*, **DATATYPE** = *datatype-value*, [ **PRECISION** = *field-precision-value* ], [ **SCALE** = *field-scale-value* ], [ **LENGTH** = *field-length-value* ], [ **CODEPAGE** = *codepage-value* ], [ **CPTYPE** = { **DB2ZOS** | **ODBC** |**ANY** |**NONE** } ] ) , [ **CODEPAGE** = *codepage-value* ] ,

Use these parameters for CHAR data:

```
 [ LENGTH = { n | MAX } ]  ,
 [ COPY = (copy-start,copy-len [,"copy-lit"]) ...  ]  ,
```

The following parameters are for Numeric data:

```
 [ RULE = { NUMERIC | BINARY | SCALED } ]  ,
```

Source value parameters are as follows:

```
 [ CASE = UPPER ] ,
 [ REMOVE = "remove-chars" ] ,
 [ TRIM = RIGHT ] ,
```

Processing parameters:

```
 [ DISCARDLIMIT = discard-limit-value ],
```

## Masking parameters

Parameters that determine how to mask data.

**PROVIDER (or PRO)**
Required. Enter the provider name, AFFINITY (or AFF).

**Note:** The PRO parameter must be first in the masking string. All other parameters can appear in any order.

**ALGO**
The algorithm to use. Enter one of the following options:

**DEFAULT (or DEF)**
Mask data by using the default algorithm.

ALGO=DEF is not compatible with the parameters KEY, ITERATIONS, and TWEAKS.

**FPE** Mask data by using the FPE algorithm. Use the KEY parameter to enter the encryption key.

ALGO=FPE is not compatible with the parameters SEED and MTD=HASH.

**METHOD (or MTD)**
The masking method to use. This parameter is required. For small or less complex data sets, the HASH method has a better strength of masking compared to the REPEATABLE method. The HASH method might provide slightly slower processing compared to the REPEATABLE method. Enter one of the following options:

**REPEATABLE (or REP)**
Generates consistently repeatable and unique output values. The same masked value is generated for every instance of a given source value, and each unique source value has a unique output value.

**HASH**
Generates output values by using a hash algorithm. The output values might not be unique nor repeatable between masking operations.

MTD=HASH is not compatible with the parameter ALGO=FPE.

## ALGO=FPE parameters

Parameters for use with ALGO=FPE only.

**KEY** The encryption key. To produce repeatable masked data, use the same key value.

Enter one of the following options:

**"***literal***"**
A 1 - 32 character alphanumeric string (including special characters such as $, @, and #). A NULL value is not valid.

*@environment-variable-name*
Name of an environment variable that provides the encryption key. The environment variable name must be prefixed with the at sign (@) and assigned a value before invoking the masking provider.

**ITERATIONS**
The number of times the FPE algorithm processes each source value to avoid creating a masked value that matches the source value. Enter an integer in the range 1 - 99. The default is 12. A value less than 12 has a negligible improvement on performance.

**TWEAKS**

If two values contain identical strings, determines whether to mask the identical strings differently for each combination of non-identical characters.

TWEAKS is not applicable to numeric data types or with the parameters ALGO=DEF and RULE.

Enter one of the following options:

**YES** Default. Use tweaks to mask identical strings in source values by using other strings within each source value to influence the masking process. For example, if TWEAKS=YES, the provider might mask values AB-1234 and CD-1234 to DX-4795 and NR-3687, rather than DX-4795 and NR-4795.

**NO** Do not use tweaks.

## ALGO=DEF parameters

Parameters for use with ALGO=DEF only.

**SEED** Use the SEED parameter to influence the masking process and to produce repeatable or non-repeatable output values. To produce repeatable masked data, use the same seed value.

Enter one of the following options:

**Parameters for producing repeatable values:**

**"***literal***"**

A character string, within enclosing double quotation marks. Any special characters in the string are ignored. The string is case-sensitive. Note that there is no particular advantage to providing a lengthy string. Using a literal potentially exposes the seed value and might allow reverse engineering to discover the original values.

*@variable*

The name of an environment variable, preceded by the at sign (@). To prevent reverse engineering to discover the original values, the variable can be created by the Optim administrator or other authorized person and secured from unauthorized access.

**Parameter for producing non-repeatable values:**

**RANDOM (or RAN)**

Produces a non-repeatable output value. Although this option might produce duplicate values each time the process is run, this option can be used to generate test data from a limited set of source data by combining the output of multiple runs.

**RANDOM (or RAN)**

Specifies a random seed value, which produces a non-repeatable output value. This option can be used as a test data generator by concatenating the output of multiple runs. This option might produce duplicate values each time the process is run.

## CHAR parameters

Parameters for use with CHAR character types.

**LENGTH (or LEN)**
For character data types only. Specifies a fixed length for each destination value, regardless of the length of the source value. The source value is repeated or truncated to match the LENGTH value before processing. If this parameter is not specified, the length of the destination value is the same as the source value.

The trailing blanks of an unfilled CHAR input value are considered part of a source value, resulting in an output value that also has trailing blanks. To include only non-blank characters in the output value, also specify TRIM=RIGHT.

The destination value might not be unique when the LENGTH parameter is specified. For example, if LEN=2 and the unique values `test1` and `test2` are input, the provider masks only the first two characters in each value (te) and generates duplicate output values.

LEN is not compatible with the parameter RULE.

Enter one of the following options:

*n* An integer value that specifies the destination value length. The value cannot exceed the length of the destination column.

**MAX** Generate a value that matches the length of the destination column.

**COPY** For character data types only. Specifies one or more pairs of substrings to copy to the destination value. The provider can also replace characters with a literal string. Note that this parameter might cause non-unique destination values.

If the literal string is shorter than the substring, the provider replicates the literal until the lengths match. If the literal is longer than the substring, it is truncated.

Spaces are supported within literals only (for example, `X Y`).

COPY is not compatible with the parameter RULE.

Enter the following options:

*copy-start*
The starting position of the substring to copy to the destination.

*copy-len*
The length of the substring to copy to the destination.

*copy-lit*
A literal string that replaces the source characters in the specified positions. If the literal string is shorter than the substring, the provider replicates the literal until the lengths match. If the literal is longer than the substring, it is truncated.

## Numeric parameters

Parameters for use with numbers and numeric data types.

**RULE** Specifies processing rules to ensure that numeric strings are properly masked as numeric values.

Enter one of the following options:

**NUMERIC (or NUM)**
For numeric values within character data types only, specifies that numeric values generate masked numeric values instead of other characters. Without this option, the value 12345 in a character type could produce a masked value that contains alphabetic characters. If a source value is not a number, the provider processes the value normally without an error or warning message. Use this option for the following situations:

- To transform integers in a character data type column to match the values of a numeric data type column. This usage maintains DBMS-type foreign key integrity across differing data types.
- To correctly transform floating point numbers in character data type columns. Without this parameter, the exponent character E (or e) is transformed into another letter.
- To maintain leading zeros that might otherwise be removed. For example, to prevent a value such as 00015 from being masked as a two-digit value.

RULE=NUM is not compatible with the parameters COPY and LEN.

**BINARY (or BIN)**
Specifies that source values of a numeric data type are transformed into values that conform to the provider data type restrictions. The output value might differ from the source value in order of magnitude and sign. Without this parameter, source values in numeric data type columns are converted to character strings and then converted back to the original data type.

Without RULE=BIN, there is a discrepancy between the number of digits that are required to represent a numeric value in character format and the maximum value that a given number of digits can express. For example, an unsigned short integer needs five-digit positions to hold the maximum unsigned value of 65535, but five-digit positions can express a maximum string value of 99999, which when converted back to an unsigned short integer results in an overflow. Without RULE=BIN, boundary source values in numeric data type columns might be transformed into values that use this discrepancy and do not conform to the storage constraints.

RULE=BIN is not compatible with the parameters COPY, LEN, TRIM, REMOVE, and CASE.

**SCALED (or SCA)**
Specifies that the output value has the same sign and approximately the same magnitude as the source value of the numeric data type. For example, when masking data such as salary figures, use this option to ensure that a value such as 12,345.67 does not produce a value such as -3.1.

## Source value parameters

Parameters for handling source values.

**CASE** Specifies that the provider converts the source values to uppercase before processing.

CASE is not compatible with the parameters RULE=BIN and RULE=SCALED.

Enter the following option:

**UPPER (or UP)**
Convert the source values to uppercase before processing.

**REMOVE**
Specifies the characters, within enclosing double quotation marks, to remove from the source value before processing.

REMOVE is not compatible with the parameters RULE=BIN and RULE=SCALED.

**TRIM** Specifies to remove trailing spaces from the source value before processing.

TRIM is not compatible with the parameters RULE=BIN and RULE=SCALED.

Enter the following option:

**RIGHT**
Right-trim the source value to remove trailing spaces before processing.

## Processing parameters

Parameters for managing provider processes.

**DISCARDLIMIT (or DLIM)**
Specifies the number of failed rows to discard before the provider stops processing.

## Data definition parameters

Parameters for defining source and destination data.

**FLDDEF1**
Required. Describes a source column to be processed. See "Field definition parameter" on page 215.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the source column. The default codepage in z/OS is derived from the operating system default value, for example, 037. Codepage 037 is the basic EBCDIC codepage in the U.S. The CP parameter within the FLDDEF parameter overrides this value.

**CPTYPE (or CPT)**
The codepage type of the source column. The CPT parameter within the FLDDEF parameter overrides this value.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

Enter one of the following values:

| Value | Description |
|---|---|
| DBZ (or DB2zOS) | DB2 for z/OS |
| ODBC | ODBC |
| ANY | Any DBMS |
| NONE | No DBMS |

## Affinity provider - Supported data types

The Affinity provider supports the following data types. Note that some data types are not compatible with RULE=BIN and RULE=SCALED.

| DB2 data type | ODPP equivalent | Description | RULE=BIN and RULE=SCALED |
|---|---|---|---|
| CHAR | CHAR | Fixed size character data that is left justified and space padded. | No |
| INT | INTEGER | A 4 byte signed integer value in the range -2,147,483,648 to 2,147,483,647. | Yes |

| DB2 data type | ODPP equivalent | Description | RULE=BIN and RULE=SCALED |
|---|---|---|---|
| VARCHAR | VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. | |
| SMALLINT | SMALLINT | A 2 byte signed integer value in the range -32,768 to 32,767. | Yes |
| DECIMAL | DECIMAL_370 | Packed decimal encoded buffer. | No |
| FLOAT | FLOAT or DOUBLE based on precision | Double precision floating point number in the range 1.7E +/- 308 (15 digits). Floating point number in the range 3.4E +/- 38 (7 digits). | Yes |
| BIGINT | LONG_LONG or U_LONG_LONG | An 8 byte signed numeric value in the range -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. An 8 byte unsigned numeric value in the range 0 to 18,446,744,073,709,551,615. | Yes |

# Age privacy provider

Use the age privacy provider to age values in a source column. The source values can contain character, numeric, date, or timestamp data.

The aging process can increment or decrement a date value. Aging can be specific to the number of years, months, weeks, or days. Optionally, aging can be a combination of these units. Aging can also be based upon a specific 4-digit year value.

The provider uses the following default date format for source and destination columns: DD/MM/YYYY HH24:MI:SS:FFFFFF. To override the default format, use the SRCDF or DSTDF parameters to match the format used by the source and destination columns.

## Examples

**Source format**

The following example ages dates one month and uses the YYYY/MM/DD format for the source value.

```
trans pro=age, mon=1, srcdf="%YYYY/%MM/%DD", flddef1=(name=agevarchar, dt=date)
```

This example uses the following parameters:

**MON=1** This parameter ages the date by adding one month.

**SRCDF="%YYYY/%MM/%DD"**
This parameter specifies the YYYY/MM/DD format for the source value.

**Destination format**

The following example ages dates one year and three months. The destination value uses the YYYY/MM/DD HH:MI:SS.FFFFFF date and time format.

```
trans pro=age, year=1, mon=3, dstdf="%YYYY/%MM/%DD %HH:%MI:%SS.%FFFFFF",
flddef1=(name=agetimestamp, dt=datetime_char)
```

This example uses the following parameters:

**YEAR=1**
: This parameter ages the date by adding one year.

**MON=3** This parameter ages the date by adding three months.

**DSTDF="%YYYY/%MM/%DD %HH:%MI:%SS.%FFFFFF"**
: This parameter specifies the YYYY/MM/DD HH:MI:SS.FFFFFF format for the destination value.

## Syntax

The age privacy provider uses the following syntax:

**Masking parameter**

```
TRANS PROVIDER = AGE,
```

**Data definition parameters**

```
 FLDDEF1 = ( NAME = colname,
  DATATYPE = datatype-value,
  [ PRECISION = field-precision-value ],
  [ SCALE = field-scale-value ],
  [ LENGTH = field-length-value ],
  [ CODEPAGE = codepage-value ],
  [ CPTYPE = { DB2ZOS | ODBC |ANY |NONE } ] ) ,
 [ DISCARDLIMIT = discard-limit-value ] ,
[ CODEPAGE = codepage-value ]
 ,
```

**Formatting parameters**

```
 [ SRC_DATE_FORMAT = "src-date-format-expression" ],
 [ DST_DATE_FORMAT = "dest-date-format-expression" ],
```

**Aging parameters**

```
 [ YEAR = n-years | "specific-year" } ],
 [ MONTH = n-months ],
 [ WEEK = n-weeks ],
 [ DAY = n-days ],
 [ PIVOT =  century-pivot-value ],
```

## Masking parameters

Parameters that determine how to mask data.

**PROVIDER (or PRO)**
: Required. Enter the provider name, AGE.

  **Note:** The PRO parameter must be first in the masking string. All other parameters can appear in any order.

## Data definition parameters

Parameters for defining source and target data.

**FLDDEF1**
: Required. Specifies the attributes of the source column to use for processing.

**DISCARDLIMIT (or DLIM)**
Specifies the number of failed rows to discard before the provider stops processing.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the source column. The default codepage in z/OS is derived from the operating system default value, for example, 037. Codepage 037 is the basic EBCDIC codepage in the U.S. The CP parameter within the FLDDEF parameter overrides this value.

**CPTYPE (or CPT)**
The codepage type of the source column. The CPT parameter within the FLDDEF parameter overrides this value.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

Enter one of the following values:

| Value | Description |
|---|---|
| DBZ (or DB2zOS) | DB2 for z/OS |
| ODBC | ODBC |
| ANY | Any DBMS |
| NONE | No DBMS |

## Formatting parameters

Parameters that determine how to format dates.

**SRC_DATE_FORMAT (or SRCDF)**
Specifies, within enclosing double quotation marks, the format of the date string values in the source column.

Use any of the following format specifiers for formatting the date-time strings. All specifiers start with a percent (%) sign. If SRCDF is not specified, the provider uses the following default date format: %DD/%MM/%YYYY %HH24:%MI:%SS:%FFFFFF.

**Year**

%YYYY
%YY

**Month**

%MONTH
%MMM
%MM

**Day**

%DD

**Time**

%HH
%MI
%SS
%HH24

**Fraction of a second**

%FFFFFF

%FFFFF

%FFFF

%FFF

%FF

%F

**AM/PM**

%PM

For example, to format a date string as four-digit year, three-character month, two-digit date with dash-type (-) separators, the format string would be "%YYYY-%MMM-%DD".

**DST_DATE_FORMAT (or DSTDF)**

Specifies, within enclosing double quotation marks, the format of the output date string values in the destination column.

Use any of the following format specifiers for formatting the date-time strings. All specifiers start with a percent (%) sign. By default, the source date format is used as the destination date format if the source data type is CHAR. If DSTDF is not specified, the provider uses the following default date format: %DD/%MM/%YYYY %HH24:%MI:%SS:%FFFFFF.

**Year**

%YYYY

%YY

**Month**

%MONTH

%MMM

%MM

**Day**

%DD

**Time**

%HH

%MI

%SS

%HH24

**Fraction of a second**

%FFFFFF

%FFFFF

%FFFF

%FFF

%FF

%F

**AM/PM**

%PM

For example, to format a date string as a four-digit year, three-character month, two-digit date with dash-type (-) separators, the format string would be "%YYYY-%MMM-%DD".

## Aging parameters

Parameters that determine how to age dates.

**YEAR (or YR)**
Specifies either the number of years to increment or decrement a source date value or a specific year to use for a date value. Enter one of the following options:

*n-years* Increments or decrements the source date value by a specific number of years. A positive value increments the year, and a negative value decrements the year. Enter an integer value within the range -2500 to +1581. The plus sign is optional.

*"specific-year"*
Specifies, within enclosing double quotation marks, a four-digit year value to replace the source year value. The other parts of the date value and format remain the same. The valid values are "1582" to "3999".

YR=*specific-year* is not compatible with the parameters MONTH, WEEK, and DAY.

**MONTH (or MON)**
Increments or decrements the source date value by a specific number of months. A positive value increments the month, and a negative value decrements the month. Enter an integer value within the range -30000 to +30000. The plus sign is optional.

MON is not compatible with the parameter YR=*specific-year*.

**WEEK (or WK)**
Increments or decrements the source date value by a specific number of weeks. A positive value increments the weeks, and a negative value decrements the weeks. Enter an integer value within the range -30000 to +30000. The plus sign is optional.

WK is not compatible with the parameter YR=*specific-year*.

**DAY** Increments or decrements the source date value by a specific number of days. A positive value increments the number of days, and a negative value decrements the number of days. Enter an integer value within the range -30000 to +30000. The plus sign is optional.

DAY is not compatible with the parameter YR=*specific-year*.

**PIVOT (or PIV)**
Specifies the appropriate century for source dates with two-digit years. Enter a value within the range 0 to 99. The default value is 65. The following rules apply to PIV:

- All two-digit years equal to or greater than the PIV value are placed in the 20th century (19xx).
- All two-digit years less than the PIV value are placed in the 21st century (20xx).

## Supported data types

The age privacy provider supports the following data types for source and destination columns:

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| CHAR | DATETIME_CHAR | A fixed size character data value that is left justified with space padded and contains date-time values. |
| VARCHAR | DATETIME_VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. |

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| DATE | ODBC_DATE | A data type that is used when the source value is a ODPP_ODBC_DATE structure of the Optim data masking API. |

## Credit card privacy provider

Use the credit card privacy provider to generate a valid and unique credit card number (CCN). The provider uses a repeatable method that algorithmically generates a consistently altered CCN based on the source CCN.

A CCN, as defined by ISO 7812, consists of a six-digit issuer identifier followed by a variable length account number and a single check digit as the final number. The check digit verifies the accuracy of the CCN and is generated by passing the issuer identifier and account numbers through the Luhn algorithm. The credit card privacy provider supports a CCN up to a maximum length of 16 digits.

### Examples

The following example preserves invalid values from the source field. The provider will copy the first four digits of the source issuer identifier values and mask the fifth and sixth digits.

```
trans pro=ccn, wheninv=pre, flddef1=(name=ccnchar, dt=char)
```

This example uses the following parameter:

**WHENINV=PRE**
The parameter copies invalid source values to the destination field.

The following example copies the six-digit issuer identifier, and preserves invalid values from the source field.

```
trans pro=ccn, pat=6c, wheninv=pre, flddef1=(name=ccnchar, dt=char)
```

This example uses the following parameters:

**PAT=6C**
This parameter copies all six digits of the issuer identifier.

**WHENINV=PRE**
The parameter copies invalid source values to the destination field.

### Syntax

The credit card privacy provider uses the following syntax:

**Masking parameter**

```
TRANS PRO = CCN  ,
```

**Data definition parameters**

```
 FLDDEF1 = ( NAME = colname,
  DATATYPE = datatype-value,
  [ PRECISION = field-precision-value ],
  [ SCALE = field-scale-value ],
  [ LENGTH = field-length-value ],
  [ CODEPAGE = codepage-value ],
  [ CPTYPE = { DB2ZOS | ODBC |ANY |NONE } ] ) ,
[ CODEPAGE = codepage-value ]
 ,
```

**Processing parameters**

```
[ PATTERN = { 4C2R | 6C } ]  ,
[ WHENINVALID = PRESERVE ]  ,
[ DISCARDLIMIT = discard-limit-value ] ,
```

## Masking parameters

Parameters that determine how to mask data.

**PROVIDER (or PRO)**
Required. Enter the provider name, CCN.

**Note:** The PRO parameter must be first in the masking string. All other parameters can appear in any order.

## Data definition parameters

Parameters for defining source and target data. For further information see Supported Data Types.

**FLDDEF1**
Required. Specifies the attributes of the source column to use for processing. See "Field definition parameter" on page 215

**DISCARDLIMIT (or DLIM)**
Specifies the number of failed rows to discard before the provider stops processing.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the source column. The default codepage in z/OS is derived from the operating system default value, for example, 037. Codepage 037 is the basic EBCDIC codepage in the U.S. The CP parameter within the FLDDEF parameter overrides this value.

**CPTYPE (or CPT)**
The codepage type of the source column. The CPT parameter within the FLDDEF parameter overrides this value.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

Enter one of the following values:

| Value | Description |
|---|---|
| DBZ (or DB2zOS) | DB2 for z/OS |
| ODBC | ODBC |
| ANY | Any DBMS |
| NONE | No DBMS |

## Processing parameters

Parameters for managing provider processes.

**PATTERN (or PAT)**
Options for overriding the masking default, which includes the first four digits of the issuer identifier from the source CCN and alters the remaining two digits based on the source value.

Enter one of the following options:

**4C2R** The provider copies the first four digits of the issuer identifier and randomly generates the fifth and sixth digits.

**6C** The provider copies all six digits of the issuer identifier.

**WHENINVALID (or WHENINV)**
Determines how to process invalid source values. If this parameter is omitted, the provider does not copy invalid source values to the destination field and skips rows that contain these values.

Enter the following option:

**PRESERVE (or PRE)**
Copy invalid source values to the destination field.

### Supported data types

The credit card privacy provider supports the following data types for source and destination fields:

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| CHAR | CHAR | Fixed size character data that is left justified and space padded. |
| VARCHAR | VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. |
| DECIMAL | DECIMAL_370 | Packed decimal encoded buffer. |
| BIGINT | U_LONG_LONG | An 8 byte unsigned numeric value in the range 0 to 18,446,744,073,709,551,615. |

## Email privacy provider

Use the email privacy provider to generate an email address. An email address consists of two parts, a user name and a domain name, which is separated by '@'. For example, user@domain.com.

The email privacy provider generates an email address with a user name based on either source data or a literal that is concatenated with a sequential number. The user name can be formed with data from two columns (name1 and name2). The domain name can be based on an email address in the source data, a literal, or randomly selected from a list of large email service companies. The email address can also be converted to uppercase or lowercase.

If the USERPFX and PARTS parameters are not specified, the user name is formed by the literal "email" concatenated with a sequential number.

The provider offers two hashing algorithms, a default algorithm and the SHA-256 (secure hashing algorithm 256 bits) algorithm. The default algorithm usually outperforms the SHA-256 algorithm in processing time. The SHA-256 algorithm is a cryptographic-strength hashing algorithm that offers stronger masking than the default algorithm and ensures that masked values cannot be revealed by reverse engineering. The HMAC (hash-based message authentication code) algorithm, which is based on

the SHA-256 algorithm, is also available with a user-supplied exit only and provides stronger encryption than the SHA-256 algorithm alone by using an encryption key that is provided by the exit. The provider prevents unauthorized access to the key value by not keeping the key in memory.

## Examples

The following example preserves invalid values from the source field. The provider will generate user names with the literal "email" concatenated with a sequential number, and the provider will use the source domain name for the output.

```
trans pro=eml, wheninv=pre, flddef1=(name=emlvarchar, dt=varchar)
```

This example uses the following parameter:

**WHENINV=PRE**
The parameter copies invalid source values to the destination field.

The following example uses a registered domain name.

```
trans pro=eml, hashdom=reg, flddef1=(name=emlvarchar, dt=varchar)
```

This example uses the following parameter:

**HASHDOM=REG**
This parameter selects domain names from a list of large email service companies.

**User name parts**

The following example generates a lowercase email address with a two-part user name that is separated by a period. The user name is taken from fields that are identified by the PARTS parameter.

```
trans pro=eml, case=low, parts="(emlfirstname, emllastname)", sep=dot,
flddef1=(name=emlnamecol, dt=varchar)
```

This example uses the following parameters:

**CASE=LOW**
This parameter generates email addresses in lowercase.

**PARTS="(***emlfirstname,emllastname***)"**
This parameter forms a two-part user name from the specified fields.

**SEP=DOT**
This parameter separates the two-part user name with a period.

**HMAC seed**

The following example uses the SHA-256 algorithm, and a seed value that is provided by a user exit. The domain name is provided by a list of large email service companies.

```
trans pro=eml, algo=sha256, seed=hmac, hashdom=reg,
          flddef1=(name=emlvarchar, dt=varchar)
```

This example uses the following parameters:

**ALGO=SHA256**
The SHA-256 algorithm is a cryptographic-strength hashing algorithm and ensures that masked values cannot be revealed by reverse engineering.

**SEED=HMAC**
This parameter uses a seed value to influence the masking process that is provided by the user exit. Use the same value each time the provider is used to produce the same masked values for a given source value.

**HASHDOM=REG**
This parameter selects domain names from a list of large email service companies.

## Syntax

The email privacy provider uses the following syntax:

**Masking parameters**

```
PROVIDER = EML  ,
 [ ALGORITHM = { SHA256 | DEFAULT } ] ,
```

**Data definition parameters**

```
FLDDEF1 = ( NAME = colname,
 DATATYPE = datatype-value,
 [ PRECISION = field-precision-value ],
 [ SCALE = field-scale-value ],
 [ LENGTH = field-length-value ],
 [ CODEPAGE = codepage-value ],
 [ CPTYPE = { DB2ZOS | ODBC |ANY |NONE } ] ) ,
```

**User name and formatting parameters**

```
[ CASE = { UPPER  | LOWER  } ]  ,
[ SEPARATOR = { DOT | . | UNDERSCORE | _ | }    ]  ,
[ USERPREFIX = username-prefix ] ,
[ PARTS = " ( {name1fld-index[ ,name2fld-index]|
                    name1fld-name [ ,name2fld-name ] }  ) " ]  ,
[ FCPART1 = { Y | N } ]  ,
```

**Processing parameters**

```
[ SEED = { "seed-value" | HMAC } ]  ,
[ HASHDOMAIN = { KEEP | REGISTERED | UNREGISTERED  } ] ,
  [ DOMAIN = domain-name ] ,[ WHENINVALID = PRESERVE ]  ,
```

## Masking parameters

Parameters that determine how to mask data.

**PROVIDER (or PRO)**
Required. Enter the provider name, EML.

**Note:** The PRO parameter must be first in the masking string. All other parameters can appear in any order.

**HASH**
Generates an email address by using a hashing algorithm. The user name is composed of alphanumeric characters. Use the HASHDOM parameter to generate an email address that includes the source domain name, a random domain name, or domain names from a list of large email service companies. The output values are based on the source values.

**ALGORITHM (or ALGO)**
ALGO is not compatible with the parameters PARTS, SEP, FCPART1, USERPREFIX, DOMAIN, and CASE.

Enter one of the following options:

**SHA256**
Specifies to use the SHA-256 (secure hashing algorithm 256 bits) algorithm, or if a user-supplied exit is available, the HMAC (hash-based message authentication code) algorithm.

**DEFAULT (or DEF)**
Specifies to use the default ODPP hash algorithm.

## Data definition parameters

Parameters for defining source and target data. For further information, see “Supported data types” on page 238.

**FLDDEF1**
Required. Specifies the attributes of the source column to use for processing. See “Field definition parameter” on page 215.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the source column. The default codepage in z/OS is derived from the operating system default value, for example, 037. Codepage 037 is the basic EBCDIC codepage in the U.S. The CP parameter within the FLDDEF parameter overrides this value.

**CPTYPE (or CPT)**
The codepage type of the source column. The CPT parameter within the FLDDEF parameter overrides this value.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

Enter one of the following values:

| Value | Description |
|---|---|
| DBZ (or DB2zOS) | DB2 for z/OS |
| ODBC | ODBC |
| ANY | Any DBMS |
| NONE | No DBMS |

## User name and formatting parameters

Parameters that determine how to manage user names and formatting.

**CASE** Indicates whether the output email address is generated in uppercase or lowercase.

CASE is not compatible with the parameters HASHDOM, ALGO, and SEED.

Enter one of the following options:

**UPPER (or UP)**
Convert the output email address to uppercase.

**LOWER (or LOW)**
Convert the output email address to lowercase.

**SEPARATOR (or SEP)**
The separator character between the name1 and name2 values.

SEP is not compatible with the parameters USERPREFIX, HASHDOM, ALGO, and SEED.

Enter one of the following options:

**DOT (or .)**
Separate the values with a period (.). For example, name1.name2@ibm.com.

**UNDERSCORE (or _)**
Separate the values with an underscore (_). For example, name1_name2@ibm.com.

**USERPREFIX (or USERPFX)**
A literal, up to 512 characters, that is concatenated with a sequential number to form the user name.

USERPFX is not compatible with the parameters PARTS, SEP, FCPART1, HASHDOM, ALGO, and SEED.

**PARTS**
The fields that provide the name1 and name2 values of the user name. Within enclosing double quotation marks, enter either the field names or the FLDDEF numeric values that are assigned to the fields. Enter the name1 value first. For example, if name1 is defined in FLDDEF1 and name 2 is defined in FLDDEF3, enter the following parameter: `PARTS="(1,3)"`.

PARTS is not compatible with the parameters USERPFX, MTD=HASH, HASHDOM, ALGO, and SEED.

**FCPART1**
Indicates whether only the first character of the name1 field value is used. Enter one of the following options:

**Y** Use only the first character of the name1 field.

**N** Use all the characters of the name1 field.

## Processing parameters

Parameters for managing provider processes.

**DOMAIN (or DOM)**
A literal, up to 31 characters, that forms the domain name. If this parameter is not specified, the source domain name is used for the output.

DOM is not compatible with the parameters HASHDOM, ALGO, and SEED.

**HASHDOMAIN (or HASHDOM)**
Indicate how the domain name is formed. HASHDOM is not compatible with the parameters PARTS, FCPART1, USERPREFIX, DOMAIN, ALGO, and CASE.

Enter one of the following options:

**KEEP** Use the source domain name.

**REGISTERED (or REG)**
Use a domain name from a list of registered email service providers.

**UNREGISTERED (or UNREG)**
Generate an alphanumeric domain name.

**SEED** Enter a value to manage the hashing process. To generate repeatable masked values for the same input, use the same SEED value.

SEED is not compatible with the parameters PARTS, SEP, FCPART1, USERPREFIX, DOMAIN, and CASE.

Enter one of the following options:

*seed value*
Enter a literal seed value, up to 31 characters and within enclosing double quotation marks, to use for hashing.

If ALGO=SHA256, the seed value must be a numeric value in the range 0 - 2,000,000,000.

**HMAC**
Specifies to use the seed value that is provided by the user exit. A user exit is required.

SEED=HMAC is not compatible with the parameter ALGO=DEF.

**WHENINVALID (or WHENINV)**
Determines what to do with invalid source values. If this parameter is omitted, invalid source values are not copied to the destination field and rows that contain these values are skipped.

Enter the following option:

**PRESERVE (or PRE)**
Copy invalid source values to the destination field.

### Supported data types

The email privacy provider supports the following data types for source and destination fields:

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| CHAR | CHAR | Fixed size character data that is left justified and space padded. |
| VARCHAR | VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. |

## NID privacy provider

Use the NID privacy provider to mask national ID numbers. The provider includes several separator options for the output values (dashes, periods, spaces, or no separators).

The provider masks the following national ID numbers:

- Canadian Social Insurance Number (SIN)
- French National Institute for Statistics and Economic Studies Number (INSEE)
- Italian Fiscal Code Number (CF)
- Spanish Fiscal Identification Number (NIF)
- United Kingdom National Insurance Number (NINO)
- United States Social Security Number (SSN)

### Examples

The following example masks a United States SSN by using a repeatable method with separator characters. The NID privacy provider will mask the area, group, and serial numbers.

```
trans pro=nid, swi=us, fmt=(US=3X-2X-4X), flddef1=(name=nidvarchar, dt=varchar)
```

This example uses the following parameters:

**SWI=US**
This parameter masks United States SSN values.

**FMT=(US=3X-2X-4X)**
This parameter masks all values in the SSN and includes a dash separator character between the subfields.

The following example masks a Canadian SIN. The NID privacy provider copies the header values and masks the remaining digits. The output values use spaces as separators. Validation is performed and invalid source values are preserved.

```
trans pro=nid, switch=ca, fmt=(ca=3c 3x 3x), wheninv=pre, val=y,
 flddef1=(name=nid_fld, dt=char,len=20)
```

This example uses the following parameters:

**SWI=CA**
This parameter masks Canadian SIN values.

**FMT=(CA=3c 3x 3x)**
This parameter copies the header values and masks the remaining digits of the SIN. The output values include spaces as separators.

**WHENINV=PRE**
The parameter copies invalid source values to the destination field.

**VAL=Y** The parameter performs country-specific validation on the source values.

The following example masks a Canadian Social Insurance Number (SIN) by using a random method. The NID privacy provider copies the header values and mask the remaining digits. The output value includes dash (-) separators.

```
trans pro=nid, swi=ca, fmt=(ca=3x-3x-3x), flddef1=(name=nid_fld, dt=char)
```

This example uses the following parameters:

**SWI=CA**
This parameter masks Canadian SIN values.

**FMT=(ca=3x-3x-3x)**
This parameter masks all values in the SIN and includes a dash separator character between the subfields.

## Syntax

The NID privacy provider uses the following syntax:

**"Masking parameters" on page 240**

```
PROVIDER = NID  ,
```

**"Data definition parameters" on page 240**

```
 FLDDEF1 = ( NAME = colname,
  DATATYPE = datatype-value,
  [ PRECISION = field-precision-value ],
  [ SCALE = field-scale-value ],
  [ LENGTH = field-length-value ],
  [ CODEPAGE = codepage-value ],
  [ CPTYPE = { DB2ZOS | ODBC |ANY |NONE } ] ) ,
[ CODEPAGE = codepage-value ]
 ,
```

**"NID parameters" on page 240**

```
 SWITCH = country-identifier  ,
 [ FORMAT = ( format-expression ) ]  ,
```

**"SWI=US parameters" on page 241**

```
 [ AREATABLE = { MAXGROUP | DIST | ZOS } ]  ,
```

**"Processing parameters" on page 241**

```
[ VALIDATE = { Y | N } ] ,
[ WHENINVALID = PRESERVE ]  ,
[ DISCARDLIMIT = discard-limit-value ] ,
```

## Masking parameters

Parameters that determine how to mask data.

**PROVIDER (or PRO)**
Required. Enter the provider name, NID.

**Note:** The PRO parameter must be first in the masking string. All other parameters can appear in any order.

## Data definition parameters

Parameters for defining source and destination data. For further information, see "Supported data types" on page 246.

**FLDDEF1**
Required. Specifies the attributes of the source column to use for processing. See "Field definition parameter" on page 215.

**CODEPAGE (or CP)**
An integer value that specifies the codepage or character-set identifier of the source column. The default codepage in z/OS is derived from the operating system default value, for example, 037. Codepage 037 is the basic EBCDIC codepage in the U.S. The CP parameter within the FLDDEF parameter overrides this value.

**CPTYPE (or CPT)**
The codepage type of the source column. The CPT parameter within the FLDDEF parameter overrides this value.

When the origin of the data is DBMS-specific but not tied to any one DBMS, specify the value as ANY. When the origin of the data is from a non-DBMS source, specify the value as NONE.

Enter one of the following values:

| Value | Description |
|---|---|
| DBZ (or DB2zOS) | DB2 for z/OS |
| ODBC | ODBC |
| ANY | Any DBMS |
| NONE | No DBMS |

## NID parameters

Parameters that identify the type of NID to mask and define formatting options specific to the NID.

**SWITCH (or SWI)**

Required. A two-character value that indicates the type of national ID to mask. Only one switch value is permitted.

Enter one of the following options:

**CA** Canadian Social Insurance Number (SIN)

**FR** French National Institute for Statistics and Economic Studies Number (INSEE)

**IT** Italian Fiscal Code Number (CF)

**ES** Spanish Fiscal Identification Number (NIF)

**UK** United Kingdom National Insurance Number (NINO)

**US** United States Social Security Number (SSN)

**FORMAT (or FMT)**
Specifies the output format and the parts of the source value to mask. The national ID determines the syntax for this parameter.

- Canadian Social Insurance Number (SIN)
- French National Institute for Statistics and Economic Studies Number (INSEE)
- Italian Fiscal Code Number (CF)
- Spanish Fiscal Identification Number (NIF)
- United Kingdom National Insurance Number (NINO)

## SWI=US parameters

Parameters for use with SWI=US only.

**AREATABLE (or ATAB)**
For use with SWI=US only. Specifies the area list to use for United States Social Security Number (SSN) masking. Enter one of the following options:

**MAXGROUP**
Default. Use the high group list, which is dated 10/10/2010. This list represents the new method of SSN allocation by the Social Security Administration, which uses all of the area code and group IDs (99 is the maximum value) for each area.

**DIST** Use the Optim area list, which is dated 11/01/2006. This list represents the old method of SSN allocation, which uses the area codes and group IDs assigned until 11/01/2006.

**ZOS** Use the Optim for z/OS area list, which is dated 01/02/2008. This list represents the old method of SSN allocation, which uses the area codes and group IDs assigned until 11/01/2008.

## Processing parameters

Parameters for managing provider processes.

**VALIDATE (or VAL)**
Specifies if the provider performs country-specific validation on the source values. If this parameter is omitted, the provider does not perform validation (VAL=N). Enter one of the following options:

**Y** Validate the source values. The provider performs the following validations, according to the NID:

**Canadian Social Insurance Numbers (SIN)**

- The first digit must not be 8.

- There must not be three consecutive zeros (000) at positions 1-3, 4-6 or 7-9.
- The check digit must be valid.

**French National Institute for Statistics and Economic Studies numbers (INSEE)**

- The commune value must be valid.
- The check digit must be valid.

**Italian Fiscal Codes (CF)**
The check digit must be valid.

**Spanish Fiscal Identification Numbers (NIF) and Foreign Identification Numbers (NIE)**
The suffix must be valid.

**United Kingdom National Insurance Numbers (NINO)**
No validations.

**United States Social Security Numbers (SSN)**

- The serial number must not be 0.
- The SSN must not be a reserved value such as 078-05-1120 and 457-55-5462.
- The group number must be in use for the source area number.

**N** Default. Do not validate the source values.

**WHENINVALID (or WHENINV)**
Determines how to process invalid source values. If this parameter is omitted, the provider does not copy invalid source values to the destination field and skips rows that contain these values.

Enter the following option:

**PRESERVE (or PRE)**
Copy invalid source values to the destination field.

**DISCARDLIMIT (or DLIM)**
Specifies the number of failed rows to discard before the provider stops processing.

## Output formats

The following output formats are available, according to NID type:

**Canadian Social Insurance Numbers (SIN)**

An SIN is a nine-digit number that consists of a one-digit region code number followed by an eight-digit serial number. The first three digits are called the header. The last digit of the serial number is a check digit.

The NID privacy provider generates a masked SIN with a check digit that is calculated based on the preceding masked eight digits of the output value.

The following output formats are available for an SIN. *C* indicates values to copy. *X* indicates values to mask. For example, *3C4X* indicates that the first three characters are copied and the next four characters are masked.

| Fields to mask | Format without separator | Format with dash separator | Format with space separator | Format with period separator |
|---|---|---|---|---|
| Serial number without header digits | CA=3C6X | CA=3C-3X-3X | CA=3C 3X 3X | CA=3C.3X.3X |

| Fields to mask | Format without separator | Format with dash separator | Format with space separator | Format with period separator |
|---|---|---|---|---|
| Serial number and header digits | CA=9X | CA=3X-3X-3X | CA=3X 3X 3X | CA=3X.3X.3X |

**French National Institute for Statistics and Economic Studies numbers (INSEE)**

An INSEE number is a 15-digit number with the following format: *SYYMMDDCCCOOOKK*.

**S** Sex and citizenship information.

**YY** Last two digits of the year of birth.

**MM** Month of birth.

**DD** Department of origin.

**CCC** Commune of origin.

**OOO** Order number.

**KK** Control key or check digits.

The NID privacy provider masks an INSEE according to the following rules:

- If the provider masks the department field, the provider also masks the commune field with a compatible value.
- The provider always masks the order field.
- The provider calculates the check digit field, which is based on the preceding masked 13 digits of the output value.

The following output formats are available for an INSEE. All formats mask the order and check digit fields.

*C* indicates values to copy. *X* indicates values to mask. For example, *3C4X* indicates that the first three characters are copied and the next four characters are masked.

| Fields to mask (in addition to Order and Check Digit) | Format without separator | Format with dash separator | Format with space separator |
|---|---|---|---|
| Sex, Year, Month, Commune | FR=5X2C8X | FR=5X2C6X-2X | FR=5X2C6X 2X |
| Sex | FR=1X9C5X | FR=1X9C3X-2X | FR=1X9C3X 2X |
| Sex, Year | FR=3X7C5X | FR=3X7C3X-2X | FR=3X7C3X 2X |
| Sex, Month | FR=1X2C2X5C5X | FR=1X2C2X5C3X-2X | FR=1X2C2X5C3X 2X |
| Sex, Commune | FR=1X6C8X | FR=1X6C6X-2X | FR=1X6C6X 2X |
| Sex, Department | FR=1X4C8X | FR=1X4C6X-2X | FR=1X4C6X 2X |
| Sex, Year, Month | FR=5X5C5X | FR=5X5C3X-2X | FR=5X5C3X 2X |
| Sex, Year, Commune | FR=3X4C8X | FR=3X4C6X-2X | FR=3X4C6X 2X |
| Sex, Year, Department, Commune | FR=3X2C10X | FR=3X2C8X-2X | FR=3X2C8X 2X |
| Sex, Month, Commune | FR=1X2C2X2C8X | FR=1X2C2X2C6X-2X | FR=1X2C2X2C6X 2X |
| Sex, Month, Department, Commune | FR=1X2C12X | FR=1X2C10X-2X | FR=1X2C10X 2X |
| Sex, Year, Month, Department, Commune | FR=15X | FR=13X-2X 13X | FR=13X 2X |
| Year | FR=1C2X7C5X | FR=1C2X7C3X-2X | FR=1C2X7C3X 2X |

| Fields to mask (in addition to Order and Check Digit) | Format without separator | Format with dash separator | Format with space separator |
|---|---|---|---|
| Year, Month | FR=1C4X5C5X | FR=1C4X5C3X-2X | FR=1C4X5C3X 2X |
| Year, Commune | FR=1C2X4C8X | FR=1C2X4C6X-2X | FR=1C2X4C6X 2X |
| Year, Department | FR=1C2X2C10X | FR=1C2X2C8X-2X | FR=1C2X2C8X 2X |
| Year, Month, Commune | FR=1C4X2C8X | FR=1C4X2C6X-2X | FR=1C4X2C6X 2X |
| Year, Month, Department | FR=1C14X | FR=1C12X-2X | FR=1C12X 2X |
| Month | FR=3C2X5C5X | FR=3C2X5C3X-2X | FR=3C2X5C3X 2X |
| Month, Commune | FR=3C2X2C8X | FR=3C2X2C6X-2X | FR=3C2X2C6X 2X |
| Month, Department | FR=3C12X | FR=3C10X-2X | FR=3C10X 2X |
| Commune | FR=7C8X | FR=7C6X-2X | FR=7C6X 2X |
| Department | FR=5C10X | FR=5C8X-2X | FR=5C8X 2X |

**Italian Fiscal Codes (CF)**

A CF is a 16-character alphanumeric value with the following format: *FFF-NNN-YYMDD-RRRRC*.

**FFF** Encoded surname.

**NNN** Encoded given name.

**YY** Year of birth.

**M** Month of birth.

**DD** Date of birth.

**RRRR** Region code.

**C** Control character.

The NID privacy provider masks a CF according to the following rules:

- The provider masks any consonant that appears in the given name or surname fields as a consonant and masks any vowel as a vowel. If an X appears after a vowel, the provider copies it to the output value.
- The provider calculates the control character field, which is based on the preceding masked 15 digits of the output value.

The following output formats are available for a CF. *C* indicates values to copy. *X* indicates values to mask. For example, *3C4X* indicates that the first three characters are copied and the next four characters are masked.

| Fields to mask | Format without separator | Format with dash separator | Format with space separator |
|---|---|---|---|
| Date of birth, Region | IT=6C10X | IT=3C-3C-5X-5X | IT=3C 3C 5X 5X |
| Surname, Given name, Region | IT=6X5C5X | IT=3X-3X-5C-5X | IT=3X 3X 5C 5X |
| Surname, Given name, Date of birth | IT=11X4C1X | IT=3X-3X-5X-4C1X | IT=3X 3X 5X 4C1X |
| Surname, Given name | IT=6X9C1X | IT=3X-3X-5C-4C1X | IT=3X 3X 5C 4C1X |
| Date of birth | IT=6C5X4C1X | IT=3C-3C-5X-4C1X | IT=3C 3C 5X 4C1X |
| Region | IT=11C5X | IT=3C-3C-5C-5X | IT=3C 3C 5C 5X |
| Surname, Given name, Date of birth, Region | IT=16X | IT=3X-3X-5X-5X | IT=3X 3X 5X 5X |

### Spanish Fiscal Identification Numbers (NIF) and Foreign Identification Numbers (NIE)

An NIF is an eight character value in the following format *NNNNNNN-A*, where the first seven characters are a serial number and the final character is an alphabetic suffix. The suffix is a check digit.

Foreign Spanish nationals use a Foreign Identification Number (NIE), which is a nine character value that uses the same format as an NIF, but is preceded by an X. An NIE uses the following format: *X-NNNNNNN-A*.

The NID privacy provider generates a masked NIF or NIE with a check digit that is calculated based on the preceding masked 7 digits of the output value.

The following output formats are available for an NIF and NIE. The provider masks all characters for each format. NIF and NIE numbers use the same format options. An NIE source value always includes an X prefix in the output value.

| Fields to mask | Format without separator | Format with dash separator | Format with space separator |
|---|---|---|---|
| Serial, Suffix | ES=8X | ES=7X-1X | ES=7X 1X |

### United Kingdom National Insurance Numbers (NINO)

A NINO consists of three parts: two letters (the prefix), six digits (the number), and one optional letter (the suffix).

The NID privacy provider can mask a NINO without a separator or with a separator in either a three- or five-part format.

The following output formats are available for a NINO. *C* indicates values to copy. *X* indicates values to mask. For example, *3C4X* indicates that the first three characters are copied and the next four characters are masked.

To create a NINO without a separator, use the following parameters:

| Fields to mask | Format without separator |
|---|---|
| Prefix, Number | UK=8X1C |
| Number | UK=2C6X1C |
| Prefix, Number, Suffix | UK=9X |

To create a NINO with either a three- or five-part format, use the following parameters:

| Fields to mask | Format with dash separator | Format with space separator | Format with period separator |
|---|---|---|---|
| Prefix, Number (three-part) | UK=2X-6X-1C | UK=2X 6X 1C | UK=2X.6X.1C |
| Prefix, Number (five-part) | UK=2X-2X-2X-2X-1C | UK=2X 2X 2X 2X 1C | UK=2X.2X.2X.2X.1C |
| Number (three-part) | UK=2C-6X-1C | UK=2C 6X 1C | UK=2C.6X.1C |
| Number (five-part) | UK=2C-2X-2X-2X-1C | UK=2C 2X 2X 2X 1C | UK=2C.2X.2X.2X.1C |
| Prefix, Number, Suffix (three-part) | UK=2X-6X-1X | UK=2X 6X 1X | UK=2X.6X.1X |
| Prefix, Number, Suffix (five-part) | UK=2X-2X-2X-2X-1X | UK=2X 2X 2X 2X 1X | UK=2X.2X.2X.2X.1X |

### United States Social Security Numbers (SSN)

An SSN consists of 3 subfields with the following format: *AAAGGSSSS*.

**AAA** Area number. The state in which the SSN is issued generally determines the area number.

**GG** Group number. A group number is assigned based on the area number.

**SSSS** Serial number.

The NID privacy provider generates a masked SSN according to the following rules:

- The provider generates a group number that is appropriate for the area number. The provider uses the most recent group number that is issued by the Social Security Administration for the area.
- Serial numbers begin with 0001 and are incremented by 1 for each additional SSN generated for the area number. When the serial number exceeds 9999, the serial number is reset to 0001 and the provider uses the group number that precedes the number that is most recently issued for the area number.

The following output formats are available for an SSN.*C* indicates values to copy. *X* indicates values to mask. For example, *3C4X* indicates that the first three characters are copied and the next four characters are masked.

| Fields to mask | Format without separator | Format with dash separator | Format with space separator | Format with period separator |
|---|---|---|---|---|
| Group, Serial number | US=3C6X | US=3C-2X-4X | US=3C 2X 4X | US=3C.2X.4X |
| Area, Group, Serial number | US=9X | US=3X-2X-4X | US=3X 2X 4X | US=3X.2X.4X |

## Supported data types

The NID privacy provider supports the following data types for source and destination fields, according to NID type:

**Canadian Social Insurance Numbers (SIN) and United States Social Security Numbers (SSN)**

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| CHAR | CHAR | Fixed size character data that is left justified and space padded. |
| VARCHAR | VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. |
| BIGINT<br>**Note:** Negative BIGINT values are not supported. | LONG_LONG or U_LONG_LONG | An 8 byte unsigned numeric value in the range 0 to 18,446,744,073,709,551,615.<br>**Note:** Optim internally changes U_LONG_LONG to LONG_LONG. |
| DECIMAL | DECIMAL_370 | Packed decimal encoded buffer. |

**French National Institute for Statistics and Economic Studies numbers (INSEE), Italian Fiscal Codes (CF), Spanish Fiscal Identification Numbers (NIF) and Foreign Identification Numbers (NIE), and United Kingdom National Insurance Numbers (NINO)**

| DB2 data type | ODPP equivalent | Description |
|---|---|---|
| CHAR | CHAR | Fixed size character data that is left justified and space padded. |
| VARCHAR | VARCHAR | Character data starting with a short integer value that indicates the length, in bytes, of the character data to follow. |

# Chapter 7. Table Maps

A Table Map provides specifications needed to direct data from source tables to destination tables, match pairs of tables for a Compare Process, or exclude tables from processing. A Table Map may also reference a Column Map for a pair of tables in the Table Map to pair columns with different names or attributes, generate column data, or eliminate individual columns from processing.

For more information about Column Maps, see Chapter 5, "Column Maps," on page 155.

A Table Map is required for an Insert, Load, Restore, Convert, or multi-table Compare Process. You can define a Table Map while setting up a process or independently, by selecting a menu option. Except in a Batch Utility Restore Process, however, an independently created Table Map is not directly referenced by the process; you must use the APPLY command in order to use an independently created Table Map during processing. For more information about the APPLY command, see "Apply a Table Map" on page 261.

Whether a Table Map is defined independently or in setting up a process, the steps are generally the same. This section describes how to define a Table Map through a menu option, and mentions any exceptions for Compare processes. Additional information about Table Maps defined within a process is provided in the discussion of each process in the appropriate *User Manual*.

**Note:**

- An independently defined Table Map is always named and stored in the Optim Directory, while a Table Map defined for a process may be LOCAL (defined for that process only) or named and stored in the Optim Directory.
- For Compare elements that also apply to Archive and Move, references to the source table in this chapter apply to the Source 1 table, and references to the destination table apply to the Source 2 table.
- References to tables also apply to Legacy Tables and Materialized Query Tables.

## Select a Table Map

To edit or create a Table Map, select Option 6.4 Table Maps from the **Main Menu**.

You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu (see "Choose a Definition Option" on page 11 ) and select Option 4 TABLE MAPS. In either case, the Choose a Table Map panel is displayed. Use this panel to name a new Table Map or to select an existing Table Map to modify or delete.

```
----------------------------- Choose a Table Map -----------------------------
Command ===>

Table Map:
  Map ID   ===>
  Map Name ===>

Use '_' for DB2 LIKE character   ===> N     (Y-Yes, N-No)

Rules used to validate Table Map ===> M     (M-Move/Archive, C-Compare)
```

*Figure 106. Choose a Table Map*

## Panel

The Choose a Table Map panel includes:

**Table Map**
The Map ID and Map Name for the Table Map. You can enter an explicit value, DB2 LIKE syntax, or blanks for these prompts in any combination.

**Map ID**
The Map ID for the Table Map that is being created or modified. The default is the previously entered value. Specify 1 to 8 characters.

**Map Name**
The name of the map that is being defined or modified. Specify 1 to 12 characters.

**Note:** You can enter the name of an existing Table Map to use as a model for the Table Map you wish to create. After editing the Table Map, save it under a different name.

**Use '_' for DB2 LIKE character**
Use of the underscore ( _ ) character. Specify Y if the underscore is used as a DB2 LIKE character or N if it is used literally as part of the name.

For example, depending upon the use of the underscore character, A_B is a three-character name containing the characters 'A_B', as entered, or a three-character name that begins with “A” and ends with “B” with any valid character in the middle. The default is N, which means that “_” is not recognized as a DB2 LIKE character.

**Rules used to validate Table Map**
Setting to apply Move or Archive validation rules or Compare validation rules. This prompt is displayed if Compare is installed. Specify:

**M** Comply with rules for Move or Archive.

**C** Comply with rules for Compare.

## Explicit Names

When you supply an explicit **Map ID** and **Map Name** and the Table Map does not exist, the **Specify Table Map Sources** panel (for Move and Archive) or the Specify Table Map Source Types panel (for Compare) is displayed. If the Table Map exists, the Modify Table Map panel is displayed.

## Selection List

A selection list is requested by leaving one or both prompts blank or using DB2 LIKE syntax as criteria. Table Maps that match the criteria are listed on the Select Table Maps panel. If no Table Maps satisfy the criteria, a message is displayed.

## Validation Rules

When creating a Table Map, you must consider the purpose of the Table Map and the Optim component with which it is used. A single Table Map can be used with Archive to restore data, with Compare to select data for comparison, or with Move to migrate extracted data to other tables or platforms.

A Table Map defined for a Compare Process can be used in Move or Archive processing; however, the Move and Archive functions are not available to Compare, and a Table Map defined for Compare processing may not provide functions needed for Move and Archive. Therefore, you may want to use naming conventions or other means to segregate Table Maps defined for Compare processing.

### Move and Archive rules

**Source Tables**
Tables included in an Archive File, Extract File, or specified in an Access Definition.

**Destination Tables**
DB2 or Legacy Tables (Optim Move for Legacy only).

**Column Maps**
Full functionality includes expressions, literals, constants, special registers, and exit routines. Column Maps that include Move functions for aging cannot be used with Archive.

### Compare rules

**Source 1 Tables**
Tables included in an Archive File, Extract File, or specified in an Access Definition.

**Source 2 Tables**
Tables included in an Archive File, Extract File, Access Definition, or DB2.

**Column Maps**
Functionality restricted to mapping columns with dissimilar names, columns with compatible attributes, or excluding columns from processing.

## Table Map Selection List

Leave **Map ID**, **Map Name**, or both blank or use DB2 LIKE syntax to display a selection list of Table Maps on the Select Table Map panel.

- If no Table Maps match the pattern for a selection list, a message is displayed on the Choose a Table Map panel.
- Otherwise, the Select Table Map panel is displayed, listing Table Maps that match the criteria. The panel is shown in the following figure.

```
------------------------------ Select Table Map -------------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 of 9

    ----- Table Map ----- ------ Last Modified -------
Cmd  Map ID      Name        By           Date
--- -------- ------------ -------- -------------------
    ************************* TOP ******************************
___ FOPDEMO  ARCACCT      FOPDEMO  1999-06-22-14.20.29
___ FOPDEMO  ARCORDER     FOPDEMO  1999-06-22-13.42.54
___ FOPDEMO  ARCPERS      FOPDEMO  1999-06-21-23.13.10
___ FOPDEMO  ARCREST      FOPDEMO  1998-10-09-15.59.17
___ FOPDEMO  ARCITEM      FOPDEMO  1999-06-21-22.40.40
___ FOPSAMP  ARCORDER     FOPSAMP  1999-06-22-14.24.56
___ FOPTST   DATE         FOPPROD  1999-06-07-00.49.23
___ FOPTST   TEST1        FOPDEV   1998-10-02-01.46.52
___ FOPTST   TEST2        FOPDEV   1998-10-01-14.36.43
    ********************** BOTTOM ******************************
```

*Figure 107. Select Table Map*

Table Maps that match the criteria are listed. The **Map ID** and **Name** are displayed, as well as Creator ID and date of last modification.

## Description

A User Option determines whether the description of each Table Map is displayed on this panel. See "User Options" on page 372 for information about the **Selection List Format** option.

## Line Commands

The **Cmd** area of the panel is used to enter line commands. The following line commands are available:

**Cmd** The line command entry area. Valid line commands are:

- **S** Select a Table Map.
- **D** Delete a Table Map.
- **C** Copy a Table Map.
- **R** Rename a Table Map.
- **AT** Modify attributes of a Table Map.
- **I** Display information about a Table Map.

See "Object Selection List Functions" on page 14 for more information about available commands.

## Select a Table Map

Use the S line command or the SELECT primary command to select a Table Map to display in the Modify Table Map panel. If the file for the Table Map does not exist, the Specify Table Map Source panel or the Specify Table Map Source Types panel is displayed, prompting for an Archive or Extract File dataset name or Access Definition name. After you provide the name, the Define Table Map panel is displayed, allowing you to define a Table Map. (The Modify Table Map panel is the same as the Define Table Map panel and is described in "Source for a Table Map" on page 253.)

You can also use the SELECT primary command to display a Table Map that is not included on the selection list. For example, to select the Table Map named FOPDEMO.TESTMAP, enter:

```
SELECT FOPDEMO.TESTMAP
```

If the Table Map named on the SELECT primary command does not exist, the Specify Table Map Source panel or the Specify Table Map Source Types panel is displayed for you to create a new Table Map.

## Table Map Attributes

To display the attributes of a Table Map, type I in **Cmd** next to the name of the Table Map. The following figure shows the read-only Table Map Attributes panel for a Move or Archive Table Map, which includes **Source** and **Destination Creator ID** and **Type**. Compare Table Maps will include **Source 1** and **Source 2** information.

```
---------------------------- Table Map Attributes -----------------------------
Command ===>                                                 Scroll ===> PAGE

Map ID                   : FOPDEMO
Map Name                 : ARCORDER

Description              : SAMPLE TABLE MAP
Security Status          : PUBLIC

Last Modified By         : FOPDEMO
Modified On              : 1999-09-30 13.16.41

Source Creator ID        : FOPDEMO
Destination Creator ID   : RESTORED

Source Type              : Extract File
Extract File DSN         : FOPDEMO.ARCHIVE.FILE
Destination Type         : DB2 Tables
```

*Figure 108. Table Map Attributes*

If the **Source Type** is Access Definition, **AD Name** is displayed instead of the Extract File DSN prompt. Also, **Security Status** is displayed only if enabled.

## Source for a Table Map

Creating a Table Map requires information about the tables that are processed. Optim obtains preliminary table information from a designated source to populate the Table Map editor. After the editor is "primed", you can edit the information in it to suit your needs.

If you are creating a Table Map for an Insert, Convert, Load, or Restore Process, the Specify Table Map Source panel is displayed to prompt for the name of an Archive or Extract File or an Access Definition. The source file or Access Definition provides the names of tables with which Optim populates the Table Map editor. If you are creating a Table Map for a Compare Process, the Specify Table Map Source Types panel is displayed to prompt for Source 1 and Source 2 information.

## Specify Table Map Source

The Specify Table Map Source panel is displayed when you indicate that Move rules apply and provide the name of a new Table Map on the Choose a Table Map panel or select a Table Map listed on the Select Table Map panel for which the source file or Access Definition does not exist.

```
-------------------------- Specify Table Map Source --------------------------
Command ===>                                                 SCROLL ===> PAGE

Specify where Source Table List will come from.

Use Source Tables from an Extract or Archive File:
  DSN   ===>

Or, Tables Specified in Access Definition
  GROUP ===>
  USER  ===>
  NAME  ===>
```

*Figure 109. Specify Table Map Source*

### Panel

The Specify Table Map Source panel requires that you provide either an:

**Extract or Archive File**
The name of an Extract or Archive File used to populate the Table Map editor. Specify the file name explicitly by enclosing it in quotes; otherwise the default **Data Set Prefix**, specified on the User Options panel, is prefixed to the name. You can obtain a selection list of data sets by using a wildcard character in the last position.

**Access Definition**
The fully qualified name of an Access Definition used to populate the Table Map editor. Use DB2 LIKE syntax to select from a list of Access Definitions.

After entering the name of a source, use END or press ENTER to display the Define Table Map panel, or use CANCEL to return to the previous panel.

## Specify Table Map Source Types and Sources

The Specify Table Map Source Types panel is displayed when you indicate that Compare rules apply and provide the name of a new Table Map on the Choose a Table Map panel.

The Specify Table Map Source Types panel prompts for the type of source used to populate the Table Map editor.

```
----------------------- Specify Table Map Source Types -------------------------
Command ===>                                                 SCROLL ===> PAGE

COMPARE can process data saved in an Extract File or in DB2 Tables.
Specify source types as follows:

   Source 1 ===> 1    1 - Extract File
                      2 - Set of Data Defined by an Access Definition

   Source 2 ===> 3    1 - Extract File
                      2 - Set of Data Defined by an Access Definition
                      3 - All Rows from Multiple DB2 Tables
```

*Figure 110. Specify Table Map Source Types*

### Source Types

The Specify Table Map Source Types panel requires that you indicate the type for Source 1 and Source 2 as

**1 - Extract File**
An Extract or Archive File is used to populate the Table Map editor.

**2 - Set of Data Defined by an Access Definition**
An Access Definition is used to populate the Table Map editor.

**3 - All Rows from Multiple DB2 Tables**
DB2 Table names are used to populate the Table Map editor. This option is available for Source 2 only.

Any combination of source types is valid. After you select the source types, Compare displays the Specify Table Map Sources panel to prompt for the name of each source. The panel is formatted according to the source types you select.

### Specify Table Map Sources

The prompts to identify Source 1 and Source 2 are displayed on the Specify Table Map Sources panel. Your selection of source types on the Specify Table Map Source Types panel determines the specific prompts that are displayed.

For example, in the following figure, Source 1 is an Extract File and Source 2 is All Rows from Multiple DB2 Tables.

```
------------------------ Specify Table Map Sources ---------------------------
Command ===>                                                 SCROLL ===> PAGE

Source 1: Extract File
  DSN   ===> 'FOPDEMO.PSTUSER.EXTRACT1'

Source 2: All Rows from Multiple DB2 Tables
  Table Map editor will Initially use matching Table Names from the
  Extract File for the Table Map
```

After the sources are identified, use END or ENTER to display the Define Table Map panel.

## Table Map Editor

After you identify the sources and press ENTER, the Table Map editor is displayed, listing the source and destination (or Source 2) tables.

The Table Map editor is also displayed from the Archive RESTORE Process panel, Specify CONVERT Parameters and Execute panel, CREATE Process panel, INSERT Process panel, or LOAD Process panel.

The Table Map editor is presented on two screens, horizontal “pages.” **MORE** preceded or followed by two arrows indicates the presence of a horizontal first or second page. Use the primary commands LEFT and RIGHT or the assigned function keys to scroll the page horizontally.

To provide processing overrides for Restore or Insert processing, you can scroll horizontally to display the portions of the Table Map editor. See “Processing Overrides” on page 257 for more information.

The Table Map editor lists the corresponding source and destination or Source 2 tables after the appropriate headings. Each table is mapped to the table listed on the same line. This panel is called the Modify Table Map panel when you select an existing Table Map to edit.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                MORE>>
 Src 1 CID: FOPDEMO                                    Column
 Src 2 CID: SMITH                                   >>  Map ID ===>

 Source 1 Table Name      Source 2 Table Name      Type   Column Map or "LOCAL"
-------------------- ------------------------>> ------- --------------------
*********************************** TOP ************************************
CUSTOMERS             CUSTOMERS                 TABLE
DETAILS               DETAILS                   TABLE
ITEMS                 ITEMS                     TABLE
ORDERS                ORDERS                    TABLE
********************************** BOTTOM **********************************
```

*Figure 111. Define Table Map*

## Panel

The Table Map editor includes:

**Src CID or**
**Src 1 CID**
The default Creator ID for the source tables, as identified in the source object (Archive File, Extract File, or Access Definition). This value cannot be modified.

**Dest CID or**
**Src 2 CID**
The default Creator ID for the Destination or Source 2 tables. You must enter a valid value. Use the value in **Src CID** to restore, insert, convert, or update data into the source tables or, when in a different DB2 subsystem, to use tables with the same Creator ID.

**Column Map ID**
The default Map ID for any referenced Column Map. If you do not provide a value, you must include the Map ID when referencing a Column Map.

**Extract Tables or Source 1 Table Name**
Names of the source tables. These names are taken from the Archive File, Extract File, or Access Definition and cannot be modified. The fully qualified name is listed if the Creator ID is different from the **Src** or **Src 1 CID**.

**Destination or Source 2 Table Name**
Names of the destination or Source 2 tables. Specify the fully qualified name if the Creator ID is different from the **Dest** or **Src 2 CID**. A table can be referenced only once as the destination; thus, you cannot enter the names of both a table and one or more synonyms, views, or aliases for the table as a destination table.

**Type** Type of object listed in **Destination Table Name**. The possible values are:

**TABLE**
Table

**VIEW** View

**U-MQT**
User-maintained Materialized Query Table

**UNKNOWN**
Non-existent table or no value in **Src 1** or **2 CID**

**A-TABLE**
Alias of a table

**A-VIEW**
Alias of a view

**S-TABLE**
Synonym of a table

**S-VIEW**
Synonym of a view

**NOT INS**
Not insertable (e.g., a joined view or a System-maintained Materialized Query Table)

**UNUSED**
Blank

**TEMPTBL**
Temporary table

**LEGACY**
Legacy Table (for *Move* or *Compare for IMS, VSAM, and Sequential Files*)

**Note:**

- During a Create, Insert, Load, or Restore Process, the CREATE Object List panel is displayed to create UNKNOWN tables. However, you cannot save an independently defined Table Map with UNKNOWN destination tables.
- An asterisk (*) next to **Type** indicates a Restore Action was defined in the Access Definition used to archive the data.

**Column Map**
The name of any Column Maps used to map source and destination or Source 2 columns. Enter the fully qualified name, if Map ID is different from the value in **Column Map ID**. (See "Include a Column Map" on page 262 for more information.)

## Processing Overrides

To provide Insert or Restore processing overrides for individual tables, you can scroll horizontally to display the remaining portion of the Table Map editor.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                 <<MORE
 Src 1 CID: FOPDEMO                                    --Overriding--
 Src 2 CID: SMITH                                   >>  Process Delete
                                                          Mode  Before
 Source 1 Table Name      Source 2 Table Name      Type    U/I/B  Insert
-------------------- ------------------------>> ------- ------- ------
*********************************** TOP ************************************
CUSTOMERS             CUSTOMERS                  TABLE
DETAILS               DETAILS                    TABLE
ITEMS                 ITEMS                      TABLE
ORDERS                ORDERS                     TABLE
********************************** BOTTOM **********************************
```

*Figure 112. Table Map – Processing Overrides*

The following prompts are displayed on the right-hand portion of the panel, and are blank by default.

**Process Mode**
The overriding process mode for selected tables. Specify:

**U** Update only.

**I** Insert only.

**B** Both (update and insert).

**Process Mode** allows you to set the processing options for any table on an individual basis. The default processing options from the Specify RESTORE Parameters and Execute and Specify INSERT Parameters and Execute panels apply for tables with no overriding process parameters.

**Delete Before Insert**
Option to delete all rows in the destination table prior to inserting data, resulting in a set of data that exactly matches data in the Extract File. Specify:

**Y** Delete before Insert.

**N** Do not delete before Insert.

**Note:**

**Delete Before Insert** is not valid for a Restore Process.

## Available Commands

The following commands are available from the Table Map editor.

ACM[2, 4]
ACTIONS[1]
APPLY
ATTRIBUTES[3]
BOTTOM
CANCEL
CLEAR
DOWN
EXPAND
LIST ALL[3]
LIST MAPS
LIST object
LIST SUBS[1]
LIST UNUSED[3]
MAP[2]
OPTIONS
POPULATE[2]
PREFIX
REL[1]
SAVE
SUFFIX
TOP
UP

**Note:**

1. Restore Process only.
2. Not available for the Create Process.
3. Compare Process only.
4. Not available for the Convert Process.

See the *Command Reference Manual,* Primary Commands for details about these commands.

## Table Map Description

Use the ATTRIBUTES command to display the Object Attributes panel, which allows you to specify a description for the Table Map. This panel provides a 40-character area to display and edit the description. (Site management determines whether this panel also displays a prompt for Security Status.) For additional information about the Object Attributes panel, see "Specify Description and Security Status" on page 380.

## Choose Access Method

If necessary, you can use the ACM command to open the Choose Access Method pop-up dialog.

```
+-------- Choose Access Method -------+
|                             1 of 4  |
| Access Method Values:               |
|   K - Key Lookup                    |
|   S - Table Scan                    |
|   blank - Software Chooses          |
|                                     |
|                              Access |
| Destination Table Name       Method |
| -------------------------  ------   |
| ************** TOP ***************** |
| FOPDEMO.CUSTOMERS                   |
| FOPDEMO.DETAILS                     |
| FOPDEMO.ORDERS                      |
| FOPDEMO.ITEMS                       |
| ************ BOTTOM *************** |
+-------------------------------------+
```

*Figure 113. Choose Access Method*

This pop-up allows you to override the default method (scan or key lookup) for accessing each table in the Extract or Archive File. A scan reads all rows in a table at one time; while a key lookup locates rows using a WHERE clause to search for primary or foreign key values.

**Note:**

- Changing the access method to Key Lookup in a Table Map only affects tables where the processing method is Update. If the processing method for a table is Insert, the access method is Table Scan.
- To set the access method for all listed tables, use the ACM command with a B (blank), K (key lookup), or S (table scan) operand. For example, enter ACM B to blank the access method for all listed tables.

For more information, see "Table Access Strategy" on page 394.

## Save the Table Map

After editing a Table Map, you can save it, using the SAVE command with the fully qualified Table Map name as an operand. The saved Table Map is available for other processes or to other users.

To save a Table Map:

- Use the END command to exit the Table Map editor and automatically save any modifications to the Table Map.
- Use the SAVE command without a name operand to save the updated Table Map or provide a new name to save as a new Table Map.

You can save a Table Map under a new name, retaining the original, prototype version under its current name.

**Note:**
- When defining a Table Map as part of a process, you can save the Table Map by specifying a fully qualified name. If you do not explicitly save the Table Map, it is not stored in the Directory and is available for the current process only.
- If the Table Map references a Legacy Table (*Move* or *Compare for IMS, VSAM, and Sequential Files* only), the Associate Legacy Tables with Data Destinations panel is displayed when you exit the Define Table Map panel. The Associate Legacy Tables with Data Destinations panel allows you to associate a Legacy Table with a data source. For details, see "Associate Legacy Tables with Data Destinations" on page 266.

Use END to return to the Choose a Table Map panel. Use the CANCEL command to abandon any changes and return to the previous panel.

## Edit Destination or Source 2 Table List

Initially, Optim provides a list of destination or Source 2 names that match the names of tables in the source file.

To edit the list, you can:
- Type over a name or select a new name from a list. You can request a selection list using the LIST primary command.
- Prefix a name with a Creator ID different from the default in **Dest CID** or **Src 2 CID**.
- Prefix all names with a string using the PREFIX primary command.
- Append a string to all names using the SUFFIX primary command.
- Clear all names using the CLEAR primary command.
- Bypass a table in a process by replacing the Destination or Source 2 table name with blanks.
- Apply a Table Map to the list. (See "Apply a Table Map" on page 261.)

### Table Type

Optim automatically revises the **Type** value to match the destination or Source 2 table name. If you modify the **Dest CID** or **Src 2 CID**, the new CID applies to any table name entries specified without Creator ID, and the **Type** values revised accordingly.

For example, if the **Destination Table Name** entry is CUSTOMERS and **Dest CID** is OPTIM, the fully qualified table name is OPTIM.CUSTOMERS. Using the fully qualified name, the Table Map references a database table and the **Type** is shown as TABLE. If you change the **Dest CID** to FOPUSER, causing the Table Map to reference a nonexistent table named FOPUSER.CUSTOMERS, the **Type** value is changed to UNKNOWN. To avoid changing a table name when changing the **Dest CID**, you can supply the Creator ID with the **Destination Table Name**.

### LIST

Use the LIST command to obtain a selection list of available aliases, tables, synonyms, or views for **Destination Table Name**. For example, to obtain a selection list of tables with the destination Creator ID, use the command LIST TABLES. If the destination Creator ID is "SMITH," a selection list of tables with

this Creator ID is displayed when LIST TABLES is entered with no operands. You can also list tables for a different Creator ID by providing the Creator ID with the LIST TABLES command, as in LIST TABLES FOPUSER.%.

While the selection list is displayed, the Extract Tables or Source 1 Table entries are numbered, sequentially, in ascending order. You can map tables by entering the value in the **Num** for a table in the list.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                 MORE>>
 Src 1 CID: FOPDEMO                                    Column
 Src 2 CID: SMITH                                             ==>
                          +------------- Tables --------------+
 Source 1 Table Name   Type |                                 | ap or "LOCAL"
--------------------- ----- | Select Items by Matching 'Num'  | -------------
*************************** |                                 | *************
CUSTOMERS             TABLE | Num CreatorID.TableName   1 OF 5 |
DETAILS               TABLE | --- --------------------        |
ITEMS                 TABLE | ********* TOP **********        |
ORDERS                TABLE | 1__ SMITH.XXCUSTOMERS           |
*************************** | 2__ SMITH.DETAILS               | *************
                          | 3__ SMITH.ITEMS                 |
                          | 4__ SMITH.ORDERS                |
                          | 5__ SMITH.SHIP_TO               |
                          | ******** BOTTOM ********        |
                          +---------------------------------+
```

*Figure 114. List Available Destination Tables*

Scroll the list of tables using the UP, DOWN, TOP, and BOTTOM commands, or PF keys assigned to these functions. Use END to return to the Table Map editor. As shown in the following figure, Optim automatically populates the **Destination Table Name** list with the selected table names. The **Type** for the newly listed tables in this example is TABLE.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                 MORE>>
 Src 1 CID: FOPDEMO                                    Column
 Src 2 CID: SMITH                                   >>  Map ID ===>

 Source 1 Table Name      Source 2 Table Name      Type   Column Map or "LOCAL"
--------------------- ------------------------>> ------- --------------------
*********************************** TOP ************************************
CUSTOMERS             XXCUSTOMERS                TABLE
DETAILS               DETAILS                    TABLE
ITEMS                 ITEMS                      TABLE
ORDERS                ORDERS                     TABLE
SHIP_TO               SHIP_TO                    TABLE
********************************** BOTTOM **********************************
```

*Figure 115. Define Table Map - Return from LIST*

## Apply a Table Map

Use the APPLY command to overlay all or part of the Table Map editor with specifications from another Table Map. When the source table names in the applied Table Map match the source table names

displayed in the Table Map editor, corresponding destination or Source 2 table names and any Column Map entries are copied to the Table Map editor. APPLY also transfers data destination associations for source Legacy Tables if an association is not already specified.

APPLY is useful to create a new Table Map from several existing Table Maps or to create a Table Map that is modeled on an existing or independently defined Table Map. For example, a "master" Table Map can establish default table mapping for a common pool of tables. A user can apply and edit the master Table Map for any process (even though processes do not involve the same set of tables) to obtain consistent mapping with a minimum of effort.

By default, the Destination or Source 2 Creator ID and table names are replaced when names of source tables match exactly (both table names must be qualified or not qualified). Other entries in the Table Map editor remain. Operands for the APPLY command allow you to:

- Clear **Dest CID** and **Destination Table Name** before applying the Table Map.
- Limit applied entries to the blank portions of the Table Map editor.
- Limit the application to table names, Column Map names, Destination or Source 2 names, or all.

For example, to apply the Column Map entries from a Table Map named SMITH.TMAP, enter:

```
APPLY ADD COLMAPS SMITH.TMAP
```

A Column Map name from the Table Map SMITH.TMAP is inserted in blank Column Map portions when the source table is the same for both Table Maps. Irrelevant Column Maps defined in SMITH.TMAP are ignored. Any Column Maps named on the currently displayed Table Map are retained.

### Selection List for APPLY

Use DB2 LIKE syntax with the APPLY command to display a selection list of Table Maps. For example, APPLY SMITH.% generates a selection list of all Table Maps that have the Map ID SMITH; APPLY%.TMAPS generates a list of Table Maps named TMAPS for all Map IDs; and APPLY with no operands generates a selection list of all available Table Maps.

## Include a Column Map

By default, columns with the same name and matching attributes are mapped. A Column Map must be used when column names or attributes between tables do not match, when data transformations are needed, or to exclude one or more columns from processing. A Column Map used in a Convert, Insert, Load, or Restore Process can modify data or split data from one table into several tables or copy one source column to several destinations. Column Maps also allow you to create a single destination table from the data in a joined view. A Column Map cannot transform data during a Compare Process, however.

To reference a Column Map from a Table Map, you must enter the **Column Map** name for a pair of tables in the Table Map. To use a previously undefined Column Map, enter a unique name in **Column Map** and press ENTER. You are prompted to confirm that you are creating a new Column Map. Use END to return to the Table Map editor or press ENTER to display the Define Column Map panel. To create a new local Column Map, enter LOCAL for a pair of tables and press ENTER. The Define Internal Column Map panel is displayed.

You must use the MAP command to edit an existing Column Map after entering the name on the Table Map editor. See "Edit Column Map" on page 266 for information.

To enter a Column Map name on the Table Map editor, you can:

- Type the Column Map name, entering the name of an existing Column Map or of a new Column Map that you will define. You can also enter "LOCAL" to use a Column Map that is embedded in and used with only the current Table Map.

**Note:** You cannot restart or retry a process that uses a local Column Map.

- Use the LIST MAPS command to display and select from a list of Column Maps.
- Use the POPULATE command to automatically insert names of existing Column Maps for every pair of source and destination tables.

The following figure shows Column Maps entered for three of the five tables listed.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                 Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                MORE>>
 Src 1 CID: FOPCOMP                                      Column
 Src 2 CID: FOPDEMO                                   >>  Map ID ===> FOPDEMO

 Source 1 Table Name      Source 2 Table Name      Type   Column Map or "LOCAL"
-------------------- ------------------------>> ------- --------------------
*********************************** TOP ************************************
CUSTOMERS             CUSTOMERS                  TABLE   CUSMAP
DETAILS               DETAILS                    TABLE   DETMAP
ITEMS                 ITEMS                      TABLE   ITMMAP
ORDERS                ORDERS                     UNKNOWN
SHIP_TO               SHIP_TO                    TABLE
********************************** BOTTOM **********************************
```

*Figure 116. Column Maps Specified*

## Column Map ID

You can use **Column Map ID** on the Table Map editor to provide a default Map ID for any Column Map names referenced by the Table Map. If no default **Column Map ID** is specified, you must enter the fully qualified Column Map name on the Table Map editor.

## LIST MAPS

The LIST MAPS command allows you to select from a list of Column Maps. Use DB2 LIKE syntax in the operand as criteria for the selection list. If no operand is specified, the list includes all Column Maps.

To use the LIST MAPS command, type the command, position the cursor on the line for the appropriate pair of tables and press ENTER. If a single name matches the criteria, it is automatically inserted. If two or more names match the criteria, a selection list is displayed, as shown in the following figure.

```
-- Define Table Map: FOPDEMO.NEWMAP ------------------------------------------
Command ===>                                                Scroll ===> PAGE

Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,ACM,ACT,CLEAR, END when Done
Source 2 May be any DB2 Tables or Views                                MORE>>
 Src 1 CID: FOPCOMP                                      Column
 Src 2 CID: FOPDEMO                                   >>  Map ID ===> FOPDEMO

 Source 1 +--------------------Select One Column Map--------------------+ OCAL"
--------- | Cmd     MapID.MapName      Source 1    Source 2     1 OF 4  | -----
********* | --- ---------------------------------------------          | *****
CUSTOMERS | ********************** TOP ***********************          |
DETAILS   | ___ FOPDEMO.RESTORE       FULL MATCH  FULL MATCH            |
ITEMS     | ___ FOPDEV.ORDERS         NO MATCH    TABLE MATCH           |
ORDERS    | ___ FOPPROD.RESTORE       NO MATCH    TABLE MATCH           |
SHIP_TO   | ___ FOPPROD.TEST          NO MATCH    NO MATCH              |
********* | ********************** BOTTOM ********************          | *****
          +-------------------------------------------------------------+
```

*Figure 117. Select One Column Map*

The fully qualified names of Column Maps that match the criteria are listed with notation that describes the correlation between the tables mapped in the Column Map and those listed on the Table Map editor. Matches are described as:

**FULL MATCH**
The Creator ID and table name on the Table Map editor match those in the Column Map. In Figure 117, both table names in the first Column Map match those on the Table Map editor.

**TABLE MATCH**
The Creator ID on the Table Map editor does not match the Creator ID in the Column Map, but the table names match. In Figure 117, the destination table name for the second and third Column Maps matches the destination table name on the Table Map editor, but the Creator ID does not.

**NO MATCH**
Neither the Creator ID nor table name on the on the Table Map editor matches those in the Column Map. In Figure 117, the source table name and Creator ID for the second and third Column Maps do not match those specified on the **Define Table Map** panel. These maps are probably not candidates for use.

Use the S line command to select a Column Map, and press ENTER or use END to return to the **Define Table Map** panel. The name of the selected Column Map is inserted in **Column Map** for the selected pair of tables.

## POPULATE

Use the POPULATE command to insert Column Map names for every pair of source and destination tables specified on the Table Map editor, beginning with the first pair. The processing sequence for each pair of tables is as follows:

1. Optim checks for a full match. (The Creator ID and table names in the Column Map and Table Map must match exactly.) If only one Column Map is identified, the name is inserted in **Column Map** and processing continues with the next pair of tables.

   If there is a full match with more than one Column Map, a selection list of the matching Column Maps is displayed. (The selection list of Column Maps is similar to the one in Figure 117. However, FULL MATCH is indicated for the source and destination columns of each Column Map.)
2. If no full matches are found, Optim checks for table matches. (The table names of the source and destination tables on the Column Map and Table Map match exactly, but the Creator IDs for one or both tables do not match.) If only one Column Map is identified, the name is inserted in **Column Map** and processing continues with the next pair of tables.

If there is a table match with more than one Column Map, a selection list of the matching Column Maps is displayed. (The selection list of Column Maps is similar to the one in Figure 117 on page 264. However, TABLE MATCH is indicated for the source and destination columns of each Column Map.)

3. If no full or table matches are found, Optim processes the next pair of tables in the Table Map.

When POPULATE processing for a specific pair of tables is complete, the Table Map display scrolls to the next pair of tables on the list and processes that pair, until all are processed. When processing is complete, the Table Map editor is re-displayed with the appropriate **Column Map** names populated.

### POPULATE Operands

You can use DB2 LIKE syntax as criteria for Column Map names inserted by POPULATE. For example, to populate the Table Map with Column Map names having the Map ID FOPDEMO, type:

```
POPULATE FOPDEMO.%
```

To populate the Table Map with Column Map names that begin with RES and have the Map ID FOPDEMO, type:

```
POPULATE FOPDEMO.RES%
```

Other operands allow you to clear Column Map names on the panel prior to executing the POPULATE command or replace names if a new match is found. You can also limit POPULATE processing to insert only Column Map names for which there is a full match. (See the POPULATE command in the *Command Reference Manual* for details.)

### Selection List for POPULATE

When a selection list is displayed during POPULATE processing, use the S line command and press ENTER or use END to select a Column Map. The name of the selected Column Map is inserted for the current pair and processing continues with the next pair.

Use CANCEL to terminate POPULATE processing and return to the **Define Table Map** panel. Any inserted Column Map names remain on the Define Table Map panel.

### Considerations for Naming Conventions

Naming conventions can be used to identify logical groups of Column Maps and to make best use of the POPULATE command. For example, Column Maps created to insert data to a specific table might include the table name, and the Map ID could identify the data model for the extracted data. Using the POPULATE command with DB2 LIKE syntax in the name operand automatically identifies and inserts the desired set of uniquely named Column Maps for all listed table pairs.

For example, assume two sets of three Column Maps have been defined for Insert processing to the production database from tables extracted from different production data models.

**Column Map Set 1**

FOP02.ORDERS

FOP02.ACCTCUST

FOP02.DETAILS

**Column Map Set 2**

FOP03.ORDERS

FOP03.ACCTCUST

FOP03.DETAILS

Further, assume that the names of the tables defined in each of these maps are a FULL MATCH. By specifying POPULATE FOP03.% only the Column Maps with Map ID FOP03 are identified and automatically inserted. This feature is especially useful when the list of source tables on the Define Table Map panel is extensive and all desired Column Maps have been named following a specific convention.

### Edit Column Map

To edit a Column Map, use the MAP command and position the cursor on the row with the name of the desired Column Map or enter the Column Map name as an operand.

Optim compares the names of tables mapped by the Column Map to the names on the Table Map editor. If table names and column names match, the Column Map is displayed in the Column Map editor. If there is a discrepancy, however, you are prompted to confirm. For example, if one or both tables in a selected Column Map named FOPDEMO.TEST1 do not match the pair of tables on the Table Map, the following confirmation message is displayed.

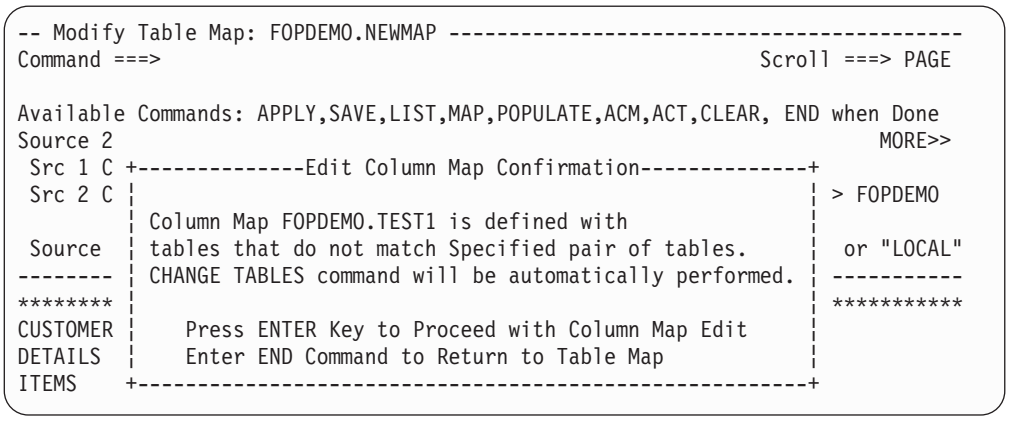



*Figure 118. Edit Column Map Confirmation*

Press ENTER to proceed or use END to return to the Table Map editor. If you proceed, Optim changes the table names in the Column Map to match those on the Table Map editor and displays the Modify Column Map panel.

See "Column Map Editor" on page 162 for detailed information on editing a Column Map.

When you have completed modifying the Column Map, use END to return to the Table Map editor.

## Associate Legacy Tables with Data Destinations

For *Move* or *Compare for IMS, VSAM, and Sequential Files*, a Legacy Table referenced in a Table Map must be associated with a specific data source in order to be used in an Insert Process. Each time you exit the Table Map editor using END, the Associate Legacy Tables with Data Destinations panel is displayed. The panel contents vary depending on whether the Legacy Tables in the Table Map reference IMS data, VSAM or sequential files, or both.

For more information about Legacy Tables, see the *Move User Manual*, Legacy Tables or *Compare for IMS, VSAM, and Sequential Files*.

In the following example, the Table Map references two Legacy Tables as the source tables (FOPLEG.LITEMS and FOPIMS.I$DETAILS) and as the destination tables (FOPLEG2.LITEMS and FOPIMS2.I$DETAILS). FOPLEG.LITEMS and FOPLEG2.LITEMS reference VSAM files and FOPIMS.I$DETAILS and FOPIMS2.I$DETAILS reference IMS data.

```
--------------- Associate Legacy Tables with Data Destinations ----------------
Command ===>                                                Scroll ===> PAGE

Source Dataset Prefix                    :                            1 of 2
Overriding Destination Dataset Prefix ===>                            MORE>>

Source Legacy Table /       Source Data /                      Dest
                            Destination Dataset                Status
Destination Legacy Table    IMS--Segment    DBD      PSB    PCB IMSID DBRC LOG
--------------------------- ---- -------- -------- -------- --- ---------- ---
********************************* TOP ************************************
FOPLEG.LITEMS               'FOPRT.LEGACY.ITEMS'               LEGACY
  FOPLEG2.LITEMS              'FOPRT.LEGACY.ITEMS'
FOPIMS.I$DETAILS            IMS  DETAILS  SALEHDAM SALHDAMA 1
  FOPIMS2.I$DETAILS                                SALHDAMA 1          Y    Y
******************************** BOTTOM ***********************************
```

*Figure 119. Associate Legacy Tables with Data Destinations*

**Note:** The labels displayed on this panel vary based on what type of legacy tables are displayed (i.e., IMS, VSAM, or sequential files). The following labels are displayed ***for IMS files only***: IMS--Segment, DBD, PSB, PCB, IMSID, DBRC, and Log. The following labels are displayed ***for VSAM or sequential files only***: Source Dataset Prefix, Overriding Destination Dataset Prefix, and Dest Status. If more than one file type is displayed, a combination of labels is displayed.

## Panel

This panel includes

**Source Dataset**
**Prefix** The prefix for datasets associated with the VSAM Legacy Tables specified on the Associate Legacy Tables with Data Sources panel.

**Overriding**
**Destination**
**Dataset Prefix**
Optional 1- to 8-character prefix used by the datasets associated with the VSAM Legacy Tables in the Table Map.

**Source Legacy Table/**
**Destination Legacy Table**
Name of the source and destination Legacy Tables referenced in the Table Map.

**Source Data/**
**Destination Dataset**
If the Table Map references a Legacy Table that describes a VSAM or sequential file, the names of the dataset for the source and destination Legacy Tables are displayed.

The **Source Data** is specified in the Access Definition used to create the Extract File and cannot be modified.

The **Destination Dataset** is specified as a default in the Destination Legacy Table. If no dataset is specified, **Destination Dataset** is blank. To enter or modify the Data Destination:

- Type the fully qualified **Destination Dataset** name, enclosing it in single quotes.
- Type a dataset name, without delimiting quotes. The dataset name is prefixed with the default prefix specified in User Options unless you provide an **Overriding Destination Dataset Prefix**. A maximum of 17 characters can be entered as the Overriding Dataset.
- Use a shortcut. Use "=" to copy the **Destination Dataset** name from the preceding entry. Use "=s" to copy the **Source Data** from the current entry into **Destination Dataset** (in Figure 119),

using the "=s" shortcut for FOPLEG.ITEMS copies the **Source Data** name, FOPRT.LEGACY.ITEMS, to the **Destination Dataset**).

**IMS--Segment**
If the Table Map references a Legacy Table that describes IMS data, **IMS** is designated and the source and destination segments within the DBD associated with the Legacy Table are displayed. These values cannot be modified.

**DBD** If the Legacy Table describes IMS data, the DBDs associated with the source and destination Legacy Tables are displayed and cannot be modified.

**PSB** If the Legacy Table describes IMS data, **PSB** lists the PSB associated with the source Legacy Table in the Table Map. You can specify the 1- to 8-character name of the PSB used to override the PSB for the destination Legacy Table specified in the Retrieval Definition. If no Retrieval Definition has been created for the destination Legacy Table, **PSB** is blank and you must enter a valid PSB.

The PSB provides access to the IMS services that Move requires to access the database records.

The PSB must be included in the PSB library referenced in the Environment Definition for the destination Legacy Table. Specify an asterisk to generate a selection list that includes PSBs referenced in the Environment Definition for the IMS Legacy Table.

**PCB** If the Legacy Table references IMS data, **PCB** lists the PCB number associated with the source Legacy Table in the Table Map. You can specify the PCB number used to override the PCB number specified in the Retrieval Definition for the destination Legacy Tables. If no Retrieval Definition has been created for the Legacy Table, **PCB** is blank and you must enter a valid PCB number.

The PCB must exist within the specified PSB and grant Move or Compare the authorization to manipulate the data. Specify an asterisk to generate a selection list that includes PCBs in the specified PSB.

**Dest Status**
If the Table Map contains a Legacy Table that references a VSAM or sequential file, **Dest Status** displays the status of the destination table. Possible values include:

**LEGACY** A **Destination Dataset** is specified.

**MISSING** A **Destination Dataset** has not been specified.

**IMSID**
If the Table Map contains a Legacy Table that references IMS data, **IMSID** lists the IMS System ID associated with the source Legacy Table in the Table Map. You can specify the IMS ID used to override the ID specified in the Environment Definition for the destination Legacy Tables. The IMS ID is required to access the IMS data when allocated to a control region (i.e., the data is online to IMS).

**DBRC** This entry is valid only for IMS processing in DL/I mode (i.e, when an IMS ID is not specified). If appropriate, enter **Y** for yes to use Database Recovery Control (DBRC) to control logging and perform database recovery; otherwise enter **N** for no. IMS uses the online log datasets (OLDS) if the database is accessed in BMP or DBB mode.

The default for a HALDB (High Availability Large DataBase), is **Y**, and that entry cannot be changed.

DBRC use is optional for a non-HALDB, such as HIDAM, HDAM, HISAM, etc. Thus, you may specify **Y** for a non-HALDB, but it is not required.

**LOG** This entry is valid only for IMS processing in DL/I mode (i.e, when an IMS ID is not specified). If appropriate, enter **Y** for yes to use an IMS log to perform database recovery; otherwise enter **N** for no.

If you specify Y, you must specify a dataset name for the IMS log on the Associate IMS Segments with IMS Database Datasets panel. The DD Name "IEFRDER" is used to allocate the log dataset on that panel.

If a PSB with a Processing Option (PROCOPT) other than G (Get) is used while accessing a HALDB in DL/I mode (i.e., an IMS region name is not specified), you must specify the name of the dataset to be allocated for DD Name IEFRDER on the Associate IMS Segments with IMS Database Datasets panel.

After you specify the IEFRDER dataset name and exit the Associate IMS Segments with IMS Database Datasets panel, the Allocate Dataset panel automatically displays. You must provide sufficient Primary and Secondary space units on that panel to allocate the IEFRDER dataset. Failing to do so will cause IMS to abort processing and lock the database from further updates until a recover/rollback is done.

If no IMS Legacy Tables are referenced in the Table Map, use END to return to the Choose a Table Map panel from the display. If one or more IMS Legacy Tables are referenced and an IMS ID was not specified, use END to display the Associate IMS Segments with IMS Database Datasets panel.

## Delete/Insert Root Segment

When you scroll horizontal, the next page of the Associate Legacy Tables with Data Destinations panel is displayed. If one of the destination tables is a Legacy Table that references IMS data, the following column is displayed:

```
--------------- Associate Legacy Tables with Data Destinations ----------------
Command ===>                                                 Scroll ===> PAGE

Source Dataset Prefix                    :                          1 of 2
Overriding Destination Dataset Prefix ===>                          <<MORE

Source Legacy Table /       Source Data /                   Delete/Insert
                            Destination                     Root Segments
Destination Legacy Table    IMS--Segment    DBD               On Update
--------------------------- ---- -------- --------          -------------
********************************* TOP ********************************
FOPLEG.LITEMS               'FOPRT.LEGACY.ITEMS'
  FOPLEG2.LITEMS              'FOPRT.LEGACY.ITEMS'
FOPIMS.I$DETAILS            IMS  DETAILS  SALEHDAM                N
  FOPIMS2.I$DETAILS
******************************** BOTTOM ********************************
```

*Figure 120. Delete/Insert Root Segments*

**Delete/Insert**
**Root Segments**
**On Update**

Option to more accurately update IMS segments. This option is used only when the processing mode for an Insert Process is update only or both (update and insert).

**Y** Update the IMS database by deleting matching entries from the destination root segment and any child segments and inserting the data from the Extract File.

**N** Use the Extract File to update the IMS database. If you try to update an IMS segment that does not have a unique key, the process fails and an error message is displayed.

## Associate IMS Segments with IMS Database Datasets

When you use END to exit the Associate Legacy Tables with Data Destinations panel and one or more Legacy Tables reference IMS data and an IMS ID was not specified, the Associate IMS Segments With IMS Database Datasets panel is displayed. This panel allows you to override the default IMS Database

Dataset Name specified in the Retrieval Definition.

```
------------- Associate IMS Segments With IMS Database Datasets --------------
Command ===>                                                 Scroll ===> PAGE

Overriding Destination Dataset Prefix ===>                            1 of 7


      DBD    Segment  DD Name  Destination IMS Database Dataset Name
    -------- -------- -------- ----------------------------------------
    ******************************** TOP ********************************
    ITEMS    ITEMSDBD ITEMDD
             -IMSLOG  IEFRDER
    ******************************** BOTTOM *****************************
```

*Figure 121. Associate IMS Segments With IMS Database Datasets*

## Panel

This panel includes:

**Overriding Destination Dataset Prefix**
Optional 1- to 8-character prefix used when specifying IMS Database Dataset Names.

**DBD** The DBDs referenced by the destination Legacy Tables in the Table Map. This value cannot be modified.

**Segment**
The segments within the specified DBD. This value cannot be modified.

**DD Name**
The names of the DD (i.e., the physical datasets) associated with each segment. This value cannot be modified. If IMS logging was requested, the DD Name IEFRDER is displayed (along with the pseudo-Segment IMSLOG) to identify the Log dataset.

**Destination IMS Database Dataset Name**
Specify the IMS Database Dataset Name to override the database dataset name specified in the Retrieval Definition.

Enter the location of the IMS Database Dataset associated with each DD Name in the DBD. This data is then associated with the named Legacy Table during processing.

A Site Option (Require IMS Data Set Names) determines whether you can omit the data set name to allow IMS to dynamically allocate the data set. All users can specify '$MDA' as the data set name to choose dynamic allocation, regardless of this Site Option.

If IMS logging was requested, you must specify the Log dataset for DD Name IEFRDER. If a default name is displayed from the Provide Retrieval Information panel, you can override that name, if needed.

After you specify the IEFRDER dataset name and exit the Associate IMS Segments with IMS Database Datasets panel, the Allocate Dataset panel automatically displays. You must provide sufficient Primary and Secondary space units on that panel to allocate the IEFRDER dataset. Failing to do so will cause IMS to stop processing and lock the database from further updates until a recover/rollback is done.

**Note:** You do not have to specify a dataset name for a HALDB because the appropriate dataset name will already be known to the IMS subsystem, but you do have to specify the IEFRDER dataset if it was not defaulted from the retrieval information specified on the Provide Retrieval Information panel.

Use END to return to the Choose a Table Map panel from the display. Use the CANCEL command to abandon any changes made on this panel and return to the Define Table Map panel.

## Archive Actions

Archive provides a facility that allows you to define actions to be executed at one or more predefined points (action events) during an Archive, Delete, or Restore Process. Each action is a single SQL statement or call to a stored procedure that may be included in the Access Definition or in the Restore Process Table Map. An action event can have only one SQL statement associated with it, although action events can share an SQL statement association.

For information about defining actions in an Access Definition, see "Archive Actions" on page 60.

You can define actions in a Table Map for the Restore Process. Restore Actions defined in the Table Map will override or disarm any Restore Actions defined in the Access Definition. The Table Map can also include Restore Actions independent of actions in the Access Definition.

### ACTIONS Primary Command

To invoke the Select an Action To Be Defined panel for a table, use the ACTIONS primary command with the desired table name as an operand, or enter the ACTIONS primary command, position the cursor on the desired table name, and press Enter.

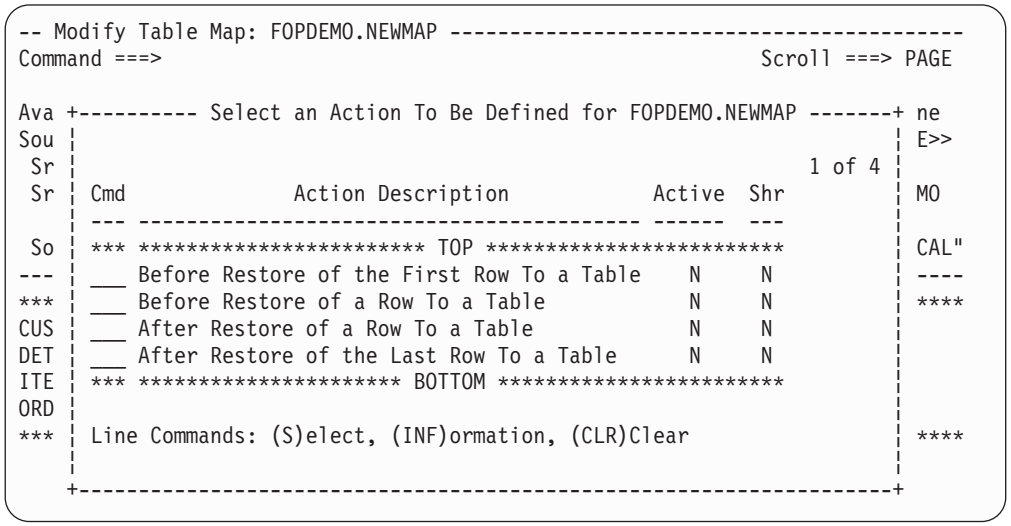



*Figure 122. Select an Action To Be Defined*

### Panel

This panel includes:

**Cmd** The line command entry area. Valid line commands are:

- **S** Select the action event.
- **INF** Display a list of action events that share the SQL statement.
- **CLR** Clear the SQL statement association for the action event.

**Action Description**
The event and time for the action. Select:

Before Restore of the First Row To a Table

Before Restore of a Row To a Table

After Restore of a Row To a Table

After Restore of the Last Row To a Table

**Active** Indicator for assignment of SQL statement:

**N** No SQL statement is assigned.

**Y** An SQL statement is assigned.

**Shr** Indicator for shared status of an assigned SQL statement:

**N** No associated SQL or SQL is not shared.

**Y** Associated SQL is shared with another action.

**Y*** Associated SQL is defined for this action and is shared with another action.

## Define the Action

Use the S or SEL line command to select one or more action events. Archive displays the Enter an SQL Statement panel for each selection, in sequence.

```
------- Enter an SQL Statement to be used by the Action Being Defined ---------
Command ===>                                                 Scroll ===> PAGE

              Before Restore of a Row To a Table
Cmd                                                                    1 of 8
--- ------------------------------------------------------------------------
*** ******************************** TOP ********************************
___
___
___
___
___
*** ***************************** BOTTOM ******************************

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the LIST VARIABLES command to add Action Variables, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
Use the SHARESQL command to select an Action whose SQL you wish to use
Use the COPYSQL command to select an Action whose SQL you wish to copy
```

*Figure 123. SQL Statement*

Use the Enter an SQL Statement panel to specify the complete SQL statement or call to a stored procedure to be executed when the selected event in the Restore Process occurs. For example, before restoring a row, you might use an action SQL statement to keep a record of restorations by inserting the primary key value into a table. As another example, you could place an indicator in a row that has been restored.

The SQL statement is displayed in segments of 8 lines of 72 positions each. A maximum of 200 lines are available in the scrollable area for specifying the SQL and self-documenting comments. Standard DB2 conventions apply to comments; each comment line must begin with two hyphens (--).

Use standard DB2 SQL format for the SQL statement. Any leading or trailing spaces typed on a line are maintained; only the spaces at the end of the last line of the SQL statement are deleted.

## Line Commands

You can perform several editing functions using line commands. Enter the desired command at the Cmd prompt. The functions and their line commands are:

**Copy** C*n*, CC

**Insert** I*n*

**Repeat**
R*n*, RR

**Delete** D*n*, DD

**Move** M*n*, MM

For Copy or Move operations, use A (After) or B (Before) to indicate the new destination. After specifying the new destination, press Enter and the copied or moved line appears, respectively, after or before the indicated location.

## Column Variables

A column variable can be used in the statement for any action event except one executed before processing the first row or after processing the last row. The column variable must be preceded by a colon (:) in the statement.

## Substitution Variables

You can use a substitution variable that has been defined on the Substitution Variable Display panel. The name of the substitution variable must be preceded by a colon (:). (For more information, see "Substitution Variables" on page 70.)

## Action Variables

Action Variables are built-in functions that can be used to return information about the specific action. A variable must be preceded by a colon (:) in the SQL Editor.

**General Variables**: The following General Variables are available to all action SQL statements. Each variable name is shown with its return value type and description.

**FOP_ACTION**
integer

A code that identifies the type of action being executed. (See return values.)

**FOP_ACTION_TEXT**
char (20)

A text string that identifies the type of action being executed. (See return values.)

**FOP_ARCHIVE_FILE_NAME**
char (44)

Name of the Archive File being processed.

**FOP_INDEX_FILE_NAME**
char (44)

Name of the index file, if a dense index exists for the Archive File.

**FOP_GROUP_NAME**
char (8)

Group name, if any, assigned to the Archive File by the user.

**FOP_USER_ID**
char (8)

TSO ID for the user that created the Archive File.

**FOP_SERVER_NAME**
char (4)

DB2 subsystem on which the operation is being executed.

**FOP_ARCHIVE_DESC**
char (40)

Description, if any, assigned to the Archive File by the user.

**FOP_START_DATETIME**
timestamp

Date and time the current operation started.

**FOP_CURRENT_DATETIME**
timestamp

Current date and time.

**FOP_START_TABLE_ID**
char (128)

Creator ID of the Start Table.

**FOP_START_TABLE_NAME**
char (128)

Name of the Start Table.

**FOP_ACCESS_DEF_NAME**
char (30)

Three-part name of the Access Definition, separated by periods.

**FOP_BATCH_ONLINE_ FLAG**
char (1)

`B` for batch execution; `O` for online execution.

**FOP_BATCH_JOBNAME**
char (8)

Job name if batch execution; TSO User ID if online execution.

**FOP_PROCESS_TYPE**
char (1)

`A` if Archive operation; `B` if Archive operation with Delete (whether or not deferred); `R` if Restore operation; `D` if Delete operation.

**FOP_RETURN_CODE**
integer

(For CALL statements only.) Return code supplied by the stored procedure.

**FOP_RETURN_MESSAGE**
char (80)

(For CALL statements only.) Optional text message supplied by the stored procedure if FOP_RETURN_CODE is 2.

**Return Values**: FOP_ACTION and FOP_ACTION_TEXT return the following values.

| FOP_ACTION | FOP_ACTION_TEXT |
|---|---|
| 0 | BefExtFirstRow |

| FOP_ACTION | FOP_ACTION_TEXT |
|---|---|
| 2 | BefExtRow |
| 3 | AftExtRow |
| 4 | AftExtLastRow |
| 5 | BefDelFirstRow |
| 7 | BefDelRow |
| 8 | AftDelRow |
| 9 | AftDelLastRow |
| 11 | BefRestFirstRow |
| 13 | BefRestRow |
| 14 | AftRestRow |
| 15 | AftRestLastRow |

**Restore/Delete Variables**: The following Restore Variables are available to Restore Action SQL statements. Each variable name is shown with its return value type and description.

**Note:**

**For users of action variables with releases of Optim prior to 7.2:** Optim 7.2 increased processing limits and changed the precision for some action variables. In Optim 6.1 and earlier, precision was **integer** for the following variables. With Optim 7.2 and later, precision for the listed variables is **bigint**. Depending on the way in which you use these variables, you may need to make changes to accommodate the difference in data type. For example, if a variable inserts a value into a DB2 table, ensure that the destination column is defined to allow a bigint value.

FOP_TOTAL_ROWS_PROCESSED

FOP_TOTAL_ROWS_REMAINING_TO_BE_PROCESSED

FOP_TOTAL_ROWS_IN_ERROR

FOP_TBL_ROWS_PROCESSED

FOP_TBL_ROWS_REMAINING_TO_BE_PROCESSED

FOP_TBL_ROWS_IN_ERROR

**FOP_ARCHIVE_ID**

A unique and arbitrary number of no logical significance assigned to the Archive File being processed. Since this value is unique for all Archive Files, it can serve as a DB2 index, if needed.

Return Value Type: integer

**FOP_TOTAL_ROWS_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows processed at this point. (See Note.)

Return Value Type: bigint

**FOP_TOTAL_ROWS_REMAINING_TO_BE_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows remaining to be processed. (See Note.)

Return Value Type: bigint

**FOP_TOTAL_ROWS_IN_ERROR**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of all rows in error at this point.

Return Value Type: bigint

**FOP_SRCTBL_CREATOR_ID**
Creator ID of the source table currently being processed.

Return Value Type: char (128)

**FOP_SRCTBL_NAME**
Name of the source table currently being processed.

Return Value Type: char (128)

**FOP_DSTTBL_CREATOR_ID**
Creator ID of the destination table currently being processed.

Return Value Type: char (128)

**FOP_DSTTBL_NAME**
Name of the destination table currently being processed.

Return Value Type: char (128)

**FOP_TBL_ROWS_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the current table processed at this point. (See Note.)

Return Value Type: bigint

**FOP_TBL_ROWS_REMAINING_TO_BE_PROCESSED**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the current table remaining to be processed. (See Note.)

Return Value Type: bigint

**FOP_TBL_ROWS_IN_ERROR**

**Note:** Prior to Optim 7.2, the return value type for this variable was integer.
Count of rows from the current table in error at this point.

Return Value Type: bigint

**FOP_AFTER_ROW_STATUS**
Status code returned by DB2 after processing a row.

Return Value Type: integer

**Note:** The value of any variable that involves a row count includes the current row for After Row actions, but does not include the current row for Before Row actions. For example, the value of FOP_TBL_ROWS_PROCESSED for the first row of a table is 0 for Before Row actions, and 1 for After Row actions.

An example of Action Variable usage is shown in the following figure.

```
------- Enter an SQL Statement to be used by the Action Being Defined ---------
Command ===>                                                 Scroll ===> PAGE

              Before Restore of a Row To a Table
Cmd                                                                    1 of 3
--- ------------------------------------------------------------------------
*** ******************************** TOP ***********************************
___ INSERT INTO FOPDEMO.ARCHIVE_ACTION
___   (ARCHIVE_CUSTID, ARCHIVE_FILE)
___   VALUES (CUST_ID, :FOP_ARCHIVE_FILE_NAME);
*** ***************************** BOTTOM ***********************************

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the LIST VARIABLES command to add Action Variables, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
Use the SHARESQL command to select an Action whose SQL you wish to use
Use the COPYSQL command to select an Action whose SQL you wish to copy
```

*Figure 124. Action Variable SQL Statement*

This SQL statement inserts CUST_ID and ARCHIVE_FILE name values (using the Action Variable FOP_ARCHIVE_FILE_NAME) into table FOPDEMO.ARCHIVE_ACTION.

## Calls to Stored Procedures

An Action statement can call a stored procedure using a one- or two-part name (i.e., *userid.procname*). If a one-part name is used, the current User ID is assumed.

Parameters passed with the CALL statement must be enclosed in parentheses. The number of parameters must be the same as the number in the procedure and must be passed in the same order. If the stored procedure does not require parameters, the CALL statement can pass an empty parameter list ( ) or omit the parentheses.

The CALL statement parameter list may include column variables, substitution variables, Action Variables, and constants.

- Type a column variable as a column name preceded by a colon (for example, :STOCK_NUMBER). The data type of the column or substitution variable must be the same as that for the parameter in the stored procedure and the variable length must equal or exceed that for the parameter. An error occurs if a column referenced by a column variable is null eligible, unless the PARAMETER STYLE specified when defining the stored procedure to DB2 is DB2SQL or GENERAL CALL WITH NULLS.
- The length of a parameter for an Action Variable must accommodate the Action Variable.
- Constants for CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, VARBINARY, DATE, TIME, and TIMESTAMP columns must be in single quotes. Constants for INTEGER, SMALLINT, and DECIMAL cannot be in quotes. The length of a parameter for a character, graphic, or binary constant must accommodate the constant. The parameter lengths for DATE, TIME, and TIMESTAMP constants are 10, 8, and 26 bytes, respectively.

A stored procedure called by an Action statement can cause restore processing to skip a row or to terminate the process by passing a value in the Action Variable FOP_RETURN_CODE. Valid FOP_RETURN_CODE values are:

**0** Normal return (default).

**1** Skip the row. This code can be returned by a stored procedure called by a Before Row statement. An error occurs if this code is returned by a stored procedure called by a Before First Row or After Action statement.

**2** Terminate the process. An optional, 80-character message can be passed in the FOP_RETURN_MESSAGE Action Variable.

**3** Normal return and Archive performs a DB2 commit at that point.

**4** Skip the row and return the table to a known state (i.e., rollback) and skip the row.

## LIST COLUMNS

Use the LIST COLUMNS command to display a selection list of column variables for the table. Use the S line command to select a column variable from the list. The selected column variable is added to the end of the SQL statement.

## LIST SUBS

Use the LIST SUBS command to display a selection list of substitution variables for the table. Use the S line command to select a substitution variable from the list. The selected substitution variable is added to the end of the SQL statement.

## LIST VARIABLES

Use the LIST VARIABLES command to display a selection list of Action Variables when formatting an SQL statement for an Archive Action. Use the S line command to select a variable from the list. The selected variable is added to the end of the SQL statement.

## SQLEDIT

You can use the SQLEDIT command to invoke an ISPF edit session for the displayed SQL statement. All standard ISPF facilities are available. Use the ISPF COPY command to insert data from a file and edit as desired. Use END to terminate the ISPF session and redisplay the Enter an SQL Statement panel. The edited data is inserted into the panel.

## SHARESQL

Use the SHARESQL command to select from a list an action whose SQL statement you wish to associate with the current action. You can scroll this list and use the S line command to select a desired action. The SQL statement associated with the selected action is inserted into the SQL statement panel for the current action, overwriting anything already entered. You can edit the shared SQL statement, but it will change for all actions for which it is active.

## COPYSQL

Use the COPYSQL command to select from a list an action whose SQL statement you wish to copy into the current action. You can scroll this list and use the S line command to select a desired action. The SQL statement from the selected action is inserted at the end of your entry. Editing the SQL statement does not affect the statement from which it is copied.

## Specification Complete

Use END to save the contents of the SQL Editor and return to the Select an Action To Be Defined panel. Use END again to return to the Define Table Map panel.

# Chapter 8. Export and Import Optim Objects

Many sites maintain similar databases on several different DB2 subsystems. For example, a site may have a database on one DB2 subsystem to support production and maintain databases on separate subsystems for testing. Another site may support multiple clients, each with its own database in a separate subsystem.

The Optim objects are stored in the Optim Directory. Each DB2 subsystem has a unique Optim Directory and objects in a Directory are available in only the subsystem that houses the Directory. The Export and Import Processes allow you to transport the Optim Directory objects (e.g., Access Definitions, Table Maps, etc.) to Directories in other DB2 subsystems. There is no need to re-create these objects on each subsystem. Thus, Export and Import save time and promote consistent, reliable data handling.

Although you can migrate table definitions, Legacy Tables, and related objects (e.g., primary keys and relationships—DB2, IMS, or OPT) by using an Extract or Archive Process in combination with a Create Process, you cannot use these processes to migrate the objects unique to Optim that are not table-related. However, the Export and Import Processes must be used to migrate Access Definitions, Table Maps, Column Maps, Compare Definitions, Archive Collections, Legacy Tables, IMS Environment Definitions, and IMS Retrieval Definitions.

The Directory objects are migrated in two steps:

1. **Export Process**

   Copy objects from the Optim Directory in the current DB2 subsystem to a data set, so the objects can be imported to an Optim Directory in any DB2 subsystem. Once copied to a data set, exported objects may be imported any number of times to any number of Optim Directories. To execute the Export Process, select Option E on the **Choose a Definition Option** menu.

2. **Import Process**

   Copy the objects in the specified data set to the current DB2 subsystem. To execute the Import Process, select Option I on the **Choose a Definition Option** menu.

Typically, you use the Export Process to copy the object definitions from the current subsystem to a data set. You can then return to the **Choose a Definition Option** menu, switch to the receiving subsystem by changing the **SUBSYS** specification and, if appropriate, the **LOCATION** specification. Once you have switched to the new subsystem, you can use the Import Process to load the object definitions from the data set. For details about the **Choose a Definition Option** menu, refer to “Choose a Definition Option” on page 11.

### About Object Security

Security status is migrated, but applies only to objects in the database, not to objects in the data set. For example, only the owner can export an Access Definition with PRIVATE status. However, the same PRIVATE status does not prevent any user from importing the Access Definition from the Export File unless an Access Definition with the same name is in the importing Directory. After the Access Definition is imported, it is assigned PRIVATE security status and, depending on an import processing option, the owner is either the importing user (default), the original owner from the Export File, or an explicit owner. (For details about assigning the owner of imported objects, see page Object Owner Processing.)

## Use the Export Process

To export Optim objects, select Option 6.E Export from the **Main Menu**.

You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu (see "Choose a Definition Option" on page 11 ) and select Option E Export. In either case, the EXPORT Process panel is displayed.

```
------------------------------- EXPORT Process -------------------------------
Command ===>

Definition Type ===>                     (See List of Types Below)

    ALL - All Types                 PK  - Primary Keys
    R   - Relationships             AD  - Access Definitions
    CM  - Column Maps               TM  - Table Maps
    CD  - Compare Definitions       LT  - Legacy Tables
    ED  - Environment Definitions   RD  - Retrieval Definitions
    AC  - Archive Collections       PR  - Column Map Procedure Definitions

Subordinate Definitions ===> NO    (Y-Yes, N-No)

Output DSN   ===>
Disposition  ===> REPLACE          (A-Append, R-Replace)

Enter Report File Name or Leave Blank for a Temporary File:
  Report DSN ===>

Delete Exported Objects ===> NO    (Y-Yes (With Confirmation Prompts), N-No)
```

*Figure 125. EXPORT Process*

## Panel

The panel includes:

**Definition Type**
The type of object to export. Specify:

**ALL** All types

**AC** Archive Collections

**AD** Access Definitions

**CD** Compare Definitions (Compare only)

**CM** Column Maps

**ED** Environment Definitions (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**LT** Legacy Tables (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**PK** Primary Keys

**PR** Column map procedure definitions

**R** Relationships

**RD** Retrieval Definitions (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**TM** Table Maps

A panel for the specified object type is displayed. Use this panel to enter the name of the desired object or use DB2 LIKE syntax to generate a selection list. You can select as many objects from the selection list as you wish. If you specify ALL, a separate panel is displayed for each object type.

**Subordinate Definitions**
Subordinate objects may be associated with a selected object. If so, these subordinate objects can also be exported. Specify:

**Y** Export subordinate objects.

**N** Do not export subordinate objects.

Each object type and its subordinate objects are as follows:

**Access Definition**
Relationship, Legacy Table

**Archive Collections**
none

**Compare Definition**
Access Definition, Relationship

**Column Map**
none

**Column Map Procedure Definition**
none

**Environment Definition**
none

**Legacy Table**
none

**Primary Key**
none

**Relationship**
none

**Retrieval Definition**
none

**Table Map**
Column Map, Legacy Table

For example, if you specify CD as the type and YES to this prompt, the Compare Definitions are copied and the subordinate Access Definitions and relationships are copied. Any additional relationships and Legacy Tables subordinate to the Access Definitions are also copied.

**Output DSN**
The name of a sequential or partitioned data set for the exported objects. You can provide the name of a new or an existing data set or place an asterisk as the last character of the name to request a selection list.

If you specify the name of a new data set, you are prompted for allocation information and the data set is allocated.

**Disposition**
Placement of objects copied to the output data set. Specify:

**A** Append objects to the contents of the output data set. (If objects are to be appended, the output data set must be a sequential file.)

**R** Replace the contents of the output data set.

**Report DSN**
The name of a sequential data set for the Export Process report. You can leave this prompt blank, provide the explicit name of a new or an existing data set, or request a selection list by including an asterisk as the last character of the name. If you specify the name of a new data set, you are prompted for allocation information and the data set is allocated.

If you do not specify a name, the report is written to a temporary file. To view the report, you must use the REPORT command from the EXPORT Summary panel.

**Delete Exported Objects**
Indicator for displaying a prompt, on which you specify whether to delete objects from the source Optim Directory after they are successfully exported. Specify:

**Y** Display the Deletion of Exported Objects pop-up window on which you can specify which objects are to be deleted:

- Delete all exported objects, including subordinate objects.
- Delete only exported objects that you explicitly selected; not subordinate objects.
- Display a list of all exported objects and select specific objects to delete.
- Do not delete any objects.

**N** Do not display a prompt and do not delete exported objects.

## Select the Objects for Export

After you have completed the entries on the EXPORT Process panel, you are prompted to select the definitions of the designated type to export. If you specify a single type, such as AD or TM, an appropriate Choose a Definition panel is displayed. However, if you specify ALL, a **Choose** panel is displayed for each object type in the order listed on the EXPORT Process panel.

- If you do not explicitly name an object on a Choose panel, a selection list is displayed.
- If you are exporting objects of one type and provide an explicit name, the Export Process is executed when you press ENTER. Alternatively, you can use END to cancel the Export Process and redisplay the EXPORT Process panel.
- If you specify ALL object types on the EXPORT Process panel and provide an explicit name on a Choose panel, the Choose panel for the next type of object is displayed when you press ENTER. You can use END to bypass any object type and proceed to the next object type. Once you complete the Choose panel for the last object type, the Export Process is invoked.

In the following figure, the Choose panel used to select Access Definitions is displayed.

```
-------------------- Choose an Access Definition to EXPORT --------------------
Command ===>


Access Definition:
  Group  ===>
  User   ===>
  Name   ===>

Use '_' for DB2 LIKE character ===> NO     (Y-Yes, N-No)
```

*Figure 126. Choose an Access Definition to EXPORT*

This panel is similar to the panel displayed when you select Option 5 ADS from the **Main Menu**. You can provide a three-part Access Definition name to select a specific Access Definition for export. You can leave one or more prompts blank or use DB2 LIKE syntax to obtain a selection list of Access Definitions.

### Selection List

Assume you have requested a selection list by specifying FOPDEMO for **Group** and SMITH for **User** on the Choose an Access Definition to EXPORT panel. The following selection list is displayed.

```
-------------------- Select Access Definitions to EXPORT ----------------------
Command ===>                                                 Scroll ===> PAGE

    Use S to Select an Entry, U to Unselect an Entry.         1 OF 6
    Use the SELECT ALL Command to Select All Entries.
    Use the END Command to Indicate that Selections are Complete.

           ------ Access Definition -----  ----- Last Modified -----
Cmd Status  Group     User       Name         By          Date
--- ------ -------- -------- ------------  -------- ----------------
*** ************************** TOP ********************************
___ SELECT FOPDEMO  SMITH    CUSTOMER      SMITH    1999-09-21-16.36
___ SELECT FOPDEMO  SMITH    DEMO1         SMITH    1999-10-15-12.15
___ SELECT FOPDEMO  SMITH    DEMO3         SMITH    1999-10-22-10.45
___ SELECT FOPDEMO  SMITH    SAMPLE        SMITH    1998-12-19-04.21
___ SELECT FOPDEMO  SMITH    SAMPLE2       SMITH    1998-12-26-10.44
*** ************************ BOTTOM *******************************
```

*Figure 127. Select Access Definitions to EXPORT*

The Access Definitions with names that satisfy the selection list criteria are displayed in alphabetical order. The user that last modified each listed Access Definition and the date it was modified are also displayed. You can scroll the list.

## Status

A blank **Status** indicates the object is not selected. When the selection list is displayed, **Status** is blank—no objects are selected. The word “SELECT” in **Status**, next to the object name, indicates an object is selected.

## Select

On the selection list, you can select one or more objects using the S (Select) line command and deselect objects using the U (Unselect) line command.

Use the SELECT and UNSELECT primary commands when many objects are listed and you want to select or deselect most of them. For example, use SELECT ALL to select all objects on the list. You can then de-select specific objects, using the U line command. Conversely, use UNSELECT ALL to de-select all objects and the S line command to select the few you wish to export.

## Selection Complete

Use END when your selections are complete. If you are exporting one type of object and:

- Have selected one or more objects from the list, the Export Process is executed.
- Have not selected an object from the list, the Choose panel is redisplayed. You can change the pattern for the selection list, specify the explicit name of an object, or use END without specifying an objects.

If you specified ALL objects on the EXPORT Process panel and:

- Have selected one or more objects from the list, the Choose panel for the next type of object is displayed.
- Have not selected an object from the list, the Choose panel is redisplayed. You can change the pattern for the selection list, specify the explicit name of an object, or use END without specifying any objects of the current type and display the Choose panel for the next object type.

### Available Commands

The following primary commands are available from a selection list:

BOTTOM
CANCEL
DOWN
END
FIND
LOCATE
OPTIONS
RESET
RFIND
SELECT
SHOW
SORT
UP
TOP
UNSELECT

## Build the Output File

At times, you may need to export objects in several steps or from several DB2 subsystems to be imported to another DB2 subsystem.

For example, you may want to export a Compare Definition with subordinate objects and a second Compare Definition with no subordinate objects. As another example, you may want to use the Access Definitions in one DB2 subsystem and supplement them with the relationships in a second and the Table Maps in a third.

In either case, the export must occur in several steps. Using one method, you can export the objects to separate output files and execute the Import Process for each file. However, if you export all objects to a single file, you can import them in one Import Process. To create a single output file, specify APPEND as the Disposition on the EXPORT Process panel for each Export Process.

The Export Process copies objects from the Optim Directory for the current DB2 subsystem. To build an output file containing objects from several DB2 subsystems, you must switch the subsystem before executing each Export Process. (You can change the DB2 subsystem designation on the **Choose a Definition Option** menu.)

## Export Summary

When the Export Process is complete, the **EXPORT Summary** is displayed.

The following is a sample summary.

```
------------------------------ EXPORT Summary --------------------------------
Command ===>

Output File DSN: FOPDEMO.EXPORT.SAMPLE
Report File DSN: FOPDEMO.EXPORT.REPORT

Summary of Results

  Primary Keys Processed           :     1
  Relationships Processed          :     1
  Column Maps Processed            :     1
  Table Maps Processed             :     1
  Access Definitions Processed     :     1
  Collection Definitions Processed :     1
  Compare Definitions Processed    :     1
  Legacy Tables Processed          :     1
  Environment Definitions Processed:     1
  Retrieval Definitions Processed  :     1

No error conditions were found

Enter REPORT Command to View the Report File
Press ENTER Key to Continue Processing
```

*Figure 128. EXPORT Summary*

This summary provides the following information:

**Output File DSN**
Name of the data set containing the exported object definitions.

**Report File DSN**
Name of the data set containing the report for the Export Process. “Temporary” indicates a data set name was not specified for the report.

**Summary of Results**
A list of the object definition types and the number of each copied to the output file.

The **Summary of Results** is followed by a message indicating any error conditions encountered in the process. Explanatory text for each error is provided in the report file.

From the EXPORT Summary panel, press ENTER to redisplay the EXPORT Process panel, or use END to return to the **Choose a Definition Option** menu.

## REPORT Command

You can use the REPORT command to invoke the ISPF browse facility and display the contents of the report file. The report lists categories of exported object definitions with the name of each exported object by category. Any error messages are also included in the report.

```
******************************** Top of Data ********************************
                          EXPORT Process Report
               Created on Friday, September 12, 2008 at 09:15 AM
                          File: FOPDEMO.EXPORT.SAMPLE

Primary Key Exported           : FOPDEMO.VENDOR
Relationship Exported          : FOPDEMO.ORDERS.PIB
Column Map Exported            : FOPDEMO.MAP10
Table Map Exported             : FOPDEMO.TESTMAP
Archive Collection Exported    : FOPWD.SALES_2007
  Archive Files in Collection  : FOPWD.SALES_01_2007
                                 FOPWD.SALES_02_2007
                                 FOPWD.SALES_03_2007
                                 FOPWD.SALES_04_2007
Legacy Table Exported          : FOPJS.CUST
Environment Definition Exported: LEGACY
Retrieval Definition Exported  : FOPJS.SALEHLD
Access Definition Exported     : FOPDEMO.FOPSP.TST01
Compare Definition Exported    : FOPDEMO.CUSTOMERS.COMPARE
******************************** Bottom of Data ********************************
```

*Figure 129. EXPORT Process Report*

Use END to terminate the ISPF browse session and return to the Optim session.

## Use the Import Process

To import Optim objects, select Option 6.I Import from the **Main Menu**.

You can also select Option 6 DEFINITIONS to display the **Choose a Definition Option** menu (see "Choose a Definition Option" on page 11 ) and select Option I Import. In either case, the IMPORT Process panel is displayed.

```
------------------------------ IMPORT Process -------------------------------
Command ===>

Definition Type ===> ALL                (See List of Types Below)

    ALL - Import All Definitions    PK  - Primary Keys
    R   - Relationships             AD  - Access Definitions
    CM  - Column Maps               TM  - Table Maps
    CD  - Compare Definitions       LT  - Legacy Tables
    ED  - Environment Definitions   RD  - Retrieval Definitions
    AC  - Archive Collections       PR  - Column Map Procedure Definitions

Input DSN    ===>

Processing Options:
  Overwrite Existing Defs      ===> NO          (Y-Yes, N-No)
  Continue Import on Error     ===> NO          (Y-Yes, N-No)
  Create Generic OPTIM Rels    ===> NO          (Y-Yes, N-No)
  Create Generic OPTIM Keys    ===> NO          (Y-Yes, N-No)
    If YES, Specify CID        ===> *           (* for All)
  Object Owner Processing      ===> USER        (U-User, S-Source, E-Explicit)
    If Explicit, Specify Owner ===>             (1-8 Alphanumeric Characters)

Enter Report File Name or Leave Blank for a Temporary File:
  Report DSN ===>
```

*Figure 130. IMPORT Process*

## Panel

The panel includes:

**Definition Type**
The type of object to import. The available types are:

**ALL** All types (all objects in the input data set are imported)

**AC** Archive Collections

**AD** Access Definitions

**CD** Compare Definitions (Compare only)

**CM** Column Maps

**ED** Environment Definitions (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**LT** Legacy Tables (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**PK** Primary Keys

**PR** Column map procedure definitions

**R** Relationships

**RD** Retrieval Definitions (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**TM** Table Maps

**Input DSN**
The name of the data set containing the objects to be imported. You can provide an explicit name or a pattern, with an asterisk as the last character of the name, to request a selection list.

**Overwrite Existing Defs**
Action taken if the name of an imported object matches the name of an object in the importing Optim Directory. Specify:

**Y** Overwrite objects in the Directory.

**N** Do not process a conflicting object. Proceed with the next object in the input data set.

The Import Process never replaces objects in the DB2 catalog. If the name of a primary key or relationship conflicts with a name in the DB2 Catalog, an error message is written to the report file and processing continues according to the **Continue Import on Error** specification.

**Continue Import on Error**
Action taken if an error is encountered during the Import Process. Specify:

**Y** Bypass processing for the object in error and continue with the next object in the input data set. Error messages are written to the report file.

**N** Stop processing.

**Create Generic OPTIM Rels**
Generic processing for all imported relationships. Specify:

**Y** Import all relationships as generic.

**N** Import all relationships as exported.

**Create Generic OPTIM Keys**
Generic processing for imported primary keys. Specify:

**Y** Import primary keys as generic primary keys.

**N** Import all primary keys as exported.

**If YES, Specify CID**
If **Create Generic OPTIM Keys** is Y, specify the identifier for primary keys to be converted to generic.

**creatorid**
Creator ID for primary keys to be imported as generic.

***** Import all primary keys in the file as generic.

**Object Owner Processing**
Determine the owner of all imported objects, as well as the date and time last modified. Specify:

**U** The owner of all imported objects is the user performing the Import Process, and the date and time last modified is the current date and time. This is the default value.

**S** The owner and the date and time last modified are set to the original values from the source Export File. This is useful for creating a mirror image of an Optim Directory.

**E** The owner of all imported objects is the value specified for **If Explicit, Specify Owner**. The date and time last modified is set to the original value from the source Export File.

**Note:**

- Only an administrator (with the authorized password specified in User Options) can modify this prompt. If you are not an administrator, this prompt is set to U and cannot be modified.
- An error occurs if you specify S or E for **Object Owner Processing** and the source Export File was created using release 5.1 or earlier of the Optim products.

**If Explicit, Specify Owner**
If **Object Owner Processing** is E, specify the owner of all imported objects. You can specify 1 to 8 alphanumeric characters; the first character must be alphabetic. (If **Object Owner Processing** is U or S, this value is ignored.)

**Report DSN**
The name of a sequential data set for the Import Process report. You can leave this prompt blank, provide the explicit name of a new or an existing data set, or specify a pattern, using an asterisk as the last character of the name, to request a selection list. If you specify the name of a new data set, you are prompted for allocation information and the data set is allocated.

If you do not specify a name, the report is written to a temporary file. Use the REPORT command from the IMPORT Summary panel to view the report.

Press ENTER to perform the Import Process. Use **END** or **CANCEL** to abandon the process. If you use END, specifications are profiled.

## Import Summary

When the Import Process is finished, the IMPORT Summary panel is displayed.

```
------------------------------ IMPORT Summary -----------------------------
Command ===>

Input File DSN:   FOPDEMO.EXPORT.SAMPLE
Report File DSN:  FOPDEMO.IMPORT.REPORT

Summary of Results

  Primary Keys Processed           :      0
  Relationships Processed          :      1
  Column Maps Processed            :      1
  Table Maps Processed             :      1
  Access Definitions Processed     :      1
  Collection Definitions Processed :      0
  Compare Definitions Processed    :      1
  Legacy Tables Processed          :      0
  Environment Definitions Processed:      0
  Retrieval Definitions Processed  :      0

No error conditions were found
Enter REPORT Command to View the Report File
Press ENTER Key to Continue Processing
Enter END Command to Exit
```

*Figure 131. IMPORT Summary*

## Panel

This panel includes the following:

**Input File DSN**
Name of the data set containing the object definitions to be imported.

**Report File DSN**
Name of the sequential data set containing the results of the Import Process. "Temporary" indicates a data set name was not specified for the report.

**Summary of Results**
A list of the object types and the total number of each added to the Directory. Conversions of primary keys and relationships to generic are shown in the **Summary of Results**.

This information is followed by a message indicating whether error conditions were encountered in processing. Explanatory text for each error is provided in the report file.

When the IMPORT Summary panel is displayed, press ENTER to redisplay the IMPORT Process panel, or use END to return to the **Choose a Definition Option** menu.

## REPORT Command

To display the contents of the report file, use the REPORT command. This command automatically invokes the ISPF browse facility and displays the contents of the report.

The report lists the names of imported objects by category. Any error messages are also included in the report.

```
******************************** Top of Data **********************
                            IMPORT Process Report
                Created on Wednesday, July 28, 2007 at 09:45 AM
                          File: FOPDEMO.EXPORT.SAMPLE

Primary Key Imported       : FOPDEMO.CUSTOMERS
Relationship Imported      : FOPDEMO.ORDERS.CUSORD
Column Map Imported        : FOPDEMO.DEMOMAP
Table Map Imported         : FOPDEMO.TESTMAP
Access Definition Imported : FOPDEMO.FOPSP.TEST1
******************************** Bottom of Data *********************
```

*Figure 132. IMPORT Process Report*

Use END to terminate the ISPF browse session and return to the Optim session.

## File Format

The Export Process generates a set of SQL-like statements for each selected Optim object definition and copies the statements to a file, which serves as the input to the Import Process. This section documents the format of these statements.

You can use ISPF facilities to browse and edit this file. Use care, however, when editing the file. Deviations from the syntax required by the Import Process cause errors.

The general format rules are:

- Each statement begins with the CREATE keyword in the first position of a line.
- Each statement ends with a semicolon. While the Export Process inserts a blank line after the semicolon, this is not required.
- At least one space between parameters is required.
- The qualifiers for individual objects in sets of specifications, such as the Table List entries for an Access Definition, are enclosed in parentheses.
- If the length of a statement exceeds the length of the line, a break occurs before or after a parameter and its qualifiers. The remaining text is continued on the next line. The Export Process indents the continued text, but this is not required. (Continuation characters are not used, except as noted for the SQL WHERE clause and selection criteria text.)
- If the first two non-blank characters on the line are dashes (-), the entire line is treated as a comment and ignored. Comments can be placed within an object definition or between definitions.

The format of the statement for each object type is documented in the following pages. The syntax used for these statements is:

**KEYWORD**
: Statement keywords are shown in uppercase and must be supplied as shown.

*text*
: Variable text is shown in lowercase italics.

**( )**
: Required as a statement delimiter to group a series of qualifiers for a parameter. For example, (*text1*, *text2*,...) or (KEYWORD1 *text1*, KEYWORD2 *text2*,...).

[ ]
: Indicates an optional parameter.

{ }
: Indicates a choice of two or more settings from which one (and only one) must be selected.

|
: Separates options.

< >
: Indicates a choice of two or more settings from which one or more may be selected.

## Primary Keys

The following statement is generated for each primary key:

```
CREATE PK tablename
  [ DESCRIPTION //description// ]
  [ SECURITY security ]
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  COLS ( columnname1 [ ,columnnamen ] )
```

### Parameters

The fully qualified table name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**COLS** The names of columns in the primary key, enclosed in parentheses and separated by commas.

Standard DDL CREATE TABLE and ALTER TABLE statements are also acceptable.

## Relationships

The following statement is generated for each relationship:

```
CREATE REL relationshipname
   [ DESCRIPTION //description// ]
   [ SECURITY security ]
   OWNER ownerid
   MODIFIED yyyy-mm-dd-hh.mm.ss
   CHI child PAR parent ( parent-expr1 = child-expr1[ , parent-exprn = child-exprn ] )
```

### Parameters

The relationship name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**CHI** The fully qualified name of the child table in the relationship.

**PAR** The fully qualified name of the parent table in the relationship.

### Relationship Expressions

Each pair of expressions must be enclosed in parentheses and separated by commas, with the parent expression before the equal sign and the child expression after the equal sign. Continuation characters are not used. Place breaks before or after an expression or the equal sign, or after a comma or a space within the expression.

Standard DDL CREATE TABLE and ALTER TABLE statements are also acceptable.

## Access Definitions

The following statement is generated for each Access Definition:

```
CREATE AD group.user.ad
  [ DESCRIPTION //description// }
  [ SECURITY security ]
  DEFCID defaultcreatorid START starttable
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  ADDTBLS { Y | N } MODCRIT { Y | N } BEGINDISP { D | S | A }
  ADCHGS { T | P } USENEW { Y | N } APPLYSELFREF { Y | N }
  [ EXPIREVALUE { ddddd | yyyy-mm-dd } ]
  [ ROWLIST dsname ]
  [ GRPCOL column GRPROWS n GRPVALS n ]
  TABLE ( table ACCESS { S | U | I | D } REF { Y | N }
    [ EXTRFREQ n ] [ EXTRLIMIT n ] PREDOP { A | O }
    [ SQL //sqlwhereclause// ] [ CORRELNAME correlname ]
    COLFLAG { Y | N }
    [ ARCIXTAB { Y | N } ]
    [ ARCDAA { Y | N } ]
    [ ARCOP { A | O } ]
    [ ARC_ACTION
       ( ( type_of_action1 share_status1 [ //sql_statement// ] )
       ( type_of_actionn share_statusn [ //sql_statement// ] ) ) ]
    [ COLUMN ( column DISP { Y | N } ACCESS { S | U }
       HEADING { N | L } { L | R | C }
       [ SORT n { A | D } ]
       [ PRED //selectioncriteria// ]
       [ ARCIXCOL { D | N | S } ]
       [ ARCFMT 'formatparam' ]
       [ ARCDATE 'explicitdate' ]
       [ ARCPIVOT n ]
       [ ARCYRS n ]
       [ ARCMOS n ]
       [ ARCWKS n ]
       [ ARCDAYS n ] ) ]
    [ LEGDSN dsname ]
    [ LEGTYPE { D | I }]
    [ LEGPSB psbname ]
    [ LEGPCB pcbnum ]
    [ LEGSEG segname ] )
  REL ( relationship STATUS { NEW | SEL | UNSEL | UNK }
       Q1 { Y | N } Q2 { Y | N }
```

```
  CHILDLIMIT { n | NOCHILDREN | UNLIMITED }
  TYPE { DB2 | PST | IMS } DUPRELS { Y | N }
  [PAR_ACCESS { K | S }]
  [PAR_KEYLIMIT n]
  [CHI_ACCESS { K | S }]
  [CHI_KEYLIMIT n]
  PAR parent CHI child )
SUBVAR ( :variablename VALUELEN 101
  [ VALUE // value // ] [ PROMPT // prompt // ] )
```

## Parameters

The Access Definition name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**DEFCID**
The default Creator ID.

**START**
The name of Start Table. The Creator ID is included only when it differs from the default Creator ID.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

The following parameters correspond to those specified on the Access Definition Parameters panel:

**ADDTBLS**
Dynamically add tables. Specify:

**Y** Users can dynamically add tables during an edit or browse session.

**N** Users cannot add tables.

**MODCRIT**
Modify selection criteria. Specify:

**Y** Users can modify selection criteria during an edit or browse session.

**N** Users cannot modify selection criteria.

**BEGINDISP**
The initial display when an Access Definition is used to edit or browse. Specify:

**D** Data

**S** Selection criteria prompt for the Start Table.

**A** Selection criteria prompt for each accessed table, including the Start Table.

**ADCHGS**
Whether changes made to an Access Definition during edit or browse are saved. Specify:

**T** (Temporary) Changes to an Access Definition during an edit or browse session are not saved, but used for the current session only.

**P** (Permanent) Changes to an Access Definition during an edit or browse session are saved..

**USENEW**
Use new relationships. Specify:

**Y** Select new relationships for traversal.

**N** Do not select new relationships.

**APPLYSELFREF**
Apply selection criteria when a table is self-referenced. This setting applies only when browsing or editing data. Specify:

**Y** Apply any criteria for a table each time the table is referenced. Default.

**N** Apply criteria to the Start Table the first time it is accessed, but not when the table is self⌂referenced.

**EXPIREVALUE**
The optional expiration parameters for Archive Files generated in Archive Processes that use the Access Definition. (Archive only) Specify:

***yyyy-mm-dd*** An explicit date, after which the data in the Archive File is considered to be expired. An error occurs if an Archive Process uses the Archive File created with this Access Definition after the specified date.

***ddddd*** The number of days until expiration.

The following parameters are included only if a Row List or group selection processing has been defined for the Start Table.

**ROWLIST**
The name of the data set containing the Row List from a Point-and-Shoot session.

**GRPCOL**
The name of the column specified for group selection processing.

**GRPROWS**
The maximum number of rows selected for each unique column value for group selection processing.

**GRPVALS**
The maximum number of unique column values for group selection processing.

## Tables

A TABLE entry is provided for each table in the Access Definition. The set of keywords for each table is enclosed in parentheses. The table name is followed by keywords that correspond to a value for a table that can be specified in the Access Definition.

**ACCESS**
Access rights for the table, specified as:

**S** Select only.

**U** Update or select.

**I** Insert, update, or select.

**D** Delete, insert, update, or select (default).

**REF** Reference table identifier:

**Y** Reference table.

**N** Not a reference table.

**EXTFREQ**
An optional numeric selection factor for rows in the table.

**EXTRLIMIT**
The optional maximum number of rows extracted or archived.

**PREDOP**
The logical operator to apply to selection criteria:

**A** Combine criteria with AND operator (default).

**O** Combine criteria with OR operator.

**SQL** Optional SQL WHERE clause text.

**CORRELNAME**
Correlation name specified on the SQL WHERE Clause panel, if applicable.

**COLFLAG**
Modifications to the Define Columns panel:

**Y** Specifications on the Define Columns panel were modified.

**N** The panel was not modified.

**ARCIXTAB**
Archive index indicator. (Archive only)

**Y** One or more columns in the table are indexed.

**N** Columns are not indexed.

**ARCDAA**
Delete after archive indicator. (Archive only)

**Y** Delete source rows after archiving.

**N** Retain rows after archiving.

**ARCOP**
The logical operator to apply to archive criteria. (Archive only)

**A** Combine criteria with AND operator (default).

**O** Combine criteria with OR operator.

**ARC_ACTION**
Archive Action definition. An entry is required for each Archive Action associated with the Access Definition. Each entry may include the following three qualifiers, which are positional and are separated from each other by blanks (not commas).

*type_of_action*
The type of Action, specified as one of the following: (Required)

BEFORE_EXT_1ST_ROW

BEFORE_EXT_ROW

AFTER_EXT_ROW

AFTER_EXT_LAST_ROW

BEFORE_DEL_1ST_ROW

BEFORE_DEL_ROW

AFTER_DEL_ROW

AFTER_DEL_LAST_ROW

BEFORE_REST_1ST_ROW

BEFORE_REST_ROW

AFTER_REST_ROW

AFTER_REST_LAST_ROW

*share_status*
The shared status for the Action, specified as either NOT_SHARED or OWNER. (Required)

*//sql_statement//*
The SQL statement for the Action. (Optional)

**Note:** An SQL WHERE clause must be delimited by double slashes, "//". If the text exceeds the length of one line, continuation is indicated by slashes at the end and at the beginning of the continuation line. The text can be broken anywhere since the continuation lines are appended without inserting blanks during the Import Process.
You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

A COLUMN entry is provided for every column in a table only when the defaults have been modified for the Access Definition. The set of keywords is enclosed in parentheses. The column name is followed by keywords that correspond to values that can be specified for a column.

**DISP** Indicator for display of column.

**Y** Display the column (default).

**N** Do not display the column.

**ACCESS**
Edit access to a column.

**U** Column can be updated.

**S** Column is display only.

**HEADING**
Two values indicating:

The column heading, displayed as:

**N** The column name (default).

**L** The label specified in the DB2 Catalog.

And the position of the heading in relation to the data length for a column, specified as:

**L** Left-justified

**R** Right-justified

**C** Centered (default)

**SORT** Two values indicating:

The sort level as a numeric value,

and the sort direction, as:

**A** Ascending (default)

**D** Descending

(Included only if sort criteria are specified.)

**PRED** The text of the selection criteria. (Included only if selection criteria are specified.)

**ARCIXCOL**
Archive index indicator for a column. (Archive only)

**D** Values in the column have a dense index.

**N** Values in the column are not indexed (default).

**S** Values in the column have a sparse index.

**ARCFMT**
The format of the date in the column. The format must be appropriate for the data type and precision of the column. (Archive only)

**ARCPIVOT**
The year used to determine whether a two-digit year value is handled as a 20th century (1900) or 21st century (2000) value. If the year is greater than or equal to this value, 1900 is assumed. (Archive only)

**ARCDATE**
The cut-off date for archived data, specified in ISO, European, or USA format. Archive converts the date to your DB2 default format. Rows with a date earlier than the specified date are selected for archiving. (Archive only)

**ARCYRS**
The number of years. This value is combined with any values for months, weeks, and days, and is subtracted from the current date to determine the cut-off date for archived data. (Archive only)

**ARCMOS**
The number of months. This value is combined with any values for years, weeks, and days, and is subtracted from the current date to determine the cut-off date for archived data. (Archive only)

**ARCWKS**
The number of weeks. This value is combined with any values for years, months, and days, and is subtracted from the current date to determine the cut-off date for archived data. (Archive only)

**ARCDAYS**
The number of days. This value is combined with any values for years, months, and weeks, and is subtracted from the current date to determine the cut-off date for archived data. (Archive only)

The following parameters apply only if TABLE is a Legacy Table. The type of legacy source file determines which parameters are included. (*Move* or *Compare for IMS, VSAM, and Sequential Files* only)

**LEGTYPE**
The type of legacy source file:

**D** VSAM or Sequential file

**I** IMS file

**LEGDSN**
The name of the legacy source data set.

**LEGPSB**
The name of the PSB that provides access to the IMS services that Optim Legacy requires to access the database records. (Included only if LEGTYPE is I.)

**LEGPCB**
The relative number of the database PCB within the specified PSB that grants Optim Legacy the authorization to manipulate the data. (Included only if LEGTYPE is I.)

**LEGSEG**
The names of the segments within the specified DBD. (Included only if LEGTYPE is I.)

## Relationships

A REL entry is provided for each relationship in the Access Definition, as listed on the Specify Relationship Usage panel. The set of keywords for each relationship is enclosed in parentheses. The relationship name is followed by keywords that correspond to values that can be specified to determine the use of a relationship by an Access Definition.

**STATUS**
The status of the relationship, specified as:

**NEW** New to the relationship list.

**SEL** Selected.

**UNSEL**
Not selected.

**UNK** Not defined for the listed tables.

**Q1** Value specified for Question 1:

**Y** Process all parent rows (default).

**N** Do not process all parent rows.

**Q2** Value specified for Question 2:

**Y** Process all child rows.

**N** Do not process all child rows (default).

**CHILDLIMIT**
The maximum number of child rows processed for each parent row, specified as:

*n* An explicit number of rows.

**NOCHILDREN**
No child rows.

**UNLIMITED**
All related child rows.

**TYPE** The type of relationship, specified as DB2, OPT (Optim), or IMS.

**DUPRELS**
Indicator for duplicate relationships:

**Y** At least one other relationship with the same parent and child tables.

**N** No duplicate relationships.

**PAR_ACCESS**
Access method used to retrieve rows from the parent table when the traversal is from child to parent. Optim determines the default if this keyword is omitted.

**K** Access by key lookup.

**S** Access using table scan.

**PAR_KEYLIMIT**
Maximum number of parent table rows that can be retrieved in one SQL call, when the traversal is from child to parent. Specify a value in the range 1 - 100. The default is 1. This parameter applies only if PAR_ACCESS is K.

**CHI_ACCESS**
Access method used to retrieve child table rows when the traversal is from parent to child. Optim determines the default if this keyword is omitted.

**K** Access by key lookup.

**S** Access using table scan.

**CHI_KEYLIMIT**
Maximum number of child table rows that can be retrieved in one SQL call, when the traversal is from parent to child. Specify a value in the range 1 - 100. The default is 1. This parameter applies only if CHI_ACCESS is K.

**PAR** The name of the parent table in the relationship.

**CHI** The name of the child table in the relationship.

### Substitution Variables

A SUBVAR entry is provided for each substitution variable that has been defined in the Access Definition. The set of keywords for each substitution variable is enclosed in parentheses. The substitution variable name must be preceded by a colon (:), and can be up to 12 characters. The substitution variable name is followed by:

**VALUELEN**
The value length is always 101. (This keyword must immediately follow the substitution variable name.)

**VALUE**
The optional default value for the substitution variable. The 1 to 100 character value must be enclosed in double slashes.

**PROMPT**
The optional prompt text to be displayed for the substitution variable. The 1 to 35 character text must be enclosed in double slashes.

## Column Maps

The following statement is generated for each Column Map:

```
CREATE CM columnmapname
  [ DESCRIPTION //description// ]
  [ SECURITY security ]
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  [ EXTDSN dsname ]
  SRC table DEST table
  VAL { ON | OFF }
  SRCTYPE code1 source-expr1 = dest-col1
    [ , SRCTYPE coden source-exprn = dest-coln ]
```

### Parameters

The Column Map name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**EXTDSN**
The optional name of the Extract or Archive File from which the column definitions are taken.

**SRC** The fully qualified name of the source table.

**DEST** The fully qualified name of the destination table.

**VAL** The validation setting, specified as:

**ON** Validation was performed when the map was created.

**OFF** Validation was not performed.

**SRCTYPE**
SRCTYPE is included for each source and destination mapping, where *code* describes the source expression as one of the following:

**A** numeric literal

**B** string, graphic, or binary literal

**C** valid column name

**D** current time register

**E** current date register

**F** current timestamp register

**H** current user register

**I** current SQLID register

**J** a default

**K** exit routine

**L** error

**M** source column matches the destination

**N** column is NULL-eligible

**U** column is unusable

**T** current TSOID register

**X** valid expression for source

*source-expr*
The specified source expression.

*dest-column*
The name of the destination column.

**Note:** See "Literals and Constants" on page 141 for examples of how to specify string, graphic, and binary literals.

## Table Maps

The following statement is generated for each Table Map:

```
CREATE TM tablemap
  [ DESCRIPTION //description// ]
  [ SECURITY { PUBLIC | READONLY | PRIVATE } ]
   OWNER ownerid
   MODIFIED yyyy-mm-dd-hh.mm.ss
   VALRULES { C | M }
   SRCCID sourceid DESTCID destid [ COLMAPID colmapid ]
   { SRCEXT dsname | SRCAD ad }
   { DESTEXT dsname | DESTAD ad }
   [ SRCTYPE { X | A | T } ] [ DESTTYPE { X | A | T } ]
   ( sourcetablename = desttablename
   [ { CM mapname | LOCALCM ( mapdef ) } ]
   [ ARC_ACTION ( ( type_of_action1 share_status1 [ //sql_statement// ] )
 ( type_of_actionn share_statusn [ //sql_statement// ] ) ) ]
   [ LEGTYPE { D | I } ] [ LEGDSN dsname ]
   [ LEGPSB psbname ] [ LEGPCB pcbnum ] [ LEGSEG segname ]
   [ PROCESSMODE { I | U | B } ] [ DELETEFLAG { Y | N } ]
   [ , sourcetablename = desttablename
   [ CM mapname | LOCALCM ( mapdef ) ]
   [ ARC_ACTION ( ( type_of_action1 share_status1 [ //sql_statement// ] )
( type_of_actionn share_statusn [ //sql_statement// ] ) ) ]
[ LEGTYPE { D | I } ] [ LEGDSN dsname ]
[ LEGPSB psbname ] [ LEGPCB pcbnum ] [ LEGSEG segname ]
[ PROCESSMODE { I | U | B } ] [ DELETEFLAG { Y | N } ] ] )
```

### Parameters

The Table Map name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**VALRULES**
The validation rules for the Table Map, specified as:

**M** Move or Archive rules.

**C** Compare rules.

**SRCCID**
The default Creator ID for the source.

**DESTCID**
The default Creator ID for the destination.

**COLMAPID**
The optional default Column Map ID.

The following parameters are included, depending on the source and destination types, as well as whether the Table Map is for Move, Archive, or Compare.

**SRCEXT**
The name of the data set containing an Extract or Archive File used as the source.

**SRCAD**
The fully qualified name of the Access Definition used as the source.

**DESTEXT**
The name of the data set containing an Extract or Archive File used as the destination.

**DESTAD**
The fully qualified name of the Access Definition used as the destination.

**SRCTYPE**
The source type as:

**X** Extract or Archive File

**A** Access Definition

**T** Table

**DESTTYPE**
The destination type as:

**X** Extract or Archive File

**A** Access Definition

**T** Table

## Table Mapping

The following source table to destination table mapping information is provided for each pair of tables referenced in the Table Map. At least one pair of tables must be specified.

**sourcetablename**
The source table name. The Creator ID is included if it differs from the SRCCID.

**desttablename**
The destination table name. The Creator ID is included if it differs from the DESTCID. The words "NOT-SPECIFIED" are inserted when the destination table in a pair is omitted.

CM or LOCALCM is included if a Column Map has been specified for a pair of tables. ARC_ACTION is included if any Archive Actions apply.

**CM** The name of the Column Map for the pair of tables. The Map ID is included if it differs from the COLMAPID.

**LOCALCM**
The local Column Map definition, enclosed in parentheses. Only the source and destination mapping (*source-expr* = *dest-column*) is included here. For details, see "Column Maps" on page 299.

**ARC_ACTION**
Archive Action definition. An entry is required for each Archive Action associated with the Table Map. Each entry may include the following three qualifiers, which are positional and are separated from each other by blanks (not commas):

**type_of_action**
The type of Action, specified as one of the following: (Required)

BEFORE_REST_1ST_ROW

BEFORE_REST_ROW

AFTER_REST_ROW

AFTER_REST_LAST_ROW

**share_status**
The shared status for the Action, specified as either NOT_SHARED or OWNER. (Required)

**//sql_statement//**
The SQL statement for the Action. (Optional)

**Note:** You must leave a space after a comma that precedes a numeric value if the DB2 setup specifies a comma as the decimal point value.

The following parameters apply only to Legacy Tables referenced in the Table Map. The type of legacy source file determines which parameters are included. (*Move* or *Compare for IMS, VSAM, and Sequential Files*only)

**LEGTYPE**
The type of legacy source file to be used with the Legacy Table:

**D** VSAM or Sequential file

**I** IMS file

**LEGDSN**
The name of the legacy source data set.

**LEGPSB**
The name of the PSB that provides access to the IMS services that *Move* or *Compare for IMS, VSAM, and Sequential Files*requires to access the database records. (Included only if LEGTYPE is I.)

**LEGPCB**
The relative number of the database PCB within the specified PSB that grants *Move* or *Compare for IMS, VSAM, and Sequential Files* the authorization to manipulate the data. (Included only if LEGTYPE is I.)

**LEGSEG**
The names of the segments within the specified DBD. (Included only if LEGTYPE is I.)

**Note:** Additionally, a Table Map referencing one or more Legacy Tables for IMS data includes parameters for any Retrieval Definitions that are referenced by the Legacy Table. (For details, see "Retrieval Definitions" on page 310.)

The following parameters are included only if the Table Map contains processing overrides:

**PROCESSMODE**
The processing mode for the table:

**I** Insert only.

**U** Update only.

**B** Both insert and update.

**DELETEFLAG**
Delete parameter for the table.

**Y** Delete all rows in the destination table prior to inserting data.

**N** Do not delete.

## Compare Definition

The following statement is generated for each Compare Definition (for Compare only):

```
CREATE CD group.user.name
  [ DESCRIPTION //description// ]
  [ SECURITY security ]
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  SRC1TYP { EXTR dsname | AD ad | LOCALAD ( addef ) }
  SRC2TYP { EXTR dsname | AD ad | LOCALAD ( addef ) | ALL }
  TM ( tablemap )
  [ REL ( relname STATUS { SEL | UNSEL | REF } SOURCE { 1 | 2 }
  TYPE { DB2 | OPT } PAR parent CHILD child ) ]
  [ CMPDSN dsname ]
```

### Parameters

The Compare Definition name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**SRC1TYP**
The type and name for Source 1, specified as one of the following:

**EXTR** Fully qualified name of the Extract or Archive File.

**AD** Fully qualified name of the Access Definition.

**LOCALAD**
Entire Access Definition specification.

**Note:** For details on the contents of the Access Definition, see "Access Definitions" on page 292 in this section.

**SRC2TYP**
The type and name for Source 2, specified as one of the following:

**EXTR** Fully qualified name of the Extract or Archive File.

**AD** Fully qualified name of the Access Definition.

**LOCALAD**
Entire Access Definition specification.

**ALL** All tables in the database. (The tables are listed in the Table Map specifications.)

**CMPDSN**
The name of the data set for the results of the Compare Process. (Included only if a data set name was specified.)

### Table Map

The complete definition of a Table Map, TM, is always included. The qualifiers are specified within parentheses. For details on the contents of the Table Map, see "Table Maps" on page 301.

### Relationships

A REL entry is provided for each relationship in the Compare Definition. The set of keywords for each relationship is enclosed in parentheses. The relationship name is followed by keywords that correspond to the relationship specifications for the Compare Definition:

**STATUS**
The status of the relationship, specified as:

**SEL** Selected.

**UNSEL**
Not selected.

**REF** For a reference table.

**SOURCE**
Indicates the source of the relationship as Source 1 or 2.

**TYPE** The type of relationship, specified as DB2 or OPT (Optim).

**PAR** The fully qualified name of the parent table in the relationship.

**CHI** The fully qualified name of the child table in the relationship.

## Legacy Tables

The following statement is generated for each Legacy Table (for *Move* or *Compare for IMS, VSAM, and Sequential Files*only).

```
CREATE LT cid.ltname
  [ DESCRIPTION //description// ]
  [ SECURITY security ]
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  LANGUAGE { P | C }
  [ CRIT //recordcriteria// ]
  LEGTYPE { D | I }
    [ DBDNAME dbdname ]
    [ SEGNAME segname ]
    [ LEGDSN dsname ]
  [ FIELD ( fieldname
    TYPE type
    LEVEL n
    LENGTH n
    [ COLNAME colname ]
    [ PRECISION value ]
    [ SCALE value ]
    [ SYNC { Y | N } ]
    [ BOUNDARY { 1 | 2 | 4 | 8 } ]
    [ SIGNED { Y | N } ]
    [ PICTURE //pictureclause// ]
    [ OCCURSFROM value ]
    [ OCCURSTO value ]
    [ OCCURSNAME name ]
```

```
  [ REDEFNAME name ] ) ]
[ LEGEXIT ioexitname ]
[ SBCS_CCSID ccsid ]
[ MBCS_CCSID ccsid ]
[ DBCS_CCSID ccsid ]
[ DETECT_ISO { NO | YES } ]
```

## Parameters

The Legacy Table name is followed by:

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**LANGUAGE**
The language for the legacy file:

**P** PL/I

**C** COBOL

**CRIT** The criteria used to match the Legacy Table with a record in the source dataset. (Included only when specified in the Legacy Table definition.)

**LEGTYPE**
The type of legacy source file.

**D** A sequential or VSAM data set.

**I** An IMS file.

**DBDNAME**
The name of the DBD for the IMS database with which the Legacy Table is associated. The DBD must be in a DBD Library referenced in the Environment Definition that matches the Creator ID of the Legacy Table. (Required if LEGTYPE is I.)

**SEGNAME**
The name of the segment in the specified DBD with which the Legacy Table is associated. (Required if LEGTYPE is I.)

**LEGDSN**
The name of the source data set.

**FIELD** The name of a field in the Legacy Table. A FIELD keyword and *fieldname* are required for each field in the Legacy Table. The allowable parameters for each field are listed as follows. (These parameters are optional unless otherwise indicated.)

**TYPE** The type is required and can be any DB2 type (e.g., SHORT, CHAR, INTEGER, TIMESTAMP, etc.). The following are also allowed:

LTTYPE_GROUP

LTTYPE_NUM

LTTYPE_VALUE

LTTYPE_LARGEINT

LTTYPE_OTHER

**LEVEL**
The field level number is required.

**LENGTH**
The field length is required.

**COLNAME**
The associated column name. If omitted, the field name becomes the column name.

**PRECISION**
The precision value.

**SCALE**
The scale value.

**SYNC** The synchronization characteristic.

**Y** Field is synchronized.

**N** Field is not synchronized.

**BOUNDARY**
The field alignment, specified as 1, 2, 4, or 8. Default is 1.

**SIGNED**
The signed status.

**Y** Field is signed.

**N** Field is not signed.

**PICTURE**
The picture clause, which must be enclosed in double slashes.

**OCCURSFROM**
The occurs-from value.

**OCCURSTO**
The occurs-to value.

**OCCURSNAME**
The name for the depending-on clause.

**REDEFNAME**
The name for the redefine clause.

**LEGEXIT**
The name of the I/O Exit Load module, if different from FOPIMIOX or FOP2DIOX or the default exit name specified in the Site Options.

**SBCS_CCSID**
Single-byte Coded Character Set Identifier (CCSID) for the Legacy Table

**MBCS_CCSID**
Multi-byte CCSID for the Legacy Table

**DBCS_CCSID**
Double-byte CCSID for the Legacy Table

**DETECT_ISO**
Specifies whether Optim looks for the $$FOPDAT, $$FOPTIM, and $$FOPSTP keywords when parsing copybooks to build Legacy Table definitions.

**NO** Optim does not look for the $$FOP*xxx* keywords when parsing copybooks (default).

**YES** Optim looks for the $$FOP*xxx* keywords when parsing copybooks. The $$FOPDAT, $$FOPTIM, $$FOPSTP, and $$FOPSTZ keywords can be included within copybook comments. The comment must appear on the line directly before the related field definition. The $$FOP*xxx* keyword must be separated by spaces from other text in the comment line.

The $$FOP*xxx* keywords indicate that the field following the comment is in ISO DATE, TIME, TIMESTAMP, or TIMESTAMP WITH TIME ZONE format. When Optim finds the $$FOP*xxx* keyword, it sets the following field to the appropriate DAT, TIM, or STP type in the Legacy Table definition. For COBOL, the field must be defined as alphanumeric. For PL/I, the field must be defined as character. The field must also adhere to the following rules.

$$FOPDAT - ISO DATE: The field must have a length of 10 and be defined in YYYY-MM-DD format.

$$FOPTIM - ISO TIME: The field must have a length of 8 and be defined in HH:MM:SS format.

$$FOPSTP - ISO TIMESTAMP: The field must have a length of 26 and be defined in YYYY-MM-DD-HH.MM.SS.NNNNNN format.

$$FOPSTZ - ISO TIMESTAMP WITH TIME ZONE:

The following examples illustrate the $$FOP*xxx* keyword comments and related field definitions.

**COBOL copybook**

```
   01 SAMPREC.
     02 CUSTNAME          PIC X(80).
*  $$FOPDAT
     02 SAMPLE-DATE       PIC X(10).
*  $$FOPTIM
     02 SAMPLE-TIME       PIC X(8).
*  $$FOPSTP
     02 SAMPLE-TIMESTAMP  PIC X(26).
*  $$FOPSTZ
     02 SAMPLE-TIMESTAMP-TZ PIC X(32).
```

**PL/I include file**

```
1 SAMPREC,
   2 CUSTNAME            CHAR(80),
/* $$FOPDAT                                    */
   2 SAMPLE_DATE         CHAR(10),
/* $$FOPTIM                                    */
   2 SAMPLE_TIME         CHAR(8),
/* $$FOPSTP                                    */
   2 SAMPLE_TIMESTAMP    CHAR(26),
/* $$FOPSTZ                                    */
   2 SAMPLE_TIMESTAMP_TZ CHAR(32);
```

## Environment Definitions

The following statement is generated for each Environment Definition (for *Move* or *Compare for IMS, VSAM, and Sequential Files* only):

```
CREATE ED environmentname
   [ DESCRIPTION //description// ]
   [ SECURITY security ]
   OWNER ownerid
   MODIFIED yyyy-mm-dd-hh.mm.ss   [ SYSLIB1 syslib1 ]
   [ SYSLIB2 syslib2 ]
   [ SYSLIB3 syslib3 ]
   DBDLIB1 dbdlib1   [ DBDLIB2 dbdlib2 ]
   [ DBDLIB3 dbdlib3 ]
   [ DBDLIB4 dbdlib4 ]
   DFSVSAMP_DSN dfsdsn   DFSVSAMP_MEM dfsmem   [ IMSID imsid ]
   [ IMSAGN imsagn ]
```

## Parameters

The Environment Definition name is followed by:

**DESCRIPTION**
The optional description of the Environment Definition. The 1 to 40 character text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**SYSLIB1, SYSLIB2, SYSLIB3**
The optional names of the IMS Program Libraries included in the Environment Definition. The IMS Program Libraries define and load the libraries required to run IMS.

**Note:**

- If your site uses IMS dynamic allocation, this includes the name of the data set containing the dynamic allocation load modules.
- If you have authorized the Optim SFOPLLIB data set, it is added to this list.

**DBDLIB1, DBDLIB2, DBDLIB3, DBDLIB4**
The names of IMS DBD and PSB Libraries included in the Environment Definition. The DBD and PSBs within the libraries define the structure of the IMS database, as well as how Optim Legacy accesses the data.

**DBDLIB1** is a required parameter, but **DBDLIB2-4** are optional.

**DFSVSAMP_DSN**
The dataset name of the DFSVSAMP containing the IMS buffer parameters to be used when running in batch mode.

**DFSVSAMP_MEM**
The DFSVSAMP member name in the specified dataset containing the IMS buffer parameters to be used when running in batch mode.

**IMSID**
The IMS System ID required to access the IMS data when allocated to a control region (i.e., the data is online to IMS).

**IMSAGN**
The IMS Application Group Name required to access the IMS data when allocated to a control region, if required by your IMS database.

## Retrieval Definitions

The following statement is generated for each Retrieval Definition (for *Move* or *Compare for IMS, VSAM, and Sequential Files* only):

```
CREATE RD environmentname.DBname
   [ DESCRIPTION //description// ]
   [ SECURITY security ]
   OWNER ownerid
   MODIFIED yyyy-mm-dd-hh.mm.ss   PSBNAME psbname   PCBNBR pcbnbr   [ DSNPREFIX dsnprefix ]
   [ IDNAME idname ]
   [ USE_DBRC { YES | NO } ]
   [ USE_IEFRDER { YES | NO } ]
   [ DBTYPE { HIDAMOS | HDAMOS | HIDAM | HDAM | HISAM | HSAM | DEDB | HALDB | SHSAM | SHISAM } ]
   DDGROUP ( DDNAME ddname DSNAME dsname TYPE type
             NUMSEG numseg SEGMENT ( SEGNAME segname )
             DCB_PARMS(UNIT esoteric unit  LRECL logical record length
             BLKSIZE block size  ALLOCSIZE allocation size
             PRIMARY primary quantity
             SECONDARY secondary quantity VOLUME (volume serial))
```

### Parameters

The Retrieval Definition name (*environmentname.DBname*) is followed by:

**DESCRIPTION**
The optional 1 to 40 character description of the Retrieval Definition. The text must be enclosed in double slashes.

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**PSBNAME**
The 1 to 8 character name of the default PSB in a PSB library referenced in the specified Environment Definition. The PSB provides access to the IMS services that Optim requires to access the database records.

**PCBNBR**
The relative number of the database PCB within the specified PSB that grants Optim the authorization to manipulate the data.

**DSNPREFIX**
The optional 1 to 8 character prefix used when specifying Associated IMS Database Dataset Names.

**IDNAME**
The optional IMS System ID required to access the IMS data when allocated to a control region (i.e., the data is online to IMS).

**USE_DBRC**
This entry is valid only for IMS processing in DL/I mode (i.e, when an IMS ID is not specified). If appropriate, enter **YES** to use Database Recovery Control (DBRC) to control logging and perform database recovery; otherwise specify **NO**. IMS uses the online log datasets (OLDS) if the database is accessed in BMP or DBB mode.

The default for a HALDB (High Availability Large DataBase) is **YES**, and that entry cannot be changed.

DBRC use is optional for a non-HALDB, such as HIDAM, HDAM, HISAM, etc. Thus, you may specify **YES** for a non-HALDB, but it is not required.

**USE_IEFRDER**
This entry is valid only for IMS processing in DL/I mode. If appropriate, enter **YES** to use an IMS log to perform database recovery; otherwise specify **NO**.

If you specify YES, you must specify the appropriate dataset name (DSNAME) for the IMS log within the DDGROUP identified by the DDNAME "IEFRDER".

**DBTYPE**
If appropriate, specify the database type as one of the following:

HIDAMOS HDAMOS HIDAM HDAM
HISAM HSAM DEDB HALDB
SHSAM SHISAM

The following information is provided for each DD Group (i.e., group of physical data sets) associated with each segment in the DBD. Each DD Group must be enclosed in parentheses.

**DDNAME**
The name of the DD (i.e., the physical data sets) associated with the segments. If IMS logging is requested, DD Name IEFRDER is used along with the Segment Name IMSLOG to identify the log dataset.

**DSNAME**
The IMS Database Dataset location associated with each DD name in the DBD.

**TYPE** The type of DD group, specified as:

**D** Dataset

**I** Index

**O** Overflow

**NUMSEG**
The number of segments within the DD group.

**SEGMENT**
Defines a segment entry within the DD group. Each segment must be enclosed in parentheses.

**SEGNAME**
The name of the segment within the DD group.

**DCB_PARMS**
This keyword is used to specify the IMS Log dataset attributes for the following:

**UNIT** Esoteric Unit

**LRECL**
Logical Record Length

**BLKSIZE**
Block Size

**ALLOCSIZE**
Allocation Size

**PRIMARY**
Primary quantity *(see Note)*

**SECONDARY**
Secondary quantity *(see Note)*

**VOLUME**
Volume serial

**Note:** If you specify an IMS Log Dataset Name (DSNAME), you must provide sufficient Primary and Secondary space units to allocate the IEFRDER dataset. Failing to do so will cause IMS to abort processing and lock the database from further updates until a recover/rollback is done.

## Archive Collections

The following statement is generated for each Archive Collection:

```
CREATE AC collectionid.collectionname
  [ SECURITY security ]
  [ DESCRIPTION //description// ]
  OWNER ownerid
  MODIFIED yyyy-mm-dd-hh.mm.ss
  DEFDATE yyyy/mm/dd
  FILES ( filename1 [, filename2 ] [ , ... ] )
```

### Parameters

The fully qualified Archive Collection name is followed by:

**SECURITY**
The security assigned to the object, specified as:

**PUBLIC**
All can use and modify (default).

**READONLY**
All can use, but only owner can modify.

**PRIVATE**
Only owner can use and modify.

**DESCRIPTION**
The optional description. The 1 to 40 character text must be enclosed in double slashes.

**OWNER**
The owner of the exported object.

**MODIFIED**
The date and time the object was last modified.

**DEFDATE**
The collection's default date. This date is used for rows in a table without a date column that is unioned with a table that includes a date column.

**FILES** The fully qualified names of Archive Files in the collection, enclosed in parentheses and separated by commas.

# Chapter 9. Convert Archive and Extract Files

The Convert Process transforms data in an Extract or Archive File. This converted file can be used as input to most Optim processes. However, since a converted file does not include object definitions needed to recreate the database, it cannot be used to create objects, and your ability to browse a converted Extract or Archive File in Table Mode is limited.

The Convert Process can be performed on an existing Extract or Archive File, using a Table Map and a set of Column Maps. An extract file stored on tape can be converted, but the converted file will be stored on disk.

**Note:** The Convert Process can also be performed as part of an Extract Process. For details, see the *Move User Manual*, Extract Data.

Because an Extract or Archive File does not have the same level of security afforded by DB2, you can use the Convert Process to mask sensitive data. The converted file can be used to create reports, while the original file can be retained for audit or restoration purposes. If desired, you can also convert the data to a comma separated values (CSV) format to be used with any application that supports CSV files.

## Convert File Panel

When you select the CONVERT Option on the **Data Migration** menu for Move, or on the **Archive and Restore** menu for Archive, the Specify CONVERT Parameters and Execute panel is displayed.

**Note:** The Convert panels shown in this section are for processing an Extract File. Any differences between the panels for processing an Extract File and for processing an Archive File are noted, where pertinent. Also, for readability, the text in this section refers to the Extract File or Archive File as the source file.

```
------------------- Specify CONVERT Parameters and Execute --------------------
OPTION ===>                                                 SCROLL ===> PAGE

  1  TABLE MAP            - Specify Table Map and Column Maps
  2  PERFORM              - Execute Extract File Conversion Process

Specify Data Set Names for Source Extract File and Control File:
  Source Extract File DSN  ===> SAMPLE.FOP.SRCEXTR
  Control File DSN         ===> EXTRACT.CTRL

Specify Data Set Name for Destination Extract File.  If Omitted,
Destination Extract DSN will be same as the Source Extract DSN:
  Destination Extract DSN  ===> SAMPLE.FOP.DESTEXTR

Age Date Values                  ===> N     (Y-Yes, N-No)
Output to External Format        ===> Y     (Y-Yes, N-No)
Limit Number of Discarded Rows   ===>       (1-4294967295,Blank-No Limit)
If Destination Tables have a Cluster Index:
  Sort Extract File Rows ===> N             (Y-Yes, N-No)
Review Propagation Key Sets      ===>       (A-Always, E-Errors)

Run Process in Batch or Online   ===> O     (B-Batch, O-Online)
  If Batch, Review or Save JCL   ===> R     (N-No, R-Review, S-Save)

Process Report Type              ===> D     (D-Detailed, S-Summary)
```

*Figure 133. Specify CONVERT Parameters and Execute*

## Panel Options

The options on this panel include:

**1 TABLE MAP**

A Table Map, listing the destination table for each table in the source file, is required to perform the Convert Process. If you do not select this option, the Convert Process Table Map panel is displayed automatically before proceeding with the selected option. When the Table Map is displayed, you must provide the destination Creator ID at the Dest CID prompt. By default, the names defined in the source file are inserted as the base destination table names. There is no default for destination Creator ID.

You may reference a Column Map for one or more destination tables. Using a Column Map, you can specify the destination data column-by-column.

For more information about Column Maps and Table Maps, see Chapter 5, “Column Maps,” on page 155 and Chapter 7, “Table Maps,” on page 249.

**2 PERFORM**

Perform the Convert Process.

## Panel

This panel includes:

**Source Extract File DSN**

The name of the source file that contains the data to be converted.

For an Archive File, this prompt is labeled **Source Archive File DSN**.

**Control File DSN**

The name of a sequential file that is to be used to accumulate information and statistics about the Convert Process.

**Destination Extract DSN**

For an Extract File, you can enter the name of the dataset for the converted Extract File. If you do not provide a name, the actual transformations are performed in a temporary file that you are prompted to allocate. After the Convert Process is successfully executed, the source Extract File is deleted and the temporary file is saved under the source Extract File name. The converted data replaces the original extracted data, and object definitions are lost.

For an Archive File, you must enter the name of a dataset for the converted Archive File, which must be different from the **Source Archive File DSN**.

**Note:**

When you convert an archive file, the destination dataset is an Extract File.

When you convert an extract file stored on tape, the destination dataset is stored on disk.

**Age Date Values**

Indicator for aging date values as part of the Convert Process. This prompt is displayed for Extract Files only. Specify:

**Y** Age date values and display the Specify Aging Parameters panel. On this panel, you can specify aging values to supplement the specifications for columns mapped with the AGE function. These values are used, if requested, to age DATE and TIMESTAMP columns that are not explicit targets of an AGE function.

**N** Do not age date values and ignore any specifications for aging that are on Column Maps included in the process.

See Age Date Values in the *Move User Manual* for details about the Specify Aging Parameters panel.

**Output to External Format**

Indicator for creating a comma delimited file based on the contents of the source file. This file can be used to input changes for any application that supports CSV files. Specify:

**Y** Display the External Format Specification panel to create a CSV file. For details, see page "Output to External Format" on page 318.

**N** Do not create a CSV file.

**Limit Number of Discarded Rows**

The maximum number of rows that can be discarded. The Convert Process terminates when this value is reached.

**Sort Extract File Rows**

For DB2 tables only. Indicator for sorting data for destination tables with a cluster index. Specify:

**Y** Sort rows.

**N** Do not sort rows.

For an Archive File, this prompt is **Sort Archive File Rows**.

**Review Propagation Key Sets**

Indicator for displaying the Propagating Key Set(s) panel before performing the Convert Process. This prompt is displayed only when one or more Column Maps include the PROP function. Specify:

**A** Always display the panel prior to performing the Convert Process.

**E** Display the panel prior to performing the Convert Process only when the PROP specifications contain errors. Default.

For information about the Propagating Key Set(s) panel, see View PROP Specifications in the *MOVE User Manual* or View PROP Specifications in the *Archive User Manual*.

**Run Process in Batch or Online**
Indicator for executing the Convert Process in batch or online. If you are converting an extract file that is stored on tape, you must execute the convert process in batch. Specify:

**B** Batch

**O** Online

If site management has established a maximum number of rows for online processing and this request exceeds that limit, this option is forced to Batch and cannot be changed. Consult site management for guidelines.

**If Batch, Review or Save JCL**
Indicator for reviewing or saving the JCL and Batch Utility control statements prior to job submission. This setting applies to batch execution only. Specify:

**N** Submit job. Do not display or save the JCL and control statements.

**R** Display the JCL and control statements for review, prior to job submission.

**S** Save the JCL and control statements. Prompts are provided for you to specify the name of a file in which to store the JCL and control statements. For details, see page "Save" on page 321.

**Process Report Type**
Indicator to include additional information in the Convert Process Report. If selected, detailed information about Column Map usage is displayed.

**D** Display detailed information in the Convert Process Report.

**S** Display summarized information in the Convert Process Report.

**Note:** You can obtain a selection list of Extract, Archive, or Control File names by using a wildcard character as the last character in the name. When the list is displayed, use the S line command to select an entry.

### Available Commands

The following primary commands are available:

CANCEL
END
OPTIONS

## Output to External Format

When you enter Yes at the Output to External Format prompt, the External Format Specification panel is displayed.

```
---------------------- External Format Specification -------------------------
Command ===>                                                Scroll ===> PAGE

                                                                    1 of 3
Extract DSN                 : SAMPLE.FOP.SRCEXTR
Output  Format              : Delimited
Generate Header          ===> N    (Y-Yes, N-No)
  Beginning Label        ===>
  Ending Label           ===>
  Header Delimiter       ===> ,
  Use Column Labels      ===> N    (Y-Yes, N-No)
Field Delimiter          ===> ,
String Delimiter         ===>
  Delimiter Escape Char  ===>
Default Destination DSN  ===> SAMPLE.FOP.DESTEXTR

Table/View                   Destination DataSet
--------------------------  ------------------------------------------
******************************** TOP ***************************************
FOPDEMO.CUSTOMERS
FOPDEMO.DETAILS
FOPDEMO.ORDERS
******************************** BOTTOM ************************************
```

*Figure 134. External Format Specification*

## Panel

This panel includes:

**Extract DSN**
Name of the source file that contains the data to be converted. You cannot modify this value.

For an Archive File, this prompt is **Archive DSN**.

**Output Format**
Type of output format. You cannot modify this value.

**Generate Header**
Indicator for generating a header for the external file.

**Y** Generate a header.

**N** Do not generate a header.

**Beginning Label**
A label placed before the first column in the table. Specify "$table" to include the table name.

**Ending Label**
A label placed after the last column in table.

**Header Delimiter**
The character used to separate column headings.

**Use Column Labels**
Indicator for using column labels in the header (if a header is generated).

**Y** Use columns labels.

**N** Do not use column labels (use column names).

**Field Delimiter**
The character used to separate values in a row.

**String Delimiter**
The character used to separate character literal values.

**Delimiter Escape Char**
The character used to generate the value of a character normally used as a delimiter.

**Default Destination DSN**
Name of the default dataset for the converted file. You can enter a file name explicitly by enclosing it in quotes; otherwise the default prefix, as specified in User Options, is prefixed to the name. You can obtain a selection list of datasets by using a wildcard character in the last position.

**Table/View**
Names of source tables or views.

**Destination Dataset**
The name of a dataset for the converted data for each source table or view. Leave blank to use the dataset named in **Default Destination DSN**.

## Perform Convert Process

Select Option 2 PERFORM on the Specify CONVERT Parameters and Execute panel to begin the Convert Process.

## Online Execution

If you execute the process online, a panel is displayed noting the progress of the Convert Process.

```
+------------------------CONVERT Process Status-------------------------+
|                                                                       |
|                          CONVERT Process in Progress                  |
|                                                                       |
|       Number of Rows Processed: 2053 of 10340                         |
|                                                                       |
|  Completed Table: FOPDEMO.CUSTOMERS                                   |
|   Converted Rows: 523                                                 |
|      Failed Rows: 0                                                   |
|                                                                       |
+-----------------------------------------------------------------------+
```

*Figure 135. CONVERT Process Status*

This panel displays the total number of rows that have been processed, the total number of rows to be processed, the name of the table that is currently processing, and the total number of rows that have been converted or failed for that table. This information is revised:

- Every 1000 rows for each table to display the current total number of processed rows.
- When the processing for a table is complete and the processing for the next table begins.

## Batch Execution

If you choose batch execution, Optim builds the necessary JCL and Batch Utility control statements. The Job card information is taken from the JCL specified on the Job Card and Print Options panel.

If you specified YES to **Prompt for Changes Before Job Submission**, the Job Card and Print Options panel is displayed prior to job submission. You can edit the job card information and specify whether your changes apply to the current job only or are to be applied permanently. (See "Job Card and Print Options" on page 386.) The information on the Job Card and Print Options panel is used, with the Convert Process parameters, to build the JCL and control statements required to perform the Convert Process.

## Review

If you specify Review to the If Batch, Review or Save JCL prompt, the complete JCL and control statements are displayed in the ISPF editor and can be edited and saved. (See the *Batch Utilities Guide*, CONVERT for the CONVERT statement keywords and values.)

After reviewing the JCL and control statements, use END. If jobs are automatically submitted when END is used, the job is submitted. Otherwise, you will have to explicitly SUBMIT the job from the ISPF editor. (See "User Options" on page 372 for information on the option to automatically submit jobs when END is used.)

## Save

If you specify Save to the If Batch, Review or Save JCL prompt, the Save JCL Parameters panel prompts for the name of the file in which to save the JCL and control statements and whether the job should be submitted after saving.

```
.--------------------------- Save JCL Parameters -------------------------.
|                                                                          |
|  DSN to Save JCL to     ===>                                             |
|  Member (if PDS)        ===>                                             |
|  Replace Existing Data  ===>       Y-Yes, N-NO                           |
|                                                                          |
|  DSN to Hold SYSIN Data ===>                                             |
|  Member (if PDS)        ===>                                             |
|  Replace Existing Data  ===>       Y-Yes, N-NO                           |
|                                                                          |
|  Submit JCL, or Review? ===>       (S-Submit, R-Review, N-Neither)       |
'--------------------------------------------------------------------------'
```

*Figure 136. Save JCL Parameters*

The following prompts are displayed:

**DSN to Save JCL to**
The name of the sequential file or partitioned data set to receive the JCL and control statements. If you specify a partitioned data set, specify the member name at the Member prompt.

**Member (if PDS)**
The name of the member in the partitioned data set specified for the DSN to Save JCL to prompt. If you specify a member name when a sequential file is specified, an error message displays.

**Replace Existing Data?**
Indicator for replacing existing data in the specified file.

**DSN to Hold SYSIN Data**
The name of the sequential file or partitioned data set to hold SYSIN data. If you specify a partitioned data set, specify the member name at the Member prompt.

**Member (if PDS)**
The name of the member in the partitioned data set specified for the DSN to Hold SYSIN Data prompt. If you specify a member name when a sequential file is specified, an error message displays.

**Replace Existing Data?**
Indicator for replacing existing data in the specified file.

**Submit JCL or Review?**
Option for handling the JCL and control statements. Specify:

**S** Save and submit the JCL and control statements.

**R** Display the JCL and control statements for review. You can use ISPF facilities to save or submit the JCL and control statements.

**N** Save the JCL and control statements; however, the JCL and control statements are not displayed for review or submitted.

### Job Card Error

If you submit the job and an error is encountered in the job card, a message is displayed. You can review the job card and correct the error, or terminate the Convert Process.

## Convert Process Report

A Convert Process Report is generated as part of the Convert Process. This report contains various statistics and information about the execution of the Convert Process.

### Display Report Contents

You can browse the contents of the Convert Process Report. When the process is executed online, the Convert Process Report is automatically displayed after the process completes. Standard ISPF scrolling functions are available.

In batch, the report is placed in the default output file specified in the JCL. You can then display the report as you would the output from any job.

### Report Format

The Convert Process Report is formatted as follows. (The report is truncated in this example; there is more data next to the data displayed in the example.)

```
 --------------------------- CONVERT Process Report ----------------------------
 Command ===>                                                  Scroll ===> PAGE
                                                               ROW 0    OF 29
******************************** Top of Data *********************************

                              CONVERT Process Report

Extract File            : FOPDEMO.SAMPLE.EXTRACT.FILE.DSN
Control File            : FOPDEMO.SAMPLE.CONTROL.FILE.DSN
User ID                 : COHEND
Time Started            : 1998-10-19 12.18.04
Time Finished           : 1998-10-19 12.18.06

       Due to Successful Completion of Process, Control File is No Longer Valid

Process Options
  File Format                   : Delimited
  Generate header               : N
  Field Delimiter               : ,

Totals:
  Number of Convert Tables        : 4
  Number of Original Extract Rows : 4972
  Number of Converted Rows        : 4972
  Number of Conversion Errors     : 0
  Number of Exit Routine Discards : 0

                                                        Data      Exit
                                  Extract  Converted Conversion  Routine
             Convert Tables        Rows      Rows      Errors   Discards  EXte
       -------------------------- --------- --------- ---------- --------- ----
 1     FOPDEMO.CUSTOMERS               132       132          0         0 FOPW
 2     FOPDEMO.ORDERS                  792       796          0         0 FOPW
 3     FOPDEMO.DETAILS                2176      2176          0         0 FOPW
 4     FOPDEMO.ITEMS                   872       872          0         0 FOPW

                           ****** End of Report ******
******************************** Bottom of Data ******************************
```

*Figure 137. Convert Process Report*

Report headings identify the information, including the source file name, the Control File name, the destination file name, the user requesting the Convert Process, and the date and time the process was executed. This information is followed by the total number of tables, number of rows in the source file to be converted, number of rows successfully converted, number of rows having conversion errors, and number of rows discarded by an exit routine.

The tables are listed after this information in the order in which they were saved in the source file. A complete breakdown of the totals for each table is provided.

## External File Format

If you created a CSV file as part of the Convert Process, processing options, including the file format, an indicator for generating a header, the field delimiter, and column labels option are displayed before the totals. Also, the name of the dataset for the converted data for each source table is provided.

## Aging Parameters

For Extract Files only, if aging parameters have been specified for the Convert Process, the report also includes:

- Any exception conditions encountered for each table are included if you specify Yes to **Report Skipped Dates** and **Report Invalid Dates**. The following is a sample list of exception conditions for aging ORDER_SHIP_DATE in the ORDERS table.

  Exception Conditions for Table FOPDEMO.ORDERS

```
Row 1: Field ORDER_SHIP_DATE (YY/MM/DD) cannot age date: X'0'
Row 78: Field ORDER_SHIP_DATE (YY/MM/DD) cannot age date: X'1'
```

- Aging parameters for the process are listed after the summary information. For example, the following specifications for a Convert Process use many of the defaults.

  Processing Parameters:

**Default Aging Amount:**
='2000/01/31'

**Default Aging Table:**
FOP2RUSA

**Default Aging Rule:**
NEXTWORKDAY

**Century Pivot Year:**
65

**Process Date Columns:**
All

**Report Invalid Dates:**
Yes

**Report Skipped Dates:**
Yes

**Output Rows With Invalid Dates:**
Yes

**Output Rows With Skipped Dates:**
Yes

- Aging results for each table include information specified on the Specify Aging Parameters panel. The information includes:
- Destination Column/Source Column names
- Number of valid dates
- Number of invalid dates
- Number of skipped dates
- Minimum value encountered
- Maximum value encountered
- Date format
- Aging Rule applied to column
- Rule Table containing Aging Rule applied to column
- Aging Amount applied to column
- Pivot year specifications for column

(DB2 DATE and TIMESTAMP columns are displayed as DB2 DATE and DB2 TMSTAMP.)

If you request a detailed process report from the Specify EXTRACT Parameters and Execute panel, the report includes the following:

## Detailed Extract Report

If you request a detailed process report from the Specify CONVERT Parameters and Execute panel, the report includes the following:

Any Column Maps specified for the Convert Process. If the Column Map is Local, no Column Map information, Column Map Name, Security Status, etc., is displayed.

```
Column Maps in Use:

Map Name        : FOPDEMO.CM2
Source File     : 'FOPDEMO.PERF.APR14'
Modfied By      : FOPDEMO
Last Modified   : 2004-04-28
Security Status : PUBLIC

Source Table: FOPDEMO.CUSTOMERS    Destination Table: FOPDEMO2.CUSTOMERS

Source Column       Data Type      Destination Column   Data Type      Status
------------------ -------------- -------------------- -------------- -------
CUST_ID            CHAR(5)        CUST_ID              CHAR(5)        EQUAL
CUSTNAME           CHAR(20)       CUSTNAME             CHAR(20)       EQUAL
ADDRESS            VARCHAR(50)    ADDRESS              VARCHAR(50)    EQUAL
CITY               VARCHAR(15)    CITY                 VARCHAR(15)    EQUAL
'PA'               LITERAL        STATE                CHAR(2)        LITERAL
ZIP                CHAR(5)        ZIP                  CHAR(5)        EQUAL
                   UNUSED         YTD_SALES            DECIMAL(7,2)   UNUSED
SALESMAN_ID        CHAR(6)        SALESMAN_ID          CHAR(6)        EQUAL
PHONE_NUMBER       CHAR(10)       PHONE_NUMBER         CHAR(10)       EQUAL
```

## Print Report

While browsing the Convert Process Report online, you can use the OUTPUT command to direct the contents of the report to an output file or SYSOUT. A panel prompts for output destination information. (For details, see the OUTPUT command in the *Command Reference Manual*.)

## Browse

After the Convert Process has executed, you can review the resulting Extract File to ensure the transformation of the data. Use Option B BROWSE on the **Data Migration** menu for Move, or on the **Archive and Restore** menu for Archive to display the contents of the file.

You can also use Option B BROWSE to browse the Control File to determine which rows were not converted. Rows are discarded when conversion cannot be performed. For example, a row is discarded if an expression used to convert the source would cause an invalid data type for the column. The discarded rows are noted in the Control File. Also, rows discarded by user-defined exit routines are included in the Control File.

# Chapter 10. Retry/Restart a Process

If Insert, Delete, or Restore processing fails for some reason, you can retry or restart a process. You can retry a process when one or more rows are discarded during the process (for example, when you attempt to insert or restore child rows and the parent rows are not in the database). You can restart a process that does not execute to completion (for example, a process can terminate abnormally when the allocated time or space resources are insufficient, or the user ends the process prematurely).

Select Option R RETRY/RESTART on the **Data Migration** menu for MOVE, or on the **Archive and Restore** menu for ARCHIVE, to display the Pending Process List panel. This panel lists any pending processes initiated by the current user.

**Note:** The Pending Process List panel lists any pending Insert or Delete Processes when invoked from the **Data Migration** menu, and lists any pending Restore or Delete Processes when invoked from the **Archive and Restore** menu.

```
---------------------- Pending Process List for FOPDEMO ----------------------
Command ===>                                                 Scroll ===> PAGE

Cmd    Timestamp   Process/Status                Control File
--- -------------- -------------- -------------------------------------------
    ******************************** TOP ***********************************
___ 12-15-98 12.30 INSERT RETRY   FOPDEMO.CONTROL.SAMPLE2
___ 12-16-98 03.23 INSERT RESTART FOPDEMO.CONTROL.SAMPLEX
    ***************************** BOTTOM ***********************************
```

*Figure 138. Pending Process List – Current® SQL ID*

## Panel

The **Pending Process List** includes:

**Cmd** The line command area. Valid line commands are:

- **D** Delete a pending process from the list.
- **I** Display information about a pending process.
- **S** Select a pending process to retry or restart.

**Timestamp**
Date and time of the last process. The list is arranged in ascending order by timestamp, but can be rearranged using the SORT command.

**Process/Status**
The process performed and the status of the process. Process is indicated as:

**DELETE**
Delete Process, including the delete portion of an Archive Process

**INSERT**
Insert Process (with insert processing)

**RSTORE**
Restore Process

**UPDATE**
Insert Process (with update or both insert and update processing)

Status is indicated as:

**RESTART**
Incomplete process can be restarted.

**RETRY**
Discarded rows can be retried.

***DELETED**
Process deleted with the D line command.

***INVALID**
You attempted to retry, restart, or display information about the listed process, but the Extract or Archive File has been modified since the process was executed (i.e, the Control File is no longer valid and the process cannot be retried or restarted).

***RUN** Process successfully retried or restarted.

**Control File**
Name of the Control File created in the process and required for the Retry or Restart Process.

**Note:** You can browse a Control File to review the status of an Insert, Delete, or Restore Process. See the *Archive User Manual*, Control Files for details about browsing a Control File for a Restore or Delete Process; and see the *Move User Manual*, Control Files for details about browsing a Control File for an Insert Process.

## Available Commands

You can scroll the list of pending processes using standard ISPF scrolling facilities. The following primary commands are available:

ALL
BOTTOM
CANCEL
DOWN
END
OPTIONS
TOP
UP

## All SQL IDs

Use the ALL command to display a list of pending processes for all SQL IDs.

```
----------------------------- Pending Process List -----------------------------
Command ===>                                                 Scroll ===> PAGE

Cmd  SQLID     Timestamp   Proc/Status              Control File
--- -------- -------------- --------- ----------------------------------------
    ******************************** TOP ***************************************
___ FOPDEMO  12-15-98 01.15 DEL RSTRT FOPDEMO.CONTROL.SAMPLE1
___ FOPDEMO  12-15-98 12.30 INS RETRY FOPDEMO.CONTROL.SAMPLE2
___ FOPDEMO  12-16-98 03.23 INS RSTRT FOPDEMO.CONTROL.SAMPLEX
___ OPTIM    12-10-98 03.39 INS RSTRT OPTIM.CONTROL.SAMPLEY
___ OPTIM    12-10-98 04.30 UPD RSTRT OPTIM.CONTROL.SAMPLE4
    ******************************** BOTTOM ************************************
```

*Figure 139. Pending Process List – All SQL IDs*

The format of the list of pending processes for all SQL IDs is similar to that of the list for the current SQL ID, shown in Figure 138 on page 327. However, the **Process/Status** column is abbreviated to accommodate the **SQLID** column. Also, the status indicator RESTART is shortened to RSTRT. The list is arranged initially by **SQLID** and **TIMESTAMP**, in ascending order. You can use the USER command to redisplay the list of pending processes for the current SQL ID only.

## Process Information

Use the I line command on the Pending Process List panel to display additional information, or attributes, about any listed pending process.

```
------------------------- Pending Process Attributes -------------------------
Command ===>


Control File: FOPDEMO.CONTROL.SAMPLE2
Status      : RETRY

Extract File: FOPDEMO.EXTRACT.SAMPLE2
Created By  :
  Job       : FOPDEMO
  SQLID     : FOPDEMO
  Run On    : 1998-12-15 10.14.22

Last Process: INSERT
  Job       : FOPDEMO
  SQLID     : FOPDEMO
  Run On    : 1998-12-15 12.30.55
```

*Figure 140. Pending Process Attributes*

The Pending Process Attributes panel includes:

- The name of the Control File and the associated Extract or Archive File.
- The status of the entry (RETRY or RESTART).
- Creation information about the Extract or Archive File.
- Information about the last process attempted using this Control File.

Use END to return to the Pending Process List panel.

## Execute Retry/Restart

Use the Select line command, S, to select a process from the list and retry or restart it, according to its status.

## Parameters Panel

The Specify Parameters and Execute panel is displayed before retrying or restarting a process. The panel title indicates the type of process (i.e., Insert, Update/Insert, Restore, or Delete) to retry or restart. You can respecify the displayed parameters, except for the names of the Extract, Archive, or Control Files and, for the Restore Process, the value for **Restore Method to Use**. The following figure shows the panel displayed when an Insert Process is restarted. (For details about these prompts, see Insert Parameters Panel in the *Move User Manual* or Specify Restore Parameters Panel in the *Archive User Manual*.)

```
---------------- Specify Restart INSERT Parameters and Execute ----------------
Command ===>

Names for Extract File and Control File:
  Extract File DSN : FOPDEMO.EXTRACT.SAMPLE2
  Control File DSN : FOPDEMO.CONTROL.SAMPLE2

Process Options:
  Age Date Values                 ===> N        (Y-Yes, N-No)
  Commit Every Nth Row            ===> 1000     (1-1000, Blank-Site Limit)
  Limit Number of Discarded Rows  ===>          (1-4294967295, Blank-No Limit)

Run Process in Batch or Online    ===> O        (B-Batch, O-Online)
  If Batch, Review or Save JCL    ===> R        (N-No, R-Review, S-Save)

Process Report Type               ===> D        (D-Detailed, S-Summary)
```

*Figure 141. Restart INSERT Parameters*

During Restart processing, the initial process commences with the rows after the last committed row. Any discarded rows from the original processing are retained. Any rows discarded in the current execution are marked accordingly. During Retry processing, only rows that were initially skipped due to an error condition are reprocessed.

Processing progress is displayed during online execution, and a Process Report is generated. If a process succeeds and there are no rows to retry or restart, the process is removed from the Pending Process List panel. A process that does not succeed remains on the list and can be retried or restarted, as needed.

## Resubmit a Batch Job

If a processing failure occurs during the execution of a batch job, the failed process can be retried or restarted by adding the RESTART operand to the end of PARM on the EXEC statement of the batch job and resubmitting the batch job with no other changes. The RESTART operand must be separated from the previous operand in PARM by a blank. The RESTART operand must be used, regardless of whether the actual condition is a retry or restart; the appropriate action is determined automatically. (Note that this is an alternative to returning to the online system to initiate a retry or restart of a failed process.)

**Note:** The RESTART operand may also be used to resubmit a batch job to perform a Restore or Archive with delete process, and to resubmit the execution of an Archive Utility that contains one or more RESTORE control statements. (See the *Batch Utilities Guide*, Processing Utilities for detailed information on the batch utilities.)

# Chapter 11. Browse Related Data

While using the Browse and Point-and-Shoot facilities, you can join data in related tables and use various techniques to manage the display while viewing the data. Then you can join related rows from multiple tables and manage the display of data according to your needs. While browsing Extract or Archive Files or using the Point-and-Shoot facility, you can also create a report about the contents of the display.

**Note:** For information about joining and managing relational data during an Access edit or browse session, see the *Access User Manual*, Join Tables.

## Browse Facility

Optim provides a full Browse facility for online review of data in files generated by Optim. You can browse the contents of the following files to obtain information or to verify the data.

- Archive Files, stored on disk, including converted and Subset Archive Files
- Extract Files, stored on disk, including converted Extract Files
- Compare Files
- Control Files

To view data from Archive Files and Extract Files, the Browse facility offers several modes: Table Mode, Report Mode, Summary Mode, and Access Definition Mode. The features highlighted in this section apply to browsing Archive File and Extract File data in Table Mode only. Table Mode provides a dynamically formatted display of related data. In Table Mode, you can display data by specifying selection criteria for rows in any table in the file and by joining related rows from remaining tables.

You can browse Control Files in Report Mode only; thus, you cannot join or manage Control File data using the Browse facility. For information about browsing in Report, Summary, and Access Definition Modes, see the *User Manual* for Archive, Move, or Compare.

**Note:**

You can browse a converted archive or extract file in Table Mode; however, since a converted file does not include object definitions, you can browse only the initially displayed table and cannot join data in related tables.

You cannot browse an archive or extract file that is stored on tape.

The features highlighted in this section also apply to browsing data in Compare Files. For information unique to browsing Compare Files, see the *Compare User Manual*, Browse Compare File.

## Point-and-Shoot Facility

The Point-and-Shoot facility allows you to select specific rows from the Start Table for an Archive, Restore, Extract, or Compare Process. Also, you can join to related tables referenced in the Access Definition and select related rows. If you have defined selection criteria for the Start Table, you can choose to override or supplement the selection criteria with the Point-and-Shoot list. For general information about the Point-and-Shoot facility, see "The Point-and-Shoot Facility" on page 72.

## About this Section

This section describes features common to both the Browse facility and the Point-and-Shoot facility. These features apply to browsing data in Archive Files, Extract Files, and Compare Files, as well as to selecting rows during a Point-and-Shoot session for an Archive, Restore, Extract, or Compare Process. Any differences between these facilities are noted.

The figures in this section show the Browse facility while browsing an Archive File. Note that the title of the panel in a Browse or Point-and-Shoot display indicates the type of session. For example, in an Archive File browse session, the title indicates "Archive Browse" and the name of the Archive File; whereas, in a Point-and-Shoot session, the title indicates "Optim: Point-and-Shoot."

# Display Basics

The display for the Browse and Point-and-Shoot facilities is very similar to an ISPF/PDF display. This section discusses the display modes, screen elements, and commands available for both Browse and Point-and-Shoot.

## Display Format

There are two basic display formats: columnar and sidelabels. Columnar format is the default. In columnar format, headings are displayed before the panel and the data is displayed in columns after the headings. Columnar format allows you to look at multiple rows in a table and join to related tables. The following figure shows data from the ORDERS table displayed in columnar format.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                  Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ========================= 1 OF 79 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___         10   00049  1998-01-26  14.22.31        9.22          WE005
___         23   00068  1998-01-26  08.16.09       14.80          WE005
___        130   00049  1998-01-26  10.12.39        7.02          WE005
___        143   00069  1998-01-26  14.22.31       17.60          WE012
___        160   00067  1998-01-26  10.22.31       13.82          WE012
___        176   00069  1998-01-26  11.28.30       11.88          WE012
___        207   00067  1997-02-24  12.12.51       48.52          WE012
___        222   00068  1998-01-26  14.22.31       19.05          WE005
___        278   00068  1998-01-26  11.51.47       21.97          WE005
___        373   00049  1998-01-26  12.08.13       27.97          WE005
___        404   00049  1998-01-26  12.18.58       23.37          WE005
___        658   00069  1998-01-26  10.00.28       13.33          WE012
___        686   00067  1998-01-26  12.08.13       21.97          WE012
___        752   00069  1998-01-26  09.11.47       33.99          WE012
___        817   00069  1998-01-26  16.00.00        7.85          WE012
___        869   00015  1998-01-26  12.08.13        8.23          WE005
___        879   00070  1998-01-26  12.08.13       11.85          WE012
```

*Figure 142. Columnar Screen Display*

## Screen Elements

The screen elements in the columnar and sidelabels display formats are the same; only the positioning is different. For more information about sidelabels format, see "Sidelabels Format" on page 356. The elements are:

**Command**
Prompt for primary commands. Refer to "Primary Commands" on page 334 for a summary of available primary commands.

**Cmd**
**or**
**LineCmd**
The line commands area. Refer to "Line Commands" on page 335 for a summary of available line commands.

In columnar format, the label is **Cmd**.

In sidelabels format, the label is **LineCmd**.

**Chg** Displayed in a Compare File browse session only. Identifies the type of change as one of the following:

**blank** No changes.

**D** Direct changes. One or more columns, other than the match key columns, are different.

**R** Related changes. One or more rows in a related table have a direct change or are unmatched.

**U** Unusual rows. Identifies orphan rows and pairs of compared rows that result in a different set of related data. Orphan rows are identified when the child relationship column does not correspond to a parent row. When the values in the parent relationship columns in a pair of matched rows are not the same, different sets of child rows are related.

Since these conditions are determined by relationships, they are detected only when sets of related data are compared.

**/**
**|**
**** Identifies rows in a single source table that have duplicate match key values. The brackets group the rows together. These rows are not compared.

For additional information about Chg flags, see the *Compare User Manual*, Compare Process.

**Src** Displayed in a Compare File browse session only. Identifies the source of the row as one of the following:

**1** Source 1.

**2** Source 2.

**12** Source 1 and Source 2. These rows are identical and only one copy of the row is displayed.

In sidelabels format, identical Source 1 and Source 2 columns are displayed once and differing or unmatched columns are displayed separately. **Src** indicates these unmatched rows as "1/2."

For additional information about Src flags, see the *Compare User Manual*, Compare Process.

**Status Flag**
Displayed in Point-and-Shoot only. Identifies selected rows.

In columnar format, the label is **F**. In sidelabels format, the label is **Row Status**.

**Table** The fully qualified name of the table or view. The name is truncated if the Creator ID and Table Name together are more than 22 characters.

In a Compare File browse session, the name is displayed without the Creator ID.

**Short Name**
An identifier showing the level for each table (T*n*) or view (V*n*). In a multiple table display, T1 identifies the first level, T2, the second level, etc. Several tables can be joined at the second or greater level. Tables joined at the same level (i.e., stacked tables) have the same level identifier. (See "Multiple Table Display" on page 341 for additional information about multi-way joins and stacked tables.) The level identifier can be substituted for the table name as a command operand.

**x OF y**
The position of the first data row on the screen (*x*) in relation to the total number of displayed rows (*y*) from the table.

**Horizontal Scroll Indicator**
Indicator, in the form **MORE** or **<<MORE**, that additional data can be displayed by scrolling LEFT or RIGHT. See "Scroll" on page 348 for additional information about scrolling.

**Column Headings**
Column names are used as headings for the rows from each table.

**Column Count**
Displayed in sidelabels format only. Indicates the current position of the first column within the set of columns.

## Available Commands

Both primary and line commands are available in the display. Many of these commands are introduced in the following sections. Detailed information about each command is provided in the *Command Reference Manual*, Primary Commands and Line Commands.

### Primary Commands

The following is a list of primary commands available using the Browse or Point-and-Shoot facility, with exceptions noted.

ANCHOR
ATTRIBUTES
BOTTOM
CANCEL
CAPS[1]
COLUMNS
COUNT[1]
DOWN
END
EXCLUDE
EXPAND
FIND
FLIP[3]
HEX
INDENT[4]
INFO[3]
JOIN[4]
LEFT
LIST[2]
LOCK
MAX FETCH ROWS[2]

ONLY
OPTIONS
REPORT[1]
RESET
RESORT[2]
RFIND
RIGHT
SELECT RELATED[2]
SELECTION CRITERIA[1]
SHOW
SHOW SQL[2]
SIDELABELS
SORT[5]
SQL[1]
START[2 4]
SWITCH[4]
TOP
UNJOIN[4]
UNLOCK
UNSELECT RELATED[2]
UP
ZOOM

**Note:**

[1]Not available in a Compare File browse session.

[2]Available only in a Point-and-Shoot session. (START is also available in a Compare File browse session.)

[3]Available only in a Compare File browse session.

[4]Not available in sidelabels format.

[5]Limits on a DB2 ORDER BY clause apply.

## Line Commands

The following line commands are available using the Browse or Point-and-Shoot facility, with the exceptions noted.

**X, X*n*, XX**
Exclude
Temporarily remove a row, a number of rows, or a block of rows from the display.

**F, F*n*** First
Redisplay the first row, or the first *n* rows, in a block of excluded rows.

**I** Info
Obtain information about tables that contain related changes during a Compare File browse session.[1]

**J** Join
Join to related tables.[2]

**L, L*n*** Last
Redisplay the last row, or the last *n* rows, in a block of excluded rows.

**SR, SSR**
Select Related
Specify a row, or block of rows, to be selected in a Point-and-Shoot session.[3]

**S, S*n*, SS**
Show
Redisplay one row, *n* rows, or all rows in a block of excluded rows.

**SID** Sidelabels
Display the row in sidelabels format, or return the display to columnar format.

**UNJ** Unjoin
Remove joined tables from the display.[2]

**UR, UUR**
Unselect Related
Specify a row, or block of rows, to be deselected in a Point-and-Shoot display.[3]

**Z** Zoom
Display all rows in the table, or return to the multiple table display.

**Note:**

[1] Available only in a Compare File browse session.

[2] Not available in sidelabels format.

[3] Available only for the Start Table in a Point-and-Shoot session.

## Join Tables

A browse session begins with the display of rows from a table selected on the Browse Parameters panel; a Point-and-Shoot session begins with the display of rows from the Start Table specified on the Select Tables/Views for AD panel or the Archive Selection Criteria - Tables panel (Restore Process only). Related rows from other tables, which are included in the Access Definition or the file, may be added to the display using a Join command. You can use various navigation aids to view a set of related data in the display at one time.

The table for which the join is executed, the *anchor table*, is joined to a related table and any related rows from that table are displayed. To perform a join, Optim requires the name of the table to join and a relationship between the named table and the anchor table. The relationship can be a DB2 or an Optim relationship and must be referenced in the Access Definition or the file. It does not matter if the anchor table is the parent or the child table; you can join from the parent to the child or vice versa.

**Note:** You cannot join to a reference table during a Point-and-Shoot session, even if a relationship exists.

## Join Command

You can use the JOIN primary command or the J line command to display simultaneously a row from the anchor table and related rows from the joined table. Use the J line command to select a specific row in the anchor table.

By default, a JOIN primary command operates on the first displayed row in the table at the lowest level in the display. This table becomes the anchor table for that operation. To specify the anchor table, use the JOIN command with the FROM operand and anchor table name. For example, JOIN FROM ORDERS or JOIN FROM T1. Additionally, you can designate the anchor table and row by positioning the cursor on the desired row in the anchor table.

In the following figure, the JOIN command is specified and the cursor is positioned on the row for ORDER_ID 211.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===> JOIN                                          Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************* TOP ************************************
___        205   00192  1997-05-24  12.12.51       48.52          NE012
___        205   00192  1997-02-24  12.12.51       48.52          NE012
___        206   00093  1997-02-24  12.12.51       48.52          SW012
___        207   00067  1997-02-24  12.12.51       48.52          WE012
___        208   03189  1997-02-24  12.12.51       48.52          NW012
___        209   00143  1997-02-24  12.12.51       48.52          SW012
___        210   00239  1997-02-24  12.12.51       48.52          NW012
___        211   00284  1997-02-24  12.12.51       48.52          SC012
___        212   00327  1997-02-24  12.12.51       48.52          SC012
___        213   00371  1997-02-24  12.12.51       48.52          NE012
___        214   00415  1997-02-24  12.12.51       48.52          NC012
___        215   02221  1997-02-24  12.12.51       48.52          SE012
___        216   00019  1997-02-24  12.12.51       48.52          SC012
___        217   00110  1997-02-24  12.12.51       48.52          SE012
___        288   00131  1997-02-24  12.12.51       48.52          SW012
___        333   01210  1997-02-24  12.12.51       48.52          SE012
___        417   00448  1997-02-24  12.12.51       48.52          SE012
___        727   00420  1997-02-24  12.12.51       48.52          NE012
```

*Figure 143. JOIN Command*

Press ENTER to execute the JOIN command. If only one table in the Access Definition or the file is directly related to the anchor table, any related rows from that table are displayed automatically. If the anchor table has no related tables, a message indicates the command cannot be satisfied. See "No Related Tables" on page 339 for more information.

### Join Using DB2 LIKE Syntax

You can use DB2 LIKE syntax to display a selection list of tables to join at any level. For example, to find tables with names beginning with "D" that are related to the ORDERS table, specify:

```
JOIN D% FROM ORDERS
```

If several tables with names that begin with "D" are directly related to ORDERS, they are included in a selection list. You can select one or more listed tables. If only one table meets the criteria, it is joined and displayed without displaying a selection list.

To minimize typing, you can use the level indicator to specify the anchor table for the FROM operand. For example, the ORDERS table in Figure 143 is assigned the level T1. You can specify the JOIN command as:

```
JOIN D% FROM T1
```

## Select Tables

Often, more than one table is related to the anchor table. When this is the case, a selection list is displayed. The selection list includes all related tables in the Access Definition or the file.

An example of the selection list, displayed after entering the JOIN command, is shown in the following figure.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************* TOP ************************************
___         205
___         206   +--------Select One or More Related AD Tables--------+
___         207   ¦ Cmd    CreatorID.TableName      From  Type  1 OF 2 ¦
___         208   ¦ --- ------------------------------------          ¦
___         209   ¦ ****************** TOP ******************          ¦
___         210   ¦ _S_ FOPDEMO.CUSTOMERS           DB2  PARENT        ¦
___         211   ¦ _S_ FOPDEMO.DETAILS             DB2  CHILD         ¦
___         212   ¦ ***************** BOTTOM ******************          ¦
___         213   +-----------------------------------------------------+
___         214
___         215   02221  1997-02-24  12.12.51       48.52          SE012
___         216   00019  1997-02-24  12.12.51       48.52          SC012
___         217   00110  1997-02-24  12.12.51       48.52          SE012
___         288   00131  1997-02-24  12.12.51       48.52          SW012
___         333   01210  1997-02-24  12.12.51       48.52          SE012
___         417   00448  1997-02-24  12.12.51       48.52          SE012
___         727   00420  1997-02-24  12.12.51       48.52          NE012
```

*Figure 144. Table Selection List for Join*

The names of related tables are listed first horizontally on the selection list. **From** identifies each relationship as a DB2 relationship, an Optim relationship, or both. **Type** indicates the role of the table in the relationship, as child or parent of the anchor table. Use the Select line command, S, to select one or more tables from the list and press ENTER or use END.

If there is only one relationship between the anchor table and the selected table, the tables are joined and displayed, as shown in the following figure.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 7 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___         211   00284  1997-02-24  12.12.51       48.52          SC012

Cmd F == Table: FOPDEMO.DETAILS(T2) ======================== 1 OF 12 ==========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- ----------------
*** ********************************* TOP ************************************
___         211   AD005           4            15.00
___         211   CH006           4            14.00
___         211   CM019           4            20.00
___         211   DR011           4            20.00
___         211   DR012           4            19.00
___         211   DR013           4            22.00
___         211   DR014           4            17.00
___         211   DR017          14            22.00
___         211   DR018           4            17.00
___         211   DR019           4            21.00
___         211   DR023           4            21.00
___         211   DR038           4            20.00
```

*Figure 145. Result of Join*

If more than one relationship exists between the anchor table and the selected table, a selection list of relationships is displayed. For more information about this selection list, see “Select Relationships.”

### No Related Tables

If the anchor table is not related to any tables in the Access Definition or the file, a join cannot be performed.

## Select Relationships

If there are two or more relationships between the anchor table and a selected table, a selection list of relationships is displayed, as in the following figure.

```
-------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ***********************************
___       205   00192  1997-05-24  12.12.51       48.52          NE012

+--------------------------Select One Relationship---------------------------+
¦ Available Line Commands: I-Info,S-Select                                   ¦
¦ Cmd Relation  From  Child Table Name            Parent Table Name          ¦
¦ --- ------>> ------------------------------------------------------------ ¦
¦ ********************************** TOP ********************************** ¦
¦ ___ RCO       DB2   FOPDEMO.ORDERS              FOPDEMO.CUSTOMERS          ¦
¦ ___ RCO1      OPT   FOPDEMO.ORDERS              FOPDEMO.CUSTOMERS          ¦
¦ ******************************** BOTTOM ******************************** ¦
+------------------------------------------------------------------------------+
```

*Figure 146. Relationship Selection List*

### Panel Elements

The panel includes:

**Cmd** Line command entry area. Valid commands are:

- **I** Display the Browse Relationship panel, allowing you to view the names and data type of the columns in the selected relationship.
- **S** Select the relationship to be used to join the anchor table and the selected table.

**Relation**
The 1 to 8 character name of the relationship.

**From** The type of relationship. Valid commands are:

- **DB2** The relationship is defined to the DB2 Catalog.
- **OPT** The relationship is defined to the Optim Directory.

**Child Table Name**
The fully qualified name (Creator ID and Table Name) of the child table in the relationship.

**Parent Table Name**
The fully qualified name (Creator ID and Table Name) of the parent table in the relationship.

When you have selected a table, press ENTER or use END to display the joined tables, as shown in Figure 145 on page 338.

### Browse Relationship Panel

In the following figure, information about the relationship RCO is displayed in the panel.

```
---------------------------- Browse Relationship -----------------------------
OPTION ===>                                                 SCROLL ===> PAGE

             Browse only Display of DB2 Relationship RCO

Parent: FOPDEMO.CUSTOMERS            Child: FOPDEMO.ORDERS
                                                                  1 OF 1
      Column Name        Data Type         Column Name        Data Type
----------------------->> ----------  ----------------------->> ----------
******************************** TOP ********************************
CUST_ID                  CH(5)       CUST_ID                  CH(5)
****************************** BOTTOM ******************************
```

*Figure 147. Browse Relationship - Join*

#### Panel

The Browse Relationship panel includes:

**Parent** The fully qualified name of the parent table.

**Child** The fully qualified name of the child table.

**Column Name**
The expressions and names of columns for each table that are used in the relationship.

**Data Type**
The data type of the matched columns. In addition to the standard DB2 data types, **Data Type** may be EXPR, to indicate that the relationship is based on an expression for which the data type is determined when the relationship is used.

Use END or press ENTER to return to the Select One Relationship panel.

## Join All Command

If you have a complex set of relationships in an Access Definition or file, you may want to join all related tables at one time so you can view all parent-child relationships. Instead of manually joining each table in the display, you can use the JOIN ALL command.

Use the JOIN ALL command to join all tables in the Access Definition or file related to the initially displayed table.

**Note:** The Point-and-Shoot facility uses the Start Table as the initial table in the display, whereas the Browse facility allows you to specify any table in the file.

You cannot use the JOIN ALL command if tables are already joined in the display. If you enter JOIN ALL when one or more tables are joined, an error message is displayed.

The following figure shows the results of executing the JOIN ALL command for the CUSTOMERS table. In this example, the ORDERS, DETAILS, ITEMS, and SALES tables are related tables.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                  Scroll ===> PAGE

Cmd F == Table: FOPDEMO.CUSTOMERS(T1) ====================== 1 OF 704 === MORE>>
      CUST_ID       CUSTNAME             ADDRESS             CITY       STATE
      ------- -------------------- ------------------- --------------- -----
___    00001  Audio-Video World    593 West 37th Street Brass Castle     NJ

Cmd F == Table: FOPDEMO.ORDERS(T2) ================ STACKED = 1 OF 4 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___        20   00001  1998-01-26  08.16.09       14.80          NE005

Cmd F == Table: FOPDEMO.DETAILS(T3) ========================== 1 OF 4 ==========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- -----------------
___        20   AD005          5            15.00

Cmd F == Table: FOPDEMO.ITEMS(T4) ============================ 1 OF 1 ==========
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== -------------- ------ ----------
*** ********************************** TOP ***********************************
___    AD005  Conan the Barbarian  Adventure       R        15.00
*** ******************************** BOTTOM **********************************
```

*Figure 148. Results of JOIN ALL*

If there are two or more relationships between a parent and child table, you are prompted to select the relationship to use for the join. See Figure 146 on page 339 for details about this selection list.

In Figure 148, the SALES table is not displayed because it is stacked. For information about the multiple table display, see “Multiple Table Display.” For information about multi-way joins and stacked tables, see “Special Considerations for Multi-way Joins” on page 343.

## Multiple Table Display

After one or more joins, the display shows related rows from multiple tables. In the following figure, the CUSTOMERS table is joined to the ORDERS table, and the SALES table is joined to the CUSTOMERS table.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 4 OF 78 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___         10   00049  1998-01-26  14.22.31        9.22          WE005

Cmd F == Table: FOPDEMO.CUSTOMERS(T2) ====================== 1 OF 1 === MORE>>
      CUST_ID       CUSTNAME             ADDRESS             CITY       STATE
      ------- -------------------- ================== --------------- -----
___    00049  Pick-a-Flick         120 Central Avenue   Blue Jay          CA

Cmd F == Table: FOPDEMO.SALES(T3) ========================== 1 OF 1 === MORE>>
      SALESMAN_ID    SALESMAN_NAME         AGE     SEX   TERRITORY
      ----------- -------------------- ----------- --- --------------
*** ********************************* TOP ***********************************
___     WE005     Captain Kangaroo              79  M  West
*** ******************************** BOTTOM *********************************
```

*Figure 149. Sample Multiple Table Display*

An information line is provided for each table in the joined display. This line includes:

**Name of the table**
FOPDEMO.ORDERS

**Identifier indicating the level of the table**
(T1)

**Row number relative to the total number of rows selected for display**
4 OF 78

The TOP and BOTTOM markers are displayed before the first line and after the last line of the lowest-level table, the SALES table in Figure 149. These markers indicate the first and last related rows in the table.

If more than one table is joined at one level (i.e., a multi-way join), the STACKED indicator is displayed. See “Stacked Tables” on page 343 for details.

The maximum number of tables that can be joined is 64. If necessary, higher-level tables are scrolled from the display to make room at the end for newly joined tables. Use the UNJOIN command to redisplay higher⌂level tables by removing lower-level tables. For more information about the UNJOIN command, see “Unjoin Tables” on page 347.

**Note:** Additionally, you can use the ZOOM command to focus the display on a single higher-level table in the display, without unjoining the table. For more information about the ZOOM command, see “Zoom a Joined Table Display” on page 346.

## Change Highest Display Levels

In a Point-and-Shoot session or Compare File browse session, you can use the START command to display the specified table at the beginning of the display, temporarily removing higher-level tables from the display. The table name or identifier of a table currently active in the display can be specified with the START command (e.g., START SALES or START T3), as long as there is sufficient screen space to display at least one line of the lowest-level table.

**Note:** The START command does not change the Start Table or remove tables from the Access Definition or Compare File.

The START command is useful when you want to display additional rows from the lowest-level table without zooming the display. Changing the highest-level table is also useful when you change from split screen to single screen and you want to focus on one or more specific tables.

You can also use the **Indented Table Display** to view or navigate the tables. For more information, see "Display Traversal Paths" on page 344.

## Special Considerations for Multi-way Joins

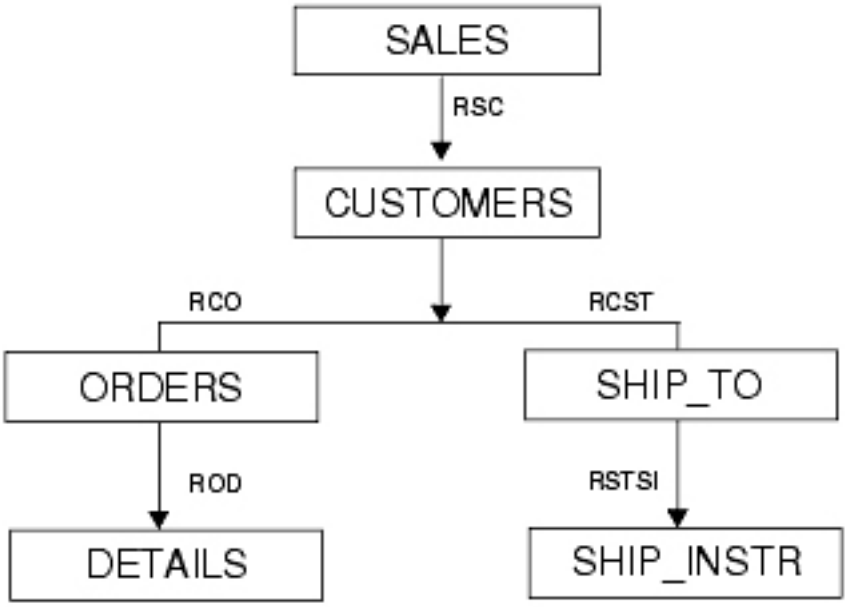
In many cases, a database table is related to two or more other tables. When multiple tables are related, several navigational paths are available for joining and browsing the data. You may find it useful to display the related data obtained by traversing several of these paths. You can use multi-way joining to browse multiple related tables without unjoining any tables.

### Stacked Tables

When multiple tables are joined to a single table, the tables are "stacked" in the order they were joined. The most recently joined table is before the second most recent, the second before the third, and so on. The table at the beginning of the stack is displayed; the other tables are hidden.

You can stack multiple tables at any display level except the first level. If tables are joined after the stacked table, these lower-level tables are hidden when the stacked table is hidden.

Consider the following example. An Access Definition or a file contains the tables shown in the following diagram with the indicated relationships.




If you are looking for information about a specific order, you would start the Browse or Point-and-Shoot session with ORDERS. When you use the JOIN command, a selection list of the related tables, CUSTOMERS and DETAILS, is displayed. You can select both tables, which are then joined in a stack with the last joined table, DETAILS, displayed. The other table in the stack, CUSTOMERS, is hidden.

### Switch the Displayed Table

You can display any table in the stack using the SWITCH command.

- If only two tables are joined in the stack, the hidden table is displayed and the previously displayed table is hidden.

- If three or more tables are joined in the stack, a selection list is displayed.
- If a single table is joined, a message is displayed.

By default, the SWITCH command operates on the lowest level of the display. You can use the cursor position to indicate the target level for the SWITCH command. You may also find it useful to assign the SWITCH command to a PF key and use the cursor position to indicate the target level.

### Using SWITCH with several tables

If three or more tables are joined in a stack, you can use operands to display a specific table in a stack at a specific level. To display a selection list of joined tables for a stack at the second level, specify:

```
SWITCH T2
```

To display the next table in a stack on the second level, specify:

```
SWITCH T2 NEXT
```

You can also specify the name of a table. To display the ORDERS table in a stack on the second level, specify:

```
SWITCH T2 ORDERS
```

### Related lower-level tables

Any table in a stack can be joined to lower-level tables. However, only related data is displayed. Any subordinate tables or stacks are hidden or displayed with the stacked table to which they are joined. If you switch the display from one table in a stack to another, the related lower-level tables also switch.

## Display Traversal Paths

Using the Browse or Point-and-Shoot facilities with multi-way joining can become very complex. You may have joined to lower-level tables for one or more tables within the stack and you may have "nested stacks" (that is, stacked tables within a stack). Even if there are no stacks, it may be difficult to keep track of joined tables and the relationships between them, especially if all levels cannot be displayed on the screen at one time.

You can use the INDENT primary command to obtain the **Indented Table Display**, which provides an indented list of all active tables. The format of this list identifies the display-level hierarchy of all joined tables and all stacks.

**Note:** The INDENT command is available while using the Browse or Point-and-Shoot facility, regardless of whether you have stacked or joined tables.

In the following figure, CUSTOMERS is the Start Table. All active tables are listed, and asterisks are used to identify the tables currently displayed on the main panel from which the indented listed was generated. The display level of each table is indicated. Tables in a stack are on the same level and, except for the active table, are identified by the word "Yes" in the **Stkd** column, as shown in the following example for the SALES and SHIP_TO tables.

```
--------------------------- Indented Table Display ---------------------------
Command ===>                                                Scroll ===> PAGE

Default Creator ID: FOPDEMO                                          1 OF 7

Cmd Lvl                      Table Name                  Stkd Relation  Type
--- --- ------------------------------------------------ ---- -------->> ----
*** ********************************* TOP ************************************
___ 1*  CUSTOMERS                                            START TABLE
___ 2*   C:ORDERS                                           RCO      DB2
___ 3*    C:DETAILS                                         ROD      DB2
___ 4*     P:ITEMS                                          RID      DB2
___ 2    C:SALES                                        Yes RSC      DB2
___ 2    C:SHIP_TO                                      Yes RCST     DB2
___ 3     C:SHIP_INSTR                                      RSTSI    DB2
*** ******************************** BOTTOM ***********************************
```

*Figure 150. Indented Table Display*

## Panel Elements

The panel includes:

**Default Creator ID**
The default Creator ID specified in the Access Definition.

**x OF y**
The relative position of the first displayed table (*x*) and the total number of active tables (*y*).

**Cmd** Line command entry area. Use the S line command to switch the display to a different table in the stack.

For example, in the preceding figure, the ORDERS, SALES, and SHIP_TO tables are stacked at level 2, and the ORDERS table is currently displayed. You can display the SHIP_TO table by typing "S" in **Cmd** for the SHIP_TO table, or any subordinate table of the SHIP_TO table (e.g., SHIP_INSTR), and pressing ENTER.

**Lvl** The display level for the table. Tables at the same level are in a stack, and the currently displayed table is identified by an asterisk next to its level number. Any stacked tables that are not currently displayed are identified by the word "Yes" in **Stkd**.

**Table Name**
The name of the table. If the Creator ID is different from the default Creator ID, it is included with the name. (In a Compare File browse session, the Creator ID is always included with the name.)

The Start Table is listed first, with other tables listed in the order displayed. Indentations reflect the display level. Tables in a stack are listed in stack order.

The names of all tables, except the Start Table, are prefixed with "C:" or "P:" to indicate whether a table is the parent or child in the relationship with the table under which it is indented.

**Stkd** Any stacked tables that are not currently displayed are identified by the word "Yes."

**Relation**
Name of the relationship between the table and the table under which it is indented. "START TABLE" is always specified for the Start Table.

**Type** Identifier to denote whether the relationship is defined in the DB2 Catalog or the Optim Directory.

## Zoom a Joined Table Display

The Zoom command switches the display between a multiple table display and a single table format without altering the joins. You can use the Zoom command when multiple tables are joined and you want to display additional rows from an upper-level table.

### Zoom Line Command

To switch from a multiple table display to focus on a single table, enter the Z line command for a row in the table. To return to a multiple table display, enter the Z line command for any row. The row on which the command is entered is displayed first in the multiple table display.

### ZOOM Primary Command

You can select a table to zoom using an operand (e.g., ZOOM ORDERS or ZOOM T1), or by positioning the cursor. If the cursor is positioned on a row, that row is the first line of data in the zoomed display. If ZOOM is executed without positioning the cursor or specifying a table name operand, the lowest-level table is displayed in single table format.

Similarly, if the cursor is positioned on a row in the zoomed display when ZOOM is requested again, the selected row is scrolled to the first position in the multiple table display. If the cursor is not positioned, the first displayed row in the zoomed display is shown in the multiple table display.

The primary command can be entered as ZOOM DETAILS or ZOOM T2, as shown in the following figure.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE ---------------------
COMMAND ===> ZOOM T2                                          Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ========================== 7 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___       211   00284  1997-02-24  12.12.51       48.52          SC012

Cmd F == Table: FOPDEMO.DETAILS(T2) ============== STACKED = 2 OF 12 ==========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- -----------------
___       211   CH006           4            14.00

Cmd F == Table: FOPDEMO.ITEMS(T3) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== -------------- ------ ----------
*** ********************************** TOP ***********************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ******************************** BOTTOM **********************************
```

*Figure 151. Zoom Multiple Table Display*

In the following figure, the display is changed from a multiple table to a single table display of the DETAILS table.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
COMMAND ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.DETAILS(T2) ================ ZOOMED = 2 OF 12 =========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- -----------------
___       211   CH006             4             14.00
___       211   CM019             4             20.00
___       211   DR011             4             20.00
___       211   DR012             4             19.00
___       211   DR013             4             22.00
___       211   DR014             4             17.00
___       211   DR017            14             22.00
___       211   DR018             4             17.00
___       211   DR019             4             21.00
___       211   DR023             4             21.00
___       211   DR038             4             20.00
*** ******************************** BOTTOM ***********************************
```

*Figure 152. Zoomed Display*

**Note:** Optim returns to the multiple-table columnar display, even if the zoom is executed in sidelabels format. See "Sidelabels Format" on page 356 for a description of sidelabels format.

### Impact on Joined Tables

All joins between tables are retained when zoom is used. The single-table zoomed display can be scrolled. If a table is scrolled during a zoomed display, any joined lower-level tables are logically scrolled. When the multiple table display is restored, the scrolled data is displayed.

## Unjoin Tables

In a multiple table display, the UNJOIN command severs the join between the specified table and the next higher-level table. The specified table and any related lower-level tables are removed from the display.

When you enter the UNJOIN command, you can either position the cursor on the desired table, regardless of level, or you can specify the table name or identifier as an operand (e.g., UNJOIN ORDERS or UNJOIN T2). The UNJOIN command removes the specified table and all related lower-level tables. If you do not specify a table, the lowest-level table is unjoined.

You can also use the UNJ line command on a row in the table you want to unjoin. Entering the UNJ command removes the specified table and any related lower-level tables from the display.

If the unjoined table is not stacked, the remaining joined tables are repositioned and the display of the lowest-level table expands to include any additional rows.

### Unjoining Stacked Tables

If the unjoined table is stacked, the next table in the stack is displayed along with any related lower-level tables.

You can use operands with the UNJOIN command to unjoin all tables in a stack. UNJOIN ALL removes all tables in a stack at the lowest level. If there is no stack at the lowest level, the table at the lowest level is unjoined. When there are multiple stacks, you can specify the target stack using the cursor position or the level indicator (e.g., UNJOIN T2 ALL).

## Manage the Display

An Access Definition or file may contain extensive data from hundreds of tables, each with dozens of columns and thousands of rows. Navigating this data to locate a specific subset of related rows can be daunting.

Optim provides many techniques and features to help you:

- Navigate large lists of data easily and efficiently.
- Find specific data and manage the display.
- Review information about columns or data.
- Arrange and format displays of wide data to accommodate browsing.

## Scroll

Optim uses familiar ISPF commands to scroll data, including DOWN, UP, BOTTOM, TOP, LEFT, and RIGHT. These commands can be entered on the command line or assigned to program function keys. ISPF SCROLL values, CSR, PAGE, DATA, HALF, MAX, and *n* (where *n* is the number of lines to scroll) are supported.

### Vertical Scroll

You can scroll using DOWN, UP, BOTTOM, and TOP to view additional rows when the number of rows retrieved exceeds the number of lines available on the screen. DOWN scrolls the display forward and UP scrolls the display backward. Operands for the DOWN and UP commands allow you to scroll by page, numeric value, or cursor position. You can also use the BOTTOM command to scroll to the last entry and the TOP command to scroll to the first entry in the table.

When a single table is displayed, vertical scrolling is exactly like vertical scrolling in ISPF. Scrolling a multiple table display is discussed in "Coordinated Scroll."

### Horizontal Scroll

You can scroll using LEFT and RIGHT to view additional columns when the combined width of the columns exceeds the width of the screen. Operands for these commands allow you to scroll by column name, numeric value, or cursor position. MORE, with an appropriate direction arrow, is displayed on the information line for a table to indicate that you can scroll horizontally.

When multiple tables are displayed, indicate the table to scroll by specifying a table name or identifier with the column name (for example, T1.CUST_ID) or by positioning the cursor. If a table is not indicated, the lowest-level table is scrolled.

### Coordinated Scroll

Scrolling a table vertically in a multiple table display automatically scrolls any joined lower-level tables. For example, scrolling the DETAILS table in Figure 151 on page 346 automatically scrolls the ITEMS table to display the related rows, if any. Scrolling the ORDERS table scrolls the DETAILS table and the ITEMS table to the related rows.

You can indicate the table you want to scroll by including its name or identifier as an operand for the command or by positioning the cursor. For example, UP T2 scrolls the table at level T2 backwards. If the cursor is not positioned in a window and no operands are given with the scroll command, the table displayed at the lowest level is scrolled.

You can also scroll by positioning the cursor on a specific row in any table and pressing a PF key assigned to scroll. All lower-level tables are scrolled accordingly.

## Navigate and Manage Display of Data

If working with many rows, you may find it difficult to find the rows you want to browse or select. The following commands allow you to search for specific rows and adjust the display according to your needs.

**Find** Locate a row containing a specified value and position the cursor to that row.

**Exclude**
Temporarily remove specified rows from the display.

**Only** Display only rows that match defined criteria, excluding rows that do not.

**Show** Redisplay rows excluded with the EXCLUDE or ONLY commands.

Operands for these commands allow you to specify how a search is conducted (e.g., where a search begins, the direction in which it proceeds, or the name of the column to search). Detailed information about available operands for each command is provided in the *Command Reference Manual*, Primary Commands.

### Working with Multiple Tables

In a multiple table display, the FIND, EXCLUDE, and ONLY commands operate on the lowest-level table, by default. To specify a particular table, enter the command and do one of the following:

- Use the cursor position to indicate the desired table.
- Use the IN operand with *creatorid.tablename.columnname*, or use the table identifier (such as T1 or T2) in place of the *creatorid.tablename.*, as indicated:

  EXCLUDE CH006 IN FOPDEMO.DETAILS.ITEM_ID

  or

  EXCLUDE CH006 IN T2.ITEM_ID

In a multiple table display, the SHOW command operates on all currently displayed tables.

### Find Specific Data

Use the FIND command to locate a row containing a specified value and position the cursor on the row and column where the value is located. A row that is not currently visible is scrolled to the beginning of the display in the window of the searched table. In a multiple table display, any lower-level tables are automatically scrolled to related rows, if any. To locate the next occurrence of the search value, use the RFIND command (usually assigned to PF5).

Several operands are available for the FIND command that allow you to specify where a search begins, the direction in which it proceeds, and whether it includes excluded rows. The Optim FIND command is similar to the ISPF FIND command. The Optim FIND command, however, is extended to support DB2 data, such as null values and floating point data types.

For example, to locate the first occurrence of the ITEM_ID DR017 in DETAILS, enter:

```
FIND FIRST DR017 IN DETAILS.ITEM_ID
or
FIND FIRST DR017 IN T2.ITEM_ID
```

As indicated in the following figure:

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===> FIND FIRST DR017 IN DETAILS.ITEM_ID             Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 7 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___       211   00284  1997-02-24  12.12.51       48.52          SC012

Cmd F == Table: FOPDEMO.DETAILS(T2) ================ STACKED = 2 OF 12 =========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- -----------------
___       211   CH006          4            14.00

Cmd F == Table: FOPDEMO.ITEMS(T3) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== -------------- ------ ----------
*** ********************************** TOP ************************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ******************************** BOTTOM ***********************************
```

*Figure 153. FIND Command*

The results of the FIND are:

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 7 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
___       211   00284  1997-02-24  12.12.51       48.52          SC012

Cmd F == Table: FOPDEMO.DETAILS(T2) ================ STACKED = 8 OF 12 =========
      ORDER_ID ITEM_ID ITEM_QUANTITY DETAIL_UNIT_PRICE
      -------- ------- ------------- -----------------
___       211   DR017         14            22.00

Cmd F == Table: FOPDEMO.ITEMS(T3) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== -------------- ------ ----------
*** ********************************** TOP ************************************
___    DR017  Chariots of Fire     Drama           PG       22.00
*** ******************************** BOTTOM ***********************************
```

*Figure 154. FIND Results*

The FIRST operand specifies that the search is to begin with the first row of data. Otherwise, the search begins at the cursor location. The IN operand specifies the search is limited to the named column. In this example, the search is limited to the ITEM_ID column. If a column name operand is omitted, all columns with a data type appropriate to the specified value are searched. (See page "Display Attributes" on page 358 for a description of attributes and data types.) If the specified column name is in more than one table (for example, ITEM_ID, in the preceding figure) and is not prefixed with the table name, the lowest-level table containing the specified column is searched.

## Case-sensitive Searching

For a case-sensitive search, qualify the search string with "C" and enclose it in quotes, as in C'DR017'. If the value includes blanks or apostrophes, you must use double quotes, as in "DR' 017". If the value includes quotation marks, use single quotes, as in 'DR"017'.

### FIND ALL

The FIND ALL command is useful in combination with EXCLUDE ALL to display all occurrences of the specified character string. (See "EXCLUDE Command.") For example, to display all rows containing "Drama," specify:

```
EXCLUDE ALL
FIND ALL DRAMA
```

## Manage Data Display

When a display includes many rows, you can use the EXCLUDE and ONLY commands to temporarily remove from the display any rows that you select or that meet specified criteria, allowing you to concentrate on the remaining rows. The SHOW command allows you to redisplay temporarily removed rows.

## EXCLUDE Command

When working with a large number of rows, you can use the Exclude command to temporarily remove rows from the display. The EXCLUDE primary command and the Exclude line command (X) allow you to exclude rows. Excluded rows are retained in the table, but are not displayed.

### EXCLUDE Primary Command

You can use the EXCLUDE primary command to exclude rows based on specific search criteria. You must specify a search value when using the EXCLUDE primary command. Several operands for the EXCLUDE command allow you to specify the direction in which the search proceeds, the name of the column to search, and whether to include all rows. For example, to exclude the ITEMS table rows for items with a PG rating, enter:

```
EXCLUDE ALL PG IN ITEMS.RATING
```

or specify the appropriate table by its table identified, such as T1:

```
EXCLUDE ALL PG IN T1.RATING
```

The EXCLUDE ALL command with no other parameters excludes all rows in a table.

### Exclude Line Command

The Exclude line command allows you to select individual rows to exclude from the display.

- To exclude a single row, type X in **Cmd** for the row.
- To exclude a block of rows, type XX in **Cmd** for the first and last rows you want to remove from the display.

In the following example, the block form of the Exclude line command is used to identify the first and last rows of a block of rows to be excluded.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___       205   00192  1997-02-24  12.12.51       48.52          NE012
___       206   00093  1997-02-24  12.12.51       48.52          SW012
___       207   00067  1997-02-24  12.12.51       48.52          WE012
___       208   03189  1997-02-24  12.12.51       48.52          NW012
___       209   00143  1997-02-24  12.12.51       48.52          SW012
___       210   00239  1997-02-24  12.12.51       48.52          NW012
XX_       211   00284  1997-02-24  12.12.51       48.52          SC012
___       212   00327  1997-02-24  12.12.51       48.52          SC012
___       213   00371  1997-02-24  12.12.51       48.52          NE012
XX_       214   00415  1997-02-24  12.12.51       48.52          NC012
___       215   02221  1997-02-24  12.12.51       48.52          SE012
___       216   00019  1997-02-24  12.12.51       48.52          SC012
___       217   00110  1997-02-24  12.12.51       48.52          SE012
___       288   00131  1997-02-24  12.12.51       48.52          SW012
```

*Figure 155. Exclude Rows*

The excluded rows are retained in the table, but are replaced with a marker indicating the location and number of excluded rows.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 16 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___       205   00192  1997-02-24  12.12.51       48.52          NE012
___       206   00093  1997-02-24  12.12.51       48.52          SW012
___       207   00067  1997-02-24  12.12.51       48.52          WE012
___       208   03189  1997-02-24  12.12.51       48.52          NW012
___       209   00143  1997-02-24  12.12.51       48.52          SW012
___       210   00239  1997-02-24  12.12.51       48.52          NW012
___    ------------------------------------------------- 4 LINE(S) EXCLUDED
___       215   02221  1997-02-24  12.12.51       48.52          SE012
___       216   00019  1997-02-24  12.12.51       48.52          SC012
___       217   00110  1997-02-24  12.12.51       48.52          SE012
___       288   00131  1997-02-24  12.12.51       48.52          SW012
___       333   01210  1997-02-24  12.12.51       48.52          SE012
___       417   00448  1997-02-24  12.12.51       48.52          SE012
```

*Figure 156. Excluded Rows Marker*

The excluded rows are no longer included in the row count; however, the excluded lines marker counts as a row. In this example, four consecutive rows are excluded and replaced with a marker, so the row count decreases by three.

## ONLY Command

Use the ONLY command to limit the display to rows that match criteria you specify, excluding all rows that do not. The excluded rows are replaced with a marker. For example, to display only the rows that contain the value NE012, specify:

```
ONLY NE012
```

To limit the selection to rows with the value in the ORDER_SALESMAN column, specify:

```
ONLY NE012 IN ORDER_SALESMAN
```

The ONLY primary command provides the same results obtained by executing EXCLUDE ALL followed by FIND ALL. The previous ONLY command is equivalent to an EXCLUDE ALL command, followed by:

```
FIND ALL NE012 IN ORDER_SALESMAN
```

### SHOW Command

Use the SHOW primary command to redisplay all rows previously hidden with the EXCLUDE or ONLY commands. When multiple tables are displayed, the SHOW command affects all currently displayed tables.

In addition, you can use the following Show line commands to redisplay specific rows.

**F, F***n* Redisplay the first row in a block of excluded rows. If *n* is specified, the first *n* rows in a block of excluded rows are redisplayed.

**L, L***n* Redisplay the last row in a block of excluded rows. If *n* is specified, the last *n* rows in a block of excluded rows are redisplayed.

**S, S*n*, SS**
Redisplay one row in a block of excluded rows. If *n* is specified, *n* rows in a block of excluded rows are redisplayed. The block form of the line command, SS, redisplays all excluded rows within the specified block of rows.

## Sort Criteria

Rows in a table are displayed in the order dictated by the Access Definition, or in an undetermined order if no sort criteria was specified in the Access Definition. (In a Compare File browse session, rows are displayed in ascending order by the match key.)

Refer to “Manage Data Displays” on page 55 for information about defining column specifications for a table.

When browsing or selecting data, you can use the SORT command to arrange rows in the display according to values in any column, noting that limits on the DB2 ORDER BY clause apply. You can apply sort criteria to a table indicated by cursor position or by specifying a table name or identifier (e.g., SORT ORDERS or SORT T2). If you do not specify a table, the sort criteria are applied to the lowest-level table. The SORT command displays the Specify Sort Criteria panel, as shown in Figure 157 on page 354.

The Specify Sort Criteria panel lists all columns in the table, prompts for sort criteria, and displays any previously specified sort criteria. You can use **Level** to specify a numeric value, 1 through 64, indicating the sort priority of a column. The value for each column must be unique and consecutive starting with 1, which is the highest priority. You can use **Asc/Desc** to specify the sort order as ascending or descending. By default, rows are sorted in ascending order.

For example, to sort rows from the ORDERS table by customer ID values, specify 1 for **Level** and A for **Asc/Desc** for the CUST_ID column, as shown in the following figure.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      ------          -                                          -------
*** ******** +------Specify Sort Criteria for FOPDEMO.ORDERS------+ ***********
___        20 |                                                    | 12
___        20 | Sort Level is a Consecutive Value Between 1 and 64 | 12
___        20 | Sort Direction is A - Ascending or D - Descending  | 12
___        20 |                                                    | 12
___        20 |                       --Sort Criteria--      1 OF 8 | 12
___        21 |      Column Name      Level  Asc/Desc              | 12
___        21 |   ------------------  -----  --------              | 12
___        21 |   *************** TOP ***************              | 12
___        21 |   ORDER_ID              _                         | 12
___        21 |   CUST_ID              1_       A                 | 12
___        21 |   ORDER_DATE            __       _                 | 12
___        21 |   ORDER_TIME            __       _                 | 12
___        21 |   FREIGHT_CHARGES       __       _                 | 12
___        28 |   ORDER_SALESMAN        __       _                 | 12
___        28 |   ORDER_POSTED_DATE     __       _                 | 12
___        28 |   ORDER_SHIP_DATE       __       _                 | 12
___        28 +-----------------------------------------------------+ 12
___
```

*Figure 157. Specifying Sort Criteria*

Once you specify the sort criteria, press ENTER or use END. Based on the sort criteria, the ORDERS rows are sorted in ascending order by customer ID, as shown in the following figure.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ======================== 1 OF 19 === MORE>>
      ORDER_ID CUST_ID ORDER_DATE ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      -------- ------- ---------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___         216   00019  1997-02-24  12.12.51       48.52          SC012
___         207   00067  1997-02-24  12.12.51       48.52          WE012
___         206   00093  1997-02-24  12.12.51       48.52          SW012
___         217   00110  1997-02-24  12.12.51       48.52          SE012
___         288   00131  1997-02-24  12.12.51       48.52          SW012
___         209   00143  1997-02-24  12.12.51       48.52          SW012
___         205   00192  1997-02-24  12.12.51       48.52          NE012
___         210   00239  1997-02-24  12.12.51       48.52          NW012
___         211   00284  1997-02-24  12.12.51       48.52          SC012
___        3398   00289  1997-02-24  12.12.51       48.52          NW012
___         212   00327  1997-02-24  12.12.51       48.52          SC012
___         213   00371  1997-02-24  12.12.51       48.52          NE012
___         214   00415  1997-02-24  12.12.51       48.52          NC012
___         727   00420  1997-02-24  12.12.51       48.52          NE012
___         417   00448  1997-02-24  12.12.51       48.52          SE012
___         333   01210  1997-02-24  12.12.51       48.52          SE012
___         215   02221  1997-02-24  12.12.51       48.52          SE012
___
```

*Figure 158. Sort Criteria Results*

For additional information on the SORT (or SORT CRITERIA) command, see the SORT command in the *Command Reference Manual*.

## Maximum Fetch Limit

When the total number of rows that satisfy the criteria for a table exceeds the maximum fetch limit specified on the Editor and Display Options panel, a message is displayed indicating that all possible rows have not been retrieved.

### COUNT Command

Use the COUNT command to determine the number of rows in a table that satisfy the criteria. This number is displayed on the panel. You can then change the selection criteria to retrieve fewer rows or use the MAX ROWS command to increase the number of rows fetched.

**Note:** The COUNT command is not available when browsing a Compare File.

### MAX ROWS Command

To view rows in excess of the maximum fetch limit, use the MAX ROWS command to increase the maximum number of retrieved rows. For example, if the maximum fetch limit is 200 rows, and the COUNT command shows that there are 300 rows to view, specify "MAX ROWS 300" and press ENTER to display the 300 rows.

**Note:** The MAX ROWS command is only available in a Point-and-Shoot session.

### Increase Maximum Fetch Limit

The MAX ROWS command increases the maximum fetch limit temporarily, until you terminate the Point-and-Shoot display. To increase the default maximum number of fetch rows, use the OPTIONS EDITOR command to display the Editor and Display Options panel. On this panel, you can change the specification for **Maximum Fetch Rows**.

## Display Hexadecimal Data

Under certain circumstances, you may need to display the hexadecimal (or hex) value of data.

For example, if a Warning indicates that the data contains nondisplayable characters, a hexadecimal display may be useful (see Figure 159 on page 356). You can use the HEX primary command to display each line of data in standard text format along with its hexadecimal format.

The HEX primary command displays each row of data on three lines. One line is in the standard text representation of the data and two are used for the hexadecimal representation. The hexadecimal representation is shown as two digits directly under each EBCDIC character. The hexadecimal display remains in effect until you use the HEX command again to return the display to character representation.

In the following figure, note that the hexadecimal representation is displayed only for columns with character data type (see "Display Attributes" on page 358 for a description of attributes and data types). The Warning in the figure refers to the character with a hexadecimal value of BB. This character is found after the word "Airplane" in the ITEM_DESCRIPTION column.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                  Scroll ===> PAGE
CAUTION - DATA CONTAINS INVALID (NON-DISPLAY) CHARACTERS
Cmd F == Table: FOPDEMO.ITEMS(T1) ========================== 1 OF 33 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== ------------- ------ ----------
*** ********************************* TOP ************************************
___    AD005  Conan the Barbarian  Adventure       R         15.00
       CCFFF  C99894A884C89889889  C8A89AA98       D444
       14005  3651503850219219915  145553495       9000
___    AD008  Dirty Harry          Adventure       R         25.00
       CCFFF  C89AA4C899A          C8A89AA98       D444
       14008  49938081998          145553495       9000
___    CH006  Willie Wonka & the C Children        G         14.00
       CCFFF  E899884E9998454A884C C8898989        C444
       38006  69339506652100038503 38934955        7000
___    CH007  Fantasia             Children        G         34.00
       CCFFF  C89A8A88             C8898989        C444
       38007  61531291             38934955        7000
___    CM009  Airplane.            Comedy          PG        14.00
       CDFFF  C8999898B            C9988A          DC44
       34009  19973155B            364548          7700
```

*Figure 159. Hexadecimal Values Displayed*

The same primary and line commands that are available in standard display are available in the hexadecimal display. Some commands, such as the FIND command, have operands specifically for a hexadecimal display.

### FIND HEX Command

To locate a specific hexadecimal value anywhere in the data, use the FIND command with the HEX operand. HEX is useful when you want to locate an invalid (non-displaying) character value. For example, to locate the first occurrence of the hex value BB, enter:

```
FIND HEX BB
```

To locate the next occurrence of the search value, use the RFIND command (usually assigned to PF5).

### Trailing Blanks and Nulls

Trailing blanks in CHAR columns are stored in the database and are displayed in hex mode as the hexadecimal value 40. Trailing blanks in VARCHAR columns are not stored in the database and are not displayed.

## Wide Data Displays

Data is sometimes too wide for display on the Browse or Point-and-Shoot panel. A table column may be wider than the maximum display width or a table may include more columns than can be displayed at once. Several features and techniques help you browse or select wide data easily.

### Sidelabels Format

When browsing tables containing multiple or wide columns, you may find it useful to use the sidelabels format instead of the standard columnar format. The sidelabels format displays a single row from a single table at one time. The column headings are displayed vertically first on the panel, and the data is displayed next to the headings. Generally, the sidelabels format displays more columns per panel and more data per row than the columnar format.

To view a specific table in sidelabels format, you can use the sidelabels primary or line command. Also, you can specify the sidelabels display format on the Editor and Display Options panel. Once specified, either by command or from the options panel, sidelabels format remains in effect until you change it.

The following figure shows a row from the ORDERS table in sidelabels format.

```
-------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

== Table: FOPDEMO.ORDERS(T1) ================================ROW    1 OF 79 ===
== LineCmd ==> ___         Row Status:                COLUMN 1 OF 8

ORDER_ID          :     10
CUST_ID           : 00049
ORDER_DATE        : 1998-01-26
ORDER_TIME        : 14.22.31
FREIGHT_CHARGES   :   9.22
ORDER_SALESMAN    : WE005
ORDER_POSTED_DATE : 1998-01-27-16.59.00.000000
ORDER_SHIP_DATE   : 98/01/29
********************************** BOTTOM ***********************************
```

*Figure 160. Sidelabels Format*

## Sidelabels Primary Command

You can view a specific table in sidelabels format by typing the SIDELABELS primary command and either specifying a table name operand (e.g., SIDELABELS ITEMS or SID T3) or positioning the cursor on the table. If the cursor is positioned on a row, that row is displayed in sidelabels format; otherwise, the first row in the table is displayed. If you do not specify a table name, the first row from the lowest-level table is displayed.

**Note:** When in sidelabels format, you can use the SIDELABELS command with a table name operand to display a different active table. During sidelabels display, joins between tables are retained and redisplayed when you return to columnar format.

To return to the columnar display, enter the SIDELABELS primary command.

## Sidelabels Line Command

To switch from columnar format to sidelabels format using the line command, enter the SID line command in **Cmd** for the row in the table. The selected row is then displayed in sidelabels format. To return to the columnar display, enter the SID line command in **LineCmd**.

## Screen Elements

The screen elements in the columnar and sidelabels formats are the same; only the positioning is different. However, only one data row can be displayed at a time in sidelabels format. In addition to the elements included in columnar format, sidelabels format shows the position of the first displayed column and the total number of columns in the table. (The screen elements are described in “Display Basics” on page 332.)

## Available Commands

Except for the INDENT, JOIN, UNJOIN, START, and SWITCH commands, the primary and line commands that are available in columnar format are also available in sidelabels format. However, since only a single row from a single table is displayed in sidelabels format, certain commands are better suited to columnar format.

### Scroll - Sidelabels

In sidelabels format, you can scroll using DOWN, UP, BOTTOM, and TOP to display another row in the table. You can also use LEFT and RIGHT to perform a logical horizontal scroll to view additional columns in the same row. In sidelabels format, a horizontal row appears as "up" and "down." See "Scroll - Wide Tables" on page 363 for further discussion of horizontal scrolling in sidelabels format.

## Wide Columns

Data is sometimes too wide to be displayed on the panel. When the width of a column exceeds the user-specified maximum column display width, the data in that column is truncated. In columnar format, this truncation is indicated by equal signs (=) after the column heading; in sidelabels format, an equal sign is displayed next to the column heading.

In the following figure, the data column ITEM_DESCRIPTION is wider than the maximum column display width. Note the equal signs under the ITEM_DESCRIPTION heading.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ITEMS(T1) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING UNIT_PRICE
      ------- ==================== -------------- ------ ----------
*** ********************************* TOP ***********************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ******************************** BOTTOM *********************************
```

*Figure 161. Wide Column*

The maximum number of characters that can be displayed per column is determined by the value specified as the **Columnar Max Display Width** and the **Sidelabel Max Display Width** on the Editor and Display Options panel. If the display of data from a column is truncated because of the maximum display width, several techniques make it possible for you to view the complete data. You can:

- Use the OPTIONS EDITOR primary command to display the Editor and Display Options panel, where you can increase the maximum column display width.
- Use the EXPAND primary command to invoke a scrollable pop-up window of all data in the column.
- If in columnar format, use the SIDELABELS primary command or the SID line command to display a row of data in a vertical format that allows a maximum display width (of 50 to 32,767 characters per column) greater than the columnar format.

### Display Attributes

It may be helpful to first review information about the displayed columns. If the **Display Column Attributes** prompt on the Editor and Display Options panel is Y, column attributes are shown automatically. If attributes are not shown, use the ATTRIBUTES primary command to display information about data type, length, and nullability for each column in each table in the Browse or Point-and-Shoot display.

Attributes can be displayed in any display mode (that is, columnar, sidelabels, or hexadecimal mode). In columnar format, attributes are displayed after the column heading. In sidelabels format, attributes are displayed next to the column name, which is truncated if insufficient space is available to display both the attributes and the full column name.

In the following figure, attributes for each column are displayed directly after the column heading.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ITEMS(T1) ========================== 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING  UNIT_PRICE
      -CH(5)- ======VCH(72)======= ---VCH(14)---- -CH(4)- -DEC(5,2)-
*** ********************************* TOP ***********************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ******************************** BOTTOM **********************************
```

*Figure 162. Column Attributes*

The attribute information is in the form *type*(*n,n*):N, where *type* is the data type and *n,n* is the width of the column. An N indicates the column is nullable. To conserve space on the screen, certain data type indicators are abbreviated when attributes are displayed. The data type indicators and the DB2 data types they represent include the following:

**CH** CHAR/
CHAR FOR MIXED DATA

**VCH** VARCHAR

**LVR** LONG VARCHAR

**DEC** DECIMAL

**INT** INTEGER

**SMALLINT**
SMALLINT

**DATE** DATE

**TIME** TIME

**TIMESTAMP**
TIMESTAMP

**SNGL FLOAT**
SINGLE FLOAT

**DBL FLOAT**
DOUBLE FLOAT

**GR** GRAPHIC

**VGR** VARGRAPHIC/
LONG VARGRAPHIC

**CLOB** CLOB FOR MIXED DATA

**DBCLOB**
DBCLOB

**BIN** BINARY

**VARBIN**
VARBINARY

**BIGINT**
BIGINT

**DECFLOAT**
DECFLOAT(16) or DECFLOAT(34), depending on precision

**Note:** See "Browsing with Unsupported Data Types" on page 364 for information on unsupported data types.

In Figure 162 on page 359, the attributes for the ITEM_DESCRIPTION column indicate it is 72 characters wide. Thus, you cannot display the entire column in columnar format by increasing the **Columnar Max Display Width** on the Editor and Display Options panel, which has a maximum limit of 70. To remove the column attributes from the display, use the ATTRIBUTES OFF command. You can also use the ATTRIBUTES command with no operand to toggle between displaying and not displaying column attributes.

## Maximum Column Display Width

The maximum column display width is specified by the **Columnar Max Display Width** and the **Sidelabel Max Display Width** options on the Editor and Display Options panel. You can use the OPTIONS EDITOR command to display this panel. In columnar format, you can specify the maximum column display width in the range of 2 to 70 characters. In sidelabels format, you can specify the maximum column display width in the range of 50 to 32,767 characters. (See "Editor and Display Options" on page 382 for additional information about these options.)

For VARCHAR columns, the entire width of the column, padded with trailing blanks, is displayed regardless of the actual length of the data (see Figure 165 on page 362). Therefore, when a table has several wide columns that do not contain correspondingly wide data, you may want to decrease the maximum display width so that more columns from the table are displayed per screen.

## Expand Column

When the maximum display width does not provide sufficient display space for a specific column or when changing the maximum display width is undesirable, the EXPAND primary command can be used to display all data in that column. You can use this command on columns with the following data types, regardless of whether the column width exceeds the maximum display width: CHAR, VARCHAR, LONG VARCHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, BINARY, and VARBINARY. You can use this command on columns with data types of TIMESTAMP and DECIMAL only if the column width exceeds the maximum display width.

When you enter the EXPAND command, you can either position the cursor on the desired table, regardless of level, or you can specify the column name. In a multiple table display, the EXPAND command operates on the lowest-level table by default. You can indicate the table in which the column you want to expand resides by including its table name or identifier in the command. For example, to expand the ITEM_DESCRIPTION column in the ITEMS table, type EXPAND ITEMS.ITEM_DESCRIPTION or EXPAND T1.ITEM_DESCRIPTION. You can also assign EXPAND to a program function key.

```
---------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===> EXPAND                                          Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ITEMS(T1) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING  UNIT_PRICE
      -CH(5)- ======VCH(72)======= ---VCH(14)---- -CH(4)- -DEC(5,2)-
*** ********************************** TOP ***********************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ********************************* BOTTOM **********************************
```

*Figure 163. EXPAND Command*

If the EXPAND command is entered on the command line and the cursor is positioned on the ITEM_DESCRIPTION column of the item CH006, the following pop-up window is displayed.

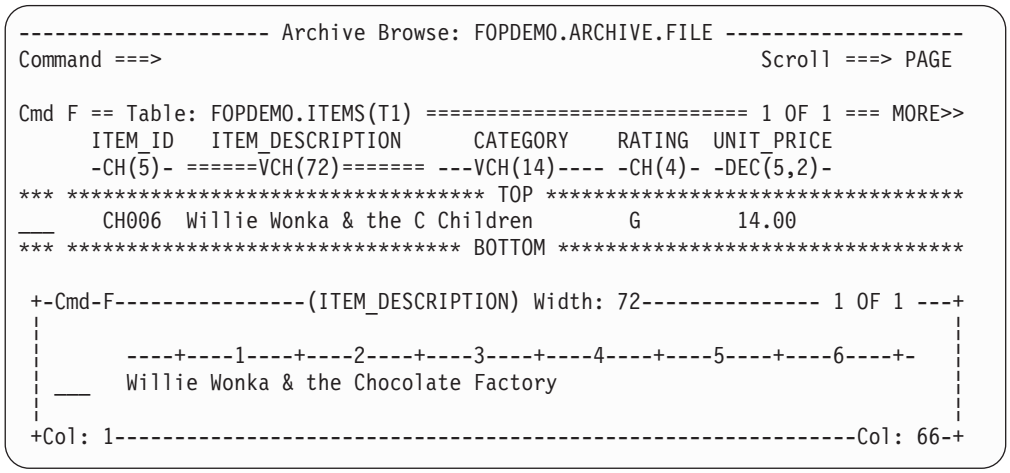



*Figure 164. Expanded Column Display*

The column name and the actual width are shown in the beginning border, as well as the position of the row and the number of rows in the Browse or Point-and-Shoot display.

In sidelabels format, positioning the cursor in a wrapped segment when executing the EXPAND command designates both the column to expand and the segment with which the expanded display begins.

### Scroll - Expanded Column Display

The currently displayed starting and ending column positions are indicated in the end border of the window. When the column width exceeds 66 characters, the expanded data can be scrolled horizontally using the LEFT and RIGHT commands.

The following line commands can also be used to scroll horizontally in an expanded column display.

**<*nn* or -***nn***
: Scroll LEFT *nn* spaces.

**>*nn* or +***nn***
: Scroll RIGHT *nn* spaces.

Use the UP and DOWN commands to scroll the data in the expanded window by row. The row counter displayed in the beginning border of the window indicates the row that is currently displayed.

When you have finished viewing the expanded data, use END to terminate the expanded column display.

## Wide Columns in Sidelabels Format

When a table contains many columns or wide columns, you may find it useful to use the sidelabels format instead of the standard columnar format. The sidelabels format displays more data per row, allowing the maximum number of characters per column to range from 50 to 32,767 characters.

When the maximum sidelabels display width permits, wide columns are automatically wrapped. The data is divided into 50-character segments and presented on multiple lines. If a column is wider than the maximum display width, it is displayed with the truncated data indicator, an equal sign (=), next to the column heading. As in columnar format, data in a truncated column can be displayed by using the EXPAND command or by increasing the maximum display width. See “Expand Column” on page 360 for details on using the EXPAND command.

In the following figure, the maximum display width is 254 characters and data in the ORDER_SHIP_INSTR column is wrapped.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

== Table: FOPDEMO.SHIP_INSTR(T1) ============================ROW   13 OF 120 ==
== LineCmd ==> ___          Row Status:                COLUMN 1 OF 4

SHIP_ID          :    106
SHIP_INSTR_ID    :  10013
ORDER_SHIP_INSTR : Federal Express.  This is the third time this orde
      (51 - 100) : r has been sent. Customer claims that first two or
     (101 - 150) : ders never arrived at store.  Make sure customer s
     (151 - 200) : igns for package.
     (201 - 250) :
     (251 - 254) :
SHIP_UPDATED     : 1998-03-09-11.27.000000
******************************* BOTTOM ***************************************
```

*Figure 165. Column Wrapping*

The column data begins next to the column name and continues, in 50-character segments, on the next five lines. The beginning and ending character positions of each segment are displayed after the column heading. For more information about sidelabels format, see "Sidelabels Format" on page 356.

## Wide Tables

Many database tables have more columns than can be displayed at once in columnar format. In addition, columns may exceed the user-specified maximum width for columnar or sidelabels displays. For example, in the following figure, the table FOPDEMO.ITEMS has more columns than can be displayed on the screen at one time, as indicated by the MORE>> designation next to the row count. Also, the column ITEM_DESCRIPTION is wider than both the user-specified maximum column display width and the 70 character maximum limit for the columnar format parameter. (Note the length attribute after the ITEM_DESCRIPTION heading.)

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ITEMS(T1) ============================ 1 OF 1 === MORE>>
      ITEM_ID   ITEM_DESCRIPTION      CATEGORY    RATING  UNIT_PRICE
      -CH(5)- ======VCH(72)======= ---VCH(14)---- -CH(4)- -DEC(5,2)-
*** ********************************* TOP ************************************
___    CH006  Willie Wonka & the C Children        G        14.00
*** ******************************** BOTTOM ***********************************
```

*Figure 166. Wide Table*

Browsing may be difficult when you can view the data in no more than a few columns at a time. If the display of data in a table is truncated because of the screen width, several techniques and features make it possible for you to view the complete data for each row or to juxtapose columns of primary interest. You can:

- Use LEFT and RIGHT to scroll a columnar or sidelabels display to view additional columns.
- Use the LOCK primary command to retain one or more columns in a columnar or sidelabels display while scrolling the remaining columns. (The ANCHOR command is a synonym for the LOCK command.)
- Use the COLUMNS primary command to arrange columns in a columnar or sidelabels display so that the columns of interest are displayed together.

- If browsing data in columnar format, use the SIDELABELS primary command or the SID line command to display a row of data in a vertical format that allows a greater maximum display width (from 50 to 32,767 characters per column) and provides room for more columns. See "Sidelabels Format" on page 356 for additional information.

## Scroll - Wide Tables

In columnar format, a column is not displayed when the display width of the column exceeds the available space for display on the screen. For example, if the display width for a column is 20 characters, but only 10 spaces remain on the screen, the column is not displayed until you scroll horizontally. When all columns do not fit on the screen, you can scroll horizontally through the columns using the LEFT and RIGHT commands (usually assigned to PF10 and PF11).

In sidelabels format, the LEFT and RIGHT commands perform a logical horizontal scroll as in columnar format; however, this appears as vertical. For example, if you use the LEFT command based on cursor location, the column containing the cursor is scrolled to the end of the display and all columns next to it are displayed vertically after this column.

## Lock Columns

Use the LOCK, or ANCHOR, command to position one or more columns at the beginning of the display. The locked columns are retained on the screen when you scroll the remaining columns.

For example, assume the combined width of the column data for the ORDERS table exceeds the width of the screen in columnar format. Thus, the display must be scrolled horizontally to view all columns. By locking the ORDER_DATE column, ORDER_DATE is repositioned horizontally first on the screen and remains in that position on the screen when scrolling to view the remaining columns. In sidelabels format, locked columns are positioned at the beginning of the display.

In the following figure, assume the LOCK command has been entered to retain the ORDER_DATE column on the display.

```
--------------------- Archive Browse: FOPDEMO.ARCHIVE.FILE --------------------
Command ===>                                                 Scroll ===> PAGE

Cmd F == Table: FOPDEMO.ORDERS(T1) ========================= 1 OF 19 === MORE>>
      ORDER_DATE ORDER_ID CUST_ID ORDER_TIME FREIGHT_CHARGES ORDER_SALESMAN
      ++++++++++ -------- ------- ---------- --------------- --------------
*** ********************************** TOP ************************************
___   1997-02-24     205   00192   12.12.51       48.52          NE012
___   1997-02-24     206   00093   12.12.51       48.52          SW012
___   1997-02-24     207   00067   12.12.51       48.52          WE012
___   1997-02-24     208   03189   12.12.51       48.52          NW012
___   1997-02-24     209   00143   12.12.51       48.52          SW012
___   1997-02-24     210   00239   12.12.51       48.52          NW012
___   1997-02-24     211   00284   12.12.51       48.52          SC012
___   1997-02-24     212   00327   12.12.51       48.52          SC012
___   1997-02-24     213   00371   12.12.51       48.52          NE012
___   1997-02-24     214   00415   12.12.51       48.52          NC012
___   1997-02-24     215   02221   12.12.51       48.52          SE012
___   1997-02-24     216   00019   12.12.51       48.52          SC012
___   1997-02-24     217   00110   12.12.51       48.52          SE012
___   1997-02-24     288   00131   12.12.51       48.52          SW012
___   1997-02-24     333   01210   12.12.51       48.52          SE012
```

*Figure 167. Repositioned Locked Column*

### Identify Locked Columns

In columnar format, a locked column is identified by a series of plus signs (+) after the column heading. The first locked column is placed in the first horizontal position with each succeeding locked column positioned next to the previously locked column.

A column that exceeds the maximum column display width is indicated by equal signs, =, after the column heading. If the truncated column is also locked, a series of dots after the column heading distinguish it from locked columns that are not truncated. For more information on truncated columns, see "Wide Columns" on page 358.

In sidelabels format, the first locked column is placed at the beginning of the display and successive locked columns after it. A locked column is marked with a plus sign (+) next to the column name.

### Lock Multiple Columns

In columnar format, any number of columns may be locked, as long as enough space remains to display at least one unlocked column. The number of positions reserved for unlocked columns is determined by the user-defined maximum display width specified on the Editor and Display Options panel. For example, if the maximum display width is 20 characters, then the horizontally last 21 character positions on the screen are reserved for unlocked columns to ensure that at least one unlocked column is displayed. More columns may be displayed if they fit, in their entirety, in the remaining area.

Use the LOCK KEY command to lock all columns that comprise the primary key.

### Unlock Columns

Use the UNLOCK command with no operands to unlock all locked columns. Use the UNLOCK command with a column name operand to unlock a single column.

## Arrange Columns

To rearrange the display order of the columns, use the COLUMNS command. The COLUMNS command displays the Describe Columns panel, discussed in "Manage Data Displays" on page 55.

On the Describe Columns panel, you can change the order of columns by using the Move (M) line command with the Before (B) and After (A) commands to designate the destination.

## Browsing with Unsupported Data Types

During a Browse session, DB2 tables that contain columns with unsupported data types are managed as follows.

- For tables containing columns with supported and unsupported data types, supported data types are displayed and unsupported data types are listed as "UNSUPPORTED."
- For tables containing columns with unsupported data types only, the Browse function is disabled and the message "UNSUPPORTED DATA TYPE" appears.

# Reports

At any time during an Extract File or Archive File browse session or a Point-and-Shoot session, you can create a report about the contents of the display.

Use the report facility to format and print or save a report that includes some or all displayed rows. To invoke the report facility, use the REPORT command. (Note that the REPORT command is not available while browsing a Compare File.) The following panel prompts for the report specifications.

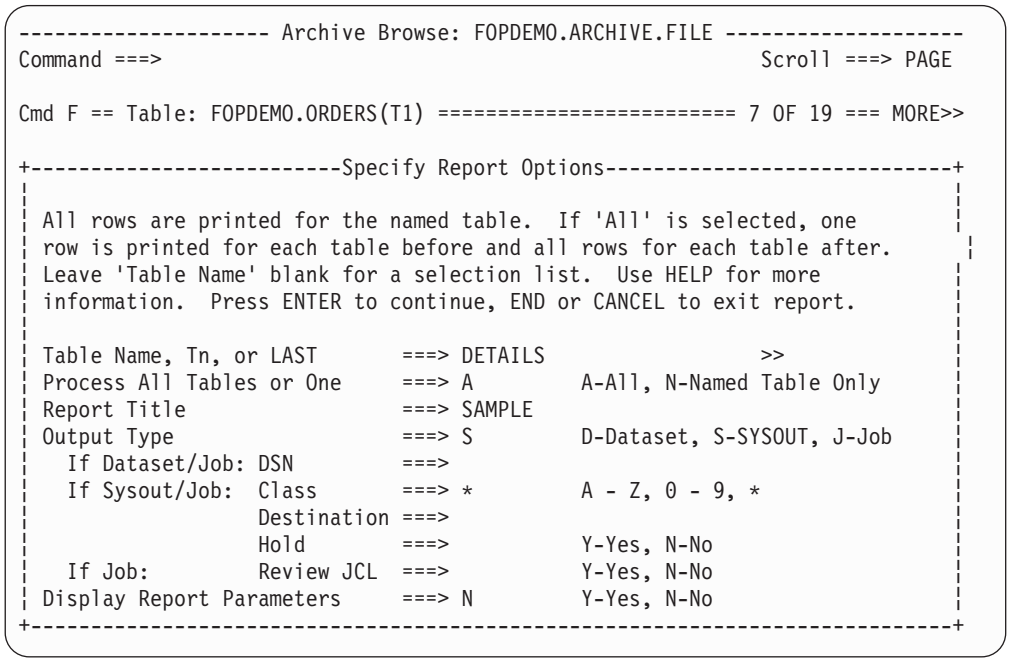



*Figure 168. Specify Report Options*

## Panel

This panel includes:

**Table Name,**
**Tn, or LAST**
The name of the table that directs the report. You can specify the name of any table in the display by explicit name, assigned T*n* or V*n*, or the word LAST (for the lowest-level table in the display). Leave **Table Name** blank or use a pattern to choose from a selection list.

This specification, combined with **Process All Tables or One**, determines the set of data in the report. For more information about the effect of this specification on the contents of the report, refer to "Report Contents" on page 366.

**Process All Tables or One**
Indicates whether one or all tables are included in the report. Specify:

**A** All tables.

**N** Only the named table.

**Report Title**
Specify a title (up to 40 characters) to be included in the report heading. Any characters, including uppercase and lowercase letters, are valid.

**Output Type**
Indicates the output type. Specify:

**D** Dataset

**S** SYSOUT

**J** Job (a batch job that produces both dataset and SYSOUT output)

**If Dataset/Job**
If the **Output Type** is D or J, specify the name of the dataset. If the dataset does not exist, Optim prompts for allocation information and allocates the file before generating the report.

**If Sysout/Job**

If the **Output Type** is S or J, specify SYSOUT parameters:

**Class** The output class for the printed output. Specify an alphabetic or numeric character (A – Z, 0 – 9) or an asterisk (*).

**Destination**

The SYSOUT destination. Specify a valid local or remote terminal, a node in the JES network, a local or remote printer or workstation, or a TSO User ID.

**Hold** Specify Y or N.

**If Job: Review JCL**

If the **Output Type** is J, specify:

**Y** Review JCL.

**N** Do not review JCL.

**Display Report Parameters**

Indicates whether you want to review the prompts for report parameters. (For more information, refer to "Report Format Parameters" on page 367.) Specify:

**Y** Review prompts.

**N** Do not review prompts.

## Report Contents

The **Table Name** and **Process All Tables or One** specifications determine the contents and layout of the report. For example, assume you are browsing four tables joined for display as follows:

```
ORDERS
  C:DETAILS
    P:ITEMS

P:CUSTOMERS (STACKED)
```

CUSTOMERS and DETAILS are joined at the same level (stacked) and ITEMS is joined to DETAILS. (For an explanation of stacked tables, refer to "Multiple Table Display" on page 341.)

1. To generate a report on the rows from a single table, enter the table name and specify "N" for **Process All Tables or One**. For example, to include only the DETAILS rows in the report, specify:

   ```
   Table Name, Tn, or LAST                ===> DETAILS
   Process All Tables or One                ===> N
   ```

   All the rows from the DETAILS table, which are related to the displayed ORDERS row (the next highest table), are included in the report. No other tables are included in this report.

2. To limit the report to a specific set of related data, specify the name of the table for which all rows on the current level are included. For reference, the related row from each higher-level table is included. The rows from lower-level tables are always included with this specification.

   For example, to generate a report on the DETAILS rows for the currently displayed ORDERS row and the related ITEMS rows for each of the DETAILS rows, specify:

   ```
   Table Name, Tn, or LAST                ===> DETAILS
   Process All Tables or One                ===> A
   ```

   The report lists the displayed row from the ORDERS table and, one at a time, each DETAILS row with the related ITEMS rows in a control break format.

3. To include the rows from all the tables joined in the current session, specify the name of the Start Table and "A" for the Process All Tables or One prompt. For example, assume the ORDERS table is the Start Table, specify:

   ```
   Table Name, Tn, or LAST                ===> ORDERS
   Process All Tables or One                ===> A
   ```

If you want to include all information for a single ORDERS row, specify DETAILS as the table name. The ORDERS row is included with all current DETAILS rows, and related ITEMS rows.

### Format

The format of rows in the report conforms to the browse or Point-and-Shoot display format. A header identifies the table, column headings are presented before the columns, and the row(s) of data are listed after the headings. You can specify the parameters for the report format.

When there are multiple lower-level tables, the report display is similar to that of the browse or Point-and-Shoot display.

- For each table that is higher-level than the lowest-level table, only one row is displayed.
- For the lowest-level table, all rows are displayed.
- When multiple rows are included in the report for a higher-level table, each row is displayed in order with the related, lower-level rows.

## Report Format Parameters

When you indicate Y to **Display Report Parameters**, prompts for the parameters are displayed. Normally, you establish these parameters once, and use them consistently for all reporting.

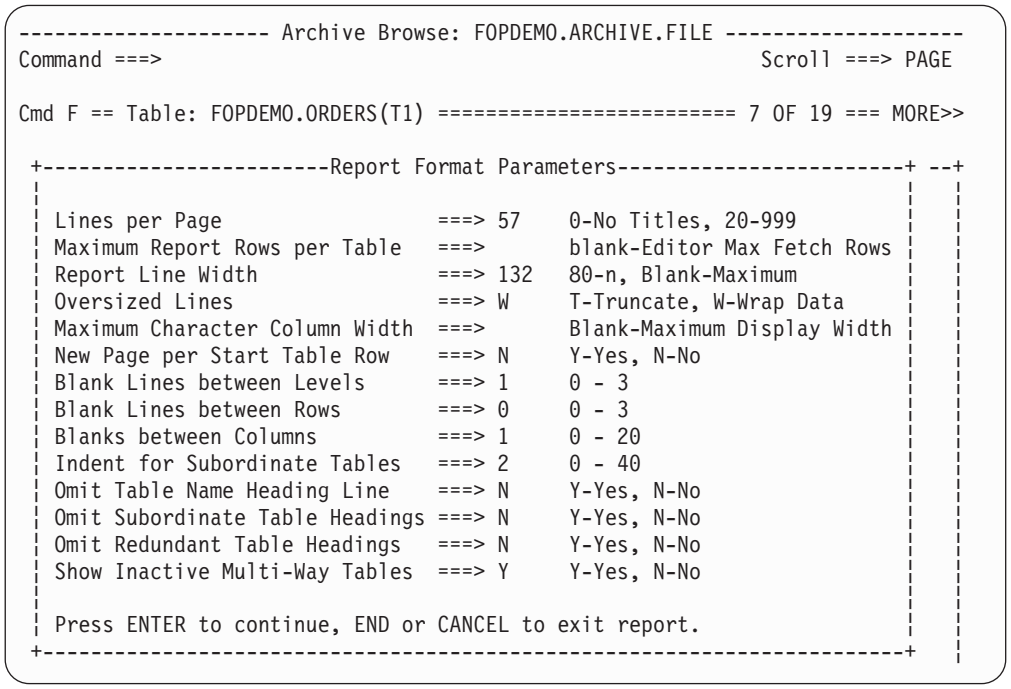



*Figure 169. Report Format Parameters*

### Panel

The following prompts are displayed on this panel.

**Lines per Page**
The maximum number of lines printed on a page. Normally, this is the number of lines that physically fit on a page, excluding lines for the vertical margins. The default is 57. Specify:

**0** Print the title line at the start of the report and do not insert page breaks. Use 0 to write to a dataset without inserting page breaks.

**20-999** Print the indicated number of lines per page. When the number of lines have been printed, Access inserts a page break, prints the title at the start of the new page, and increments the page number by 1.

**Maximum Report Rows per Table**
The maximum number of rows fetched for each table in the report. The value must be from 1 through the fetch limit specified by site management. Leave blank to use the value specified for **Maximum Fetch Rows** in the **Editor and Display Options**.

**Report Line Width**
The maximum number of characters on a line in the report. If a row exceeds this value, it is wrapped or truncated according to the **Oversized Lines** specification.

The minimum value is 80. A value less than 80 is automatically reset to 80. Use 80 if the report is directed to a data set for subsequent browsing or printing. Leave blank to set the line width equal to the longest row of any table in the report.

If the report is directed to a printer, this value is normally the line length of the printer (132 for most printers), excluding the carriage control byte in the first position of the line.

**Oversized Lines**
Indicates the handling of lines that exceed the maximum width. Specify:

**T** Data is truncated. Any columns that do not fit entirely on the line are omitted.

**W** Data is wrapped. Each row occupies as many lines as needed. Line breaks occur between columns when the column width exceeds the remaining line width.

**Maximum Character Column Width**
The maximum number of bytes that can be included in a report column.

**Note:** For GRAPHIC and DBCLOB columns, the Shift Out and Shift In characters are included in the maximum width. Therefore, a report displays the specified maximum number of GRAPHIC and DBCLOB bytes minus the number of Shift Out and Shift In characters.

Leave blank to use the value specified for **Columnar Max Display Width** on the Editor and Display Options panel.

**New Page per Start Table Row**
Indicator for a new page for each Start Table row. (If the report is for a single table, this is ignored.) Specify:

**Y** Print on a new page.

**N** Continue printing on the current page.

**Blank Lines between Levels**
Indicates the number of blank lines inserted between table levels in the report. Specify a value from 0 through 3.

**Blank Lines between Rows**
The number of blank lines inserted between rows. This parameter applies only to the tables for which all relevant rows are included; only a single line is included for other tables. Specify a value from 0 through 3.

**Blank between Columns**
Indicates the number of blanks inserted between columns. Specify a value from 0 through 20.

This value is the minimum number of blanks between any two columns. The report may display more blanks between some columns, depending on the contents of the column and the justification of that data in relation to the column heading.

**Indent for Subordinate Tables**
The number of positions to indent data from a subordinate table at each control break to distinguish it from the immediately preceding table. Use this indentation to present a clear visual representation of the display levels.

Specify a value from 0 through 40. This value is subtracted from the **Report Line Width** to determine the actual number of characters in the indented table that can fit on a line.

**Omit Table Name Heading Line**
Indicator for including the line containing the table name and level number for each level in the report. Specify:

**Y** Omit heading.

**N** Include heading.

**Omit Subordinate Table Headings**
Indicator for including header lines for all tables other than the Start Table in the report. If the subordinate table names and column titles are not necessary, use this option to eliminate them. Specify:

**Y** Omit heading.

**N** Include heading.

**Omit Redundant Table Headings**
Indicator for including table headings more than once on a page. That is, if a table is included on a page more than once, the headings are included with the first occurrence only. Specify:

**Y** Omit redundant table headings.

**N** Include table headings for all occurrences of a table on a page.

**Show Inactive Multi-Way Tables**
Indicator for including all joined tables in a multi-way join. Although all tables cannot be currently active, all must have been displayed during the session. Specify:

**Y** Include all tables in a multi-way join.

**N** Include only the currently displayed table.

## Additional Notes

The following notes apply to reports:

- Excluded rows are included in the report. When you return to the browse or Point-and-Shoot display after generating a report, any excluded rows are redisplayed.
- If you specify a table in a stack and only one table is printed, only the active table is included in the report. The hidden tables on that level are not printed, regardless of the **Show Inactive Multi-way Tables** option.
- All locked column specifications are respected; locked columns are positioned as on the browse or Point-and-Shoot display.
- The Describe Columns panel specifications for column and label handling are used to format the report lines.
- The REPORT command is not available from a ZOOMed display.

# Chapter 12. Options

Several types of options are available to manage the functions performed while using Optim.

These options can be modified by selecting Option 0 OPTIONS on the **Main Menu** or by entering the OPTIONS command from any panel with a Command prompt. The **Choose Option Type** menu is displayed, allowing you to specify User, Editor, Job Card, Compare, Archive, and Legacy options on separate panels.

```
------------------------------ Choose Option Type ----------------------------
OPTION  ===>


   1  USER           - User Options
   2  EDIT           - Editor and Display Options
   3  JOBCARD        - Job Card and Print Options
   4  COMPARE        - Compare Options
   5  ARCHIVE        - Archive Options
   6  LEGACY         - Legacy Options

   C  CUSTOM         - Custom Options
```

*Figure 170. Choose Option Type*

## Panel Options

The available options are:

**1 USER**
Display the User Options panel, which prompts for user-specific parameters that affect the current session.

**2 EDIT**
Display the Editor and Display Options panel, which prompts for parameters that affect an Access edit or browse session, a Point-and-Shoot session, or a Compare File, Extract File, or Archive File browse session.

**3 JOBCARD**
Display the Job Card and Print Options panel, which prompts for information needed to submit a batch job. (Available only if Move, Archive, or Compare is installed.)

**4 COMPARE**
Display the Compare Options panel, which prompts for parameters used for a Compare Process. (Available only if Compare is installed.)

**5 ARCHIVE**
Display the Archive Options panel, which prompts for parameters used for an Archive Process. (Available only if Archive is installed.)

**6 LEGACY**
Display the Legacy Options panel, which prompts for parameters used when creating Legacy Tables.
(Available only if *Move* or *Compare for IMS, VSAM, and Sequential Files* is installed.)

**C CUSTOM**
Display the Custom Options panel, which allows the enablement and disablement of various custom options that affect Optim behavior. (Displayed only for users with authorized passwords.)

An additional option, Option 0 SITE, is displayed for any user having an authorized password. For detailed information on Site Options, see the *Customization Guide*, Customize the Optim Site Options.

### Panel Format

Options panels are displayed in full-screen or pop-up format, depending upon the method used to access the panel: full-screen format when you select an option through the **Choose Option Type** menu, and pop-up format when you use the OPTIONS command.

**Note:** For further information, see the OPTIONS command in the *Command Reference Manual*.

### Values Profiled

Values on the Options panels are profiled.

## User Options

User options determine several user-specific parameters that govern the current Optim session. When you select Option 1 from the **Choose Option Type** menu, the User Options panel is displayed. Due to space limitations, only the first panel of User Options is shown in the following figure.

```
--------------------------------- User Options ---------------------------------
Command ===>                                                 SCROLL ===> PAGE

Caps Mode On              ===> N                Y-Yes, N-No
Confirm on Deletes        ===> N                Y-Yes, N-No
Load Utility to Use          : D                D-DB2, B-BMC
Submit Jobs with END      ===> Y                Y-Yes, N-No
Display DB2 SubSystem     ===> N                Y-Yes, N-No
Selection List Format     ===> S                S-Short, F-Full, D-Desc
Data Set Prefix           ===> USERID           PREFIX, USERID, NONE or value
User Supplied Password    ===>                  Enable Administrator Mode

Change Line Characters    ===> N                Y-Yes, N-No
Print Tables in JCL       ===> N                Y-Yes, N-No
Always Display Create     ===> U                A-Always, U-Unknown table
Use DB2 LOB Defaults      ===> N                Y-Yes, N-No
Check Create Table Chg    ===> N                Y-Yes, N-No
Display GDG Level         ===> A                A-Absolute, R-Relative
Del LOAD work DSN RC=4    ===> N                Y-Yes, N-No
Load when Zero Rows       ===> N                Y-Yes, N-No
Generate TYPRUN=HOLD      ===> N                Y-Yes, N-No
Generate Multi-Job DSN    ===> Y                Y-Yes, N-No
Display UDT Name          ===> N                Y-Yes, N-No
AD Default Creator ID     ===> S                P-Profile, S-SQLID
Review Job Card           ===> N                Y-Yes, N-No
Stage Centera File        ===> Y                Y-Yes, N-No
Storage Recall Processing ===> P                P-Prompt,A-Auto,B-Abort
Ignore Generic Rels          : N                N-Never,A-Always,S-SameCID
Max Selection List Entry  ===> 16384            1-32767
Specify Unit Parameters   ===> N                Y-Yes, N-No
Defer Auth-Chk for Batch  ===> Y                Y-Yes, N-No
Relationship Processing   ===> B                D-DB2,P-OPT,B-Both,R-Prompt
Key Lookup Limit Default  ===> 1                1-100
Allow Mismatched CCSIDs   ===> N                Y-Yes, N-No
Allow Empty Tables RC=4   ===> N                Y-Yes, N-No
No Rows Found RC=4        ===> Y                Y-Yes, N-No
Zero Rows Deleted RC=4    ===> N                Y-Yes, N-No
Delete Failures RC=4      ===> N                Y-Yes, N-No
```

*Figure 171. User Options*

## Panel

The user options apply to all Optim components, unless indicated. The user options include:

**Caps Mode On**
Case translation mode. This option applies to entries in an edit session, entries of selection criteria or SQL WHERE clauses, or entries in Column Maps. Specify:

**Y** Caps Mode is on. All entered data is translated to uppercase.

**N** Caps Mode is not on. All entered data remains as entered.

**Confirm on Deletes**
Option for confirmation prompts when deleting an Access Definition, Compare Definition, Primary Key, Relationship, Table Map, or Column Map or, for Access only, a row from a table. Specify:

**Y** Display a confirmation prompt.

**N** Do not display a confirmation prompt.

**Load Utility to Use**
For Move and Archive. Select the load utility to use for a Load Process. (Site management may establish a default load utility and prevent modification.) If modifiable, specify:

**D** Use the DB2 Load utility.

**B** Use the BMC LOADPLUS utility.

**Submit Jobs with END**
For Move, Compare, and Archive. Default method of submission for batch jobs when reviewing the JCL and Batch Utility control statements. Specify:

**Y** User can submit job using END.

**N** User cannot submit job using END, and must use the SUBMIT command to submit the job.

A message, displayed with the JCL and control statements, indicates whether to use END or the SUBMIT command to submit the job.

**Display DB2 SubSystem**
Option for displaying the name of the current DB2 subsystem on Optim panels. Specify:

**Y** Display the DB2 subsystem on every panel.

**N** Do not display the DB2 subsystem on every panel, but only on the main menus, where it can be changed.

If connected to a remote subsystem, the location of the remote subsystem is displayed instead.

**Selection List Format**
Information presented on a selection list of Optim objects (e.g., Access Definitions, Table Maps, Column Maps, Relationships, Primary Keys, Legacy Tables, etc.). Specify:

**S** Short format (i.e., the object name and modification information). (This format is used throughout the Optim documentation.)

**D** Object name and description only.

**F** Full format (i.e., object name and modification information on the first line, and the description and the security status (if other than PUBLIC) on the second line. (See "Specify Description and Security Status" on page 380 for information about security status.)

**Data Set Prefix**
The value used as the default high order qualifier for any data set name that is not enclosed in single quotes. Specify:

- PREFIX (or PRFX)

  The TSO Prefix for the current user.
- USERID (or USER)

  The TSO ID for the current user.
- NONE

  No prefix. (The data set name or pattern is accepted exactly as entered, whether or not enclosed in single quotes.)
- value

  An explicit 1- to 17-character value. This value can be a combination of PREFIX, USERID, and a literal.

**User Supplied Password**
Prompt for the Administrator Password. If the proper password is entered correctly, the current user can access the Site Options panel.

**Change Line Characters**
Characters used to define the perimeter of pop-up windows. (Site management can set the default characters, which users can override.) Specify:

**Y** Display the User Line Drawing Characters panel to specify characters. (For information, see "User Line Drawing Characters" on page 379.)

**N** Do not display the User Line Drawing Characters panel.

**Print Tables in JCL**
For Move and Archive. Option to list tables that participate in a batch process, as comments in the JCL. The list of tables is displayed when you review JCL and control statements before submitting a job, or when you browse the saved JCL and control statements. Specify:

**Y** Display the Print Tables in JCL panel, which prompts you for the processes for which tables are printed in the JCL. (For information, see "Print Tables in JCL" on page 379.)

**N** Do not print tables in the JCL (default).

**Always Display Create**
For Move and Archive. Default behavior for displaying the CREATE Object List panel. Specify:

**A** Always display the CREATE Object List panel, regardless of table status.

**U** Display the CREATE Object List panel only when a table is in Unknown status (default).

**Use DB2 LOB Defaults**
For Move and Archive. Tablespace for Large Objects (LOBs) when the Create Process generates an auxiliary table. (This option is available only if enabled by a Site Option.) Specify:

**Y** Use DB2 default LOB tablespace.

**N** User explicitly creates a tablespace for LOBs (default).

**Check Create Table Chg**
For Move and Archive. Default status displayed on the CREATE Object List panel if a destination table definition is altered before executing the Create Process. Specify:

**Y** Display the CHANGED status.

**N** Display the EXISTS status (default).

**Display GDG Level**
Display of GDG Level. Specify:

**A** Absolute (default)

**R** Relative

**Del LOAD work DSN RC=4**
For Move and Archive. Action taken if the LOAD operation return code is 4. Specify:

**Y** Delete the data set name.

**N** Do not delete the data set name (default).

**Load when Zero Rows**
For Move and Archive. Action taken when the Load Process will insert no rows. Specify:

**Y** Perform the Load Process.

**N** Do not perform the Load Process (default).

On the Specify LOAD Parameters and Execute panel, if Delete all rows in tablespace=Y, any tables in the input file with zero rows will be included in the Load Process, regardless of the Load when Zero Rows value.

**Generate TYPRUN=HOLD**
For Move and Archive. Option to include the TYPRUN=HOLD statement in JCL generated for a multi-step Load Process that requires multiple jobs. (See the following **Generate Multi-Job DSN** option.) Specify:

**Y** Include TYPRUN=HOLD in the JCL (default).

**N** Do not include TYPRUN=HOLD in the JCL.

**Generate Multi-Job DSN**
For Move and Archive. If the number of job steps in the Load Process exceeds 256, Optim automatically splits the process into multiple jobs. These jobs must be executed in the proper sequence to maintain the system-determined order. Specify:

**Y** Include the DD statement PSDFFLAG in the batch process (default). PSDFFLAG contains information used to maintain the correct processing sequence.

**N** Do not include PSDFFLAG. The user must submit the jobs manually, in the correct sequence.

**Display UDT Name**
Option for display of user-defined type (UDT) names. Specify:

**Y** Display the UDT name in addition to the base type name.

**N** Display only the base type name (default).

**AD Default Creator ID**
The default Creator ID (CID) displayed in the Access Definition Editor. Specify:

**P** Display the most recently used (i.e., profiled) Creator ID as the CID.

**S** Display the SQLID of the current user as the CID (default).

**Review Job Card**
For Move, Compare, and Archive. Option to display the Job Card and Print Options panel, to review or change the job statement or other job parameters before submitting a batch job. (See "Job Card and Print Options" on page 386.) Specify:

**Y** Display the Job Card and Print Options panel.

**N** Do not display the Job Card and Print Options panel.

**Stage Centera File**
For Archive only. Option for recalling an Archive File in its entirety or in 2 MB stages.

**Y** Recall the Archive File in stages (default).

**N** Recall the Archive File in its entirety.

**Storage Recall Processing**
For Archive only. Determine how Archive Files are managed when recalled from external storage (e.g., Centera or Tivoli® Server) through Archive processing (e.g., browse or restore). Specify:

**P** An allocation panel is displayed when a recall of the Archive File is attempted.

**A** The Archive File is automatically recalled to the original location, with no user intervention.

**B** If the original Archive File does not exist, the request is aborted.

**Note:** This option is available if the **Recall Processing** option on Storage Site Options is set to U. See Specifying Storage Options in the *Customization Guide* for more information.

**Ignore Generic Rels**
Option for ignoring generic relationships to increase performance. Specify:

**N** Use generic relationships to retrieve data from related tables.

**A** Do not use generic relationships to retrieve data.

**S** Use generic relationships to retrieve data, only if the Creator ID is the same for both the parent and child tables.

**Note:** This option is available if the **Ignore Generic Rels** option in Site Options is set to U. See Customize the Optim Site Options in the *Customization Guide* for more information.

**Max Selection List Entry**
The maximum number of items that can be selected by a user from a selection list. Specify a number from 1 to 32767.

**Specify Unit Parameters**
For Move and Archive. Option to provide default volume information that can be used when allocating Optim-generated files that span multiple volumes.

**Y** Display the File Unit Parameters panel to prompt for unit, volume number, and one to six volume serial numbers. (For information, see "File Unit Parameters" on page 381.)

**N** Do not display the File Unit Parameters panel.

**Defer Auth-Chk for Batch**
Timing of authorization checks on objects when creating JCL and Batch Utility control statements for batch Extract and Archive Processes.

**Note:** This option is available if the **Defer Auth-Check for Batch** Site Option is set to U. See Customize the Optim Site Options in the *Customization Guide* for more information.

**Y** Defer authorization checks.

**N** Do not defer authorization checks.

**Relationship Processing**
For Move and Archive. Type of relationships processed when the GET TABLES RELATED command is issued from the Select Tables/Views for AD panel.

**D** Process DB2 relationships only.

**P** Process Optim relationships only.

**B** Process both DB2 and Optim relationships.

**R** Display the Get Related Processing Options panel, which allows you to select the relationship type.

**Key Lookup Limit Default**
Default value for the maximum number of key lookups performed at one time for a table. This value is used whenever a key lookup is the access method used to scan a table and no specific key lookup limit is specified. Valid values are 1 through 100.

For more information, see "Key Lookup Limit" on page 397.

**Allow Mismatched CCSIDs**
Controls the action Optim should take if the Coded Character Set Identifier (CCSID) of a source column does not match the CCSID of a target column, or the terminal CCSID does not match that of the DB2 subsystem.

**Note:** This option is available if the **Allow Mismatched CCSIDs** Site Option is set to U. See Customize the Optim Site Options in the *Customization Guide* for more information.

**Y** Allow the Optim process to continue despite the mismatched CCSIDs if the user chooses to proceed. (The process may fail if the CCSIDs are not compatible.)

**N** Terminate the Optim process when mismatched CCSIDs are detected (default).

**Allow Empty Tables RC=4**
Indicate the action taken and return code (RC) setting when empty tables have been detected during Insert, Restore, Delete, or Load batch processing.

**Y** **Restore/Insert** - Empty tables will be processed and a warning message will be issued with RC=4. **Delete/Load** - Empty tables will be processed and processing will end with RC=4.

**N** **Restore/Insert** - Empty tables will cause termination with an error and RC=12. **Load** - Empty tables will be processed and processing will end with RC=0. **Delete** - Empty tables will be ignored and processing will end with RC=0. (Default)

**No Rows Found RC=4**

Indicate the action taken and return code (RC) setting when no rows meet the selection criteria during Restore or Subset batch processing or when rows have not been found during Delete batch processing.

**Y** The program will end with RC=4 (default).

**N** The program will end with RC=0.

**Zero Rows Deleted RC=4**

Indicate the action taken and return code (RC) setting during Delete batch processing when a table has rows in an Archive File or Extract File but no rows have been deleted. Possible reasons include

- The rows cannot be deleted because there are no matching rows in DB2 tables
- A condition existed that prevented it (for example, a violation of Referential Integrity would occur).

**Y** Processing will end with RC=4.

**N** Processing will end with RC=0 (default).

**Delete Failures RC=4**

Indicate the action taken and return code (RC) setting when some rows of a table have failed to be deleted because a condition existed that prevented it (for example, a violation of Referential Integrity would occur).

**Y** Processing will end with RC=4.

**N** Processing will end with RC=0 (default).

**Resolve DB2 Rel Change**

For batch execution of extract and archive processes, determines whether Optim rebuilds any DB2 relationship if the columns in the relationship have changed since the time when the relationship was originally defined.

**Note:** This option is available if the **Resolve DB2 Rel Change** Site Option is set to U. See Customize the Optim Site Options in the *Customization Guide* for more information.
Specify:

**Y** Yes. Rebuild a DB2 relationship if the columns have changed.

**N** No. Do not rebuild a DB2 relationship if the columns have changed. If an extract or archive process encounters a relationship with columns that have changed, it will fail with a return code of 12. This is the default.

**Register DASD Extracts**

Controls whether Optim automatically creates an entry in the Directory when an extract file is created on disk.

**Note:** This option is available if the **Register DASD Extracts** Site Option is set to U (User). See Customize the Optim Site Options in the *Customization Guide* for more information.

**Y** Yes. Automatically create an entry in the Directory when an extract file is created.

**N** No. Do not create an entry in the Directory when an extract file is created. This is the default.

**Replace Extract Dir**
Action taken if a batch extract process creates a duplicate entry in the Directory.

**Note:** This option is available if the **Replace Extract Dir** Site Option is set to U (User). See Customize the Optim Site Options in the *Customization Guide* for more information.

**Y** Yes. Proceed with the extract process, replacing the entry in the Directory. This is the default.

**N** No. Stop the extract process with an error message.

**Change Data Privacy**
Option to use a different data privacy environment variable file than the site-specified default file.

**Note:** This option is available only if the **Change Optim Data Privacy** Site Option is set to U (User). See Customize the Optim Site Options in the *Customization Guide* for more information.

**Y** Yes. User can specify a data privacy environment variable file.

**N** No. User cannot specify a data privacy environment variable file and must use the site-specified default.

## User Line Drawing Characters

If you enter Yes for the **Change Line Characters** option, the User Line Drawing Characters panel is displayed.

```
------------------------ User Line Drawing Characters -------------------------
Command ===>                                                 SCROLL ===> PAGE

Top Left Corner     ===>       Any Displayable Character
Top Right Corner    ===>       Any Displayable Character
Bottom Left Corner  ===>       Any Displayable Character
Bottom Right Corner ===>       Any Displayable Character
Horizontal Bar      ===>       Any Displayable Character
Vertical Bar        ===>       Any Displayable Character
Forward Slash       ===>       Any Displayable Character
Backward Slash      ===>       Any Displayable Character
```

*Figure 172. User Line Drawing Characters*

This panel allows you to indicate the characters used to draw borders around pop-up windows, and to replace certain line characters. You can enter any displayable character for any position or character. Site management sets the default characters. To reset any value to the site default, leave an entry blank. (Only overrides to the default characters are displayed.) When you have completed this panel, press ENTER or use END to return to the User Options panel.

## Print Tables in JCL

If you enter Yes for the **Print Tables in JCL** option, the following panel is displayed.

```
---------------------------- Print Tables in JCL -----------------------------
Command ===>                                                 SCROLL ===> PAGE

Extract Process     ===> N     Y-Yes, N-No
Archive Process     ===> N     Y-Yes, N-No
Insert Process      ===> N     Y-Yes, N-No
Restore Process     ===> N     Y-Yes, N-No
Convert Process     ===> N     Y-Yes, N-No
Delete Process      ===> N     Y-Yes, N-No
Load Process        ===> N     Y-Yes, N-No
```

*Figure 173. Print Tables in JCL*

This panel allows you to select processes for which tables in the JCL are printed. For each process, you can specify:

**Y** Print the relevant tables in the JCL.

**N** Do not print the tables in the JCL (default).

**Note:** When you return to the User Options panel, the **Print Tables in JCL** value is displayed as N, but the option will work as specified.

## Specify Description and Security Status

While the setting for **Selection List Format** governs whether the description and security status are displayed on a selection list of Optim objects, you must use the ATTRIBUTES command from the object editor or the AT line command on the selection list to modify these values.

The prompts are displayed on the Object Attributes panel. In the following figure, the attributes for an Access Definition named FOPDEMO.TEST.SAMPLE are displayed.

```
------------------------ Select Access Definitions --------------------------
Command ===>                                                 Scroll ===> PAGE

    Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 8

     +-----------------------Object Attributes----------------------+
Cmd  |                                                              |
---  | Object Name:  FOPDEMO.TEST.SAMPLE                            |
     |                                                              | ***
___  | Modify the attributes as needed.                             |
___  |                                                              |
     | Description     ===>                                         |
AT_  |                                                              |
___  | Security Status ===> PUBLIC     (PUBLIC, PRIVATE, READONLY)  |
___  |                                                              |
     | Use END command to accept any changes and return.            |
     | Use CANCEL command to ignore any changes and return.         |
     |                                                              |
     +--------------------------------------------------------------+
```

*Figure 174. Object Attributes*

### Panel

The name of the object is displayed. You can modify the following:

**Description**
A 1 to 40 character description of the object.

**Security Status**
Access privileges for the named object as:

**PUBLIC**
Any user can access and edit the object.

**PRIVATE**
Only the user that created the object can access and edit the object.

**READONLY**
Any user can access the object, but only the user who created the object can edit it.

Site management determines whether security status is available. If it is not available, **Security Status** is not displayed.

## File Unit Parameters

If you enter Yes for the Specify Unit Parameters option, the following panel is displayed.

```
---------------------------- File Unit Parameters -----------------------------
Command ===>                                                 Scroll ===> PAGE

Enter values for each file type as required.  You may specify either the
number of volumes or the volume serial(s) for each file type, but not both.
Values specified on the defaults line are used when the corresponding values
for the type of file being allocated are omitted.

 File      Unit   Nbr
 Type      Name   Vols   Volume Serial(s)
-------  -------- ---- ------------------------------------------------------
Default                >        >        >        >        >        >

Archive                >        >        >        >        >        >

Extract                >        >        >        >        >        >

Control                >        >        >        >        >        >

Load                   >        >        >        >        >        >

Maximum DASD Tracks ===>          In Range (16695 - 65535)

Press END to return and apply any changes.
Press CANCEL to return without applying any changes.
```

*Figure 175. File Unit Parameters*

The File Unit Parameters panel allows you to provide information that generated files use to allocate new data sets. You can specify information for multiple volumes as a default and individually for specific file types (e.g., Archive, Control, Extract, or Load Files).

Optim allocates the data set on the first volume with available space. If the data set does not fit on a single volume, additional space is allocated on the next volume with available space.

**File Type**
The Optim file type for which the data set is to be allocated.

**Default**
Volume information for allocating data sets for Optim file types not listed (such as, Point-and-Shoot Files).

Additionally, if you do not specify volume information for an individual file type listed on the panel, the default volume information is used.

**Archive**
Volume information for allocating data sets for Archive Files, as well as Subset Extract Files.

**Extract**
Volume information for allocating data sets for Extract Files.

**Control**
Volume information for allocating data sets for Control Files.

**Load** Volume information for allocating data sets for Load files.

**Unit Name**
The 1 to 8 character name of a group of devices, such as TAPE, SYSDA, or SYSALLDA. Consult site management to obtain a valid list of names for your installation. If a unit name is not specified, the default unit is used.

**Nbr Vols**
The maximum number of volume serials the data set may span.

**Volume Serial(s)**
Up to six specific volume serial numbers to be used for allocating data sets.

**Note:** You can enter the number of volumes or the volume serial, but not both.

**Maximum DASD Tracks**
The maximum number of tracks allowed to be allocated for a data set on a DASD device. Specify a value in the range 16695 to 65535. Consult with site management before entering a value.

For more information about allocating data sets, see Appendix B, “Allocating External Files,” on page 417.

## Editor and Display Options

A set of editor and display options are available. When you select Option 2 from the **Choose Option Type** menu, the Editor and Display Options panel is displayed.

```
------------------------ Editor and Display Options ----------------------
Command ===>

NULL Value Indicator         ===> ?        ?,!,&,%,#,@
DB2 VarChar Delimiter        ===> ;        ;,:,?,!,&,%,#,@
AutoCommit Mode              ===> A        A-Automatic, M-Manual
Deleted Rows Display         ===> S        S-Show, H-Hide
Audit Mode                      : OFF      Forced OFF
Maximum Fetch Rows           ===> 1000     In Range (1 - 100000)
Display Column Attributes    ===> N        Y-Yes, N-No
Display Data in Hex          ===> N        Y-Yes, N-No
Editor Display Format        ===> C        C-Columnar, S-Sidelabels
Columnar Max Display Width   ===> 20       In Range (2 - 70)
Sidelabel Max Display Width  ===> 50       In Range (50 - 32767)
Columnar LOB Display Width   ===> 15       In Range (2 - 70)
User Supplies Defaults       ===> N        Y-Yes, N-No
NULL as Insert Default       ===> N        Y-Yes, N-No
Editor Exit Processing       ===> E        E-Entry Panel, P-Prompt Panel
ENTER Exits Criteria Panels  ===> A        A-Always, N-Never, U-Unchanged
AutoSwitch Mode              ===> Y        Y-Yes, N-No
SelCrit w/Self Ref Rel       ===> Y        Y-Yes, N-No
Single View Mode             ===> N        Y-Yes, N-No
```

*Figure 176. Editor and Display Options*

## Panel

The editor and display options apply to all Optim components, unless otherwise indicated. They include:

**NULL Value Indicator**
The character used to represent the null value. Specify one of the following characters:

? ! & % # @

The **NULL Value Indicator** must be different from the **DB2 VarChar Delimiter**. The default character is question mark (?).

**DB2 VarChar Delimiter**
The character used to delimit the end of a variable-length character string when trailing blanks are to be retained. Specify one of the following characters:

; : ? ! & % # @

The **DB2 VarChar Delimiter** must be different from the **NULL Value Indicator**. The default character is the semicolon (;).

For example, if the string “ABC ” is entered in a variable-length character field, it is retained as “ABC” unless a delimiter is used to indicate otherwise. If the delimiter character is “;”, the string entered as “ABC ;” is retained as “ABC ”.

**AutoCommit Mode**
For Access only. Mode for committing changes to the database. Specify:

**A** AutoCommit (default). Changes to the data are automatically committed to the database every time you press a program function key or ENTER.

**M** Manual. You must use the COMMIT command to commit database changes.

You can use the AUTOCOMMIT command during a session to change this option.

**Deleted Rows Display**
For Access only. Indicator for displaying deleted rows when editing data. Specify:

**S** Show deleted rows. (The D, Delete, status flag indicates deleted rows.)

**H** Hide deleted rows.

You can use the SHOW and HIDE commands to override this option during an edit session.

**Audit Mode**
For Access only. Indicator for the Audit facility during an edit session. (Site management may force this option to ON or OFF, and prevent its modification. If so, ON indicates the Audit facility is active for all users; OFF indicates that the Audit facility is not active for all users.) If modifiable, specify:

**Y** Activate the Audit facility.

**N** Deactivate the Audit facility.

**Maximum Fetch Rows**
The maximum number of rows that can be obtained from one table at one time during a session. This value can be from 1 to the site-defined maximum. The figure shows the distributed default maximum, 1000 rows. (This value may be different at your site.)

You can use the MAX FETCH ROWS command to modify this value for the current edit, browse, or Point-and-Shoot session and refetch all data.

**Note:** This limit does not apply during a Compare File browse session. See "Compare Options" on page 388.

**Display Column Attributes**
Indicator for the display of column attributes (data type, length, and nullability). Specify:

**Y** Display attributes after each column heading in columnar display, or next to each column heading in sidelabels display.

**N** Do not display attributes.

You can use the ATTRIBUTES command during the session to toggle the display of column attributes.

**Display Data in Hex**
Indicator for display of hexadecimal data. Specify:

**Y** Display in hexadecimal format. Each line of text character data requires three display lines: one for text characters and two for the corresponding hexadecimal values.

**N** Display text characters only.

You can use the HEX command during the session to toggle the display of hexadecimal format.

**Editor Display Format**
Indicates the panel display format for the data. Specify:

**C** Display data in columnar format. (Column headings are arranged horizontally, with multiple rows of data displayed after.)

**S** Display data in sidelabels format. (Column headings are arranged vertically on the panel, and one row of data is displayed at a time.)

During the session, you can use the SIDELABELS command to toggle between columnar and sidelabels format.

**Columnar Max Display Width**
The maximum number of characters (from 2 through 70) per column that are displayed in columnar format. This value does not affect the actual data in the column, but may truncate the display.

**Note:** To edit a column, all data must be displayed. Specify a maximum display width that is greater than or equal to the length of data in the column, or use the EXPAND command to display the entire column.

**Sidelabel Max Display Width**
The maximum number of characters (from 50 through 32767) per column that are displayed in sidelabels format. This value does not affect the actual data in the column, but may truncate the display.

**Note:** To edit a column, all data must be displayed. Specify a maximum display width that is greater than or equal to the data length of the column, or use the EXPAND command to display the entire column.

**Columnar LOB Display Width**
The maximum number of characters (from 2 through 70) displayed for an LOB column. This setting does not affect the actual data in the column, but may truncate the display.

**User Supplies Defaults**
For Access only. Source of default values for columns defined to DB2 without a default value. (Site management may force this option to Yes or No, and prevent its modification.) If modifiable, specify:

**Y** User must supply values for every column that cannot receive a default.

**N** Access supplies default values, based on the column data type, including blank, zero, current date, current time, and current timestamp.

**NULL as Insert Default**
For Access only. Default handling of blank nullable columns when inserting a row. (Site management may force this option to Yes or No, and prevent its modification.) Specify:

**Y** Nullable columns default to the NULL value.

**N** Access supplies default values (other than NULL), based on the column data type, including blank, zero, current date, current time, and current timestamp.

**Editor Exit Processing**
For Access only. Default behavior when END is used to terminate an edit or browse session. Specify:

**E** Display the panel from which the session was invoked. This panel may be the Describe Columns panel, the Specify Selection Criteria panel, or the SQL WHERE Clause panel.

**P** Display the prompt panel with which the session was initially begun. This panel may be a Choose panel or a selection list.

**ENTER Exits Criteria Panels**
For Access only. Action taken when the ENTER key is used to exit the Specify Selection Criteria panel, the SQL WHERE Clause panel, and the Describe Columns panel during an edit or browse session. Specify:

**A** ENTER or END exits the panel, unless errors are detected (default).

**N** ENTER never exits the panel; only END exits the panel.

**U** ENTER exits the panel if values are not changed. If values are changed, END exits the panel.

**AutoSwitch Mode**
For Access only. AutoSwitch mode for multi-way joining during an edit or browse session. Specify:

**Y** AutoSwitch mode. When scrolling a table that is joined to two or more lower-level tables in a stack and the currently displayed lower-level table has no related rows, the first table in the stack that has at least one related row is displayed instead.

**N** No AutoSwitch mode (default). If the lower-level table has no related rows, do not switch to another table in the stack.

**SelCrit w/Self Ref Rel**
For Access only. Application of selection criteria when a table is self-referenced while browsing or editing data. Specify:

**Y** Apply any selection criteria each time the table is referenced. (This setting is the default Access Definition parameters setting.)

**N** Apply selection criteria to the table the first time it is accessed, but not when the table is self-referenced.

**Single View Mode**
For Access only. Indicator for joining views when browsing and editing data. Specify:

**Y** Views cannot be joined.

**N** Views can be joined (default).

Browsing and editing views in single view mode is faster and more efficient because relationship processing is bypassed. However, joining is required to display or edit a segment of related data. (You cannot modify this option from an edit or browse session.)

## Job Card and Print Options

The Job Card and Print Options panel determines several user-specific parameters that govern batch execution.

When you select Option 3 from the **Choose Option Type** menu or when you enter Yes at the **Review Job Card** prompt on the User Options panel, the following panel is displayed. The information displayed on the panel is initially set to the user specifications for ISPF option 0 (select option 2 from the Log/List Menu).

```
---------------------- Job Card and Print Options ----------------------------
Command ===>                                                 SCROLL ===> PAGE

Job Statement:
  ===> //SPECHTR  JOB  'ACCT-INFO',MSGCLASS=8,CLASS=6,
  ===> //         NOTIFY=SPECHTR
  ===>
  ===>
  ===>

Report Parameters:
  Report Type     ===> D              D-Dataset, S-SYSOUT
  If Dataset: DSN ===> FOP.PRINT
              Disposition  ===> M     M-Mod, O-Old
  If SYSOUT:  SYSOUT Class ===> *     A - Z, 0 - 9, *
              Destination  ===> LOCAL
              Hold         ===> Y     Y-Yes, N-No

Job Changes Permanent      ===> N     Y-Yes, N-No
```

*Figure 177. Job Card and Print Options*

## Panel

The job card and print options apply to Move, Compare and Archive, and include the following parameters.

**Job Statement**
The default job statement for batch jobs. Type a valid job statement of up to five lines of 66 characters each. The first line must be in the form:

//name JOB

where *name* is the job name.

**Report Type**
Indicate the type of report output generated. Type:

- **D** to write the report to a dataset.
- **S** to route the report to SYSOUT.

**If Data Set: DSN**
Specify the name of the dataset to receive the report. Type the name of a sequential file.

**If Data Set: Disposition**
Specify the disposition of the dataset. Type:

- **M** to append the new report to existing data.
- **O** to overwrite existing data with the new report.

**If SYSOUT: SYSOUT CLASS**
Specify the SYSOUT class. Type:

- **A-Z** for any alphabetic character.
- **0-9** for any numeric character.
- * (an asterisk) to use the msgclass for the job.

**If SYSOUT: Destination**
Specify the destination of the SYSOUT file.

**If SYSOUT: Hold**
Indicate the hold job output status. Type:

- **Y** to hold the SYSOUT file for online viewing.

- **N** to indicate the SYSOUT file is available for printing when execution is complete.

If not specified, the site default is used.

**Job Changes Permanent**
Indicate whether the job changes are permanent. Type:

- **Y** if the job changes are permanent.
- **N** to if the job changes are not permanent.

### Job Card Profiled

The job card information is profiled. If no information is profiled, the job card is filled in with ISPF variables, set from ISPF option 0 (then select option 4 from the Log/List Menu). Use the RESET command to reset the job card to these original values.

**Note:** If the profiled information exceeds the display width of 66 characters, it is truncated.

## Compare Options

The compare options determine several user-specific parameters that govern the Compare Process. When you select Option 4 from the **Choose Option Type** menu, the Compare Options panel is displayed.

```
------------------------------ Compare Options --------------------------------
Command ===>                                                 SCROLL ===> PAGE

Source Flags Display      ===> D         D-Display, B-Blank
Maximum Fetch Rows        ===> 100000    In Range (1 - 2000000000)
Display Unused Cols       ===> Y         Y-Yes, N-No
```

*Figure 178. Compare Options*

### Panel

The compare options apply to Compare, and include:

**Source Flags Display**
Option to display the "12" source flag for equal rows, while browsing the Compare File. Specify:

**D** Display the source flag.

**B** Do not display the source flag. (**Src** is blank for equal rows.)

**Maximum Fetch Rows**
The maximum number of rows that can be obtained from one table at one time when browsing a Compare File. Specify a value from 1 through the site-defined maximum. (The possible range of values is displayed on the panel.)

**Display Unused Cols**
Option to display unused columns in Compare Browse or the Compare File Report. Unused columns are specified in a Column Map used with the Compare Process. Specify:

**Y** Display unused columns.

**N** Do not display unused columns.

# Archive Options

Several options are provided for archive and restore processing. When you select Option 5 from the **Choose Option Type** menu, the following panel is displayed.

```
------------------------------ Archive Options -------------------------------
Command ===>                                                SCROLL ===> PAGE

Sort Archive List by      ===> 1         1-Date, 2-User, 3-Name, 4-Grp, 5-Desc
Default Sort Sequence     ===> D         A-Ascending, D-Descending
Display Detail Mode       ===> Y         Y-Yes, N-No
Share Search in List      ===> Y         Y-Yes, N-No
Dual Archive File Output  ===> Y         Y-Yes, N-No
Default DAA Table in TM   ===> Y         Y-Yes, N-No
Restore Table Change      ===> N         N-None, I-Info, F-Fail
Replace Archive Dir       ===> Y         Y-Yes, N-No
```

*Figure 179. Archive Options*

## Panel

The archive options apply to Archive, and include:

**Sort Archive List by**
The default column for sorting Archive Directory entries listed on the Archive Files panel. Specify:

**1** Date

**2** User

**3** File Name

**4** Group

**5** Description

**Default Sort Sequence**
The default sequence for sorting Archive Directory entries listed on the Archive Files panel. Specify:

**A** List in ascending order, according to the value in the **Sort Archive List by** column.

**D** List in descending order, according to the value in the **Sort Archive List by** column.

**Display Detail Mode**
The default setting for Optim Directory entries listed on the Archive Files panel. Specify:

**Y** List each entry in two-line detail mode. Display the Date, File Name, Group, and Unit on the first line, and the User, Description, and Security on the second line.

**N** List each entry in single-line mode to display only the Date, File Name, Group, and Unit.

**Share Search in List**
Option to retain criteria for searching, browsing, or restoring data from the **Archive Files** for use with subsequent commands. Specify:

**Y** Retain criteria specified on the Archive Selection Criteria – Columns panel and display each time a Search, Browse, or Restore line command invokes the Archive Search Criteria – Columns panel from the Archive Files panel.

**N** Always display the Archive Selection Criteria – Columns panel without previously specified criteria.

**Dual Archive Output**
Option to write an Archive File to more than one output. Specify:

**Y** Write Archive File to more than one output.

**N** Do not write Archive File to more than one output.

**Default DAA Table in TM**
Option for populating destination table names in the Table Map editor. Specify:

**Y** Populate with names of tables for which Delete After Archive was specified in the Access Definition when data was archived.

**N** Populate with names of all tables in the Archive File.

**Restore Table Change**
Action taken when a table targeted for restoration is different from the table used to create the Archive File. Specify:

**N** No action is taken; the Restore Process proceeds.

**I** Display an informational message.

**F** The Restore Process fails.

**Replace Archive Dir**
Option for batch processing when an entry for the Archive File already exists. (This option is available only if enabled by a Site Option.) Specify:

**Y** Proceed with the Archive Process, replacing the Archive File entry in the Archive Directory (default).

**N** Halt the Archive Process with an error message.

---

# Legacy Options

The Legacy options apply when creating Legacy Tables needed to apply Move or Compare processes to legacy data. When you select Option 6 from the **Choose Option Type** menu, the Legacy Options panel is displayed.

```
---------------------------------- Legacy Options ----------------------------
Command ===>                                                 SCROLL ===> PAGE

Default Libraries:
  Copybook            ===>
PL/I Source Margins:
  From                ===> 2        In Range (1 - 100), Default value is 2
  To                  ===> 72       In Range (1 - 100), Default value is 72
Legacy Data Attributes:
  MIXED DATA             : N        (Y-Yes, N-No)
  EBCDIC CCSID           : 0        In Range (1 - 65533), 0 =  DB2 CCSID
Detect ISO keywords      : N        (Y-Yes, N-No)
```

*Figure 180. Legacy Options*

## Panel

The Legacy options apply to Optim Legacy, and include:

**Default Libraries: Copybook**
The name of the Default Library for copybooks used to create Legacy Tables. You can enter the fully qualified name by enclosing it in single quotes. Or you can specify a pattern, using * or %, to request a selection list of copybook data sets.

**PL/I Source Margins:**
The location of the PL/I statement within a source record. Specify:

**From** A value from 1 through 100. The default value is 2.

**To** A value from 1 through 100. The default value is 72.

**Legacy Data Attributes**
The Legacy Data Attributes apply to all legacy data processed by Optim. They must be set to appropriate values before creating a definition for a Legacy Table.

**Note:** The values can only be updated if the Legacy MIXED DATA site option has been set to U.

**MIXED DATA**
Specifies whether the code points X'0E' and X'0F' have special meaning as the shift-out and shift-in controls for double-byte character strings in Legacy data.

**Y** Indicates that these code points have the special meaning. Therefore, character strings can be either single-byte character set (SBCS) or MIXED data. The CCSID that is specified in the Legacy EBCDIC CCSID field must be the mixed CCSID for the encoding scheme. Using this value, Optim determines the associated SBCS and double-byte character set (DBCS) CCSIDs for the encoding scheme.

**N** Indicates that these code points have no special meaning. Therefore, all character strings in Legacy data processed by Optim are SBCS data.

**EBCDIC CCSID**
Specifies the default CCSID for EBCDIC-encoded character data that is stored in your Legacy data. Acceptable values for this field are in the range of 1 - 65533. You can specify 0 (zero) to indicate that Optim should use the default EBCDIC CCSID of the connected DB2 subsystem as the default EBCDIC CCSID of your Legacy data.

**Note:** If Legacy MIXED DATA is set to Y, you must specify a valid mixed CCSID value.

**Detect ISO keywords**
Specifies whether Optim looks for the $$FOPDAT, $$FOPTIM, $$FOPSTP keywords when parsing copybooks to build Legacy Table definitions.

**Y** Optim looks for the $$FOP*xxx* keywords when parsing copybooks. The $$FOPDAT, $$FOPTIM, and $$FOPSTP Optim keywords can be included within copybook comments. The comment must appear on the line directly before the related field definition. The $$FOP*xxx* keyword must be separated by spaces from other text in the comment line.

The $$FOP*xxx* keywords indicate that the field following the comment is in ISO DATE, TIME, or TIMESTAMP format. When Optim finds the $$FOP*xxx* keyword, it sets the following field to the appropriate DAT, TIM, or STP type in the Legacy Table definition. For COBOL, the field must be defined as alphanumeric. For PL/I, the field must be defined as character. The field must also adhere to the following rules.

$$FOPDAT - ISO DATE: The field must have a length of 10 and be defined in YYYY-MM-DD format.

$$FOPTIM - ISO TIME: The field must have a length of 8 and be defined in HH:MM:SS format.

$$FOPSTP - ISO TIMESTAMP: The field must have a length of 26 and be defined in YYYY-MM-DD-HH.MM.SS.NNNNNN format.

The following example illustrates the $$FOP*xxx* keyword comments and related field definitions in a COBOL copybook.

```
   01 SAMPREC.
     02 CUSTNAME          PIC X(80).
*  $$FOPDAT
```

```
     02 SAMPLE-DATE       PIC X(10).
*  $$FOPTIM
     02 SAMPLE-TIME       PIC X(8).
*  $$FOPSTP
     02 SAMPLE-TIMESTAMP  PIC X(26).
```

**N** Optim does not look for the $$FOP*xxx* keywords when parsing copybooks (default).

# Chapter 13. Performance

You can configure Archive as well as your database so that Archive, Restore, and Delete Processing are as efficient as possible. The configuration strategy you choose depends on your needs and the constraints of your database processing. This chapter provides some general strategies that you can employ to meet your specific performance needs.

These strategies are not intended to solve specific performance problems but are simply suggestions to aid your development of an archive strategy for your database, and can be tailored or even combined to fit your specific needs.

**Note:** This chapter focuses on methods and strategies that allow you to improve performance for Archive, Delete, and Restore Processing. You can employ these same strategies to improve the Compare, Extract, and Insert Processing of large numbers of rows.

## Processing Strategies

The following section describes strategies and tips to help you optimize processing. To help you determine the best strategy for your database, you can use the DB2 Performance Monitor to measure processing speed under different strategies. Once you decide on a strategy, you can fine-tune Archive and your database to achieve the best results.

**Note:** Changes to your database that provide maximum optimization for Archive Processing may have negative implications for your database application processing. You must determine for yourself the combination of solutions that best suits the demands upon your particular database. In doing this, you should consider Archive an important database application, and configure your database accordingly.

### Speed Processing

The most basic archiving strategy applies a single Archive Process to as many rows as possible in the shortest period of time. Speed processing is typically performed during off-peak or maintenance hours to reduce (or eliminate) competition from other application processing. In cases where the database has no competition, you can remove any unnecessary constraints from the database and from Archive. Another way to enhance efficiency is to design the partitioning of your database tables so that you can access the rows quicker. (For more information, see **Partitioned Tables**.)

To improve the performance of Delete Processing, you might remove any unnecessary indexes from database tables and, within Archive, you can disable row comparison and increase the commit frequency.

### Running Multiple Processes

A more intricate strategy for processing rows during off-peak hours is to process as many rows as possible using multiple processes, rather than a single process. Since Archive and Delete Processing is light on CPU usage, creating multiple requests to process data at one time may significantly increase processing throughput.

The recommendations for this strategy are similar to those for the Speed Processing strategy. Additionally, for the Delete Process, you could disable row comparison and table locking and maximize the commit frequency.

### Data Integrity Processing

The previously listed strategies are designed to process rows during off-peak hours. However, if you must process a significant number of rows, your off-peak window may be inadequate. Additionally, your database tables may be continuously updated, where a significant shut down of database access is difficult to accommodate.

In this case, you can adopt a strategy in which a small number of rows are processed regularly during normal operating hours. The archiving and deleting of such a small number of rows should not affect other application processing. If you want to ensure database integrity during the Delete Process you want to ensure that table locking and compare row contents are enabled.

### Partitioned Tables

An additional way to increase the speed of Archive and Delete Processing is to divide your database tables into partitions. Depending on the type of data in the tables, rows can be sorted by frequency of access and age into the partitions. Since there is less I/O traffic to an infrequently used partition, the speed of processing rows is increased. Additionally, you could more easily disable unnecessary database indexes for a partition that is used infrequently.

## Performance Tools

Archive provides various features and tools to complement the processing strategies that you establish to best enhance performance at your site. Using these features you can better diagnose and configure Archive and your database to more successfully process your database tables. These features are discussed in this chapter.

**Table Access Strategy**

Archive generally uses a key lookup when a DBMS index is available and a scan when an index is not available. You can override the default strategy:

**Note:** If accessing a significant portion of the table, Archive will use a scan, even if an index is available.

- For an Archive Process you can override the default method of accessing the parent or child table for each relationship.
- For a Delete or Restore Process, you can override the default method used to delete rows from or restore rows to the database.

**Key Lookup Limit**

When processing rows using a key lookup, Archive generally looks up one key at a time. You can specify the maximum number of key lookups performed at one time for a table.

- For an Archive Process, you can specify the maximum number of simultaneous key lookups (1 to 100) performed when processing rows from the parent and child tables.
- For a Delete or Restore Process, you can specify the maximum number (1 to 100) of key lookups performed at one time for a table.

**Index Analysis**

Analyze DBMS indexes to process rows in the Access Definition or the Archive File, and determine if you need to create any indexes.

## Table Access Strategy

Archive allows you to modify the default method (scan or key lookup) used to access tables for processing. Additionally, for an Archive or Delete Process, if Key Lookup is the method used, you can specify the maximum number of key lookups performed at one time.

You can change the access strategy for an Archive, Delete, and Restore Process, from the following panels:

## Specify Relationship Usage

From the Specify Relationship Usage panel, type ACM to display the Choose Access Method / Key Lookup Limit pop-up window.

```
+------------------ Choose Access Method / Key Lookup Limit -----------------+
|                                                                      1 of 3 |
| Access Method Values:                                                       |
|   K - Key Lookup                                              Key           |
|   S - Table Scan                                  Access      Lookup        |
|   blank - Software Chooses                        Method      Limit         |
|                                      Rel    ------------ ------------       |
|     Parent Table        Child Table  Name   Parent Child Parent Child       |
| ------------------ ------------------ ------>> ------ ----- ------ -----     |
| ********************************* TOP ********************************** |
| FOPDEMO.CUSTOMERS   FOPDEMO.ORDERS     RCO                                  |
| FOPDEMO.ORDERS      FOPDEMO.DETAILS    ROD                                  |
| FOPDEMO.ITEMS       FOPDEMO.DETAILS    RID                                  |
| ******************************** BOTTOM ********************************** |
+-----------------------------------------------------------------------------+
```

*Figure 181. Choose Access Method / Key Lookup Limit*

The window displays all of the relationships in the Access Definition. You can override the default access method and specify the maximum number of simultaneous key lookups used when archiving rows for each relationship in the Access Definition.

## Delete Rows from Archive Process

From the Delete Rows from Archive Process panel, type ACM to display the Specify Access Method and Key Lookup Limit pop-up window.

```
+--- Choose Access Method/Key Lookup Limit  --+
|                                    1 of 4 |
| Access Method Values:                     |
|   K - Key Lookup                          |
|   S - Table Scan                          |
|   blank - Software Chooses                |
|                                    Key    |
|                              Access Lookup |
| Table Names In Archive File  Method Limit  |
| ---------------------------  ------ ------ |
| ****************** TOP ******************* |
| FOPDEMO.CUSTOMERS                         |
| FOPDEMO.DETAILS                           |
| FOPDEMO.ORDERS                            |
| FOPDEMO.ITEMS                             |
| ***************** BOTTOM ****************** |
+-------------------------------------------+
```

*Figure 182. Choose Method/Limit - Delete Process*

The window displays the names of all the tables in the Archive File. You can override the default access method and specify the maximum number of simultaneous key lookups used when deleting rows for each table in the Archive File.

### Table Map

From the Restore Process Table Map panel, type ACM to display the Choose Access Method pop-up window.

```
+-------- Choose Access Method -------+
|                              1 of 4  |
| Access Method Values:                |
|   K - Key Lookup                     |
|   S - Table Scan                     |
|   blank - Software Chooses           |
|                                      |
|                               Access |
| Destination Table Name        Method |
| ---------------------------  ------ |
| ************** TOP ***************** |
| FOPDEMO.CUSTOMERS                    |
| FOPDEMO.DETAILS                      |
| FOPDEMO.ORDERS                       |
| FOPDEMO.ITEMS                        |
| ************ BOTTOM **************** |
+--------------------------------------+
```

*Figure 183. Choose Access Method*

The window displays the names of all the destination tables in the Table Map. You can override the default access method used when restoring rows to the destination tables specified in the Table Map.

**Note:** Changing the access method to Key Lookup in a Table Map only affects tables where the processing method is Update. If the processing method for a table is Insert, the access method is Table Scan.

## Access Method

For an Archive Process, you can override the default method (scan or key lookup) for accessing the parent or child table for each relationship.

For a Delete or Restore Process, you can override the default method for accessing rows from each table. You can specify the following:

**<blank>**
By default, the process automatically determines whether to use scan or key lookup.

**Note:** In general, a key lookup is used when a DBMS index is available, and a scan is used when an index is not available. However if you need to access a significant portion of the table, a scan may be used, even if an index exists.

**K - Key Lookup**
Force the process to use a key lookup to locate rows by using a WHERE clause to search for primary or foreign key values.

**Note:** Before selecting, you should verify that a DBMS index exists. For more information, see “Index Analysis” on page 397.

**S - Table Scan**
Force the process to read all rows in a table at one time.

### ACM Command

To set the access method for all listed tables, use the ACM command with a B (blank), K (key lookup), or S (table scan) operand. For example, enter ACM B to blank the access method for all listed tables.

## Key Lookup Limit

Allows you to specify the maximum number of key lookups performed at one time for a table. Valid values are 1 through 100.

If no value is specified, then the default value is used. The default value is specified by a user option. See "User Options" on page 372 for more information.

**Note:** Increasing the Key Lookup Limit only affects processes that use the Key Lookup access method.

Increasing the **Key Lookup Limit** may significantly improve performance. For example, if you specify 5 as the **Key Lookup Limit**, 5 key values are searched in a single request to the DBMS, which reduces the number of times the DBMS is called.

For most sites, increasing the **Key Lookup Limit** to the maximum (100) is most effective.

**Note:** The following conditions must be true to use multiple key lookups in a Delete Process.

- An index on the primary key is defined for the table.
- Compare Row Contents is not selected for the Delete Request.
- Row level Archive Actions are not defined for the Delete Process (e.g., Before Delete of Row).
- The table is not a parent table in a DBMS relationship.

---

## Index Analysis

Missing DBMS indexes is the most frequent cause of performance problems in an Archive or Delete Process. Archive allows you to analyze DBMS indexes for selected relationships and primary keys, allowing you to diagnose whether you need to create indexes.

If the status of an index is shown as **Partial**, or **None**, creating the necessary index may enhance processing performance.

### Relationship Index Analysis

You can analyze DBMS indexes for relationships used with the Access Definition through the Specify Relationship Usage panel. Type SHOW INDEXES to display the Relationship Index Analysis panel.

```
------------------------ Relationship Index Analysis --------------------------
Command ===>                                                 Scroll ===> PAGE

                                                                    1 of 3
AD: GRP.FOPDEMO.AD

Parent Ix    Child Ix
Stat/Needed  Stat/Needed  Relationship                  Parent Table
-----------  -----------  ----------------------------->> ---------------------
********************************** TOP ***************************************
NotAnlyzed   None       Y FOPDEMO.ORDERS.RCO            FOPDEMO.CUSTOMERS
Full       Y NotAnlyzed   FOPDEMO.DETAILS.RID           FOPDEMO.ITEMS
NotAnlyzed   Full       Y FOPDEMO.DETAILS.ROD           FOPDEMO.ORDERS
******************************** BOTTOM **************************************
```

*Figure 184. Relationship Index Analysis*

The Relationship Index Analysis panel lists each selected relationship in the Access Definition, with an analysis of DBMS indexes for the corresponding parent and child tables.

## Primary Key Index Analysis

Use Primary Key Index Analysis window to determine the tables in an Archive File that have an index on the primary key. The Primary Key Index Analysis pop-up window lists each table in the Archive File, with an analysis of DBMS indexes for the table.

From a Specify Parameters and Execute panel or the Delete Rows from Archive Process panel, use the SHOW INDEXES command to display the Index Analysis pop-up window listing the tables and the status of supporting indexes.

```
+----------------------- Index Analysis ----------------------+
¦                                          1 of 3             ¦
¦ Table Name                 Index Name         Index Status  ¦
¦ -------------------------- ------------------ ------------  ¦
¦ ************************* TOP ***************************** ¦
¦ FOPDEMO.ITEMS              XITEMPK            DBPK          ¦
¦ FOPDEMO.ORDERS             XORDRPK            Unique        ¦
¦ FOPDEMO.DETAILS                               Partial       ¦
+-----------------------------------------------------------+
```

*Figure 185. Index Analysis*

## Status

The possible status of indexes for the Relationship Index Analysis and Index Analysis windows are:

**DBPK** Index exactly matches the DBMS primary key definition for the table. (Index Analysis only)

**Unique**
A unique index is defined for the table; however, no primary key is defined. (Index Analysis only)

**Full** An index on an Optim primary key is defined for the table. The index includes all primary key columns at the beginning of the index, in any order, but may include additional columns.

**Note:** On the Relationship Index Analysis window, the Full status describes any DBPK, Unique, or Full index on the primary key.

**Partial** An index on an Optim primary key is defined for the table. The index includes at least one primary key column at the beginning of the index, but may include additional columns.

**None** No index for the necessary columns exists.

**NotAnalyzed**
Index is not needed. (Relationship Index Analysis only)

# Appendix A. Sample Database Tables and Structure

The sample database, created during installation of Optim, is described in this Appendix. As a group, the database tables include information on customers, orders, shipping instructions, and so on.

Two files for demonstrating *Move* or *Compare for IMS, VSAM, and Sequential Files*are also provided and you can use the FOPNPRIV Extract File to create tables for use with the Column Map lookup functions or for other data privacy or data masking needs.

The data model contains related tables and Legacy files with the following names. Each table name has the Creator ID FOPDEMO.

- OPTIM_CUSTOMERS
- OPTIM_DETAILS
- OPTIM_FEMALE_RATES
- OPTIM_ITEMS
- OPTIM_MALE_RATES
- OPTIM_ORDERS
- OPTIM_SALES
- OPTIM_SHIP_INSTR
- OPTIM_SHIP_TO
- OPTIM_STATE_LOOKUP
- OPTIM_ACTIONS
- VENDITEM
- VENDOR
- DEPARTMENT
- EMPLOYEE
- POSITION
- JOBCODE

The database also contains the following tables with the Creator ID FOPDEMO2. The tables with the Creator ID FOPDEMO2 are not populated during installation. These tables are used to demonstrate the facilities provided by Optim.

- OPTIM_CUSTOMERS
- OPTIM_DETAILS
- OPTIM_ITEMS
- OPTIM_ORDERS

Another table, FOPDEMO3.ACCOUNTS, is also used to demonstrate Optim. There are no distributed relationships for this table.

## SALES Table (FOPDEMO.OPTIM_SALES)

The SALES table identifies each salesperson by name, ID number, and manager.

It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_SALES
  (SALESMAN_ID CHAR(6) NOT NULLABLE,
  FIRST_NAME VARCHAR(15) NOT NULLABLE,
```

```
   LAST_NAME VARCHAR(15) NOT NULLABLE,
   NATIONALITY VARCHAR(30) NULLABLE,
   NATIONAL_ID VARCHAR(30) NULLABLE,
   PHONE_NUMBER VARCHAR(20) NOT NULLABLE,
  AGE INTEGER NOT NULLABLE,
  SEX CHAR(1) NOT NULLABLE,
  TERRITORY VARCHAR(14) NOT NULLABLE
   EMAIL_ADDRESS VARCHAR(70) NOT NULLABLE,
  MANAGER_ID CHAR(6)NULLABLE)
 IN FOPDEMOB.FOPDEMOS
 CCSID EBCDIC;
```

## Content

The SALES table contains the following columns:

**SALESMAN_ID**
CHAR; up to 6 characters; cannot contain null

**FIRST_NAME**
VARCHAR; up to 15 characters; cannot contain null

**LAST_NAME**
VARCHAR; up to 15 characters; cannot contain null

**NATIONALITY**
VARCHAR; up to 30 characters; nullable

**NATIONAL_ID**
VARCHAR; up to 30 characters; nullable

**PHONE_NUMBER**
VARCHAR; up to 20 characters; cannot contain null

**AGE** INTEGER; cannot contain null

**SEX** CHAR; 1 character; cannot contain null

**TERRITORY**
VARCHAR; up to 14 characters; cannot contain null

**EMAIL_ADDRESS**
VARCHAR; up to 70 characters; cannot contain null

**MANAGER_ID**
CHAR; up to 6 characters; nullable

## Indexes and Primary Keys

The SALES table has one index and primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XSALESPK | SALESMAN_ID | Primary, ascending | DB2 |

## Relationships to other tables

The SALES table is a parent of:

- The CUSTOMERS table, through a foreign key on column SALESMAN_ID.
- The MALE_RATES table, through an Optim data-driven relationship on column AGE when SEX = 'M'.

- The FEMALE_RATES table, through an Optim data-driven relationship on column AGE when SEX = 'F'.
- The STATE_LOOKUP table, through an Optim substring relationship using SUBSTR(SALESMAN_ID,1,2).
- The ORDERS table, through an Optim relationship on column SALESMAN_ID.

The SALES table is a child of the EMPLOYEE Legacy Table, through an Optim relationship on column SALESMAN_ID.

## CUSTOMERS Table (FOPDEMO.OPTIM_CUSTOMERS)

The CUSTOMERS table contains customer names, ID numbers, and addresses.

It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_CUSTOMERS
   (CUST_ID CHAR(5) NOT NULL,
    CUSTNAME CHAR(60) NOT NULL,
    ADDRESS1 VARCHAR(100) NULLABLE,
    ADDRESS2 VARCHAR(100) NULLABLE,
    LOCALITY VARCHAR(56) NULLABLE,
    CITY VARCHAR(60) NULLABLE,
    STATE VARCHAR(30) NULLABLE,
    COUNTRY CODE CHAR(2) NULLABLE,
    POSTAL_CODE VARCHAR(15) NULLABLE,
    POSTAL_CODE_PLUS4 CHAR(4) NULLABLE,
    EMAIL_ADDRESS VARCHAR(70) NULLABLE,
    PHONE_NUMBER VARCHAR(20) NULLABLE,
    YTD_SALES DECIMAL(7,2) NOT NULL,
    SALESMAN_ID CHAR(6) NULLABLE,
    NATIONALITY VARCHAR(30) NULLABLE,
    NATIONAL_ID VARCHAR(30) NULLABLE,
    CREDITCARD_NUMBER VARCHAR(19) NULLABLE,
    CREDITCARD_TYPE VARCHAR(30) NULLABLE,
    CREDITCARD_EXP CHAR(4) NULLABLE,
    CREDITCARD_CVV VARCHAR(4) NULLABLE,
    DRIVER_LICENSE VARCHAR(30) NULLABLE
    CREDITCARD_HISTORY CLOB(10K) NULLABLE)
    IN FOPDEMOB.FOPDEMOS
CCSID EBCDIC;
```

### Content

The OPTIM_CUSTOMERS table contains these columns:

**CUST_ID**
CHAR; up to 5 characters; cannot contain null

**CUSTNAME**
CHAR; up to 60 characters; cannot contain null

**ADDRESS1**
VARCHAR; up to 100 characters; nullable

**ADDRESS2**
VARCHAR; up to 100 characters; nullable

**LOCALITY**
VARCHAR; up to 56 characters; nullable

**CITY** VARCHAR; up to 60 characters; nullable

**STATE**
VARCHAR; up to 30 characters; nullable

**COUNTRY_CODE**
CHAR; 2 characters; nullable

**POSTAL_CODE**
VARCHAR; up to 15 characters; nullable

**POSTAL_CODE_PLUS4**
CHAR; 4 characters; nullable

**EMAIL_ADDRESS**
VARCHAR; up to 70 characters; nullable

**PHONE_NUMBER**
VARCHAR; up to 20 characters; nullable

**YTD_SALES**
DECIMAL; dollar amount; cannot contain null

**SALESMAN_ID**
CHAR; up to 6 characters; nullable

**NATIONALITY**
VARCHAR; up to 30 characters; nullable

**NATIONAL_ID**
VARCHAR; up to 30 characters; nullable

**CREDITCARD_NUMBER**
VARCHAR; up to 19 characters; nullable

**CREDITCARD_TYPE**
VARCHAR; up to 30 characters; nullable

**CREDITCARD_EXP**
CHAR; 4 characters; nullable

**CREDITCARD_CVV**
VARCHAR; up to 4 characters; nullable

**DRIVER_LICENSE**
VARCHAR; up to 30 characters; nullable

**CREDITCARD_HISTORY**
CLOB; up to 10K; nullable

## Indexes and Primary Keys

The OPTIM_CUSTOMERS table has one index and primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XCUSTPK | CUST_ID | Primary, ascending | DB2 |

## Relationships to other tables

The OPTIM_CUSTOMERS table is a parent of:

- The OPTIM_ORDERS table, through a foreign key on column CUST_ID.
- The OPTIM_SHIP_TO table, through an Optim relationship on column CUST_ID.

It is a child of the OPTIM_SALES table, through its foreign key on column SALESMAN_ID.

## ORDERS Table (FOPDEMO.OPTIM_ORDERS)

The ORDERS table contains information for customer orders, including order number, customer ID, and salesman.

The ORDERS table is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_ORDERS
   (ORDER_ID DECIMAL(10,0) NOT NULL,
    CUST_ID CHAR(5) NOT NULL,
    ORDER_DATE TIMESTAMP NOT NULL WITH DEFAULT,
    ORDER_TIME TIMESTAMP NOT NULL WITH DEFAULT,
    FREIGHT_CHARGES DECIMAL(4,2),
    ORDER_SALESMAN CHAR(6),
    ORDER_POSTED_DATE TIMESTAMP NOT NULL WITH DEFAULT,
    ORDER_SHIP_DATE CHAR(8) NOT NULL WITH DEFAULT)
  IN FOPDEMOB.FOPDEMOS
  CCSID EBCDIC;
```

### Content

The ORDERS table has these columns:

**ORDER_ID**
DECIMAL; order ID number; cannot contain null.

**CUST_ID**
CHAR; customer ID number; up to 5 characters; cannot contain null.

**ORDER_DATE**
TIMESTAMP; cannot contain null; has default value.

**ORDER_TIME**
TIMESTAMP; cannot contain null; has default value.

**FREIGHT_CHARGES**
DECIMAL; dollar amount; can contain null value.

**ORDER_SALESMAN**
CHAR; up to 6 characters; can contain null value.

**ORDER_POSTED_DATE**
TIMESTAMP; cannot contain null; has default value.

**ORDER_SHIP_DATE**
CHAR; date when order is shipped; up to 8 characters; cannot contain null; has default value.

### Indexes and Primary Keys

The ORDERS table has one index and primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XORDRPK | ORDER_ID | Primary, ascending | DB2 |

### Relationships to other tables

The ORDERS table is a parent of the DETAILS table, through a foreign key on column ORDER_ID.

The ORDERS table is a child of:

- The CUSTOMERS table, through its foreign key on column CUST_ID.
- The SALES table, through an Optim relationship on column ORDER_SALESMAN.

## DETAILS Table (FOPDEMO.OPTIM_DETAILS)

The DETAILS table contains additional information for each order in the ORDERS table.

It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_DETAILS
   (ORDER_ID DECIMAL(5,0) NOT NULL,
    ITEM_ID CHAR(5) NOT NULL,
    ITEM_QUANTITY SMALLINT NOT NULL,
    DETAIL_UNIT_PRICE
    DECIMAL(5,2) NOT NULL)
   IN FOPDEMOB.FOPDEMOS
   CCSID EBCDIC;
```

### Content

The DETAILS table has the following columns:

**ORDER_ID**
DECIMAL; order ID number; cannot contain null.

**ITEM_ID**
CHAR; item ID number; cannot contain null.

**ITEM_QUANTITY**
SMALLINT; number of items; cannot contain null.

**DETAIL_UNIT_PRICE**
DECIMAL; unit price; dollar amount; cannot contain null.

### Indexes and Primary Keys

The DETAILS table has one index and primary key:

| Name | On Columns | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XDETLPK | ORDER_ID<br>ITEM_ID | Primary, ascending | DB2 |

### Relationships to other tables

The DETAILS table is a child of:
- The ORDERS table, through its foreign key on column ORDER_ID.
- The ITEMS table, through its foreign key on column ITEM_ID.

## ITEMS Table (FOPDEMO.OPTIM_ITEMS)

The ITEMS table contains information about each item for an order, including description, price, and quantity in inventory. It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_ITEMS
      (ITEM_ID CHAR(5) NOT NULL,
       ITEM_DESCRIPTION VARCHAR(72) NOT NULL,
       CATEGORY VARCHAR(14) NOT NULL,
```

```
 RATING CHAR(4) NOT NULL,
 UNIT_PRICE DECIMAL(5,2) NOT NULL,
 ON_HAND_INVENTORY INTEGER NOT NULL)
IN FOPDEMOB.FOPDEMOS
CCSID EBCDIC;
```

### Content

The ITEMS table has these columns:

**ITEM_ID**
CHAR; up to 5 characters; cannot contain null.

**ITEM_DESCRIPTION**
VARCHAR; up to 72 characters; cannot contain null.

**CATEGORY**
VARCHAR; up to 14 characters; cannot contain null.

**RATING**
CHAR; up to 4 characters; cannot contain null.

**UNIT_PRICE**
DECIMAL; dollar amount; cannot contain null.

**ON_HAND_INVENTORY**
INTEGER; cannot contain null.

### Indexes and Primary Keys

The ITEMS table has one index and primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XITEMPK | ITEM_ID | Primary, ascending | DB2 |

### Relationships to other tables

The ITEMS table is a parent of:

- The DETAILS table, through a foreign key on column ITEM_ID.
- The VENDITEM Legacy Table, through an Optim relationship on column ITEM_ID.

## SHIP_TO Table (FOPDEMO.OPTIM_SHIP_TO)

The SHIP_TO table contains order shipping information.

It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_SHIP_TO
   (CUST_ID CHAR(5) NOT NULL,
    SHIP_ID DECIMAL(10,0) NOT NULL,
    ADDRESS1 VARCHAR(100) NULLABLE,
    ADDRESS2 VARCHAR(100) NULLABLE,
    LOCALITY VARCHAR(56) NULLABLE,
    CITY VARCHAR(30) NULLABLE,
    STATE VARCHAR(20) NULLABLE,
    COUNTRY_CODE CHAR(2) NULLABLE,
    POSTAL_CODE VARCHAR(15) NULLABLE,
    POSTAL_CODE_PLUS4 CHAR(4) NULLABLE,
```

```
    IN_CARE_OF VARCHAR(31) NULLABLE,
    SHIPPING_CHANGE_DT TIMESTAMP NOT NULLABLE)
IN FOPDEMOB.FOPDEMOS
CCSID EBCDIC;
```

## Content

The SHIP_TO table has these columns:

**CUST_ID**
CHAR; up to 5 characters; cannot contain null

**SHIP_ID**
DECIMAL; cannot contain null

**ADDRESS1**
VARCHAR; up to 100 characters; nullable

**ADDRESS2**
VARCHAR; up to 100 characters; nullable

**LOCALITY**
VARCHAR; up to 56 characters; nullable

**CITY** VARCHAR; up to 30 characters; nullable

**STATE**
VARCHAR; up to 20 characters; nullable

**COUNTRY_CODE**
CHAR; 2 characters; nullable

**POSTAL_CODE**
VARCHAR; up to 15 characters; nullable

**POSTAL_CODE_PLUS4**
CHAR; 4 characters; nullable

**IN_CARE_OF**
VARCHAR; up to 31 characters; nullable

**SHIPPING_CHANGE_DT**
TIMESTAMP; cannot contain null

## Indexes and Primary Keys

The SHIP_TO table does not have an index. It has the following primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| | SHIP_ID | | OPT |

## Relationships to other tables

The SHIP_TO table is a parent of the SHIP_INSTR table, through an Optim relationship on column SHIP_ID.

It is a child of the CUSTOMERS table, through its Optim relationship on column CUST_ID.

## SHIP_ INSTR Table (FOPDEMO.OPTIM_SHIP_INSTR)

The SHIP_INSTR table contains detailed information for order shipping.

It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_SHIP_INSTR
    (SHIP_ID DECIMAL(10,0) NULLABLE,
     SHIP_INSTR_ID INTEGER NULLABLE,
     ORDER_SHIP_INSTR VARCHAR(254) NULLABLE,
     SHIP_UPDATED TIMESTAMP NOT NULLABLE)
   IN FOPDEMOB.FOPDEMOS
   CCSID EBCDIC;
```

### Content

The SHIP_INSTR table contains these columns:

**SHIP_ID**
DECIMAL; nullable

**SHIP_INSTR_ID**
INTEGER; nullable

**ORDER_SHIP_INSTR**
VARCHAR; up to 254 characters; nullable

**SHIP_UPDATED**
TIMESTAMP; cannot contain null

### Indexes and Primary Keys

The SHIP_INSTR table has one index and primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| FOPDEMO.XSHIPIPK | SHIP_INSTR_ID | Unique, ascending | OPT |

### Relationships to other tables

The SHIP_ INSTR table is a child of the SHIP_TO table, through its Optim relationship on column SHIP_ID.

## MALE_RATES Table (FOPDEMO.OPTIM_MALE_RATES)

The MALE_RATES table contains insurance rates, based on age. It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_MALE_RATES
        (AGE INTEGER, RATE_PER_1000 DECIMAL(5,0))
       IN FOPDEMOB.FOPDEMOS
       CCSID EBCDIC;
```

### Content

The MALE_RATES table has these columns:

**AGE** INTEGER; can contain a null value.

**RATE_PER_1000**
DECIMAL; rate in dollar amount; can contain a null value.

### Indexes and Primary Keys

The MALE_RATES table does not have an index. It has the following primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| | RATE_PER_1000 | | OPT |

### Relationships to other tables

The MALE_RATES table is a child of the SALES table, through its Optim data-driven relationship on column AGE.

## FEMALE_RATES Table (FOPDEMO.OPTIM_FEMALE_RATES)

The FEMALE_RATES table contains insurance rates based on age. It is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_FEMALE_RATES
       (AGE INTEGER,
        RATE_PER_1000 DECIMAL(5,0))
      IN FOPDEMOB.FOPDEMOS
      CCSID EBCDIC;
```

### Content

The FEMALE_RATES table has the following columns:

**AGE** INTEGER; can contain a null value.

**RATE_PER_1000**
DECIMAL; rate in dollar amount; can contain a null value.

### Indexes and Primary Keys

The FEMALE_RATES table does not have an index. It has the following primary key:

| Name | On Column | Type of Index | Type of Primary Key |
|---|---|---|---|
| | RATE_PER_1000 | | OPT |

### Relationships to other tables

The FEMALE_RATES table is a child of the SALES table, through its Optim data-driven relationship on column AGE.

## STATE_LOOKUP Table (FOPDEMO.OPTIM_STATE_LOOKUP)

The STATE_LOOKUP table is created as follows:

```
CREATE TABLE FOPDEMO.OPTIM_STATE_LOOKUP
           (DIST_CODE CHAR(3) NOT NULL,
            DISTRICT CHAR(2) NOT NULL)
          IN FOPDEMOB.FOPDEMOS
          CCSID EBCDIC;
```

### Content

The STATE_LOOKUP table has the following columns:

**DIST_CODE**
CHAR; 3 characters; cannot contain a null value.

**DISTRICT**
CHAR; 2 characters; cannot contain a null value.

### Indexes and Primary Keys

The STATE_LOOKUP table does not have an index or a primary key.

### Relationships to other tables

The STATE_LOOKUP table is a child of the SALES table through an Optim substring relationship on column DISTRICT using SUBSTR(SALESMAN_ID,1,2).

---

## VENDOR File (Legacy Table FOPDEMO.VENDOR)

The VENDOR Legacy Table is created from the following copybook:

```
01 VENDOR.
05 VEND-ID PIC S9(8) COMP.
05 VEND-NAME PIC X(20).
05 VEND-ADDR PIC X(50).
05 VEND-CITY PIC X(15).
05 VEND-STATE PIC X(2).
05 VEND-ZIP PIC X(5).
05 VEND-PHONE PIC X(10).
```

### Content

The VENDOR Legacy Table contains these fields:

**VEND-ID**
Fullword binary.

**VEND-NAME**
Character; up to 20 characters.

**VEND-ADDR**
Character; up to 50 characters.

**VEND-CITY**
Character; up to 15 characters.

**VEND-STATE**
Character; 2 characters.

**VEND-ZIP**
Character; 5 characters.

**VEND-PHONE**
Character; up to 10 characters.

VENDOR is a KSDS VSAM file with a key defined on the VEND-ID field.

### Relationships to other Tables

The VENDOR Legacy Table is a parent of the VENDITEM Legacy Table through an Optim relationship on field VEND-ID.

## VENDITEM Table (Legacy Table FOPDEMO.VENDITEM)

The VENDITEM Legacy Table is created from the following copybook:

```
01 VENDITEM.
05 VEND-ITEM-ID PIC X(5).
05 VEND-ID PIC S9(8) COMP.
05 SKU PIC X(10).
05 PRICE PIC S9(5)V99 COMP-3.
```

### Content

The VENDITEM Legacy Table contains the following fields:

**VEND-ITEM-ID**
Character; up to 5 characters.

**VEND-ID**
Fullword binary.

**SKU** Character; up to 10 characters.

**PRICE**
Packed Decimal with up to 5 significant digits and 2 decimal places.

VENDITEM is a KSDS VSAM file with a key defined on the
VEND-ITEM-ID, VEND-ID, and SKU fields.

### Relationships to other tables

The VENDITEM Legacy Table is a child of the ITEMS table, through its Optim relationship on field VEND-ITEM-ID.

The VENDITEM Legacy Table is a child of the VENDOR Legacy Table through its Optim relationship on field VEND-ID.

## DEPARTMENT Table (IMS Legacy Table FOPDEMO.DEPARTMENT)

The DEPARTMENT Legacy Table is created from the following copybook.

```
01 DEPARTMT.
05 DEPTCODE PIC X(3).
05 DEPTNAME PIC X(30).
05 LOCATION PIC X(30).
05 LOCCODE PIC X(3).
```

### Content

The DEPARTMENT Legacy Table contains the following fields:

**DEPTCODE**
Character; up to 3 characters.

**DEPTNAME**
Character; up to 30 characters.

**LOCATION**
Character; up to 30 characters.

**LOCCODE**
Character; up to 3 characters.

DEPARTMENT is the root segment of the FOPDEPDB DBD. DEPTCODE is the sequence field.

### Relationships to other tables

The DEPARTMENT Legacy Table is a parent of the EMPLOYEE Legacy Table through a physical relationship in the DBD.

## EMPLOYEE Table (IMS Legacy Table FOPDEMO.EMPLOYEE)

The EMPLOYEE Legacy Table is created from the following copybook:

```
01 EMPLOYEE.
05 EMPNO PIC X(6).
05 FRSTNAME PIC X(15).
05 INITIAL PIC X(1).
05 LASTNAME PIC X(15).
05 HIREDATE PIC X(8).
05 MANAGER PIC X(6).
```

### Content

The EMPLOYEE Legacy Table contains the following fields:

**EMPNO**
Character; up to 6 characters.

**FRSTNAME**
Character; up to 15 characters.

**INITIAL**
Character; up to 1 character.

**LASTNAME**
Character; up to 15 characters.

**HIREDATE**
Character; up to 8 characters.

**MANAGER**
Character; up to 6 characters.

EMPLOYEE is a segment in the FOPDEPDB DBD. EMPNO is the sequence field.

### Relationships to other tables

The EMPLOYEE Legacy Table is dependent on the DEPARTMENT Legacy Table through a physical relationship in the DBD.

The EMPLOYEE Legacy Table is a parent of the POSITION Legacy Table through a physical relationship in the DBD.

The EMPLOYEE Legacy Table is a parent of the SALES table through an Optim relationship on field EMPNO.

## POSITION Table (IMS Legacy Table FOPDEMO.POSITION)

The POSITION Legacy Table is created from the following copybook:

```
01 POSITION.
05 JOBCODE-STARTDATE
10 JOBCODE PIC X(3).
10 STARTDATE PIC X(8).
05 SALARY PIC S9(7)V9(2) USAGE COMP-3.
05 ENDDATE PIC X(8).
```

### Content

The POSITION Legacy Table contains the following fields:

**JOBCODE**
Character; up to 3 characters.

**STARTDATE**
Character; up to 8 characters.

**SALARY**
Packed Decimal with up to 7 significant digits and 2 decimal places.

**ENDDATE**
Character; up to 8 characters.

POSITION is a segment in the FOPDEPDB DBD. JOBCODE is the sequence field.

### Relationships to other tables

The POSITION Legacy Table is dependent on the EMPLOYEE Legacy Table through a physical relationship in the DBD.

The POSITION Legacy Table is dependent on the JOBCODE Legacy Table through an Optim relationship on field JOBCODE.

## JOBCODE Table (IMS Legacy Table FOPDEMO.JOBCODE)

The JOBCODE Legacy Table is created from the following copybook:

```
01 JOBS.
05 JOBCODE PIC X(3).
05 TITLE PIC X(15).
05 DESC PIC X(30).
05 GRADE PIC 9(2).
```

### Content

The JOBCODE Legacy Table contains the following fields:

**JOBCODE**
Character; up to 3 characters.

**TITLE** Character; up to 15 characters.

**DESC** Character; up to 30 characters.

**GRADE**
Number; up to 2 digits.

JOBCODE is the root segment of the FOPJOBDB DBD. JOBCODE is the sequence field.

### Relationships to other tables

The JOBCODE Legacy Table is a parent of the POSITION Legacy Table through an Optim relationship on field JOBCODE.

## Sample Extract File

A sample Extract File, *qual1.qual2*.DEMO.EXTRACT, is provided and contains data from the sample database to demonstrate Compare.

For additional information, see the *Compare User Manual*, Specify the Data to Compare.

## Sample Legacy File

A sample Legacy File, *qual1.qual2*.LEGACY.UNLOAD, is provided and contains data from the sample database to demonstrate VSAM files for *Move* or *Compare for IMS, VSAM, and Sequential Files*.

## Sample data privacy tables

Optim provides enhanced sample data privacy tables that can be used with the Optim data privacy providers for masking data. You can mask company and personal data such as employee names, customer names, social security numbers, credit card numbers, and email addresses. Use the tables to generate transformed data that is both unique and valid within the context of the application.

You can use the data privacy tables to:

- Prevent internal privacy breaches by de-identifying or masking the data available to developers, quality assurance testers, and other personnel.
- Improve privacy compliance initiatives by substituting customer data with contextually accurate, but fictionalized data.
- Protect confidential customer information and employee data in your application development and testing environments.
- Ensure valid test results by propagating masked elements across related tables to ensure the referential integrity of the database.

Each category of personal data is provided in a separate table for the following countries (abbreviations are in parentheses): Australia (AU), Canada (CA), France (FR), Germany (DE), Italy (IT), Japan (JP), Spain (ES), United Kingdom (UK), and United States (US). Each table includes a column of sequential numbers that is used with lookup policies that use hash values to select a row in the lookup table.

Each table name is composed of a country abbreviation prefix and the category (*countryabbreviation_category*). For example, the address table for Canada is named CA_ADDRESSES and the address table for Germany is named DE_ADDRESSES.

The schema includes the following categories:

**ADDRESSES**
Tables that include columns for street address, city, locality (e.g., state or province), and postal code.

**FIRSTNAME**
Tables that include a column with male and female given names.

**FIRSTNAME_F**
Tables that include a column with female given names.

**FIRSTNAME_M**
Tables that include a column with male given names.

**LASTNAME**
Tables that include a column with family names.

**PERSON**
Tables that include columns for birth date, given name, family name, gender, phone number, national ID number, company name, and email address.

**CCN** Tables that include a credit card number for the associated issuer (MasterCard, VISA, etc.).

**DOMAIN_NAMES**
Table that includes domain names for masking email addresses.

To create and load the sample data privacy tables, modify and run the following sample jobs in the SFOPSAMP library:

- FOPPDENG (English)
- FOPPDLA (English, Spanish, French, German, Italian)
- FOPPDJPN (Japanese)

# Appendix B. Allocating External Files

Optim requires several types of files, depending on the component you are using and the process you are performing.

Your site may have any or all of the following file types:

- Access Edit or Browse Report
- Control Browse
- Archive
- Export
- Archive Browse
- Extract
- Archive Dense Index
- Extract Browse
- Compare
- OUTPUT
- Compare Process Report
- Output SQL
- Control
- Point-and-Shoot

Optim automatically prompts for information to allocate these external files if they do not exist. Some of the values, based on the type of file to be allocated, are required as specified and cannot be changed. For Space Units values may vary depending on use of Profiled or Stored values and Optim's prior calculations). The following figure shows the prompts for data sets. The values for each prompt for each type of external file are shown in the chart "Defaults for Specific Files" on page 421.

```
+----------------------------Allocate Dataset-----------------------------+
|                                                                         |
| Data Set Name: Z13600MP.TEST.DSN                                        |
| Management Class     ===>          (Blank for Default Management Class) |
| Storage Class        ===>          (Blank for Default Storage Class)    |
|   Use Stored Values  ===>          (Y-Yes, N-No, V-View Stored Values)  |
|   Volume Serial      ===>          (Blank for Authorized Default Volume)|
|   Device Type        ===>          (Generic Unit or Device Address)     |
| Data Class           ===>          (Blank for Default Data Class)       |
|   Space Units           :          (BLKS, CYLS, TRKS, KB or MB)         |
|   Primary Quantity   ===>          (In Above Units)                     |
|   Secondary Quantity ===>          (In Above Units)                     |
|   Directory Blocks      :                                               |
|   Record Format         :                                               |
|   Record Length         :          (Value Will Be Set to Block Size)    |
|   Block Size         ===>          (Minimum 7944;0=Determine at Runtime)|
|   Data set name type ===>          (BASIC, LARGE, EXTREQ, EXTPREF or    |
|                                     Blank)                              |
|                                                                         |
| Calculated Values Shown, Change to Profiled Values ===> N (Y-Yes, N-No) |
+-------------------------------------------------------------------------+
```

*Figure 186. Allocate Dataset*

## Panel

The prompts include:

**Management Class**
The management class used to obtain the data management-related information (migration, backup, and retention criteria) for the allocation of the data set. Leave blank for default. **Management Class** is active only if SMS is active.

**Storage Class**
The storage class used to obtain the storage-related information for the allocation of the data set. Leave blank for default. **Storage Class** is active only if SMS is active.

**Use Stored Values**
The source of volume information.

**Y** Use the volume information specified on the File Unit Parameters panel, accessed from User Options. See "File Unit Parameters" on page 381 for more information.

**Note:** If you select **Y**, but no explicit or default volume information was specified on the File Unit Parameters panel, an error message is displayed.

**N** Use the volume information specified on the Allocate Dataset panel.

**V** Display the File Unit Parameters panel to specify values for unit, volume number, and one to six volume serial numbers. (For information, see Figure 187 on page 421.)

**Volume Serial**
The serial number of the volume on which the data set is to reside. This value must contain only alphanumerics, national characters, and the hyphen. Alternatively, you can leave **Volume Serial** blank to use a default volume for which you are authorized. You cannot provide both **Volume Serial** and **Device Type**.

**Device Type**
The generic unit or device address of the device on which the data set is to reside. Only alphanumeric characters are valid. Alternatively, you can leave **Device Type** blank to use the default device type. You cannot provide both **Device Type** and **Volume Serial**. This prompt is not displayed for a VSAM file.

**Data Class**
The data class used to obtain the data-related information (SPACE, LRECL, etc.) for the allocation of the data set. Leave blank for default. **Data Class** is active only if SMS is active.

**Space Units**
The DASD space for the data set to be allocated. When Optim provides the suggested sizes, this value may be BLKS, KB or MB. If you are supplying the value, select one of the following:

**BLKS** Blocks: abbreviations are BLK or B. This value is not allowed for a VSAM file.

**TRKS** Tracks: abbreviations are TRK or T.

**CYLS** Cylinders: abbreviations are CYL or C.

**KB** Kilobytes

**MB** Megabytes

Site management may provide guidelines regarding the units to use.

**Primary Quantity**
The primary quantity of space to be allocated in the indicated space units. Site management may provide guidelines regarding appropriate values.

**Secondary Quantity**
The secondary quantity of space to be allocated in the indicated space units. Site management may provide guidelines.

**Directory Blocks**
The number of directory blocks to be allocated. This value is required for partitioned data sets and must be 0 for sequential data sets. This prompt is not displayed for a VSAM file. Site management may provide guidelines.

**VSAM Type**
Archive Index File (Dense Index) only. The value is L (LINEAR) for an SMS-managed VSAM file and cannot be modified. This prompt is displayed only if you are creating a data set for an Archive Dense Index.

**Record Format**
The record format depends on the type of file and is specified as:

**FB** Export file, Point-and-Shoot file, or when using the OUTPUT command to direct browsed output, SQL, or a Process Report to a data set.

**FBS** Compare only. Required for Compare Files.

**FS** Extract Files and Control Files.

**FSA** Archive only. Required for sequential data sets created for Archive Files.

**VB** Files being browsed (e.g., browsing an Extract File, Control File or Archive File) and for Compare Report Files.

**FBM** Access Edit or Browse Report Files.

Site management may provide guidelines.

**Record Length**
The logical record length. Note requirements in the chart "Defaults for Specific Files" on page 421 for default values. It is recommended that you specify the default of 0 to allow the system to calculate the actual value at run time. Site management may provide guidelines.

**Block Size**
The block size in bytes. Note requirements in the chart "Defaults for Specific Files" on page 421 for default values.

It is recommended that you specify the default of 0 to allow the system to calculate the actual value at run time. Site management may provide guidelines.

**Data set name type**
The type of data set being allocated. Specify:

**BASIC**
The data set is not an extended or large format data set.

**LARGE**
Allocates a large format sequential data set.

**EXTREQ**
Specifies that an extended data set is required.

**EXTPREF**
Specifies that an extended data set is preferred.

*blank* Allocates a partitioned or sequential data set based on the values entered on the panel.

**Calculated Values Shown, Change to Profiled Values**
The values used to allocate the file. Specify:

**Y** Use values previously specified and profiled. Also, if you have modified the values on this panel, they are profiled and replace any previously specified values.

**N** Use the values calculated by Optim.

Calculated values are based on the "worst case scenario." For example, for an Extract File, the value is determined by the maximum number of rows that can be extracted times the length of the longest row. When available, a table's DB2 RUNSTATS values are used for the computation of rows. If valid RUNSTATS are not available, the DB2 Table Space Statistics, when activated and access granted to the user, may also be used to estimate the number of rows to be processed. Reasonable primary and secondary extents are also calculated from this value. When the estimated size of the extents exceeds the maximum value allowed by DFSMS then the allocation size will be adjusted and possibly spread over multiple disk volumes. Any unused space is freed when the process using the data set is complete.

When you have entered the necessary information on this panel, press ENTER to allocate the data set. To return to the previous panel without allocating a data set, use END or CANCEL.

## SMS Considerations

If Managed Data Support for the underlying operating environment is present, the panel displays the prompts needed to create an SMS data set.

Extract Files, Archive Files, Control Files, and Compare Files can be physical sequential data sets (DSNTYPE=BASIC), Physical Sequential Extended data sets (DSNTYPE=EXTENDED) or large format data sets (DSNTYPE=LARGE).

Optim will use DFSMS compression, which requires special DFSMS setup, if compression is available on Physical Sequential Extended format data sets. A compressed data set can save from 20 to 40 percent of DASD storage, compared to an uncompressed data set.

Restrictions:

- You cannot allocate a compressed data set for a Control, Compare, Point-and-Shoot, trace, or Report file.
- Large format data sets are not supported for Print or Report files.
- For a multi-volume extended data set, you must define a Stripe Count greater than 1. The Stripe Count is determined by the Sustained Data Rate (SDR) value. Use these guidelines to define the SDR:
  - For 3390 devices, SDR must be greater than or equal to 8
  - For 3380 devices, SDR must be greater than or equal to 6

For additional information, refer to *z/OS V1R12.0 DFSMSdfp Storage Administration notes*:
http://publib.boulder.ibm.com/infocenter/zos/v1r12/index.jsp?topic=%2Fcom.ibm.zos.r12.idas200%2Fstripvs.htm

## File Unit Parameters

If you enter **V** for the **Use Stored Values** option, the following panel is displayed.

```
---------------------------- File Unit Parameters -----------------------------
Command ===>                                                 Scroll ===> PAGE

Enter values for each file type as required.  You may specify either the
number of volumes or the volume serial(s) for each file type, but not both.
Values specified on the defaults line are used when the corresponding values
for the type of file being allocated are omitted.

 File      Unit   Nbr
 Type      Name   Vols   Volume Serial(s)
-------  -------- ---- ------------------------------------------------------
Default                >        >        >        >        >        >

Archive                >        >        >        >        >        >

Extract                >        >        >        >        >        >

Control                >        >        >        >        >        >

Load                   >        >        >        >        >        >

Press END to return and apply any changes.
Press CANCEL to return without applying any changes.
```

*Figure 187. File Unit Parameters*

The File Unit Parameters panel allows you to view or edit information used to allocate a data set for Optim generated files, that is Archive, Extract, Control, and Load Files. You can specify information for multiple volumes as a default or individually for each file type.

For more information see, “File Unit Parameters” on page 381.

## Defaults for Specific Files

The following chart lists each external file type and its defaults. The modifiable defaults are shown in *Italics*. Other values cannot be modified.

| External File | DSORG | Dir Blks | RECFM | LRECL | BLKSIZE |
|---|---|---|---|---|---|
| Archive File | SEQ | 0 | FSA | a | *0* |
| Batch Archive File search data set | SEQ | 0 | FB | 80 | *3200* [e] |
| Control File or Extract File | SEQ | 0 | FS | a | *0* |
| Compare File | SEQ | 0 | FBS | a | *0* |
| Archive Index File (Dense Index) | VSAM Linear [b] | n/a | n/a | n/a | n/a |
| Point-and-Shoot/Export/ Output SQL | SEQ | 0 | FB | 80 | *3200* |
| Point-and-Shoot/Export/ Output SQL | PDS | *0* [c] | FB | 80 | *3200* |
| Browsed File (Archive, Control or Extract) | SEQ | 0 | VB | *0* | *0* [d] |
| Compare Report File | SEQ | 0 | VB | *0* | *0* [d] |
| Process Report Output | SEQ | 0 | FB | 132 | *1320* [e] |
| Access Edit or Browse Report | SEQ | 0 | FBM | [f] | [f] |

**Note:**

- [a] LRECL=BLKSIZE; minimum is 7944. (This value is based on the number of columns.)
- [b] To release unused space, the file must be SMS-managed using SMS 1.5 and allocated in extended format using an SMS dataclass with DSNTYPE=EXT.
- [c] If DSORG=PDS, Dir Blks must be a value other than 0.
- [d] BLKSIZE>=LRECL+4; min LRECL=84; min BLKSIZE=88.
- [e] BLKSIZE=multiple of LRECL.
- [f] LRECL and BLKSIZE values are determined at run time.

### About Point-and-Shoot File

The Point-and-Shoot file, by default, is allocated as a sequential file with a RECFM of FB. To allocate this file as a member of a PDS, you must specify a value other than 0 for directory blocks.

For more detailed information, refer to the IBM Job Control Language documentation for your operating system.

## Allocating Image Copy Data Sets

To archive, extract or compare data from DB2 image copy data sets in multiple partitions of the same tablespace stored on the same tape volume, you must manually edit the JCL to allocate the data sets. Multiple image copy data sets cataloged on the same tape volume can not be allocated using dynamic allocation. This is a z/OS limitation. If you attempt to use dynamic allocation, the archive, extract, or compare process fails with a dynamic allocation error.

1. If you are using Optim v7.1 or earlier, apply the provided fix: http://www.ibm.com/support/docview.wss?uid=isg1PM52820#more

   Fixed component name: OPTIM MOVE FOR

   Fixed component ID: 5655V0700

   Applicable component levels: R710 PSY UK78105 and UP12/04/21 P F204
2. For image copy data sets of one or more partitioned tablespaces on the same tape, either:
   - concatenate image copy data sets under one DD Name of your choice, or
   - code individual DD Names for allocation of image copy data sets. Use a pattern, such as *xxxxPPPP* where *xxxx* is the DD Name prefix and *PPPP* is the maximum number of partitions in that tablespace. For example, if a tablespace contains 5 partitions, the DD Name would be IMGC0001, IMGC0002, and so on.

Example:

| Table | Partition | Tape Allocation Requirement |
|---|---|---|
| SALES | TAPEVL1 - LABEL(1,SL) | // T01$0001 DD DISP=OLD,DSN=(dsname) IC Dataset for SALES Table |
| CUSTOMER | PART1 TAPEVL1 - LABEL(2,SL) | // T02$0001 DD DISP=OLD,DSN=(dsname) IC Dataset for CUSTOMER Table Part #1 |
| CUSTOMER | PART2 TAPEVL1 - LABEL(3,SL) | // T02$0002 DD DISP=OLD,DSN=(dsname) IC Dataset for CUSTOMER Table Part #2 |

| Table | Partition | Tape Allocation Requirement |
|---|---|---|
| CUSTOMER | PART3 TAPEVL1 - LABEL(4,SL) | // T02$0003 DD DISP=OLD,DSN=(dsname) IC Dataset for CUSTOMER Table Part #3 |
| CUSTOMER | PART4 TAPEVL1 - LABEL(5,SL) | // T02$0004 DD DISP=OLD,DSN=(dsname) IC Dataset for CUSTOMER Table Part #4 |
| CUSTOMER | PART5 TAPEVL1 - LABEL(6,SL) | // T02$0005 DD DISP=OLD,DSN=(dsname) IC Dataset for CUSTOMER Table Part #5 |
| ORDERS | TAPEVL1 - LABEL(7,SL) | // T03$0001 DD DISP=OLD,DSN=(dsname) IC Dataset for ORDERS Table |
| DETAILS | TAPEVL2 - LABEL(1,SL) | JCL allocation not required because it resides on a different tape |

# Appendix C. Create a Row List File

The Point-and-Shoot facility is used with Archive, Compare, or Move to select the primary key values for rows in the Start Table when you want to extract data from a database or restore it to a database. These values are automatically stored in a Row List file.

The Point-and-Shoot facility is available for DB2 tables only. If the data does not reside in DB2, you can create a Row List file with ISPF or a similar editor. The file must conform to the format generated by Point-and-Shoot, as follows:

- The file must be a member of a partitioned dataset or a sequential dataset.
- The file must contain 80-character records. (Note that any line numbers in positions 73-80 are interpreted as data. Do not use line numbers.)
- The data in the file must conform to DB2 syntax and rules for column data types. Additionally, the data must match, by data type and length, the attributes of the primary key column(s) in the Start Table.
- The DD name of the file must be PSDFPNS.

## Example

For example, assume you want to archive or extract specific rows from the DETAILS table using a list created by a method other than Point-and-Shoot. The primary key for the DETAILS table consists of two columns, ORDER_ID and ITEM_ID. These columns are defined as follows:

| Column Name | Data Type |
|---|---|
| ORDER_ID | DEC(5,0) |
| ITEM_ID | CHAR(5) |

The following figure shows an example of the structure of the list. In the example, the file containing the list is displayed in the ISPF editor. The value in the ORDER_ID column is followed by the value in the ITEM_ID column for each row. Commas separate the column values for each row, and semicolons separate each row.

```
EDIT ---- FOPDEMO.FOPPK.KEY ------------------------- COLUMNS 001 072
COMMAND ===>                                          SCROLL ===> CSR
****** ********************** TOP OF DATA *************************
000001 00123, 'CH001'; 00124, 'CH002'; 00125, 'CH003'; 00126, 'CH004';
000002 00133, 'CH001'; 00134, 'CH002'; 00135, 'CH003'; 00146, 'CH004';
000003 00153, 'CH001'; 00154, 'CH002'; 00155, 'CH003'; 00156, 'CH004';
****** ********************** BOTTOM OF DATA ************************
```

*Figure 188. Primary Key Values*

The format requirements are:

- Values must be separated by commas (,) and one or more spaces.
- Primary key values for each row must be separated by semicolons (;) and one or more spaces.

## Character Data

- Character data must be enclosed in single quotes (e.g., ‘abc'). Embedded quotes must be in the form of two single quotes.

Character data in a column can be wrapped to the next line. Each portion of the data must be enclosed in quotes, without an intervening semicolon. For example:

**‘Data for Row 1'**
**‘Col 1', ‘Data for'**
**‘Row 1 Col 2'**

The value for a column exceeds the length of the line and wraps to the next line. Each portion of a value is enclosed in single quotes (‘ '). Commas separate column values.

**‘Data for Row 1 Col 1',**
**‘Data for Row 1 Col 2';**
**‘Data for Row 2 Col 1',**
**‘Data for Row 2 Col 2';**

The value for a column fills the entire line length. Each column value is enclosed in single quotes (‘ '). Commas separate column values. Semicolons (;) separate column values for each row.

Character data that is stored in fixed-length columns is truncated or padded appropriately to fit the column. Character data that is stored in variable-length columns is truncated if necessary, but is not padded.

## Date/Time

- All date and time data must be enclosed in single quotes. Any valid DB2 format for these values is acceptable and is handled appropriately.

## Numeric Data

- Numeric data is not enclosed in quotes. A decimal can be indicated by either a comma or a period, and it is handled appropriately.

## Partial Primary Key

You can use this information to create a Point-and-Shoot file to specify an alternate key or a partial primary key. This is very useful for selecting non-unique values or values that do not correspond to the primary key. To indicate that the data in a file contains values for some set of the columns, prefix the file with:

COLUMN-LIST

Names of the columns for which data is supplied

END-COLUMN-LIST

The data for the columns is specified in the order in which the columns are listed.

**Example 1:** Assume you have a set of ITEMS rows that are not in your DB2 database. However, you want to extract the DETAILS rows from your DB2 database for specific ITEMS. The primary key for the DETAILS rows is comprised of two columns: ORDER_ID and ITEM_ID. However, you prefer to extract rows based on the ITEM_ID only. You can generate a file to use as the Point-and-Shoot file to extract the several DETAILS rows for each specified ITEM_ID value, regardless of the ORDER_ID value.

```
EDIT ---- FOPDEMO.FOPPK.KEY --------------------------- COLUMNS 001 07
COMMAND ===>                                            SCROLL ===> CSR
****** ********************** TOP OF DATA ************************
000001 COLUMN-LIST
000002  ITEM_ID
000003 END-COLUMN-LIST
000004 'CH001'; 'CH002'; 'CH003'; 'CH004';
000005 'CH005'; 'CH006'; 'CH007'; 'CH008';
000006 'CH009'; 'CH010';
****** ********************** BOTTOM OF DATA ************************
```

*Figure 189. Specify Partial Primary Key - Example 1*

**Example 2:** Assume that two columns comprise the partial primary key: ORDER_ID and ITEM_ID. Specify the column values in the order they are listed. For each row in this example, the values for ORDER_ID are followed by the values for ITEM_ID, separated by commas. Semicolons separate the rows. Since commas and semicolons delimit the values, entries can span multiple lines and multiple entries can be specified on a single line.

```
EDIT ---- FOPDEMO.FOPPK.KEY --------------------------- COLUMNS 001 07
COMMAND ===>                                            SCROLL ===> CSR
****** ********************** TOP OF DATA ************************
000001 COLUMN-LIST
000002  ORDER_ID
000003  ITEM_ID
000004 END-COLUMN-LIST
000001 00123, 'CH001'; 00124, 'CH002'; 00125, 'CH003'; 00126, 'CH004';
000002 00133, 'CH001'; 00134, 'CH002'; 00135, 'CH003'; 00146, 'CH004';
000003 00153, 'CH001'; 00154, 'CH002'; 00155, 'CH003'; 00156, 'CH004';
****** ********************** BOTTOM OF DATA ************************
```

*Figure 190. Specify Partial Primary Key - Example 2*

## Use the Row List File

Do the following to use a Row List file for a process:

1. For an Archive or Extract Process, choose Option 1 TABLES from the **EXTRACT Process** or **ARCHIVE Process** menu to display the Select Tables/Views for AD panel.

   For a Compare Process that uses an Access Definition as one or both sources, specify that the Access Definition is to be modified and display the Select Tables/Views for AD panel.

   For an Archive Selective Restore Process, choose Option 3 SELECTION CRITERIA to display the Archive Selection Criteria - Tables panel.
2. Enter the POINT command on the panel to display a pop-up window that will prompt you for an Input DSN. Specify the name of the Row List file you created and press END. When the process is performed, that file is used to select rows from the Start Table.

# Appendix D. Compatibility Rules

When you define an Optim relationship or a Column Map, corresponding columns must be compatible.

For relationships, column compatibility rules are the same for all Optim components. There are differences in the compatibility rules for Column Maps, depending on the Optim component with which you plan to use a Column Map.

## Relationship Column Compatibility

The following list shows compatibility for each column data type when you are creating or modifying a relationship.

The columns listed are compatible and may be defined as corresponding columns. Here, the term "column" refers to actual columns, as well as expressions containing concatenation, substrings, and literals.

**Numeric**

- Numeric
- CHAR [2]
- VARCHAR [2]
- Numeric Constant [3]

**CHAR**

- Numeric [2]
- CHAR [4]
- VARCHAR [4]
- String Literal
- String Expression[1]
- CLOB [4]

**VARCHAR**

- Numeric [2]
- CHAR [4]
- VARCHAR [4]
- String Literal
- String Expression[1]
- CLOB [4]

**DATE**

- DATE

**TIME**

- TIME

**TIMESTAMP**

- TIMESTAMP

**TIMESTAMP WITH TIME ZONE**

- CHAR
- VARCHAR

- DATE
- TIME
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE

**StringLiteral**

- CHAR
- VARCHAR
- String Expression[1]

**Graphic Literal**

- DBCLOB
- VARGRAPHIC

**Numeric Constant**

- Numeric [3]

**String Expression[1]**

- CHAR
- VARCHAR
- String Literal
- String Expression[1]

**Binary Expression**

- BINARY
- VARBINARY

**CLOB**

- CHAR [4]
- VARCHAR [4]
- CLOB [4]

**DBCLOB**

- Graphic Literal
- DBCLOB
- GRAPHIC
- VARGRAPHIC

**GRAPHIC**

- Graphic Literal
- DBCLOB
- GRAPHIC
- VARGRAPHIC

**VARGRAPHIC**

- Graphic Literal
- DBCLOB
- GRAPHIC
- VARGRAPHIC

**BINARY**

- Binary Literal
- BINARY

- VARBINARY

**VARBINARY**

- Binary Literal
- BINARY
- VARBINARY

**Note:** [1] "String Expression" is any comparand containing a concatenation operator or a substring function.

[2] The "Numeric" column must be an INTEGER, SMALLINT, BIGINT column or a DECIMAL column with a scale of 0 (the numeric value cannot have digits after the decimal point).

[3] The "Numeric" column must be an INTEGER, SMALLINT, BIGINT, or DECIMAL column.

[4] If the parent column is for mixed data, the child column must also be for mixed data.

### Verification

When you press ENTER on the panel to define or modify the relationship, Optim verifies that corresponding columns are compatible. If they are not, an error message is displayed. You must correct the incompatibility before saving the relationship definition.

### Comparing Character to Numeric Columns

Character columns are compatible with numeric columns when defined in an Optim relationship. To compare character data to numeric data, it may be necessary to convert the numeric data to character format. This may involve padding the data prior to the comparison. Optim pads the data with beginning zeros so the two strings are equal in length. This operation is performed internally and does not affect your DB2 data. To demonstrate, assume the following columns are defined in a relationship.

```
COL1  CHAR(5)   COL2  DECIMAL(2)
```

If the data in COL2 is 43, Optim pads the value with three leading zeros to form "00043," prior to comparing that entry to the data in COL1, which is five characters long. This automatic padding may affect how Optim retrieves related rows of data.

Character-to-numeric conversion will occur only if the character column contains all digits.

### Comparing Different Length Columns

Corresponding CHAR and VARCHAR columns do not require the same length attribute. However, if the length of one column exceeds the length of its corresponding column, Optim truncates any trailing blanks until the column is equal in length to the corresponding column. If the data in the longer column does not have enough trailing blanks to truncate, that data is not considered related and is not retrieved. Truncation also occurs when the length of a concatenated expression exceeds the length of its corresponding column.

## Column Map Compatibility

When you create a Column Map to use with Archive or Move, you can map columns with compatible attributes, including the use of literals, constants, expressions, exit routines, and special registers. Compare allows you to map columns with compatible attributes only. Literals, constants, expressions, exit routines, and special registers can not be used. Column Maps are not used with Access.

The columns listed are compatible and may be defined as corresponding columns. Here, the term "column" refers to actual columns, as well as expressions containing concatenation, substrings, and literals.

**Numeric**

- Numeric
- CHAR [2]
- VARCHAR [2]
- Numeric Constant [3]

**CHAR**

- Numeric [2]
- CHAR
- VARCHAR
- String Literal
- String Expression1
- Graphic Literal [4]
- String Expression1
- CLOB
- DBCLOB [4]
- GRAPHIC [4]
- VARGRAPHIC [4]

**VARCHAR**

- Numeric [2]
- CHAR
- VARCHAR
- String Literal
- String Expression1
- Graphic Literal [4]
- String Expression1
- CLOB
- DBCLOB [4]
- GRAPHIC [4]
- VARGRAPHIC [4]

**DATE**

- DATE
- TIMESTAMP

**TIME**

- TIME
- TIMESTAMP

**TIMESTAMP**

- DATE
- TIME
- TIMESTAMP

**StringLiteral**

- CHAR

- VARCHAR
- String Expression1

**Binary Literal**

- BINARY
- VARBINARY

**Numeric Constant**

- Numeric [3]

**String Expression1**

- CHAR
- VARCHAR
- String Literal
- String Expression1

**Binary Expression**

- BINARY
- VARBINARY

**BLOB**

- BLOB
- CLOB

**CLOB**

- CHAR
- VARCHAR
- BLOB
- CLOB
- DBCLOB [4]
- GRAPHIC [4]
- VARGRAPHIC [4]

**DBCLOB**

- DBCLOB
- GRAPHIC
- VARGRAPHIC

**GRAPHIC**

- Graphic Literal
- DBCLOB
- GRAPHIC
- VARGRAPHIC

**VARGRAPHIC**

- DBCLOB
- GRAPHIC
- VARGRAPHIC

**BINARY**

- Graphic Literal
- BINARY
- VARBINARY

**VARBINARY**

- Graphic Literal
- BINARY
- VARBINARY

**DECFLOAT**

- DECFLOAT [5]

**Note:**

[1] "String Expression" is any comparand containing a concatenation operator or a substring function.

[2] The "Numeric" column must be an INTEGER, SMALLINT or BIGINT column or a DECIMAL column with a scale of 0 (the numeric value cannot have digits after the decimal point).

[3] The "Numeric" column must be an INTEGER, SMALLINT, BIGINT or DECIMAL column.

[4] Only "mixed" source data can be inserted. Data cannot be compared.

[5] A DECFLOAT column can be mapped only to a column with matching data type and precision.

## Move and Archive Column Map Compatibility

Optim converts source data when necessary.

### Character Data

Optim converts source data in character format to any character data type for the destination column, padding or truncating the source data to fit the destination column. It also converts a valid numeric value in character format to a numeric format if the length of the converted value is accommodated in the destination column.

### Numeric Data

Optim converts source data in a numeric format to any numeric or character data type in a destination column, truncating decimal positions, if necessary.

A destination column with character data type must be defined with a minimum width of five, CH (5), when converting single or double floating point data types.

If significant positions for numeric data would be truncated by the transformation, the transformation is not performed and a conversion error results.

### Date, Time, or Timestamp Data

Optim cannot convert date, time, or timestamp values to a character or numeric data type. Conversely, character or numeric data types cannot be converted to date, time, or timestamp data types.

Optim can convert a timestamp value in a source column to a time or date, dropping the extraneous portion of the timestamp, or convert a date or time value to a timestamp. The time portion of the destination timestamp is filled with zeros, when converting a date, or the date portion with the current date, when converting a time.

### Special Registers

Optim supports the DB2 special registers, CURRENT TIME or CURRENT_TIME, CURRENT DATE or CURRENT_DATE, CURRENT TIMESTAMP or CURRENT_TIMESTAMP, CURRENT SQLID or CURRENT_SQLID, and USER as source data specifications.

The Optim special register, CURRENT TSOID, is also available. This special register contains the TSO ID of the current user.

## Compare Column Map Compatibility

The Compare rules for Column Maps allow you to map columns with dissimilar name and columns with compatible data types.

Compatible data types are determined as follows:

- Character data types CHAR, VARCHAR, and LONGVARCHAR can be compared, regardless of length. The shorter value is padded with blanks before the comparison. When browsing a Compare File, the longer data type is assumed.
- Character data types GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC can be compared. The shorter value is padded with blanks before the comparison. When browsing a Compare File, the longer data type is assumed.
- BINARY and VARBINARY can be compared. Binary values are padded with binary zeroes. When browsing a Compare File, the longer data type is assumed.
- INTEGERS, SMALLINT, and BIGINT columns can be compared. When browsing the Compare File, the data is displayed as INTEGER.
- DECIMALS of any scale and precision can be compared. When comparing DECIMALS of different scale, precision, or both, the larger in each dimension is used and displayed when browsing the Compare File. The resulting number of digits cannot exceed 31. For example, if comparing a column DEC(4,2) with a column DEC(4,4), the resulting dimension and precision is DEC(6,4).
- SINGLE PRECISION FLOATING POINT and DOUBLE PRECISION FLOATING POINT columns can be compared. When browsing the Compare File, the values are displayed as DOUBLE PRECISION FLOATING POINT.
- DECFLOAT columns can be compared only to DECFLOAT columns with the same precision.
- Comparisons cannot be performed on LOB or XML data.
- A DATE, TIME, TIMESTAMP, and TIMESTAMP WITH TIME ZONE data type column can be compared only with a column of the same data type.

# Appendix E. Optim Exit Parameter Lists

Before compiling each Security Exit, Column Map Exit, Archive Exit, or API, locate and edit the corresponding parameter list, indicated by the DSect name.

In the parameter list, locate the specified field or fields and edit as indicated (to change the length for any fields except PDP-STRUCTID). After saving the updates, proceed to recompile the exit. Value shown for "Old Length" represents the length of the field for releases prior to Release 5.5.3.

```
Exit Name:  PSXOSCxx
Language:   Assembler
DSECT Name: XPARMS
Field Name: XPLEV1
Old Length:  18
New Length: 128

Exit Name:  PSXOSCxx
Language:   Assembler
DSECT Name: XPARMS
Field Name: XPLEV2
Old Length:  18
New Length: 128

Exit Name:  PSXOSCxx
Language:   Assembler
DSECT Name: XPARMS
Field Name: XPLEV3
Old Length:  18
New Length: 128

Exit Name:  PSXOSCxx
Language:   Assembler
DSECT Name: XPARMS
Field Name: XPSQLID
Old Length:   8
New Length: 128

Exit Name:  PSXFSCxx
Language:  Assembler
DSECT Name: RTPARMS
Field Name:  SQLID
Old Length:   8
New Length: 128

Exit Name:  PSXASCxx
Language:   Assembler
DSECT Name: XPARMS
Field Name:  XPSQLID
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name:  I_OPTIMREL
Old Length: --
New Length: 4

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name:  I_SQLID
```

```
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name:  I_SCRIDV
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name: I_STBNMV
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name:  I_DCRIDV
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   Assembler
DSECT Name: IDENT
Field Name:  I_DTBNMV
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: PARMAREA
Field Name:  [1]PARMOPTIMREL
Old Length: --
New Length: 4

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: PARMAREA
Field Name: PARMSQLID
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: PARMAREA
Field Name: PARMSCRIDV
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: PARMAREA
Field Name: PARMSTBNMV
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: PARMAREA
Field Name: PARMDCRIDV
Old Length:   8
New Length: 128

Exit Name:  Column Map Exit
```

```
Language:   COBOL
DSECT Name: PARMAREA
Field Name: PARMDTBNMV
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: COLWKAREA
Field Name: CW-TABLE
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: COLWKAREA
Field Name: FILLER
Old Length: 208
New Length:  98

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: ROWWKAREA
Field Name: RW-TABLE
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: ROWWKAREA
Field Name: FILLER
Old Length: 208
New Length:  98

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: TBLWKAREA
Field Name: TW-TABLE
Old Length:  18
New Length: 128

Exit Name:  Column Map Exit
Language:   COBOL
DSECT Name: TBLWKAREA
Field Name: FILLER
Old Length: 208
New Length:  98

Exit Name:  SAMPAPI2/SAMPAPI3
Language:   COBOL
DSECT Name: PARMLIST
Field Name: 2PDP-STRUCTID
Old Length: 8
New Length: 8

Exit Name:  SAMPAPI2/SAMPAPI3
Language:   COBOL
DSECT Name: PARMLIST
Field Name: PDP-TABLE
Old Length:  18
New Length: 128

Exit Name:  SAMPAPI2/SAMPAPI3
Language:   COBOL
DSECT Name: PARMLIST
Field Name: PDP-CREATOR
Old Length:   8
```

```
New Length: 128

Exit Name:  SAMPAPI2/SAMPAPI3
Language:   COBOL
DSECT Name: PARMLIST
Field Name: PDP-SQLID
Old Length:   8
New Length: 128

Exit Name:  SAMPAPI2/SAMPAPI3
Language:   COBOL
DSECT Name: PARMLIST2
Field Name:
Old Length: 342
New Length: 692
```

[1]PARMOPTIMREL should be added after PARMSSN in the DSECT.

[2]Change value of PDP-STRUCTID from API00100 to API00200

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing*
*IBM Corporation*
*North Castle Drive*
*Armonk, NY 10504-1785*
*U.S.A.*

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing*
*Legal and Intellectual Property Law*
*IBM Japan Ltd.*
*1623-14, Shimotsuruma, Yamato-shi*
*Kanagawa 242-8502 Japan*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation*
*Software Interoperability Coordinator*
*Director of Engineering, Information Management*
*111 Campus Drive*
*Princeton, NJ 08540*
*USA*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Java™ is a trademark of Sun Microsystems, Inc.

Linux is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of The Open Group.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other company, product, or service names may be trademarks or service marks of others.

# Index

## Special characters



## A



## B

# C

# D



# E

# P



# R



# S

# T



# U



# V



# W



# Z

Printed in USA