CA IDMS™

Logical Record Facility Guide

Release 18.5.00

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Introduction to the Logical Record Facility

This section contains the following topics:

Introduction

The Logical Record Facility (LRF) is a runtime facility that allows application programmers to access CA IDMS data without having to know the physical structure of the database. Under LRF, programmers do not have to use database navigation statements to access information. This is because the DBA predefines database access logic that is typically coded by programmers.

Advantages: LRF offers many advantages for the corporate information system:

- Enhances runtime efficiency. Database access through LRF often requires less operating-system overhead than database access through navigational DML commands. LRF can save overhead by reducing the number of program calls.

  For batch programs running under the CA IDMS central version (CV), LRF can also minimize supervisor calls (SVCs). This results in faster and more efficient database access.
- Allows for increased data integrity. With LRF, the DBA can write all database navigation instructions in the subschema. This helps to ensure that the logical relationships of the data are preserved.
- Allows for data security. The DBA can use LRF to:
  - Restrict the records and fields viewed by the application program
  - Restrict the database record occurrences viewed by the application program
  - Restrict the operations that the application program can perform on records and fields
- Provides a flexible way to present data to different application programs. With LRF, the DBA can use standard relational operations to:
  - Select the record occurrences that the program can access
  - Project the fields that appear in the program's view

- Join together information from two or more database records
- Compute new fields based on existing field values

These relational operations let the DBA establish relationships that do not exist in the schema. They also let the DBA tailor logical views of data to individual programs. No matter how the DBA chooses to construct a view, the application program will see the data as a single table.

- Simplifies a program's access to the database. LRF eliminates the need for programmers to learn the database structure. By using LRF, a programmer does not have to be familiar with database navigation techniques or keep track of database currency.
- Facilitates program maintenance. Because LRF insulates application programs from the database, changes to the logical and physical database structures have minimal impact on existing programs. For example, the DBA can change selection criteria for a record, and the program need not be recompiled
- Reduces programming redundancy. Because all database navigation instructions are placed in a path, the programmer does not have to code these instructions. Applications that require similar information can access the path rather than issue the database navigation statements themselves.

To use LRF successfully, you must understand your role as the DBA in defining logical-record subschemas. You must also understand how the applications programmer will use the subschemas that you define.

The DBA's Role

As the database administrator (DBA), you control a program's access to the database by defining a subschema that contains one or more **logical records**. A logical record is a logical grouping of fields selected from one or more database records.

For example, you might want to use a logical record in the following business situation. Suppose that application programs that process employee information frequently require similar information. This information often comes from different database records. Some programs, for example, require an employee's id, name, start date, and status, as well as information about that employee's department and office.

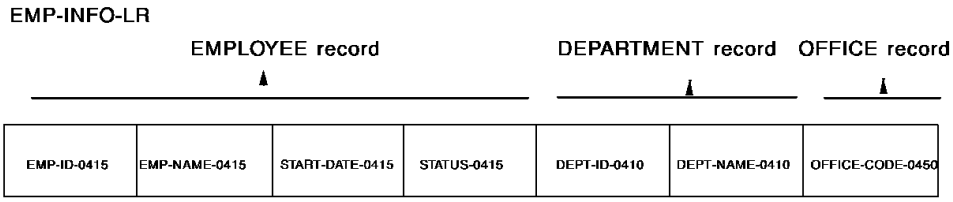
Defining the logical record

By using LRF, you can create a logical record that accesses all of the information required by an application program. For example, you can define a logical record called EMP-INFO-LR, which contains selected fields from the EMPLOYEE, DEPARTMENT, and OFFICE records.

Once the EMP-INFO-LR logical record is defined, a program can issue the following logical-record request:

```
OBTAIN FIRST EMP-INFO-LR
  WHERE EMP-ID-0415 EQ '0015'.
```

As you can see from the diagram below, EMP-INFO-LR presents a view of three different database records to the application program.



The following example compares two program requests for employee information: program A requests information through the logical record facility; program B requests information by using navigational DML statements.

Program A

```
OBTAIN FIRST EMP-INFO-LR
    WHERE EMP-ID-0415 EQ '0015'.
```

Program B

```
MOVE '0015' TO EMP-ID-0415.
OBTAIN CALC EMPLOYEE.
IF DEPT-EMPLOYEE MEMBER
     OBTAIN OWNER WITHIN DEPT-EMPLOYEE.
FIND CURRENT EMPLOYEE.
IF OFFICE-EMPLOYEE MEMBER
     OBTAIN OWNER WITHIN OFFICE-EMPLOYEE.
```

Defining paths and path groups

Once you have defined the EMP-INFO-LR logical record, you can define paths and path groups for the logical record. A **path** is a collection of Data Manipulation Language statements (DML statements) that are designed to process program requests for database access. Paths are grouped into **path groups**, according to the DML verb used to access the path.

The following example shows how you can associate paths with the EMP-INFO-LR logical record. The EMP-INFO-LR logical record is associated with paths that are grouped into a single path group. This logical record is defined in the EMPLR35 subschema.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   .
   .
   .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR                                  ┐
  ELEMENTS ARE                                                      │
     EMPLOYEE                                                       │
     DEPARTMENT                                                     │ Logical
     OFFICE                                                         │ record
  COMMENTS                                                          │
     '*************************************************************  │
     'THE EMP-INFO-LR RECORD ACCESS INFORMATION FROM THE'           │
     'EMPLOYEE DATABASE RECORD AND ALSO ACCESS INFORMATION'         │
     'FROM THE ASSOCIATED DEPARTMENT AND OFFICE RECORDS.'           │
   .                                                                ┘
   .
   .
ADD
PATH-GROUP NAME IS OBTAIN EMP-INFO-LR                               ┐
                                                                    │
  SELECT FOR KEYWORD ON-LEAVE                ┐                     │
  OBTAIN EACH EMPLOYEE WITHIN EMP-NAME-NDX   │                     │
     WHERE STATUS-0415 EQ '04'               │                     │
  IF DEPT-EMPLOYEE MEMBER                    │ Path                │
     ON 1601 RETURN NO-DEPT                  │                     │
  OBTAIN OWNER WITHIN OFFICE-EMPLOYEE        ┘                     │
                                                                    │ Path group
  SELECT FOR FIELDNAME-EQ EMP-ID-0415        ┐                     │
   .                                         │ Path                │
   .                                         │                     │
   .                                         ┘                     │
                                                                    │
  SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT ┐              │
   .                                                 │ Path         │
   .                                                 │              │
   .                                                 ┘              ┘
```

For more information on defining logical records, path groups, and paths, refer to Chapter 4, Starting to Define the Subschema, Chapter 5, Defining Path Groups, and Chapter 6, Specifying Path Selectors.

How The Programmer Uses LRF

Logical-record requests

Application programmers request services from LRF by issuing special logical-record requests. Each of these requests consists of a DML verb and the name of the desired logical record:

- **OBTAIN logical-record** should be used to request that a logical record be retrieved.
- **MODIFY logical-record** should be used to request that a logical record be modified.
- **STORE logical-record** should be used to request that a logical record be stored.
- **ERASE logical-record** should be used to request that a logical record be erased.

When a programmer issues a logical-record request, it maps to a path defined in the subschema. The outcome of the request depends on how the path is coded. For example, a MODIFY logical-record request may or may not cause a logical record to be changed.

Request options

Programmers can include the following options in a logical-record request:

- **Specific selection criteria**, in the form of a program WHERE clause. This clause requests that logical-record occurrences be selected according to specified boolean selection criteria.
- **A request to check the outcome of LRF processing**, in the form of an ON clause. This clause specifies an action to be taken based on a status returned from LRF.

For more information on accessing data through LRF, see Chapter 19, LRF Programming Techniques.

Chapter 2: How the Logical Record Facility Works

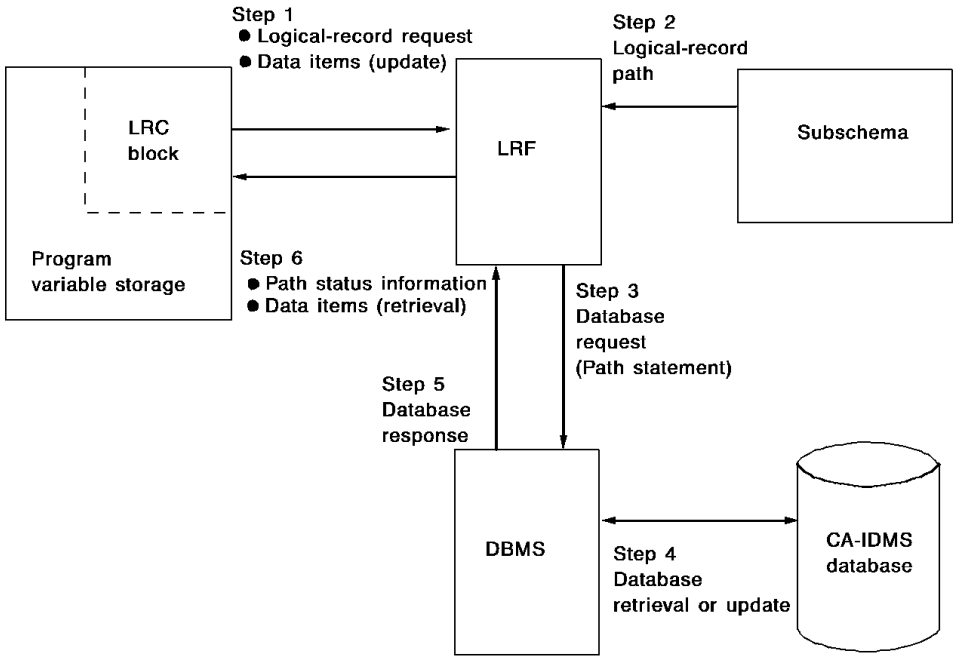
This section contains the following topics:

Overview Of Logical Record Facility Processing

A program request for a logical record is processed in the following manner:

1. **The application program requests the services of LRF** by issuing a logical-record request. The request includes a DML verb (OBTAIN, MODIFY, STORE, or ERASE) and the name of the desired logical record. It can also include a WHERE clause that contains selection criteria for the logical record and data to be used for updating the logical record. All information associated with the logical-record request is sent through the program's logical-record request control (LRC) block.
2. **LRF interprets the program request** by following these steps:
    a. **Locates the appropriate logical-record definition in the subschema**.
    b. **Locates the appropriate logical-record path**. LRF uses the logical-record name and DML verb to match to the path group, and the contents of the program request's WHERE clause (if one exists) to map to the path.
    c. **Determines where to enter the path**. LRF always enters the beginning of the path unless the path is being iterated. For a discussion of path iteration, refer to Chapter 11, Controlling Path Execution.
3. **LRF issues a database access request to the DBMS**, based on the current path-DML statement.
4. **The DBMS executes the path-DML statement**.
5. **The DBMS returns control to LRF**. Steps 3, 4, and 5 are repeated for each path-DML statement until all appropriate statements have been executed.
6. **LRF returns path-status information and data** to the application program. The program regains control at the ON clause (if present) or at the statement that immediately follows the logical-record request.

The following drawing shows how a logical-record request is processed. LRF processes a program's logical-record request according to the steps outlined above. Steps 3, 4, and 5 are repeated until all appropriate path-DML statements in the program have been executed.



Types of paths

There are two types of paths: retrieval and update. These paths are discussed below, followed by discussions of how communication occurs between the application program and LRF, and between LRF and the DBMS.

Processing Retrieval Paths

A **retrieval path** services program requests to OBTAIN a logical record. Typically, the path-DML commands in a retrieval path retrieve each database record that participates in the requested logical record. For example, a retrieval path for the EMP-INFO-LR logical record (described in Chapter 1, Introduction to the Logical Record Facility) would probably contain path-DML commands that retrieve the EMPLOYEE, DEPARTMENT, and OFFICE database records.

When LRF processes a retrieval request, the way it returns the data depends on the environment:

- **For batch programs running under the CA IDMS central version** , LRF returns the entire logical record in one packet of data. The packet is created when the program issues a bind request. It overlays program variable storage when all path-DML commands have been executed. By returning all of the logical-record components in a single packet, LRF reduces the number of SVC calls required for program/database communication.
- **For online programs running under central version and batch programs running in local mode**, LRF returns data directly to program variable storage, one record at a time.

The transfer of data to program variable storage is transparent to the application program. Regardless of the type of processing, the logical record is available to the program when all path-DML commands have been executed successfully and all selection criteria have been satisfied.

Processing Update Paths

An **update path** services program requests to MODIFY, STORE, or ERASE a logical record. To MODIFY or STORE a logical record, data must be made available to the DBMS. LRF passes the required data from program variable storage to the DBMS when a request is issued. The data present in program variable storage can be placed there either by the program or by the execution of a previous path-DML command.

You can code an update path to direct LRF to update some or all of the database records that participate in the logical record. As the DBA, you determine what database records, if any, will be affected by a program's logical-record request.

For example, suppose a program needs to modify the STATUS-0415 field in the EMP-INFO-LR logical record. You can code a path that modifies the EMPLOYEE database record (which contains this field) but does not affect the DEPARTMENT or OFFICE records.

Communication Between The Program and LRF

LRC block

Communication between the application program and LRF occurs through the subschema control block and the **logical-record request control (LRC) block**. When a program issues a request for a logical record, the LRC block passes the logical-record verb, logical-record name, and WHERE clause selection criteria to LRF. Once the request is processed, the LRC block passes path-status information back to the program.

SUBSCHEMA-LR-CTRL

The data description of the LRC block is identified as **SUBSCHEMA-LR-CTRL** in the application program. This description is copied into the program when it is compiled.

Examining fields

After every call to LRF, the program should examine the LR-STATUS field of the LRC block. If the value returned is LR-ERROR, the program should examine the ERROR-STATUS field of the IDMS (or IDMS-DC) communications block, as described below.

Communication Between LRF and The DBMS

Communication between LRF and the DBMS occurs through either the IDMS communications block or the IDMS-DC communications block:

- **The IDMS communications block** is used when the operating mode is either BATCH or BATCH-AUTOSTATUS.
- **The IDMS-DC communications block** is used when the operating mode is either IDMS-DC or DC-BATCH.

SUBSCHEMA-CTRL

The data description of the IDMS or IDMS-DC communications block is identified as **SUBSCHEMA-CTRL** in the application program. SUBSCHEMA-CTRL is copied into the program when it is compiled. The program can use the ERROR-STATUS field of SUBSCHEMA-CTRL to check the outcome of the last path-DML statement executed. This field should only be used if the LR-STATUS field of the LRC block returns LR-ERROR.

For more information on the IDMS and IDMS-DC communications blocks, refer to the *CA IDMS Navigational DML Programming Guide*.

Chapter 3: Preliminary Analysis and Design

This section contains the following topics:

Introduction

The key to using LRF successfully is to analyze the information needs of your organization and to design logical-record subschemas that meet those needs.

Considerations

Before you begin to define a subschema, you should consider:

- The data requirements of the application programs that will use the subschema
- How the data should be secured
- Whether you want the selection of logical-record occurrences to occur in the program or the path
- Whether you want the subschema to be used for LRF processing only, or for both LRF and navigational DML processing
- Whether you want LRF to return complete logical records only, or both partial and complete logical records

Each of these considerations is described below, followed by a list of specific logical-record design suggestions.

Identifying Data-Access Requirements

Steps to determine data requirements

Follow these steps to determine the data requirements for programs that will use the subschema:

1. **Identify all anticipated user requests**. You should determine exactly what data the users require for data processing.
2. **Analyze each anticipated request** to determine the required data elements. Then identify the database records that contain these data elements.
3. **Group together all requests that require the same (or almost the same) data elements**. Each group of requests will be the basis of one logical record.
4. **Analyze each group of requests** identified in step 3 to determine required selection criteria. Requests that use the same selection criteria can be serviced by a single logical-record path.

Establishing priorities

You should use your discretion to determine which requests are most important. Priorities are typically defined by the users and determined by the frequency and type of request.

Example

To illustrate how to identify data-access requirements, let's look at an example from the sample non-SQL defined employee database shown in the following diagram.

Note: Most of the examples in this document are based on the sample non-SQL defined employee database.

Some users frequently require access to the following data elements:

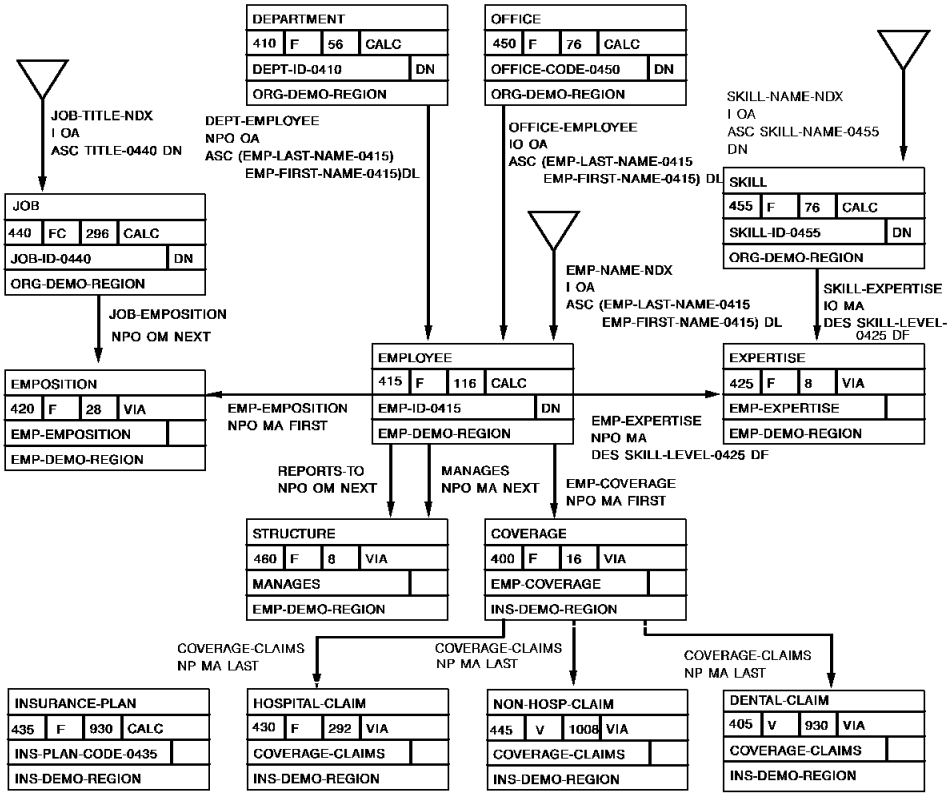
```
FIELD NAME          DESCRIPTION

EMP-ID-0415         employee id
EMP-NAME-0415       employee name
START-DATE-0415     employee start date
STATUS-0415         employee status
DEPT-ID-0440        department id
DEPT-NAME-0440      department name
OFFICE-CODE-0450    office code
```

Because these data elements are frequently accessed together, it is efficient to include them in a single logical record (the EMP-INFO-LR logical record).

Suppose that users want to select this data according to the following criteria:

- For a specific employee, by employee id
- For a specific employee, by employee last name
- For all employees who work in a specific department
- For all employees who work in a specific office
- For all employees who are on leave

To service these requests, you could define five separate paths within the EMP-INFO-LR logical record. Each path would service one type of data selection. For information on defining paths, refer to Chapter 5, Defining Path Groups.



Identifying Data-Security Requirements

Once you have identified and analyzed the data-access requirements of your organization, you should decide how the data should be secured. Using logical records, you can secure data at three levels:

- The subschema level
- The path-group level
- The path level

Securing data at the subschema level

A subschema determines which database records and elements are available to the application programs that use the subschema. When you define a subschema, you can include only those database records and elements that you want the programs to access.

For example, suppose that two groups of application programs have similar data requirements for employee information. One group needs access to the employee salary amount (SALARY-AMOUNT-0420), while the other group is not authorized to access this field.

To secure this salary information from unauthorized users, you could:

- **Define a subschema for each of the two groups**. The first subschema would include the SALARY-AMOUNT-0420 field on an ELEMENTS ARE clause; the second subschema would not include this field.
- **Register application programs for the appropriate subschema** by using the Integrated Data Dictionary (IDD). Program registration is discussed in the *CA IDMS IDD DDDL Reference Guide*.

LR usage mode

Another way you can secure data at the subschema level is by defining a subschema usage mode of LR. This usage mode allows programs to access logical records only; the programs cannot access database records through navigational DML calls. Subschema usage modes are described later in this chapter.

Other forms of subschema-level security include access restrictions for users, areas, sets, and records. This type of security is described in *CA IDMS Database Administration Guide*.

Securing data at the path-group level

By limiting the path groups defined in a subschema, you can control the operations a program can perform on a logical record. If you define a logical record that has an OBTAIN path group only, application programs will be able to retrieve the logical record but not update it.

Similarly, you can allow a program to retrieve a logical record, add a new logical record, and modify an existing logical record by including OBTAIN, STORE, and MODIFY path groups in the logical record definition. The program, however, will not be able to erase the logical record.

Path groups are discussed in Chapter 5, Defining Path Groups.

Securing data at the path level

Selection criteria

You can implement occurrence-level security at the path level by specifying selection criteria within the path. Path selection criteria are evaluated before the selection criteria specified within the program WHERE clause. Therefore, you can use path selection criteria to control the programmer's awareness of the actual data within the database.

For example, suppose a group of application programs require salary information for nonexecutive employees. You can code path selection criteria that request logical-record occurrences where the contents of the EMP-SALARY-0440 field is less than $100,000. The programs will only be able to view those logical records that meet the path selection criteria. Furthermore, the programs will not know that the selection criteria have been applied.

The path can also space or zero out sensitive data before the logical record is returned to the program.

Path selection criteria are discussed in more detail below.

Controlling the Selection Of Logical-Record Occurrences

Determining control

Before you code a logical-record subschema, you should determine how much control you want to maintain over the selection of logical-record occurrences. The amount of control you maintain depends on whether the selection criteria are specified in the program or in the path:

- The programmer controls the selection of logical-record occurrences by coding selection criteria in the program's WHERE clause.
- You control the selection of logical-record occurrences by coding selection criteria in the path.

Logical-record selection criteria

Logical-record selection criteria can be coded in the program, in the path, or in both the program and the path. *You should be aware that path selection criteria are evaluated before program selection criteria*. Therefore, programs never see data that doesn't meet path selection criteria.

Advantages

Selecting logical-record occurrences in the path rather than the program provides the following advantages:

- **You, rather than the programmer, have control over the logical-record occurrences returned to satisfy a program request**. This helps to minimize programmer error because it ensures that all programs that require the same logical-record occurrences will use the same selection criteria. It also provides data security, since programmers need not be aware of the selection criteria specified in the path.
- **You can change selection criteria defined in the path without requiring that programs be modified and recompiled**. This is especially useful for selection criteria that change often.

  For example, the MINIMUM-SALARY and MAXIMUM-SALARY fields of the JOB record are typically subject to frequent change. Any selection of JOB records based on salary levels should be placed in the path. That way, when the salary levels change, application programs do not have to be recompiled.
- **There is less data transfer activity involved**. A record occurrence that doesn't meet path selection criteria is not moved from the DBMS buffers to program variable storage.

Determining Subschema Usage Modes

Before you code a logical-record subschema, you should decide what usage mode you want the subschema to have. The usage mode determines the types of requests that a program using the subschema can issue:

LR usage mode

A usage mode of LR specifies that the program can issue logical-record requests only. Programs that access logical records only do not have to perform database navigation or be aware of currency. Therefore, *for most subschemas, you should specify a usage mode of LR*.

When defining logical-record components for these subschemas, make sure you include all of the database records that are required by the programs. Because the programs cannot retrieve database records directly, the required data must be part of the logical record.

MIXED usage mode

A usage mode of MIXED specifies that the program can issue requests for both logical records and database records. Programs that use the subschema can access database records through logical records *and* they can use navigational DML commands to access database records directly.

These programs are also more dependent on the database structure. Therefore, the programmer must be aware of database currencies.

Note: A usage of MIXED should be specified only if it is *absolutely necessary* for programs using the subschema to issue navigational DML commands.

Controlling How LRF Returns Data

Before you code a logical-record subschema, you need to decide how you want LRF to return data to the program. Whenever possible, LRF tries to construct a complete logical record. This means that LRF will return data for all components of the logical record, as specified by the path.

LR-NOT-FOUND

If LRF cannot construct a complete logical record, it automatically returns a path status of LR-NOT-FOUND. It also returns any data it was able to retrieve.

Accessing partial data

If you want a program to have access to the partial data, you should code your own path statuses to alert the program that a partial logical record is being returned. You should also initialize the unused portions of the logical record to ensure that the returned data is accurate.

For more information on complete and partial logical records, refer to Chapter 11, Controlling Path Execution.

Logical-Record Design Suggestions

There are a number of steps you can take to develop a good logical-record design:

- Customize each logical record to service one category of information.
- Customize each logical-record path to service one type of program request.
- Provide efficient database navigation in the path.
- Define path selectors carefully.

Customizing a logical record

Each logical record you construct should contain only one category of information. For example, the sample EMP-INFO-LR logical record is designed to service program requests for employee information. All programs that require the same same (or similar) employee information will request this logical record.

Too global

A logical record that is too global often processes and returns data elements that are not required by the program. This can result in:

- Wasted space in program variable storage
- Unnecessary resources being allocated to maintain buffer areas for each logical record component
- An increased load module size for the subschema
- An excessive number of unrelated paths in the subschema

Too sparse

A logical record that is too sparse may return fewer data elements than a program requires. One program execution must then access many logical records, resulting in:

- More complex LRF design and coding activity
- Increased program-LRF communication, with a corresponding increase in operating system overhead
- Increased programmer awareness of currency
- A reduction in data structure independence

Customizing a logical-record path

Each logical-record path you define should service only one type of program request. In general, the primary purpose of an application program is to perform either retrieval or update activity. Thus, the subschemas you code should contain the retrieval or update functions needed to perform the required activity. (A subschema designed for update activity will usually contain both retrieval and update functions.)

You should always use a single path to retrieve an entire table. Whenever possible, you should also use the same program WHERE clause to access this path.

If you do not use the same program WHERE clause, you force a program to switch paths to get the information it requires. This can cause a 2040 error, which occurs when the WHERE clauses in successive OBTAIN statements direct LRF to different paths.

Navigating the database efficiently

Considerations

When you code logical-record paths, you should navigate the database in the most efficient manner possible. Efficient database navigation depends on the following factors.

Entering the database

You can enter the database by using the following access strategies:

- **CALC**, based on a record's CALC key value
- **DIRECT**, based on a record's database key value
- **INDEXED**, based on a system-owned index or an indexed set
- **AREA**, based on a record's physical location in a database area

Your choice of an access strategy will depend on two factors:

- The information needs of the program
- The structure of the database

In general, CALC, DIRECT, and INDEXED entries are more efficient than AREA entry.

Progressing through the database

Once the database is entered, the accessed record becomes current of run unit, area, record type, and all sets in which it participates as an owner or member.

A good strategy uses the interconnections that already exist in the database to minimize the total number of database records accessed.

Using FIND or OBTAIN

FIND locates a record occurrence in the database; **OBTAIN** locates a record occurrence in the database *and* moves that occurrence to program variable storage.

Using FINDs instead of OBTAINs makes LRF processing more efficient, but sometimes you must OBTAIN database records. For a discussion of when to use OBTAIN statements, refer to Chapter 5, Defining Path Groups.

Currency options

The DBMS refers to and updates **currency** while processing path-DML statements, just as it does while processing program-DML commands. LRF automatically keeps track of currency between program requests. As a result, programmers do not have to know the currency of records, sets, or areas when they rerequest a logical record.

You can specify the following currency options in the ADD SUBSCHEMA DDL statement:

- **LR CURRENCY RESET (default)** causes LRF to reobtain all of the logical-record elements it placed in program variable storage during the previous execution of the path. This occurs each time LRF reenters an OBTAIN path.

  LR CURRENCY RESET also causes LRF to restore the currency tables in the DBMS to what they were when control was returned to the program.

- **LR CURRENCY NO RESET** causes LRF to restore currency only for the last iterated verb. LRF does not reobtain any logical-record elements.

In most cases, you will want to specify a currency option of LR CURRENCY NO RESET. This option is more efficient than LR CURRENCY RESET because it saves processing time and may also save I/O.

You should only allow LR currency to default to RESET when:

- The program will be modifying the logical-record area of program variable storage during path iteration
- Either the program or another path will be modifying a logical-record component in the database (by means of a MODIFY or ERASE command), and you want LRF to react to the changes
- You are using subschemas defined for the exclusive use of CA OLQ

For a detailed discussion of LRF currency considerations, refer to Chapter 15, Currency Considerations. For a discussion of using LRF with CA OLQ, refer to Chapter 17, Using LRF with Other Facilities.

Defining path selectors carefully

Every path in a path group must begin with at least one SELECT clause. This clause marks the beginning of a path definition. Selectors in the clause are used to match a program request to the appropriate path.

Considerations

When you define path selectors, you should keep these guidelines in mind:

- **Some path selectors are better than others for certain types of database access**:
  - Use the FIELDNAME-EQ selector for CALC or direct entry. This selector ensures that the path will be selected if the program references the key field in the appropriate way.
  - Use the FIELDNAME selector to allow for nonkey retrieval or for generic key retrieval through an index.
  - Use ELEMENT or null selectors if you want to do an area sweep.
- **KEYWORD selectors are very versatile**. You can use them to:
  - Guarantee a match from the program to a given path.
  - Reduce the need for programmers to code detailed comparisons. For example, you can put selection criteria in the path and then use the keyword to map to the path.

  You can also combine KEYWORD selectors with other types of selectors.

- **LRF evaluates SELECT clauses in the order in which they are coded**; it services program requests with the first matching path. You should, therefore, always sequence SELECT clauses in order, from most specific to least specific.

For a detailed discussion of path selectors, refer to Chapter 5, Defining Path Groups.

Chapter 4: Starting to Define the Subschema

This section contains the following topics:

Introduction

Once you have analyzed your organization's information needs and thought about how you will design your logical records, you are ready to define a subschema.

This chapter describes how to start defining a subschema that includes one or more logical records. In this chapter, you will:

- Specify a usage mode for the subschema
- Specify a currency option for the subschema
- Include the necessary records, sets, and areas
- Define the logical records you want to include in the subschema

As you read this chapter, you should refer to the *CA IDMS Database Administration Guide* for a description of subschema compiler syntax.

Specifying a Subschema Usage Mode

One of the first steps in defining a logical-record subschema is to specify a subschema usage mode. For a subschema that contains logical records, the usage mode can be either LR or MIXED.

LR usage mode

A usage mode of **LR** allows programs using the subschema to issue requests for logical records (through LRF DML statements). Database records are accessed as components of logical records.

Additionally, these programs can issue the following commands, if appropriate:

- ACCEPT DATABASE STATISTICS
- BIND PROCEDURE
- COMMIT
- RETURN (for indexed sets only)
- ROLLBACK

MIXED usage mode

A usage mode of **MIXED** (the default) allows programs using the subschema to issue LRF DML statements *and* navigational DML statements. These programs can access logical records and single database records.

Specifying the usage mode

You specify a subschema usage mode by using the USAGE IS clause of the ADD SUBSCHEMA Data Description Language (DDL) statement. The following example shows how to specify a subschema usage mode for the sample EMPLR35 subschema.

The EMPLR35 subschema has a usage mode of LR. Programs that use this subschema can issue requests for logical records only.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   DESCRIPTION IS 'SAMPLE SUBSCHEMA FOR LRF MANUAL'
   PUBLIC ACCESS IS ALLOWED FOR ALL
   USAGE IS LR  ◄--------
   .
   .
   .
```

Specifying a Subschema Currency Option

When a subschema contains logical records, you can specify whether LRF is to reset currency and restore the logical-record area of program variable storage. You specify these currency options in the LR CURRENCY clause of the ADD SUBSCHEMA DDL statement.

NO RESET currency option

NO RESET directs LRF to restore currency for the last path-DML statement. LRF will restore currency before it reexecutes an OBTAIN path.

RESET currency option

RESET (default) directs LRF to reset currency *and* to restore the logical record's program variable storage area before it reexecutes an OBTAIN path.

The following example shows how to specify a currency option for the EMPLR35 subschema. The EMPLR35 subschema has a currency option of NO RESET.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   DESCRIPTION IS 'SAMPLE SUBSCHEMA FOR LRF MANUAL'
   PUBLIC ACCESS IS ALLOWED FOR ALL
   USAGE IS LR
   LR CURRENCY NO RESET  <-----
   .
   .
   .
```

For more information on subschema currency options, refer to Chapter 15, Currency Considerations.

Including Records, Sets, and Areas

You must include the following database components in a subschema that contains logical records:

- All database records that contain data to be used in application processing
- Any other database record referenced in a path
- All sets (including indexed sets) referenced in a path
- All areas that contain the specified database records and sets
- All sets and areas that will be affected by update activity

You include database components in a subschema by using the ADD RECORD, ADD SET, and ADD AREA DDL statements.

Restricting the subschema's view

You should restrict the subschema's view of any database records included in your subschema definition. You can restrict this view in two ways:

- By using a VIEW ID clause to copy a predefined view of the record description into the subschema
- By using an ELEMENTS ARE clause to identify the fields to be included in the record description

If you don't use one of these clauses, the record description will include all fields contained in the database record. This could:

- Increase the size of the subschema load module, which can result in wasted space in program variable storage.
- Make the programs that use the subschema more vulnerable to changes in the database record definition. For example, if new fields are added to the record, the new record description gets copied into the logical-record area of program variable storage. Programs that use the subschema then need to be recompiled, whether or not they access the new fields.

The following example shows how to include database components for the sample EMPLR35 subschema.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
    .
    .
    .
ADD
AREA NAME IS EMP-DEMO-REGION
    .
ADD
AREA NAME IS ORG-DEMO-REGION
    .
ADD
RECORD NAME IS EMPLOYEE
   ELEMENTS ARE EMP-ID-0415 EMP-NAME-0415 START-DATE-0415
   STATUS-0415
    .
ADD
RECORD NAME IS DEPARTMENT
   ELEMENTS ARE DEPT-ID-0410 DEPT-NAME-0410
    .
ADD
RECORD NAME IS OFFICE
   ELEMENTS ARE OFFICE-CODE-0450
    .
ADD
SET NAME IS EMP-NAME-NDX
    .
ADD
SET NAME IS DEPT-EMPLOYEE
    .
ADD
SET NAME IS OFFICE-EMPLOYEE
    .
    .
    .
```

Defining Logical Records

Now you are ready to define the logical records that you want to include in the subschema. You define a logical record by using the ADD LOGICAL RECORD DDL statement. LRF uses the record layout described by this statement to reserve an area in program variable storage.

Steps in defining a logical record

To define a logical record, you should:

1. Name the logical record
2. Name the elements of the logical record
3. Specify whether the logical-record description in program variable storage will be reinitialized when certain path statuses are returned
4. Document the logical record

Step 1: Name the logical record

The logical record name must be from 1 through 16 characters and must be unique for the current subschema; it can't duplicate the name of another logical record or database record described in the same subschema.

You name a logical record by using the NAME IS clause of the ADD LOGICAL RECORD DDL statement. The following example shows how to name the EMP-INFO-LR logical record.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
    .
    .
    .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
    .
    .
    .
```

Step 2: Name the logical-record elements

You identify the records that participate in the logical record by using the ELEMENTS clause of the ADD LOGICAL RECORD DDL statement. Program variable storage will contain a description of each record you name as a logical-record element.

Note: Fields that are excluded from the subschema view will *not* be included in a logical-record description.

Records to include

You should include the following records as logical-record elements:

- All database records used to construct the logical record.
- Any other database record that contains fields used in one of the following ways:
  - To pass data between the program and the path
  - As operands in EVALUATE and COMPUTE operations
- Any IDD-defined work record that contains fields used in one of the following ways:
  - To pass data between the program and the path
  - As operands in EVALUATE and COMPUTE operations

  Be sure you indicate which version of the work record you want to include in the logical record.

  Note: Do not code an ADD RECORD statement for any IDD-defined work record included in your logical-record description.

Using role names

To include an element that occurs more than once in a single logical record, you can use role names. Role names must be included in the ELEMENTS clause of the ADD LOGICAL RECORD statement. Once a role name has been assigned, both the path and the program must refer to the role rather than to the associated record element. For complete information on the use of roles, refer to Chapter 13, Using Role Names.

Naming record elements for EMP-INFO-LR

The following example shows how to name the record elements for the EMP-INFO-LR logical record.

Note: All logical-record components are double-word aligned when they appear in program variable storage.

The EMP-INFO-LR logical record includes the EMPLOYEE, DEPARTMENT, OFFICE, and PATHREC records.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   .
   .
   .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
   ELEMENTS ARE
       EMPLOYEE
       DEPARTMENT
       OFFICE
       PATHREC VERSION 1  ◄------ IDD-defined work record
   .
   .
   .
```

The following example shows the description of the EMP-INFO-LR logical record that will appear in program variable storage. The logical-record description for EMP-INFO-LR includes selected fields from the EMPLOYEE, DEPARTMENT, and OFFICE database records, and all fields from the PATHREC work record.

```
01  EMP-INFO-LR.
    02  EMPLOYEE.
        03  EMP-ID-0415              PIC 9(4).
        03  EMP-NAME-0415
            04  EMP-FIRST-NAME-0415  PIC X(10).
            04  EMP-LAST-NAME-0415   PIC X(15).
        03  START-DATE-0415
            04  START-YEAR-0415      PIC 9(2).
            04  START-MONTH-0415     PIC 9(2).
            04  START-DAY-0415       PIC 9(2).
        03  STATUS-0415              PIC 9(4).
        03  FILLER                   PIC X(1).
    02  DEPARTMENT.
        03  DEPT-ID-0410             PIC 9(4).
        03  DEPT-NAME-0410           PIC X(45).
        03  FILLER                   PIC X(7).
    02  OFFICE.
        03  OFFICE-CODE-0450         PIC 9(3).
        03  FILLER                   PIC X(5).
    02  PATHREC.
        03  WORK-PATH-ID             PIC X(10).
```

Step 3: Specify initialization options for program variable storage

You can request that LRF clear program variable storage automatically when the following path statuses are returned:

- LR-ERROR
- LR-NOT-FOUND

You specify these options in the ON LR-ERROR and ON LR-NOT-FOUND clauses of the ADD LOGICAL RECORD DDL statement. The default for each option is NOCLEAR.

Specifying CLEAR

By specifying CLEAR, you ensure that the data in program variable storage made available to the user meets program WHERE clause selection criteria. This option is recommended if you want LRF to return complete logical records to the program.

The following example shows how to specify initialization options for the EMP-INFO-LR logical record. These initialization options ensure that program variable storage will be cleared when LRF returns a path status of LR-NOT-FOUND or LR-ERROR.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   .
   .
   .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
   ELEMENTS ARE
       EMPLOYEE
       DEPARTMENT
       OFFICE
       PATHREC VERSION 1
   ON LR-ERROR CLEAR          ┐  Initialization
   ON LR-NOT-FOUND CLEAR      ┘  options
   .
   .
   .
```

For more information on complete and partial logical records, refer to Chapter 11, Controlling Path Execution.

Step 4: Document the logical record

To give programmers the information they need to use the subschema successfully, you should document each logical record thoroughly. You can document a logical record by using the COMMENTS clause of the ADD LOGICAL RECORD DDL statement. The information you provide will be copied along with the logical record to the appropriate area of program variable storage.

Coding preliminary comments

It is recommended that you code preliminary comments at this stage of the subschema definition process. Once you have defined the associated path groups and paths, you can make your comments more complete.

For the preliminary comments, you can:

- List the database records that the logical record will access
- List the DML verbs that can be issued for the logical record
- Describe the program selection criteria that will map to each path

The following example shows how to code preliminary comments for the EMP-INFO-LR logical record. For instructions on coding comments, refer to Chapter 14, Documenting the Subschema.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
    .
    .
    .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
    ON LR-ERROR CLEAR
    ON LR-NOT-FOUND CLEAR
    ELEMENTS ARE
        EMPLOYEE
        DEPARTMENT
        OFFICE
        PATHREC VERSION 1
    COMMENTS
```

```
      '*************************************************************'
-     'THE EMP-INFO-LR LOGICAL RECORD ACCESSES INFORMATION FROM THE'
-     'EMPLOYEE DATABASE RECORD AND ALSO ACCESSES INFORMATION'
-     'FROM THE ASSOCIATED DEPARTMENT AND OFFICE RECORDS.'
-     '*************************************************************'
-     ' '
-     'LR VERBS ALLOWED: OBTAIN'
-     ' '
-     '*************************************************************'
-     ' '
-     'SELECTION CRITERIA (TOTAL OF FIVE PATHS)'
-     ' '
-     '   OBTAIN PATH GROUP:'
-     ' '
-     ' '
-     '    PATH 1)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR ALL EMPLOYEES WHO ARE'
-     '             ON LEAVE.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF THE PROGRAM'
-     '             REQUEST INCLUDES THE KEYWORD ON-LEAVE.'
-     ' '
-     ' '
-     '    PATH 2)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.'
-     '             IT USES THE EMP-ID-0415 FIELD AS A'
-     '             CALC KEY TO ACCESS EMPLOYEE INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             EMP-ID-0415 = A NUMERIC LITERAL'
-     '                           A PROGRAM VARIABLE'
-     '                           A FIELD IN THE LOGICAL-RECORD'
-     '                            AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
.
.
.
```

Chapter 5: Defining Path Groups

This section contains the following topics:

What Is a Path Group

Path groups

A path group is a collection of paths that service a particular type of program request. Every path defined for a logical record must be included in a path group; you define at least one path group for each logical record.

You can write up to four path groups for any given logical record:

- An **OBTAIN** path group contains paths that service program requests to OBTAIN a logical record.
- A **MODIFY** path group should contain paths that service program requests to MODIFY a logical record.
- A **STORE** path group should contain a collection of paths that service program requests to STORE a logical record.
- An **ERASE** path group should contain a collection of paths that service program requests to ERASE a logical record.

Retrieval paths

Paths included in an OBTAIN path group are called **retrieval paths**. With retrieval paths, you can:

- Issue path-DML commands to retrieve a database record
- Iterate the path at the request of the program

Note: You cannot use a retrieval path to update a database record.

Update paths

Paths included in MODIFY, STORE, and ERASE path group are called **update paths**. You can issue any path-DML command (including retrieval commands) within an update path. However, you cannot iterate the path at the request of the program.

Note: Be careful when using a program WHERE clause since it is evaluated prior to the execution of an update verb.

Creating the Definition

You define a path group by using the ADD PATH-GROUP DDL statement. To define a path group, you should:

- Determine the verbs you want the programmer to be able to issue for the logical record. You code a path-group statement for each of these verbs.
- Design and code the paths you need within each path group.

  Instructions for coding paths are presented in Chapter 6, Specifying Path Selectors, through Chapter 14, Documenting the Subschema.

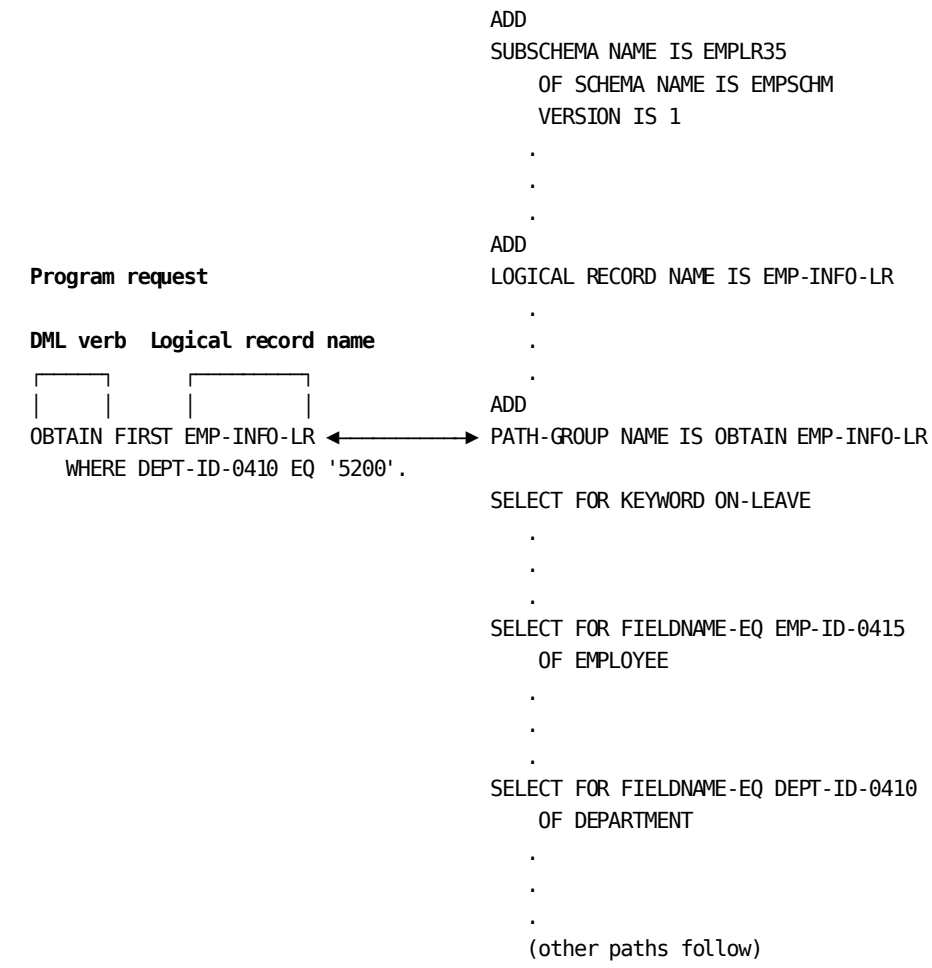
Example

The following example shows how to define the OBTAIN path group used in the sample EMP-INFO-LR logical record.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
   .
   .
   .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
   .
   .
   .
ADD
PATH-GROUP NAME IS OBTAIN EMP-INFO-LR
     SELECT FOR KEYWORD ON-LEAVE
        .
        .
        .
    SELECT FOR FIELDNAME-EQ EMP-ID-0415
        OF EMPLOYEE
        .
        .
        .
    SELECT FOR FIELDNAME-EQ DEPT-ID-0410
        OF DEPARTMENT
        .
        .
        .
    (other paths follow)
```

Locating an appropriate path group

When LRF receives a logical-record request from the program, it locates the appropriate path group based on the DML verb issued and the name of the object logical record. The following diagram shows how LRF locates an appropriate path group.



Chapter 6: Specifying Path Selectors

This section contains the following topics:

What Is a Path Selector?

Every path in a path group must contain at least one SELECT clause. A SELECT clause delimits a path. It is always coded at the beginning of a path definition.

Path selectors

Each SELECT clause can contain any number of path **selectors**. These selectors describe comparison criteria for the path. After locating the appropriate path group, LRF uses selectors to locate an appropriate path for a program request:

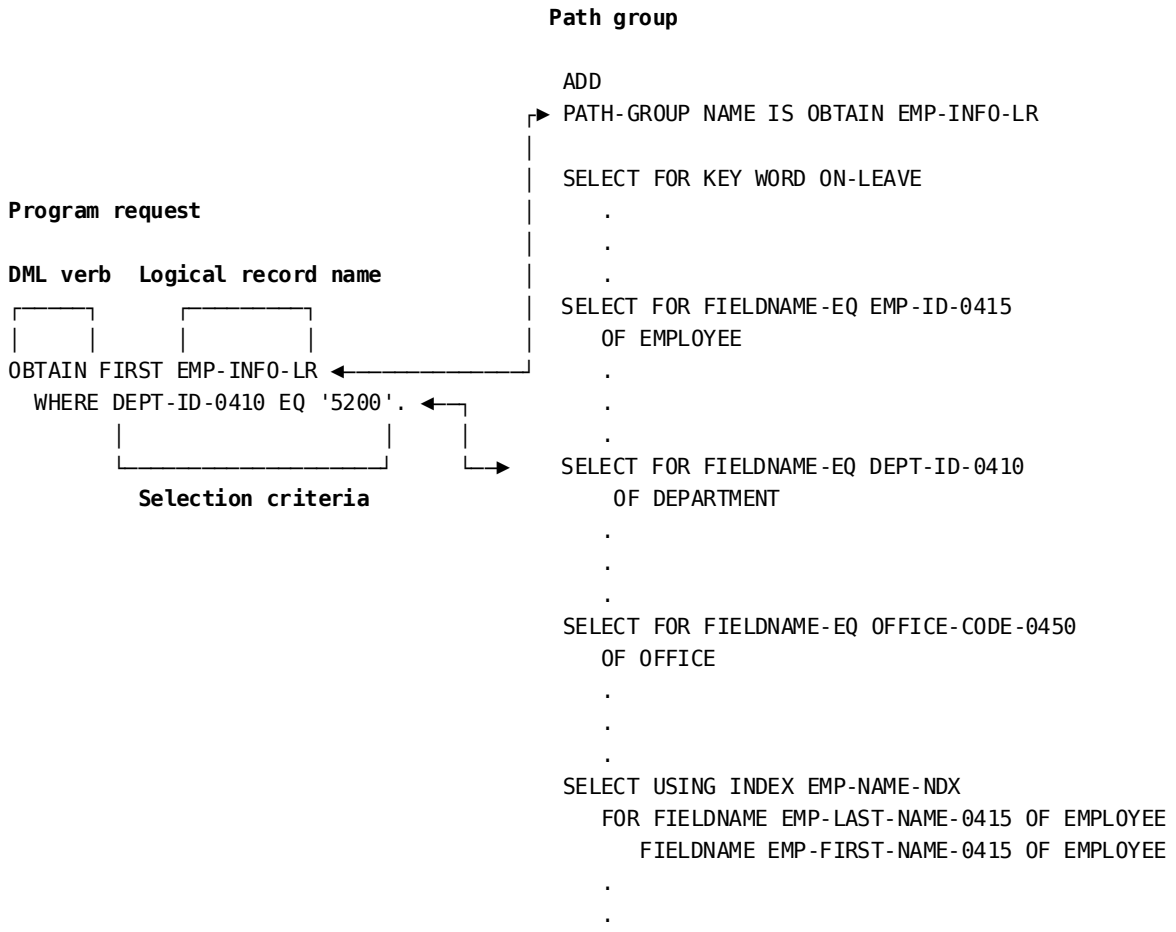
1. LRF compares the program WHERE clause with the SELECT clauses coded in the path group. LRF examines SELECT clauses in the order in which they appear. The first path whose selectors match those of the program request is the path that LRF executes.

   For a match to occur, the program request must satisfy *all* of the criteria specified by the path selectors. However, the program WHERE clause can contain additional selection criteria not matched in the path.

   For information on selection criteria, refer to Chapter 10, Specifying Selection Criteria for Logical Records.

2. If LRF does not encounter a match between the program WHERE clause and path selectors, and there is no null SELECT clause, the program request will not be processed.

The following diagram shows how LRF locates an appropriate path for a program request by matching the selection criteria specified in the program request to the selectors specified in the path.



Types of path selectors

There are four path selectors that you can include in a SELECT clause.

Path selector	Description
KEYWORD *keyword*	The WHERE clause of a program request must include the named keyword. The comparison must be in the affirmative (that is, NOT KEYWORD isn't valid), and the KEYWORD must not participate in an OR operation.

Path selector	Description
FIELDNAME-EQ *lr-field-name*	The WHERE clause of a program request must reference the named logical-record field in an equality comparison. The field must be compared to a single value and must not participate in an OR operation.
FIELDNAME *lr-field-name*	The WHERE clause of a program request must reference the named logical-record field in any manner; no restrictions apply.
ELEMENT *lr-element-name*	The WHERE clause of a program request must reference a field that is in a named logical-record element.

SELECT clauses

Using the four path selectors, you can code these SELECT clauses in a path:

- SELECT FOR KEYWORD
- SELECT FOR FIELDNAME-EQ
- SELECT FOR FIELDNAME
- SELECT FOR ELEMENT

In addition, you can code:

- A SELECT clause that has no selectors (a null SELECT)
- A SELECT clause that specifies an index
- Multiple selectors in one SELECT clause
- Multiple SELECT clauses in a single path

The remainder of this chapter discusses how to use each of the four path selectors and the various types of SELECT clauses. These discussions are followed by guidelines for determining the order of your paths.

Using the KEYWORD Selector

What the KEYWORD selector does

You use the KEYWORD selector to:

- **Guarantee a match from the program to a given path**. With the KEYWORD selector, the program request doesn't have to contain detailed selection criteria to be matched to a path.
- **Insulate the program from logical-record selection criteria**. With the KEYWORD selector, you can put all logical-record selection criteria in the path. This allows you to change selection criteria without requiring that the program be recompiled. It also makes data more secure, since programmers need not be aware of the selection criteria used.

You can also use a keyword to specify which path is servicing a particular request. This is very useful for the DBAs in debugging situations and can be useful to the programmer, as well.

For a program request to match to a KEYWORD selector, the program must include the named keyword in its WHERE clause.

Examples

You have coded the following SELECT clause in a path:

```
SELECT FOR KEYWORD ON-LEAVE
```

The examples that follow show some program requests that can successfully access this path as well as program requests that will not be successful.

Successful path access

This program request includes the named keyword in its WHERE clause.

```
OBTAIN FIRST EMP-LR WHERE ON-LEAVE.
```

This program request includes the named keyword in its WHERE clause; the keyword participates in an AND operation.

```
OBTAIN FIRST EMP-LR WHERE ON-LEAVE AND START-DATE-0415
   GE '800101'.
```

Unsuccessful path access

This program request includes the named keyword in a negative comparison. The keyword must be in the affirmative to match to the path.

```
OBTAIN FIRST EMP-LR WHERE NOT ON-LEAVE.
```

This program request includes the named keyword in an OR operation. The keyword must be either by itself or in an AND operation to match to the path.

```
OBTAIN FIRST EMP-LR WHERE ON-LEAVE OR START-DATE-0415
   GE '800101'.
```

Using the FIELDNAME-EQ Selector

What the FIELDNAME-EQ selector does

You use the FIELDNAME-EQ selector to:

- **Guide a program request to a path that enters the database based on a CALC key or sort key value**. The FIELDNAME-EQ selector guarantees that the path will be selected if the program references the key field in an appropriate way.
- **Limit the number of logical-record occurrences returned to the program**. Because the path is accessing a database record type based on a key value, fewer record occurrences will be found and returned.

For a program request to match to a FIELDNAME-EQ selector, the program's comparison must equate the logical-record field named in the selector to one of the following:

- A literal value
- A program variable defined in the data dictionary
- A field described in the logical-record area of program variable storage

Examples

You have coded the following SELECT clause in a path:

```
SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
```

The examples that follow show some program requests that can successfully access this path as well as program requests that will not be successful.

Successful path access

This program request equates the named field to a literal.

```
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 EQ '0015'.
```

This program request equates the named field to a variable defined in the data dictionary.

```
MOVE '0015' TO WORK-FIELD.
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 OF EMPLOYEE EQ WORK-FIELD.
```

This program request equates the named field to a field defined in the logical-record area of program variable storage.

```
MOVE '0015' TO EMP-ID-0415.
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 EQ
   EMP-ID-0415 OF LR.
```

This program request equates the named field to a literal and includes the field in an AND operation.

```
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 EQ '0015' AND
   EMP-LAST-NAME EQ 'SMITH'.
```

Unsuccessful path access

This program request includes the named field in a comparison other than an equality comparison. The field must be referenced in an equality comparison to match to the path.

```
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 GT '0010'.
```

This program request includes the named field in a OR operation. The field can be either by itself or included in an AND operation to match to the path.

```
OBTAIN FIRST EMP-LR WHERE (EMP-ID-0415 EQ '0015') OR
   (EMP-LAST-NAME EQ 'JONES').
```

This program request equates the named field to an expression rather than to a single value. The field must be compared to a single value to match to the path.

```
OBTAIN FIRST EMP-LR WHERE EMP-ID-0415 EQ (WORK-FIELD - 1).
```

Using the FIELDNAME Selector

What the FIELDNAME selector does

You use the FIELDNAME selector as follows:

- To perform retrieval based on a nonkey field
- To perform indexed retrieval based on a generic key, as described in Coding Path Database Retrieval Commands (see page 61).

For a program request to match to a FIELDNAME selector, the program's comparison must reference the logical-record field named in the selector. This field can be referenced in any manner.

Examples

You have coded the following SELECT clause in a path:

```
SELECT FOR FIELDNAME START-YEAR-0415 OF EMPLOYEE
```

The examples that follow show some program requests that can successfully access this path.

Successful path access

This program request includes the named field in an AND operation.

```
OBTAIN FIRST EMP-LR WHERE (START-YEAR-0415 GT '80') AND
   (TERMINATION-YEAR-0415 LT '85').
```

This program request includes the named field in a MATCHES comparison.

```
OBTAIN FIRST EMP-LR WHERE (START-YEAR-0415 MATCHES '8#')
   AND (DEPARTMENT-NAME-0410 EQ 'DATA PROCESSING').
```

This program request equates the named field to an expression.

```
OBTAIN FIRST EMP-LR WHERE START-YEAR-0415 EQ
   (START-YEAR-WORK + 1).
```

Using the ELEMENT Selector

What the ELEMENT selector does

Paths associated with an ELEMENT selector typically perform an area sweep. This selector can be used to service a broad range of program requests.

For a program request to match an ELEMENT selector, the program's comparison must reference a field that's included in the named logical-record element. The field can be referenced in any manner.

Examples

You have coded the following SELECT clause in a path:

```
SELECT FOR ELEMENT EMPLOYEE
```

The examples that follow show some program requests that can successfully access this path.

Successful path access

This program request references a field contained in the named logical-record element. The field is used in an equality comparison.

```
OBTAIN FIRST EMP-LR WHERE EMP-LAST-NAME-0415 EQ
   EMP-LAST-NAME-0415 OF LR.
```

This program request references a field contained in the named logical-record element. This field is included in an AND operation.

```
OBTAIN FIRST EMP-LR WHERE STATUS-0415 EQ '05' AND
   START-YEAR-0415 GE '80'.
```

Using a Null SELECT Clause

What a null SELECT clause does

A null SELECT clause contains no selectors. This type of SELECT clause can service any logical-record request, including:

- All logical-record requests for which no specific path has been defined
- A logical-record request that doesn't contain a WHERE clause

Because a null SELECT clause can match any program request, you should use it judiciously. If you choose to code a path with a null SELECT clause, always place this path at the end of the path group.

Examples

You have coded the following SELECT clause in a path:

```
SELECT
```

The examples that follow show some program requests that can successfully access this path.

Successful path access

This program request doesn't match any other path.

```
OBTAIN FIRST EMP-LR WHERE (DEPT-NAME-0410 EQ DEPT-NAME-0410
   OF LR) AND (EMP-ID-0415 GT '0010' AND EMP-ID-0415 LT '0050').
```

This program request doesn't include a WHERE clause.

```
OBTAIN FIRST EMP-LR.
```

Using a SELECT Clause That Names an Index

What a SELECT clause naming an index does

A SELECT clause that names an index tells LRF to use the named index in path processing. You can associate an index with any type of SELECT clause, including a null SELECT.

A program request matches this type of SELECT clause when it meets the requirements of the various selectors (if any are included). The program request does not have to specify an index name.

For more information on database entry based on indexes, refer to Chapter 7, Coding Path Database Retrieval Commands.

Example

You have coded the following SELECT clause in a path:

```
SELECT USING INDEX EMP-NAME-NDX
  FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
```

The examples that follow show some program requests that can successfully access this path.

Successful path access

This program request meets the requirements of the FIELDNAME selector.

```
OBTAIN FIRST EMP-LR WHERE EMP-LAST-NAME-0415 EQ
   EMP-LAST-NAME-0415 OF LR.
```

This program request also meets the requirements of the FIELDNAME selector.

```
OBTAIN FIRST EMP-LR WHERE EMP-LAST-NAME-0415
   MATCHES 'B'.
```

Using Multiple Selectors In a Single SELECT Clause

A SELECT clause that contains multiple selectors combines the capabilities of each of its selectors. The following combinations of selectors are especially useful:

- **A KEYWORD selector in combination with a FIELDNAME-EQ selector**. This combination guarantees that the path will be selected if the program references the key field in an appropriate way, *and* it gives the path control over the selection of logical-record occurrences.
- **Multiple FIELDNAME-EQ selectors**. This combination is useful for paths that enter the database based on a concatenated key. The various FIELDNAME-EQ selectors guarantee that the path will be selected if the program references each key field in the appropriate way.

For a program request to match a SELECT clause that contains multiple selectors, the program request must meet the requirements of *all* the named selectors.

Examples

You have coded the following SELECT clause in a path:

```
SELECT FOR KEYWORD MOD-EMP
           FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
```

To successfully access this path, the request shown below meets the requirements of both the FIELDNAME-EQ selector and the KEYWORD selector.

```
OBTAIN FIRST EMP-LR WHERE MOD-EMP AND EMP-ID-0415 EQ
   EMP-ID-0415 OF LR.
```

You have coded the following SELECT clause in a path, where each FIELDNAME-EQ selector references a component of a concatenated key:

```
SELECT FOR FIELDNAME-EQ RACE-TITLE
           FIELDNAME-EQ RACE-DATE
```

To successfully access this path, the request shown below meets the requirements of both FIELDNAME-EQ selectors.

```
OBTAIN FIRST RACE-LR WHERE RACE-TITLE EQ 'DERBY' AND
   RACE-DATE EQ '860801'.
```

Using Multiple SELECT Clauses For One Path

You can specify more than one SELECT clause for a path when you want to access the path with different types of program requests. You may want to specify multiple SELECT clauses in the following cases:

- In combination with the USING INDEX clause, to process a concatenated index
- To eliminate identical paths that are delimited by different SELECT clauses
- To service different types of program requests that have not been matched to previous paths in the subschema

To access a path delimited by multiple SELECT clauses, the program request need only meet the requirements of *one* SELECT clause.

Examples

You have coded the following SELECT clauses for a path:

```
SELECT USING INDEX EMP-NAME-NDX
   FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
       FIELDNAME EMP-FIRST-NAME-0415 OF EMPLOYEE
SELECT USING INDEX EMP-NAME-NDX
   FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
```

The examples that follow show some program requests that can successfully access this path.

Successful path access

This program request references both EMP-LAST-NAME-0415 and EMP-FIRST-NAME-0415. Therefore, it meets the requirements of the first SELECT clause.

```
OBTAIN FIRST EMP-LR WHERE (EMP-LAST-NAME-0415 EQ 'SMITH')
   AND (EMP-FIRST-NAME-0415 EQ 'JANET').
```

This program request references EMP-LAST-NAME-0415 but does not reference EMP-FIRST-NAME-0415. Therefore, it meets the requirements of the second SELECT clause.

```
OBTAIN FIRST EMP-LR WHERE EMP-LAST-NAME-0415 MATCHES 'H'.
```

Determining Path Order

When LRF receives a request to retrieve or update a logical record, it attempts to match the selection criteria specified by the program's WHERE clause to the selectors in a path. LRF begins this process with the first path in the path group. It then searches the path group from top to bottom until it finds a path whose selectors match the program's selection criteria. The first matching path is the one that LRF selects.

Sequencing paths

Because LRF searches the path group sequentially, the order in which you code paths can determine which paths will be chosen. To facilitate efficient database entry, be sure you code paths in order, from most specific to least specific.

As a rule, you should sequence paths by selector in the following order:

- KEYWORD
- FIELDNAME-EQ
- FIELDNAME
- ELEMENT

A path that has multiple selectors is often one of the more specific paths in a path group, while a path with a null SELECT clause is *always* the least specific path. Because a null SELECT clause matches all requests, it should be the last SELECT clause coded in a path group.

Examples

The following examples shows the sequence of retrieval paths coded in a sample logical record, along with some matching program requests. The paths are coded in order from most specific to least specific. By coding the paths in this sequence, efficient database entry is guaranteed.

```
ADD
LOGICAL RECORD NAME IS SAMPLE-LR
   .
   .
   .
ADD
PATH-GROUP NAME IS OBTAIN SAMPLE-LR
    SELECT FOR KEYWORD ALL-EMP
      .
      .
      .
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      .
      .
      .
   SELECT USING EMP-NAME-NDX
      FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
          FIELDNAME EMP-FIRST-NAME-0415 OF EMPLOYEE
      .
      .
      .
   SELECT FOR FIELDNAME START-DATE-0415 OF EMPLOYEE
      .
      .
      .
   SELECT FOR ELEMENT EMPLOYEE
      .
      .
      .
   SELECT
```

Chapter 7: Coding Path Database Retrieval Commands

This section contains the following topics:

Introduction

Path database retrieval commands locate data in the database and make it available to the application program. You can use the following path-DML statements to retrieve database records.

Statement	Action
FIND/OBTAIN WHERE CALCKEY	Accesses a database record by using its CALC key value.
FIND/OBTAIN WITHIN SET WHERE SORTKEY	Accesses a database record in a sorted set by using its sort key value.
FIND/OBTAIN WHERE DBKEY	Accesses a database record directly by using its database key (db-key) value.
FIND/OBTAIN WITHIN SET/AREA	Accesses a database record based on its logical location within a set or its physical location within an area.
FIND/OBTAIN EACH USING INDEX	Accesses a database record based on the index specified in the path SELECT clause.

Statement	Action
FIND/OBTAIN OWNER	Accesses the owner record of a set occurrence.
FIND/OBTAIN CURRENT	Accesses a database record by using previously established currencies.

Unique LRF options

Path retrieval commands can include a number of options that are unique to LRF:

- All FIND/OBTAIN commands can include a WHERE clause to select the occurrences of the database record to be accessed. (For the FIND/OBTAIN WHERE CALCKEY, FIND/OBTAIN WITHIN SET WHERE SORTKEY, and FIND/OBTAIN WHERE DBKEY commands, the WHERE clause is mandatory.)
- The FIND/OBTAIN WHERE CALCKEY, FIND/OBTAIN WHERE SORTKEY, and FIND/OBTAIN WITHIN SET/AREA commands can include an EACH option, which directs LRF to iterate the path. Path iteration is discussed in Chapter 11, Controlling Path Execution.
- The FIND/OBTAIN WHERE CALCKEY, FIND/OBTAIN WHERE SORTKEY, and FIND/OBTAIN WHERE DBKEY commands can contain the clauses OF REQUEST or OF LR. These clauses indicate the location of the key value. They're discussed under "Passing key values", later in this chapter.

In general, you use path retrieval commands in the same way that you use CA ADS database retrieval commands or navigational DML retrieval commands.

Note: In this document, all references to CA ADS apply to the Application Development System process language, which is used by CA ADS and CA ADS Batch.

Information on using CA ADS retrieval commands can be found in the *CA ADS Reference Guide*. Information on using navigational DML commands can be found in the *CA IDMS Navigational DML Programming Guide*.

Table of similarities and differences

The table below summarizes the similarities and differences between path retrieval commands and CA ADS retrieval commands.

Path command	Comparison	CA ADS command
FIND/OBTAIN WHERE CALCKEY	Path command has no DUPLICATE option. To retrieve records with duplicate CALC keys, you must use the FIRST/NEXT or EACH option of the path command.	FIND/OBTAIN CALC/DUPLICATE
FIND/OBTAIN WITHIN SET WHERE SORTKEY	The commands are functionally equivalent.	FIND/OBTAIN WITHIN SET USING SORTKEY
FIND/OBTAIN WHERE DBKEY	Path command must specify a record name. For the CA ADS command, this is optional.	FIND/OBTAIN DB-KEY
FIND/OBTAIN WITHIN SET/AREA	Path command must specify a record name. For the CA ADS command, this is optional.	FIND/OBTAIN WITHIN SET/AREA
FIND/OBTAIN EACH USING INDEX	There is no CA ADS command for this path command.	
FIND/OBTAIN OWNER	Path command can specify record name.	FIND/OBTAIN OWNER
FIND/OBTAIN CURRENT	Path command must specify one of the following: A record name WITHIN set name WITHIN area name For the CA ADS command, this is optional.	FIND/OBTAIN CURRENT
	There is no path command for this CA ADS command.	ACCEPT
	There is no path command for this CA ADS command.	RETURN DB-KEY

Note: CA ADS database retrieval commands are similar in syntax and function to COBOL DML commands.

For an in depth discussion of each path retrieval command, refer to the *CA IDMS Database Administration Guide*.

Special considerations

When coding path retrieval commands, you should be aware of special considerations that apply to:

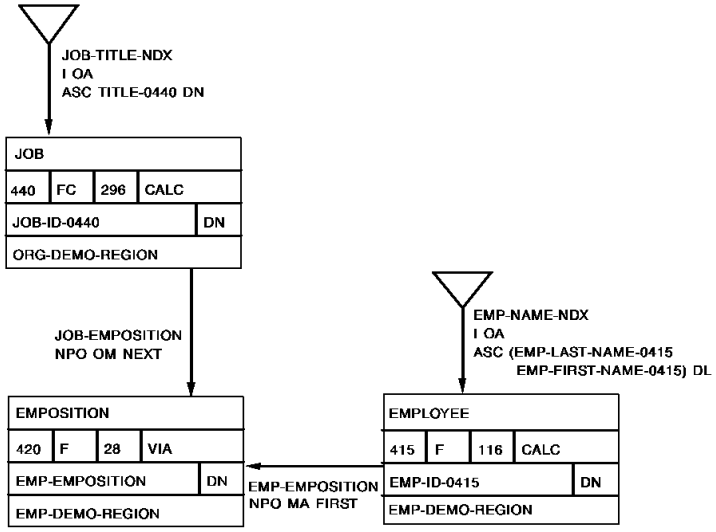
- Using FINDs and OBTAINs
- Passing key values
- Retrieving CALC records
- Retrieving indexed records
- Retrieving records directly

Using FINDs and OBTAINs

FIND

The **FIND** format of the path FIND/OBTAIN commands locates a record occurrence in the database but does not move the record occurrence to program variable storage. You should issue a FIND command whenever possible, to avoid moving data unnecessarily.

For example, you could use a FIND command to access a database record that the path will use for navigational purposes only. Suppose the EMPOSITION record is needed only to navigate from the EMPLOYEE record to the owner JOB record. Because the application program does not require any data from the EMPOSITION record, you could locate it with a FIND command.



Path code:

```
SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
   OBTAIN FIRST EMPLOYEE
      WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
   FIND FIRST EMPOSITION WITHIN EMP-EMPOSITION
   OBTAIN OWNER WITHIN JOB-EMPOSITION.
```

Program request:

```
OBTAIN FIRST EMP-NAME-LR
   WHERE EMP-ID-0415 EQ '0015'.
```

Note: You must *always* use a FIND command to access a database record that is not included in the logical record. There is no reserved space for this record in program variable storage; therefore, you can't OBTAIN the record.

OBTAIN

The **OBTAIN** format of the path FIND/OBTAIN commands locates a record occurrence in the database *and* moves the record's contents to program variable storage. You should OBTAIN a database record in the following situations:

- When you want to return the contents of the record to the application program.
- If you intend to issue an EVALUATE or COMPUTE statement against a field in the record. The EVALUATE statement is discussed in Chapter 10, Specifying Selection Criteria for Logical Records. The COMPUTE statement is discussed in Chapter 12, Manipulating Logical-Record Data.

Passing Key Values

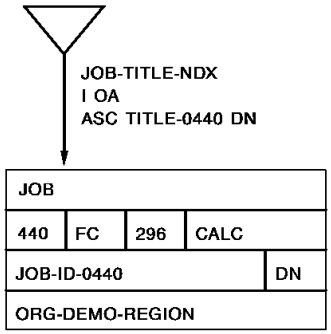
Specifying a key value

You can specify a value for a CALC key, sort key, or db-key in four ways:

- As a literal coded in the path
- As a logical-record field name, with the OF REQUEST clause
- As a logical-record field name, with the OF LR clause
- As an arithmetic expression

Specifying the key value as a literal

You can tell LRF to use a designated, numeric literal as a CALC key, sort key, or db-key value. For example, the path shown below retrieves the JOB record occurrence where the CALC key (JOB-ID-0440) equals '5031'.



Path code:

```
SELECT
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ '5031'.
```

Program request:

```
OBTAIN FIRST EMP-JOB-LR.
```

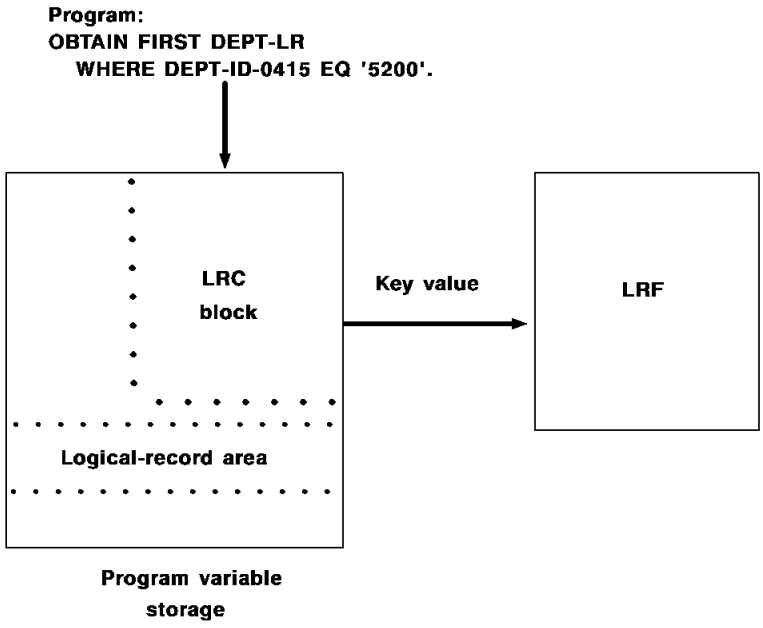
Specifying the key value with the OF REQUEST clause

The OF REQUEST clause tells LRF to use the key value contained in the LRC block (that is, the value passed through the program's WHERE clause). This value can equate to a literal, a program variable, or another field in the logical record. You typically use the OF REQUEST clause to access a key value passed by the application program.

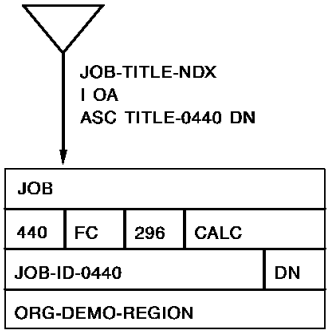
When you issue an OF REQUEST:

1. The value to be passed is coded in the program's WHERE clause
2. At runtime, that value is placed in the LRC block
3. The OF REQUEST clause in the path tells LRF to look for the key field value in the LRC block

The example below shows how the OF REQUEST clause works. LRF will use the key value contained in the program's WHERE clause. It examines the LRC block to find this value.



Examples



Using a value that equates to a literal

This path retrieves the JOB record occurrence where the JOB-ID-0440 field is equal to '5100'.

Path code:

```
SELECT FOR FIELDNAME-EQ JOB-ID-0440 OF JOB
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST.
```

Program request:

```
OBTAIN FIRST EMP-JOB-LR
   WHERE JOB-ID-0440 EQ '5100'.
```

Using a value that equates to a program variable

This path retrieves the JOB record occurrence where the JOB-ID-0440 field is equal to the value of JOB-WORK-FIELD.

Path code:

```
SELECT FOR FIELDNAME-EQ JOB-ID-0440 OF JOB
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST.
```

Program request:

```
MOVE '5100' TO JOB-WORK-FIELD.
OBTAIN FIRST EMP-JOB-LR
    WHERE JOB-ID-0440 EQ JOB-WORK-FIELD.
```

Using a value that equates to another field in the logical record

This path retrieves the JOB record occurrence where the JOB-ID-0440 field is equal to the value of the JOB-ID-0440 field in program variable storage.

Path code:

```
SELECT FOR FIELDNAME-EQ JOB-ID-0440 OF JOB
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST.
```

Program request:

```
MOVE INPUT-JOB TO JOB-ID-0440.
OBTAIN FIRST EMP-JOB-LR
   WHERE JOB-ID-0440 EQ JOB-ID-0440 OF LR.
```

Note: The OF LR clause in the **Program request** points to the key field's location in program variable storage. This clause will always correspond to an OF REQUEST clause in the path.

Specifying the key value with the path OF LR clause

The OF LR clause (used in the path) tells LRF to use the key value contained in the logical-record area of program variable storage. To use this clause successfully, you should make sure that the named key field has been initialized to a value. The field can be initialized by the path:

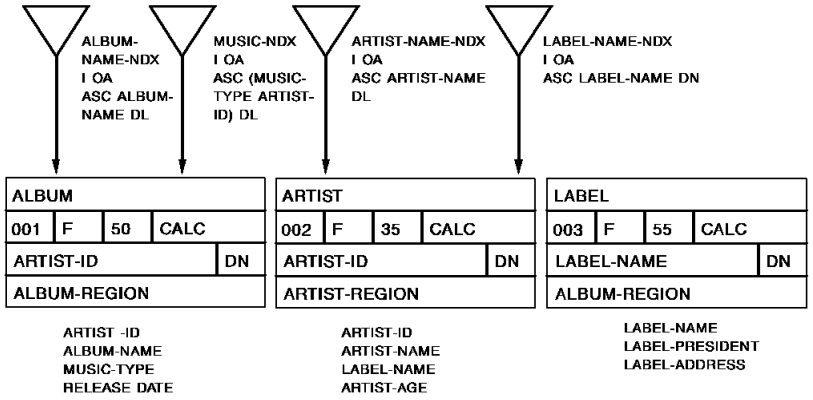
- The **path** can move a value into the LR buffer through either a prior database retrieval command or through a COMPUTE command.

You typically use the OF LR clause to access a key value that is set up in the path. This technique is especially useful for processing records that have shared keys.

The example below shows how the OF LR clause works. LRF will use the key value contained in the logical-record buffer. The key field must be initialized by the path.

Examples

When using the OF LR clause, be sure to qualify the logical-record field name in both the path *and* the program if the field name is not unique within the logical record.



Using a value the path has moved into program variable storage

This path retrieves the LABEL record occurrence where the LABEL-NAME field is equal to the value of the LABEL-NAME field in the ARTIST record occurrence. The LABEL-NAME field is qualified by record name because it is not unique within the logical record.

Path code:

```
SELECT
   OBTAIN FIRST ARTIST
      WHERE CALCKEY EQ '0415'
   OBTAIN FIRST LABEL
      WHERE LABEL-NAME OF LABEL EQ LABEL-NAME OF ARTIST OF LR.
```

Program request:

```
OBTAIN FIRST ARTIST-LR.
```

Using a value the program has moved into program variable storage

This path retrieves the ALBUM record occurrence where the ARTIST-ID field is equal to a value passed by the program.

Path code:

```
SELECT
   OBTAIN FIRST ALBUM
      WHERE CALCKEY EQ ARTIST-ID OF ARTIST OF REQUEST.
```

Program request:

```
MOVE INPUT-ARTIST TO ARTIST-ID OF ARTIST.
OBTAIN FIRST ARTIST-LR WHERE ARTIST-ID EQ ARTIST-ID OF ARTIST OF LR
```

Comparison of OF REQUEST and OF LR use in path

The table below shows the primary differences between the action of OF REQUEST and OF LR.

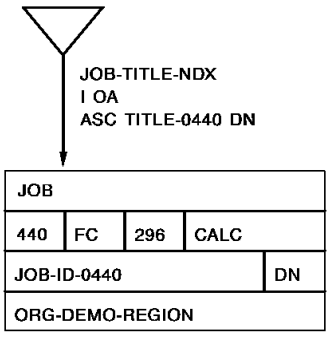
Using	Value comes from	Value at runtime	Message to LRF at runtime
OF REQUEST	Program WHERE clause	Put in LRC block	Look in LRC block
OF LR	■ Prior path retrieval ■ Path COMPUTE	Placed in LR buffer	Look in LR buffer

Specifying the key value as an arithmetic expression

You can tell LRF to use the result of an arithmetic expression as a CALC key, sort key, or db-key value. The expression can be a simple or compound arithmetic operation. The operands in the expression can be either a numeric literal or a logical-record field specified in the OF LR clause.

Example

For example, if the EMP-ID-0415 field is initialized to '0100', the following path will retrieve the JOB record occurrence where the JOB-ID-0440 field contains the value '0101'.



Path code:

```
SELECT
   OBTAIN EACH EMPLOYEE
      WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST.
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ (JOB-ID-0440 OF LR + 1).
```

Program request:

```
OBTAIN FIRST EMP-JOB-LR
   WHERE EMP-ID-0415 EQ '0100'.
```

Retrieving CALC Records

To retrieve a record based on its CALC key value, use the FIND/OBTAIN WHERE CALCKEY path-DML command. The WHERE clause of this command is mandatory and must identify the CALC key value of the appropriate record occurrence.

You specify the CALC key value in the same way that you specify any key value:

- As a literal
- As a logical-record field name, with the OF REQUEST clause
- As a logical-record field name, with the OF LR clause
- As an arithmetic expression

Retrieving Indexed Records

LRF provides a variety of ways to retrieve records that are members of an indexed set:

- **For a sorted indexed set**, you can use one of the following methods:
  - Issue a FIND/OBTAIN EACH USING INDEX command
  - Issue a FIND/OBTAIN WITHIN SET WHERE SORTKEY command
  - Issue a FIND/OBTAIN WITHIN SET command
- **For an unsorted indexed set**, you would issue a FIND/OBTAIN WITHIN SET/AREA command. You cannot use a sort key to access an unsorted indexed set.

In all cases, you will be able to add on an index at a later date without recompiling the program.

The three methods for accessing a sorted indexed set are discussed in detail below.

Using the FIND/OBTAIN EACH USING INDEX command

The FIND/OBTAIN EACH USING INDEX command directs LRF to retrieve a record occurrence based on the record occurrence's index entry. When this command is issued, LRF requests that the db-key and sort key for the specified record occurrence be returned from the DBMS. LRF then uses the db-key to retrieve the data portion of the indexed record, if needed.

This method of retrieving indexed records allows you to:

- Associate an index with a path SELECT clause to give you more flexibility in coding paths
- Build the sort key from both the path and program WHERE clauses
- Set up paths that can process complete, partial, or concatenated sort keys

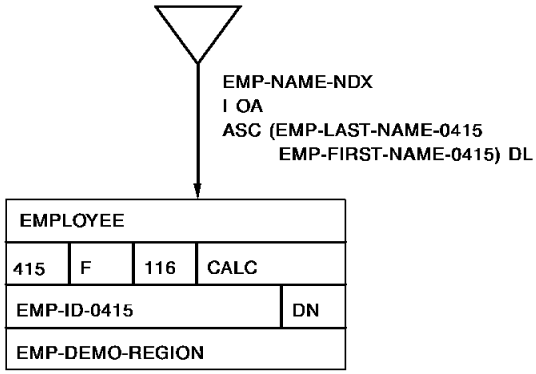
Each of these capabilities is discussed below.

Associating an index with a SELECT clause

You can associate an index with a path SELECT clause by naming the index in the SELECT statement. LRF will use the named index whenever it finds an appropriate FIND/OBTAIN EACH USING INDEX command in the path. To be appropriate, the command must specify a record that is a member of the named indexed set.

The FIND/OBTAIN EACH USING INDEX command in the path code below appropriately specifies the EMPLOYEE record, which is a member of the EMP-NAME-NDX set.

Examples



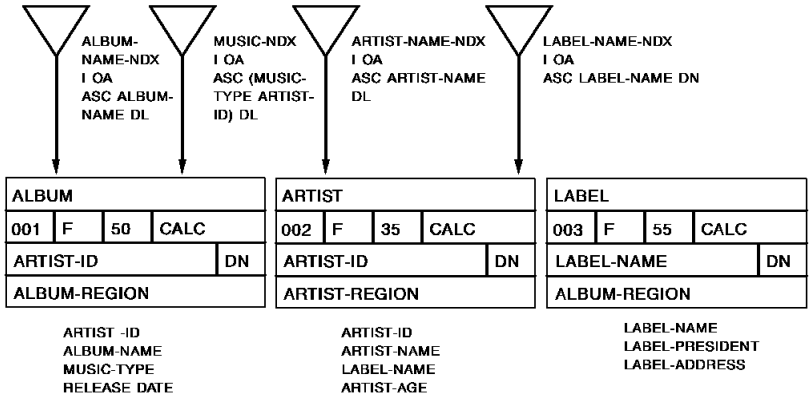
Path code:

```
SELECT USING INDEX EMP-NAME-NDX
   FOR FIELDNAME EMP-LAST-NAME-0415
   OBTAIN EACH EMPLOYEE USING INDEX.
```

Program request:

```
OBTAIN FIRST EMP-NAME-LR
   WHERE EMP-LAST-NAME-0415 EQ 'BOWER'.
ON LR-FOUND
   REPEAT.
   PUT DETAIL.
    OBTAIN NEXT EMP-NAME-LR
       WHERE EMP-LAST-NAME-0415 EQ 'BOWER'.
   END.
DISPLAY.
```

By associating an index with a SELECT clause, you can use the same path code to service each SELECT clause, as illustrated in the example below.



Path code:

```
SELECT USING INDEX ARTIST-NAME-NDX
   FOR FIELDNAME ARTIST-NAME OF ARTIST
SELECT USING INDEX LABEL-NAME-NDX
   FOR FIELDNAME LABEL-NAME OF ARTIST
   OBTAIN EACH ARTIST USING INDEX.
```

Program request:

```
OBTAIN FIRST ARTIST-LR
   WHERE ARTIST-NAME OF ARTIST MATCHES 'L'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT ARTIST-LR
         WHERE ARTIST-NAME OF ARTIST MATCHES 'L'.
   END.
```

Or

```
OBTAIN FIRST ARTIST-LR
   WHERE LABEL-NAME OF ARTIST EQ 'COLUMBIA'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT ARTIST-LR
         WHERE LABEL-NAME OF ARTIST EQ 'COLUMBIA'.
   END.
DISPLAY.
```

In this example, a single path-DML command services two separate path SELECT clauses. If you want to use a third index to process the ARTIST record, you would simply add a new path selector. You would not have to change the path-DML command or add additional path code.

Specifying the sort key

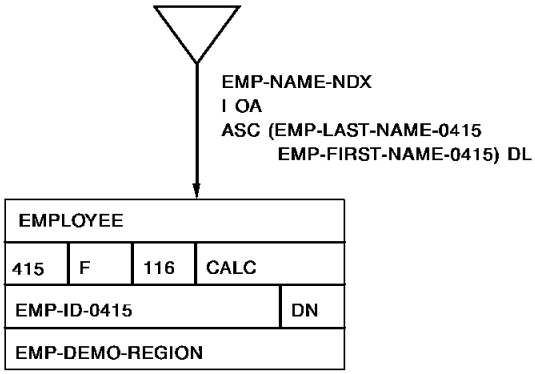
You specify a sort key value for the FIND/OBTAIN EACH USING INDEX command by using the path and program WHERE clauses. Whenever possible, LRF builds a sort key from information provided in both of these WHERE clauses. LRF builds the key as follows:

1. LRF first looks at the *path WHERE clause* (if one exists) for references to sort key fields. LRF uses these references to build a low and high range for the key.
2. LRF then looks at the *program WHERE clause* (if one exists) for references to sort key fields. LRF uses these references to update the ranges that were established in step 1.

The sort key that LRF uses must meet the sort criteria specified by both the path and program WHERE clauses. If there are no WHERE clauses specified in the path or the program, LRF walks the indexed set.

Examples

For example, suppose you have the following path code and program request:



Path code:

```
SELECT FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
   OBTAIN EACH EMPLOYEE USING EMP-NAME-NDX
      WHERE (EMP-LAST-NAME-0415  > 'H')
         AND (EMP-LAST-NAME-0415 < 'M').
```

Program request:

```
OBTAIN FIRST EMP-NAME-LR
    WHERE EMP-LAST-NAME-0415 < 'L'
WHILE LR-FOUND
   REPEAT.
      PUT DETAIL.
       OBTAIN NEXT EMP-NAME-LR
          WHERE EMP-LAST-NAME-0415 < 'L'.
   END.
DISPLAY.
```

In the above example, the sort key that LRF uses will be:

Low value	High value
H	L

If there were no program WHERE clause, the sort key would have been:

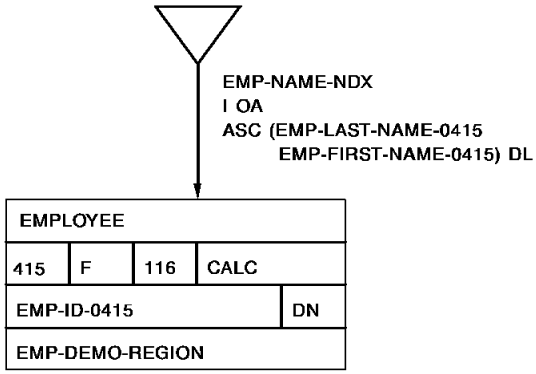
Low value	High value
H	M

Passing sort key values

You can set up paths to accept complete, partial (generic), and concatenated sort keys to be passed from the program. This depends on how you code your SELECT clauses:

- **To allow for complete and partial (generic) sort keys**, you should code a FIELDNAME selector. This selector guarantees that the path will be matched if the program request references the specified field in any manner. With this selector, the program can pass a string of characters, up to the length of the sort key.
- **To allow for concatenated sort keys**, you should code a FIELDNAME selector for each portion of the sort key.

Examples



Allowing for complete and partial sort keys

This path code allows the program to pass a string of characters, up to the length of the sort key, for the EMP-FIRST-NAME-0415 component of the EMP-NAME-NDX.

Path code:

```
SELECT USING INDEX EMP-NAME-NDX
   FOR FIELDNAME EMP-FIRST-NAME-0415
OBTAIN EACH EMPLOYEE USING INDEX.
```

Program request:

```
OBTAIN NEXT EMP-NAME-LR
   WHERE (EMP-FIRST-NAME-0415 GE 'H') AND (EMP-FIRST-NAME-0415 LT 'M').
```

Allowing for concatenated sort keys

This path code allows the program to pass a concatenated sort key for the EMP-NAME-NDX. Because a FIELDNAME selector has been associated with *each* component of the concatenated key, a program request must reference both components to match to the path.

Path code:

```
SELECT FOR FIELDNAME EMP-LAST-NAME-0415
           FIELDNAME EMP-FIRST-NAME-0415
OBTAIN EACH EMPLOYEE USING EMP-NAME-NDX INDEX.
```

Program request:

```
OBTAIN FIRST EMP-INFO-LR
   WHERE (EMP-LAST-NAME-0415 EQ 'MURDOCH') AND
       (EMP-FIRST-NAME-0415 EQ 'CATHY').
```

Index processing considerations

The presences of an OR operand in the WHERE clause — even if the syntax references only key fields — results in the index being swept:

Efficient path

```
OBTAIN EACH PLAYER USING INDEX
   WHERE LAST-NAME GE WORK-LAST-NAME OF LR
      ON 0000 DO
          EVALUATE FIRST-NAME OF LR GT
             WORK-FIRST-NAME OF LR
            ON 2001 ITERATE
            ON 0000 NEXT
              END
      ON 2001 NEXT
```

If your index is defined with KEY = *composite of elementary items*, an OBTAIN USING INDEX with the key equal to a group item which is equivalent to the composition of the elementary items, results in the index being swept.

Examples

Path code:

```
SELECT USING INDEX IX-A FOR FIELDNAME GE-KEY OF A
OBTAIN EACH A USING INDEX
```

Program request:

```
OBTAIN FIRST LR WHERE GE-KEY GE F-SEARCH-KEY1
     AND GE-KEY LE F-SEARCH-KEY2
```

If your index has concatenated keys, a program request that passes a non-high-order key only results in a full sweep of the index, with all entries being returned:

Sortkey:

```
LAST-NAME
FIRST-NAME
```

Path code:

```
OBTAIN EACH PLAYER USING INDEX
```

Program request:

```
OBTAIN FIRST PLAYER-LR WHERE FIRST-NAME EQ 'DAVID'.
```

Unpredictable results occur if a program request for a logical record compares a sortkey with a field in the same physical record. In the example below, assume HIRE-DATE is a sortkey and both fields are in the player record:

Program request:

```
OBTAIN FIRST PLAYER-LR
 WHERE HIRE-DATE EQ TERMINATION DATE.
```

A workable method of applying this type of selection criteria is with an evaluate command in the path:

Path code:

```
OBTAIN EACH PLAYER USING INDEX
EVALUATE HIRE-DATE OF LR EQ
    TERMINATION-DATE OF LR
  ON 2001 ITERATE
  ON 0000 NEXT
```

Note: LRF does *not* have enough information to build the internal high-low table needed for fencing in the index search, because the actual value for TERMINATION-DATE is not available until the player record is obtained. LRF expects the information before the return against the INDEX SORTKEY is issued.

Using the FIND/OBTAIN WITHIN SET WHERE SORTKEY command

The FIND/OBTAIN WITHIN SET WHERE SORTKEY command directs the DBMS to search the index until it finds a sort key value that matches the value specified. When a match is found, the DBMS returns the data portion of the indexed record to LRF.

Complete sort key

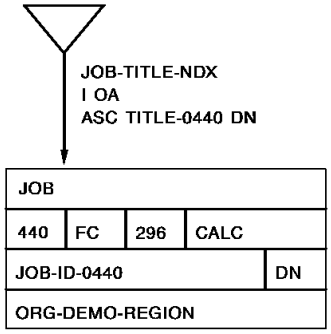
If you use this method to retrieve records in an indexed set, you must specify a *complete* sort key value. You specify a sort key value in the same way that you specify a CALC key value in path:

- As a literal coded in the path
- As a logical-record field name, with the OF REQUEST clause
- As a logical-record field name, with the OF LR clause
- As an arithmetic expression

These specifications are described earlier in this chapter, under Passing Key Values (see page 66).

Example

For example, the path shown below retrieves the first JOB record occurrence where the TITLE-0440 field is equal to 'PROGRAMMER'.



Path code:

```
SELECT FOR FIELDNAME EQ TITLE-0440 OF JOB
   OBTAIN FIRST JOB WITHIN JOB-TITLE-NDX
      WHERE SORTKEY EQ TITLE-0440 OF REQUEST.
```

Program request:

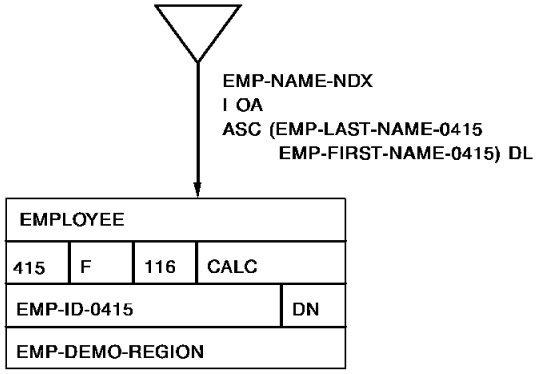
```
OBTAIN FIRST EMP-JOB-LR
   WHERE TITLE-0440 EQ 'PROGRAMMER'.
```

Using the FIND/OBTAIN WITHIN SET command

The FIND/OBTAIN WITHIN SET command directs LRF to walk the indexed set until it meets path and program selection criteria. This option is recommended in the following situations:

- If you want to retrieve all of the record occurrences in the set
- If you want to select a record occurrence on the basis of a nonkey field

Examples



Retrieving all record occurrences in an indexed set

This path retrieves all EMPLOYEE records in sorted order, based on last name.

Path code:

```
SELECT
    OBTAIN EACH EMPLOYEE WITHIN EMP-NAME-NDX.
```

Program request:

```
OBTAIN FIRST EMP-NAME-LR.
ON LR-FOUND
   REPEAT.
   PUT DETAIL.
      OBTAIN NEXT EMP-NAME-LR.
   END.
DISPLAY.
```

Selecting a record occurrence on the basis of a nonkey field

This path retrieves all employees who are on leave.

Path code:

```
SELECT FOR KEYWORD ON-LEAVE
    OBTAIN EACH EMPLOYEE WITHIN EMP-NAME-NDX
       WHERE STATUS-0415 EQ '04'.
```

Program request:

```
OBTAIN FIRST EMP-INFO-LR WHERE ON-LEAVE.
ON LR-FOUND
   REPEAT.
   PUT DETAIL.
      OBTAIN NEXT EMP-INFO-LR WHERE ON-LEAVE.
   END.
DISPLAY.
```

Retrieving Records Directly

To retrieve a record directly, based on its db-key value, use the FIND/OBTAIN WHERE DBKEY path-DML command. The WHERE clause of this command is mandatory and must identify the db-key value of the appropriate record occurrence.

You can specify a db-key value in the same way that you specify a sort key or CALC key value:

- As a literal (the db-key value must be a numeric literal)
- As a logical-record field name, with the OF REQUEST clause
- As a logical-record field name, with the OF LR clause
- As an arithmetic expression

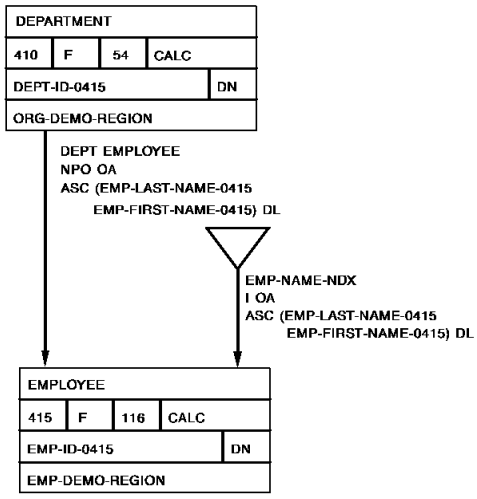
These specifications are described earlier in this chapter under Passing Key Values (see page 66).

Because the ACCEPT DB-KEY and RETURN DB-KEY statements are not valid in paths, the required db-key information is usually passed from the program. The program can:

- **Pass the db-key value to the path through any field in the logical record, including an IDD-defined work field**. The field used must be included in the subschema view. The record that contains the field must be defined as a logical-record element.
- **Issue a RETURN DB-KEY statement** if the record is indexed.
- **Issue an ACCEPT DB-KEY statement** if the subschema has a usage mode of MIXED.

Examples

For example, the following path will obtain the DEPARTMENT record occurrence where the db-key equals the value of WORK-FIELD in program variable storage.



Path code:

```
SELECT
OBTAIN FIRST DEPARTMENT
   WHERE DBKEY EQ WORK-FIELD OF REQUEST.
```

Program request (mixed mode only):

```
MOVE INPUT-ID TO EMP-ID-0415.
FIND CALC EMPLOYEE.
ACCEPT DB-KEY INTO WORK-FIELD
   FROM DEPARTMENT-EMPLOYEE CURRENCY.
OBTAIN FIRST EMP-INFO-LR
   WHERE WORK-FIELD EQ WORK-FIELD OF LR.
```

Chapter 8: Coding Path Database Update Commands

This section contains the following topics:

Introduction

You use path database update commands to update database record occurrences. These commands can only be issued in update paths (that is, STORE, MODIFY, and ERASE paths).

Path-DML statements

You can use the following path-DML statements to update database records:

- **STORE** adds a new database record occurrence to the database by using data present in program variable storage.
- **MODIFY** changes the contents of an existing database record occurrence by using data present in program variable storage.
- **ERASE** deletes the current occurrence of a database record.
- **CONNECT** establishes the current occurrence of a database record as a member of the current occurrence of a set.
- **DISCONNECT** disconnects the current occurrence of a database record from the current occurrence of a set.

Path database update commands are similar in syntax and function to CA ADS database update commands and navigational DML update commands. For a complete description of path database update commands, refer to the *CA IDMS Database Administration Guide*.

Note: For update requests, ensure that the LR buffer contains the values needed to fulfill the WHERE clause.

The remainder of this chapter describes how to store, modify, erase, connect, and disconnect database records by using LRF.

Storing Database Records

Usage

You use the STORE command to:

- Acquire space in the database and a database key for a new record occurrence
- Transfer the values of the record elements from program variable storage to the object record occurrence in the database
- Connect the object record to all sets for which it's defined as an automatic member

Conditions

Before a STORE command can be executed in LRF, the following conditions must be satisfied:

- All areas affected either directly or indirectly by the STORE command must be readied in an update usage mode.
- All fields contained in the record to be stored should be included in the subschema. Any fields that are not included in the subschema will be initialized to low values.
- The record type of the record to be stored must be defined as a logical-record element.
- All CALC and sort key fields must be initialized.
- Appropriate currencies must be established if:
  - The object record has a location mode of VIA
  - The object record participates as an automatic member of a set
  - The set order is NEXT or PRIOR

You can design paths that store database records in a variety of ways, depending on the following factors:

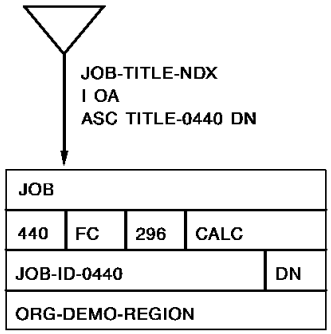
- Whether currency needs to be established in the database for associated records
- Whether you want to use LRF to implement data integrity rules

Design considerations based on currency needs are discussed below. The implementation of data integrity rules through LRF is discussed in Chapter 16, Implementing Data Integrity Rules.

When you don't have to establish currency

If currency doesn't need to be established in the database prior to the STORE, you can code a STORE path that simply stores the object record. Because the path does not require key information, you can use either a KEYWORD selector or a null SELECT clause to delimit the path.

For example, to store a new JOB record occurrence, you could code the following path:



Path code:

```
ADD
PATH-GROUP NAME IS STORE EMP-JOB-LR
   SELECT FOR KEYWORD STORE-JOB
      STORE JOB.
```

Program request:

```
MOVE WORK-JOB TO EMP-JOB-LR.
STORE EMP-JOB-LR
   WHERE STORE-JOB.
```

When you must establish currency

If currency needs to be established in the database prior to a STORE, the program will have to pass appropriate key information to the path. This information should be passed through the program WHERE clause.

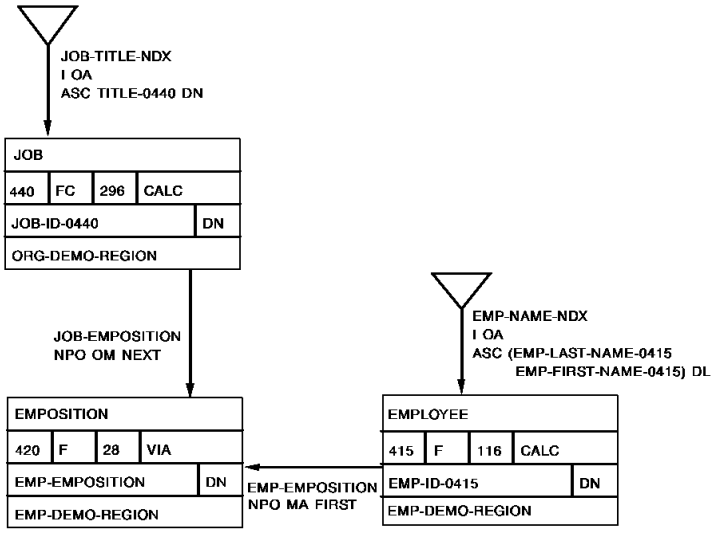
You generally establish currency by issuing an OBTAIN command in a path. There are two recommended ways to issue this command:

- Issue an OBTAIN command in an OBTAIN path. Then issue a STORE command in a separate STORE path.
- Issue an OBTAIN command in a STORE path.

In either case, you should use FIELDNAME-EQ selectors to force the program to pass the necessary key information.

Examples

These examples show two ways to store the EMPOSITION record. To store this record, you must establish currency on the appropriate EMPLOYEE record.



Establishing currency in an OBTAIN path

This path establishes currency on the appropriate EMPLOYEE record in an OBTAIN path.

Path code:

```
ADD
PATH-GROUP NAME IS OBTAIN EMP-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
       OBTAIN FIRST EMPLOYEE
          WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST.

ADD
PATH-GROUP NAME IS STORE EMP-LR
   SELECT
      FIND CURRENT EMPLOYEE
      STORE EMPOSITION.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
OBTAIN FIRST EMP-LR
   WHERE EMP-ID-0415 EQ EMP-OF-0415 OF LR.
ON LR-FOUND THEN
   STORE EMP-LR.
```

Establishing currency in a STORE path

This path establishes currency on the appropriate EMPLOYEE record in a STORE path.

Path code:

```
ADD
PATH-GROUP NAME IS STORE EMP-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      STORE EMPOSITION.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
STORE EMP-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
```

Modifying Database Records

Usage

You use the MODIFY command to replace the element values of a database record occurrence with element values that are defined in program variable storage.

Conditions

Before a MODIFY command can be executed in LRF, the following conditions must be satisfied:

- All areas affected either directly or indirectly by the MODIFY command must be readied in an update usage mode.
- All fields to be modified should be included in the subschema view.
- If a sort-key field is to be modified, the sorted set in which the object record participates must also be included in the subschema.
- The record type of the record to be modified must be defined as a logical-record element.
- The record to be modified must be established as current of run unit.

Steps

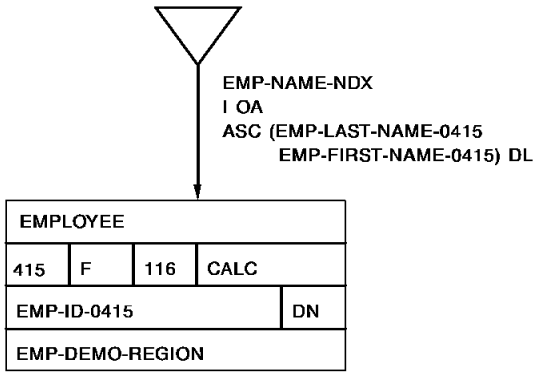
To MODIFY a record through LRF, you should follow these steps:

1. Issue an OBTAIN command in an OBTAIN path. This establishes the necessary currency and brings the object record into program variable storage.
2. Issue a MODIFY command in a MODIFY path.

If the program is passing CALC key, sort key, or db-key information, be sure to code the appropriate path selectors.

Example

For example, to modify the STATUS-0415 field of a specific EMPLOYEE record occurrence, you could code the following path:



Path code:

```
ADD
PATH-GROUP NAME IS OBTAIN EMP-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST.
ADD
PATH-GROUP NAME IS MODIFY EMP-LR
   SELECT FOR KEYWORD MOD-EMP
      FIND CURRENT EMPLOYEE
      MODIFY EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
OBTAIN FIRST EMP-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
ON LR-FOUND THEN
   DO.
      MOVE NEW-STATUS TO STATUS-0415.
      MODIFY EMP-LR
         WHERE MOD-EMP.
   END.
```

Erasing Database Records

You use the ERASE command to delete a record from the database. Erasure is a two-step process that first cancels a record's membership in any set occurrences and then releases for reuse the space occupied by the record.

Usage

The ERASE command performs the following functions:

- Disconnects the object record from all set occurrences in which it participates as a member; logically or physically deletes the record from the database
- Optionally erases all records that are mandatory members of set occurrences owned by the object record
- Optionally disconnects or erases all records that are optional members of set occurrences owned by the object record

Conditions

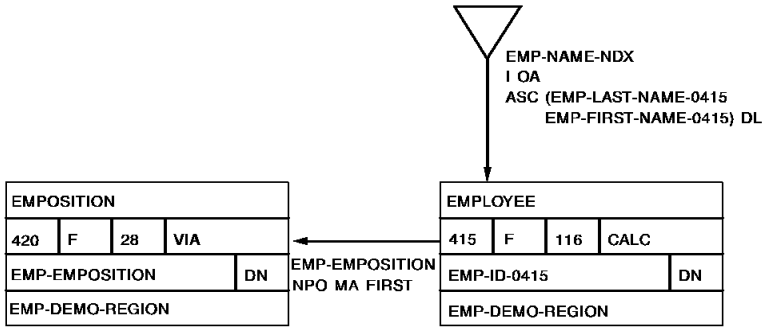
Before an ERASE command can be executed in LRF, the following conditions must be met:

- All areas either directly or indirectly affected by the ERASE command must be readied in an update usage mode.
- The record type of the record to be erased must be included in the subschema, but does not have to be defined as a logical-record element.
- All sets in which the object record participates as owner either directly or indirectly (for example, a set whose owner is a member of a set owned by the object record) and all member record types in those sets must be included in the subschema.
- If the object record participates as a member in a set with a mandatory disconnect option, the set's owner must be included in the subschema.
- The object record must be established as current of run unit.

You can design paths that ERASE database records in a variety of ways:

- Issue an OBTAIN command in an OBTAIN path to establish the necessary currency. Then erase the record by issuing an ERASE command in an ERASE path group.
- Issue the OBTAIN and ERASE commands in an ERASE path.
- Create a loop within an ERASE path group to erase all record occurrences within a set or area. In this case, a single program request may result in multiple physical ERASES.

Examples



Establishing currency in an OBTAIN path

This path erases a specific occurrence of the EMPLOYEE record.

Path code:

```
ADD
PATH-GROUP NAME IS OBTAIN EMP-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
       OBTAIN FIRST EMPLOYEE
          WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST.
ADD
PATH-GROUP NAME IS ERASE EMP-LR
   SELECT FOR KEYWORD ERASE-EMP
      FIND CURRENT EMPLOYEE
      ERASE EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
OBTAIN FIRST EMP-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
ON LR-FOUND THEN
    ERASE EMP-INFO-LR
       WHERE ERASE-EMP.
```

Establishing currency in an ERASE path

This path also erases a specific occurrence of the EMPLOYEE record.

Path code:

```
ADD
PATH-GROUP NAME IS ERASE EMP-LR
   SELECT FOR KEYWORD ERASE-EMP
              FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      ERASE EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
ERASE EMP-LR
   WHERE (EMP-ID-0415 EQ EMP-ID-0415 OF LR)
      AND ERASE-EMP.
```

Erasing records by walking a set

This path erases all EMPOSITION records within a specific EMP-EMPOSITION set occurrence. Path iteration is discussed in detail in Chapter 11, Controlling Path Execution.

Path code:

```
ADD
PATH-GROUP NAME IS ERASE EMP-LR
   SELECT FOR KEYWORD ERASE-EMPOSIT
              FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
      ERASE EMPOSITION
         ON 0000 ITERATE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
ERASE EMP-LR
   WHERE (EMP-ID-0415 EQ EMP-ID-0415 OF LR)
      AND ERASE-EMPOSIT.
```

Connecting Database Records

Usage

You use the CONNECT command to establish a record occurrence as a member of a set occurrence. Membership for the set can be defined as either optional or mandatory.

Conditions

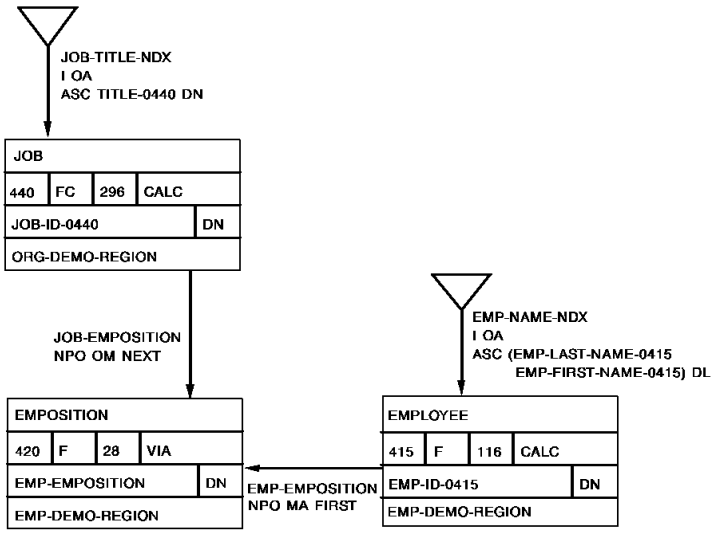
Before a CONNECT command can be executed in LRF, the following conditions must be met:

- All areas affected either directly or indirectly by the CONNECT command must be readied in an update usage mode.
- The record type of the record to be connected must be included in the subschema. This record does not have to be defined as a logical-record element.
- The named set (into which the specified record will be connected) must also be included in the subschema.
- The object record must be established as current of its record type.
- The occurrence of the set into which the specified record will be connected must be established as current of set. If set order is NEXT or PRIOR, current of set also determines the position where the specified record will be connected within the set.

In general, you would code the CONNECT command in either a STORE path or a MODIFY path, depending on how you're using it.

Examples

Connecting a record in a STORE path



This path connects the EMPOSITION record to a JOB record when the EMPOSITION record is stored.

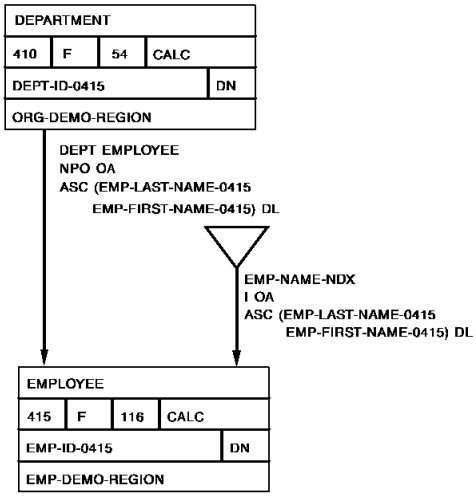
Path code:

```
ADD
PATH-GROUP NAME IS STORE EMP-JOB-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
              FIELDNAME-EQ JOB-ID-0440 OF JOB
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      STORE EMPOSITION
      FIND FIRST JOB
         WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST
      CONNECT EMPOSITION TO JOB-EMPOSITION.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
MOVE INPUT-JOB-ID TO JOB-ID-0440.
STORE EMP-JOB-LR
   WHERE (EMP-ID-0415 EQ EMP-ID-0415 OF LR)
      AND (JOB-ID-0440 EQ JOB-ID-0440 OF LR).
```

Connecting a record in a MODIFY path



This path processes the transfer of an employee by connecting the EMPLOYEE record to a new DEPARTMENT record.

Path code:

```
ADD
PATH-GROUP NAME IS MODIFY EMP-LR.
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
              FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      OBTAIN FIRST DEPARTMENT
         WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
      CONNECT EMPLOYEE TO DEPT-EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
MOVE INPUT-DEPT-ID TO DEPT-ID-0410.
MODIFY EMP-LR
   WHERE (EMP-ID-0415 EQ EMP-ID-0415 OF LR)
      AND (DEPT-ID-0410 EQ DEPT-ID-0410 OF LR).
```

Disconnecting Database Records

Usage

You use the DISCONNECT command to cancel the membership of a record occurrence in a set occurrence. Membership in the object set must be defined as optional.

Conditions

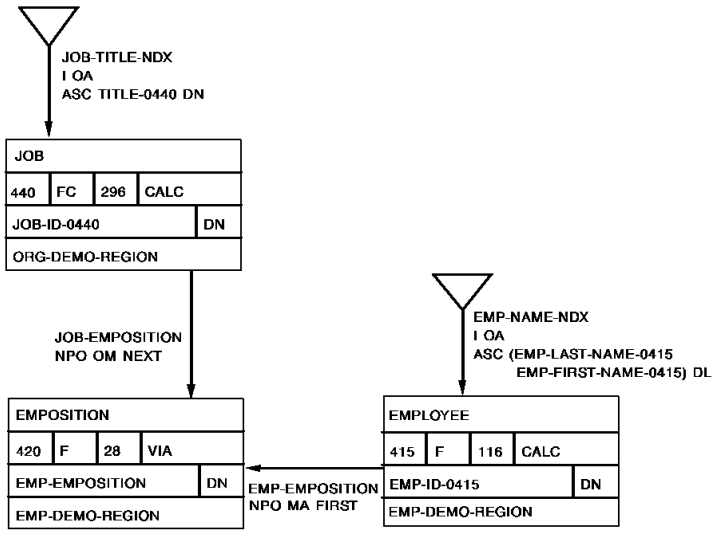
Before a DISCONNECT command can be executed in LRF, the following conditions must be met:

- All areas affected either directly or indirectly by the DISCONNECT command must be readied in an update usage mode.
- The record type of the record to be disconnected must be included in the subschema. This record does not have to be defined as a logical-record element.
- The named set (from which the specified record will be disconnected) must also be included in the subschema.
- The object record must be established as current of its record type.
- The occurrence of the set from which the specified record will be disconnected must be established as current of set.

In general, you would code the DISCONNECT command in either an ERASE path or a MODIFY path, depending on how you're using it.

Examples

Disconnecting a record in an ERASE path



This path disconnects the EMPOSITION record from a JOB record when the EMPOSITION record is erased.

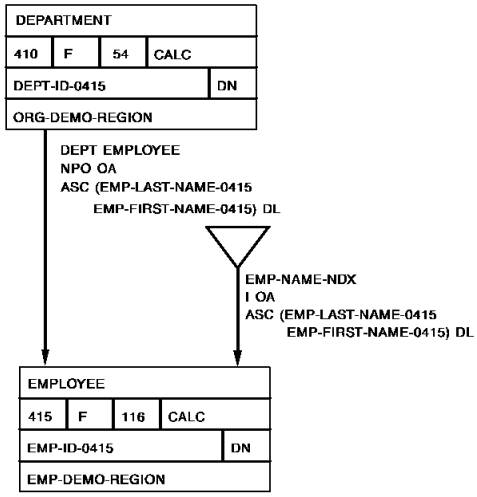
Path code:

```
ADD
PATH-GROUP NAME IS ERASE EMP-JOB-LR
  SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      FIND FIRST EMPOSITION WITHIN EMP-EMPOSITION
      FIND OWNER JOB WITHIN JOB-EMPOSITION
      DISCONNECT EMPOSITION FROM JOB-EMPOSITION
      ERASE EMPOSITION.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
ERASE EMP-JOB-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
```

Disconnecting a record in a MODIFY path



This path processes the transfer of an employee by disconnecting the EMPLOYEE record from the old DEPARTMENT record.

Path code:

```
ADD
PATH-GROUP NAME IS MODIFY EMP-LR
   SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
      OBTAIN FIRST EMPLOYEE
         WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
      FIND OWNER DEPARTMENT WITHIN DEPT-EMPLOYEE
      DISCONNECT EMPLOYEE FROM DEPT-EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
MODIFY EMP-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
```

Chapter 9: Coding Path Database Control Commands

This section contains the following topics:

Introduction

You use path database control commands to evaluate set conditions and place locks on database record occurrences. These commands can be issued in both retrieval paths and update paths.

You can use the following path-DML statements to perform database control functions:

- **IF [NOT] EMPTY** tests the current occurrence of the named set to determine whether it contains any member record occurrences.
- **IF [NOT] MEMBER** tests the record that is current of run unit to determine whether it participates as a member of the named set.
- **KEEP** places a shared or exclusive lock on the database record occurrence that is current of the named record type, set, or area.

Control command comparison

The following table compares path database control commands to CA ADS database control commands. For a complete description of path database control commands, refer to the *CA IDMS Database Administration Guide*.

Path command	Comparison	CA ADS command
IF [NOT] EMPTY	These commands are functionally equivalent.	IF SET [NOT] EMPTY
IF [NOT] MEMBER	These commands are functionally equivalent.	IF SET [NOT] MEMBER
KEEP [EXCLUSIVE]	These commands are functionally equivalent.	KEEP [EXCLUSIVE]
	There is no path command for this CA ADS command.	KEEP LONGTERM

Path command	Comparison	CA ADS command
	There is no path command for this CA ADS command.	READY
	There is no path command for this CA ADS command.	COMMIT
	There is no path command for this CA ADS command.	ROLLBACK

Note: In CA ADS, BIND and FINISH commands are performed automatically.

The remainder of this chapter describes how to check for empty-set and set-membership conditions, and how to place locks on database records.

Evaluating Empty-Set Conditions

To check for an empty-set condition, use the IF [NOT] EMPTY path-DML command. Before this command can be executed, the object set must be current of run unit and must be included in the subschema definition.

Note: Use this command when you want to check for a set with no members. To check for an end-of-set condition, it is better to use the FIND/OBTAIN NEXT WITHIN SET command.

LRF returns status codes in response to an IF [NOT] EMPTY query. You can use an ON clause to test for a particular status code and indicate what action will be taken.

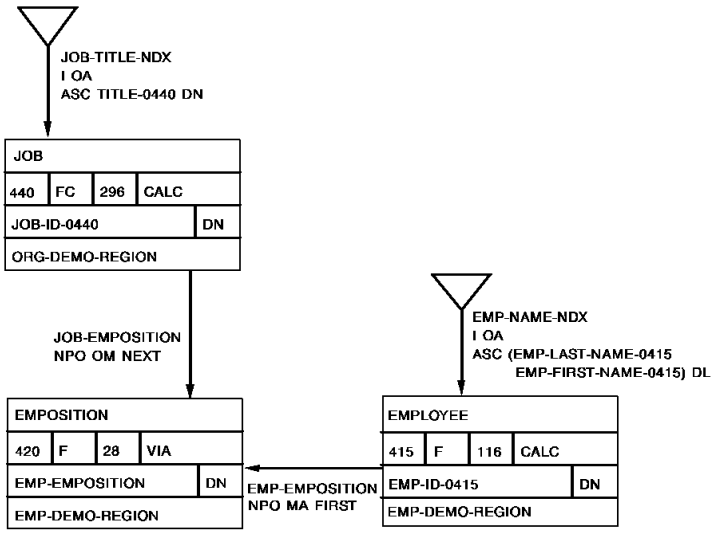
The status codes and their meanings are described below in the table below.

Status codes returned

Statement	Status code	Set condition
If *set-name* EMPTY	0000	Empty
	1601	Not empty

Example

The following example illustrates how to use the IF [NOT] EMPTY path-DML command. In this example, the path retrieves all the employees who work in a particular job. If the job has no employees, the path returns a 'JOB-NOT-FILLED' status to the program.



Path code:

```
SELECT FOR FIELDNAME-EQ JOB-ID-0440 OF JOB
   OBTAIN FIRST JOB
      WHERE CALCKEY EQ JOB-ID-0440 OF REQUEST
   IF JOB-EMPOSITION IS EMPTY
      ON 0000 RETURN 'JOB-NOT-FILLED'
      ON 1601 NEXT
   FIND EACH EMPOSITION WITHIN JOB-EMPOSITION
   OBTAIN OWNER EMPLOYEE WITHIN EMP-EMPOSITION.
```

Program request:

```
MOVE INPUT-JOB-ID TO JOB-ID-0440.
OBTAIN FIRST JOB-LR
   WHERE JOB-ID-0440 EQ JOB-ID-0440 OF LR.
WHILE LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT JOB-LR
         WHERE JOB-ID-0440 EQ JOB-ID-0440 OF LR.
   END.
DISPLAY.
```

Evaluating Set-Membership Status

To test whether a record is a member of a named set, use the IF [NOT] MEMBER path-DML command.

Conditions

The following conditions must be satisfied before this command can be executed:

- The object record must be current of run unit.
- The object record type must be included in the subschema.
- The named set must be included in the subschema.

For sets that have an optional disconnect option or a manual connect option, you should always issue an IF [NOT] MEMBER command before you issue a FIND/OBTAIN OWNER command. For these types of sets, you can never assume that the current of record or current of run unit is also current of set. For example, if you issue a FIND/OBTAIN OWNER command for a record that is *not* currently connected to the named set, you will retrieve the owner of the current record of set.

If a set is mandatory automatic, you don't have to check for an owner. Members of this set will *always* have an owner.

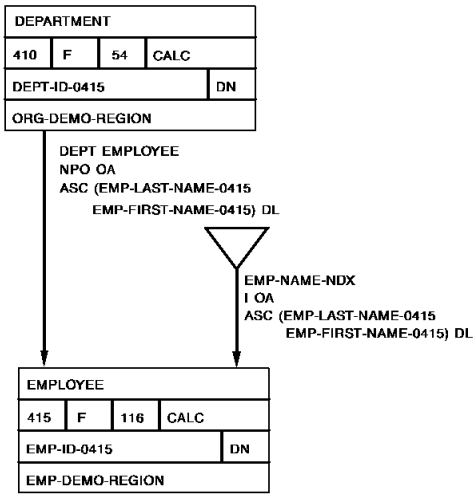
LRF returns status codes in response to an IF [NOT] MEMBER query. The status codes and their meanings are described in the following table.

Status codes returned

Statement	Status code	Member condition
If *set-name* MEMBER	0000	The current record of run unit is a member of the named set.
	1601	The current record of run unit is not a member of the named set.

Example

The following example illustrates how to use the IF [NOT] MEMBER path-DML command. In this example, the path retrieves the owner DEPARTMENT record for a particular employee. If the employee is not associated with a department, the path returns a 'NO-DEPT' path status to the program.



Path code:

```
SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
   OBTAIN FIRST EMPLOYEE
      WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
   IF DEPT-EMPLOYEE MEMBER
      ON 0000 NEXT
      ON 1601 RETURN 'NO-DEPT'
   OBTAIN OWNER DEPARTMENT WITHIN DEPT-EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
OBTAIN FIRST EMP-INFO-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
```

Locking a Database Record

If you need to place an explicit lock on a database record, use the KEEP command. The KEEP command maintains record locks for the duration of the recovery unit (that is, until released by means of the COMMIT, FINISH, or ROLLBACK statements in the program).

Alternatively, you can use the KEEP option of the FIND/OBTAIN statements to place locks on records as they are retrieved. This clause is described in the *CA IDMS Database Administration Guide*.

Note: LRF places *implicit* locks on current database record occurrences after the execution of each path database retrieval, update, or control command. This process is described in Chapter 15, Currency Considerations.

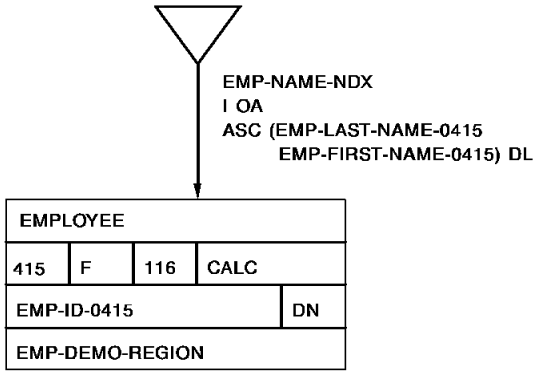
Conditions

Before the KEEP command can be executed in LRF, the following conditions must be satisfied:

- The object record must be current of the named record type, set, or area.
- The object record type, set name, and area name must be included in the subschema.

Example

The example shown below illustrates how to use the KEEP path-DML command to place an exclusive lock on an EMPLOYEE record before it is modified.



Path code:

```
ADD
PATH GROUP NAME IS OBTAIN EMP-LR
    SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
       OBTAIN FIRST EMPLOYEE WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
       KEEP EXCLUSIVE CURRENT EMPLOYEE.
ADD
PATH GROUP NAME IS MODIFY EMP-LR
    SELECT
       MODIFY EMPLOYEE.
```

Program request:

```
MOVE INPUT-EMP-ID TO EMP-ID-0415.
MOVE NEW-STATUS TO STATUS-0415.
OBTAIN FIRST EMP-LR
   WHERE EMP-ID-0415 EQ EMP-ID-0415 OF LR.
ON LR-FOUND
   THEN DO.
        MODIFY EMP-LR.
        END.
```

Chapter 10: Specifying Selection Criteria for Logical Records

This section contains the following topics:

Introduction

You can specify selection criteria for a logical record by:

- Coding a **WHERE clause** to specify attributes of the logical-record occurrence you want to access.
- Issuing an **EVALUATE command** to determine whether an expression is true or false. You can then direct LRF to perform specific path logic on the basis of this evaluation.

Using a WHERE Clause

A WHERE clause is a boolean expression that can specify one of the following:

- Criteria for selecting logical-record occurrences to be retrieved, stored, modified, or erased
- Criteria for selecting the database-records that will be used to construct the logical record

A WHERE clause can be coded in the program, in the path, or in both the program and the path. Program and path WHERE clauses are discussed below, followed by a discussion of program and path WHERE clause interactions.

Coding a program WHERE clause

Usage

The program WHERE clause serves two functions:

- **Directs the program to an appropriate path** by matching the criteria specified in the WHERE clause to the path selectors. This is described in num=6.Specifying Path Selectors.
- **Specifies selection criteria to be applied to a logical record occurrence.** The program selection criteria is evaluated *after* a logical-record occurrence has been placed in program variable storage. In the case of update verbs, the selection criteria is evaluated *before* the logical-record occurrence is moved to the database. If the selection criteria is not met, LRF returns a status of LR-NOT-FOUND.

Comparisons

The program WHERE clause consists of one or more comparisons or keywords connected by boolean operators. The following table shows the general format of the program WHERE clause. A program WHERE clause can consist of one or more operands connected by a boolean operator.

First operand	Boolean operator	Second operand
keyword	AND	keyword
[NOT] comparison		[NOT] comparison
[NOT] comparison	OR	[NOT] comparison

WHERE clause comparison format

The following table shows the suggested format of a program WHERE clause comparison. This comparison is formed by using one first operand, one conditional operator, and one second operand.

First operand	Conditional operator	Second operand
Logical-record field name	EQ (IS) (=)	Alphanumeric or numeric literal
	NE	
	GT (>)	
	LT (<)	IDD-defined variable field
	GE	
	LE	Arithmetic expression

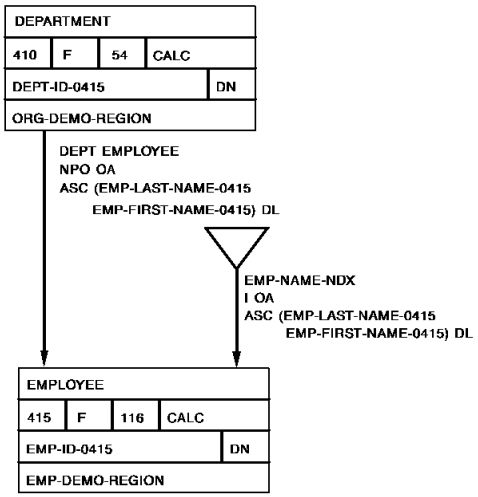
First operand	Conditional operator	Second operand
	CONTAINS MATCHES	Alphanumeric literal IDD-defined variable field Logical-record field name OF LR

For information on program WHERE clause syntax, refer to the *CA ADS User Guide*. or the *CA IDMS Navigational DML Programming Guide*.

A program WHERE clause comparison is checked against data that has been moved into program variable storage. If the path does not OBTAIN the records to be used in the comparison, LRF returns a path status of LR-NOT-FOUND.

Examples

This program request asks LRF to retrieve all employees whose status code is '01' (active), who work in a particular department.



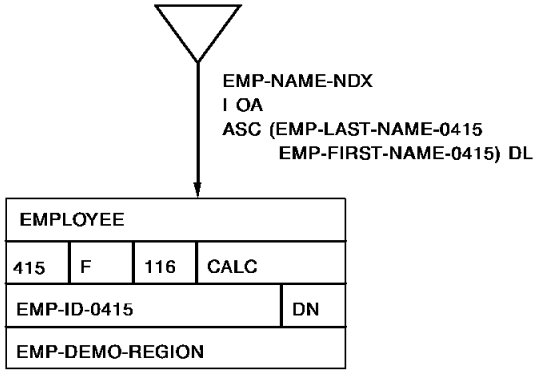
Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
   OBTAIN FIRST DEPARTMENT
      WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
   OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE.
```

Program request:

```
MOVE INPUT-DEPT-ID TO DEPT-ID-0410.
OBTAIN FIRST DEPT-LR
   WHERE (DEPT-ID-0410 EQ DEPT-ID-0410 OF LR)
       AND STATUS-0415 EQ '01'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT DEPT-LR
          WHERE (DEPT-ID-0410 EQ DEPT-ID-0410 OF LR)
             AND STATUS-0415 EQ '01'.
   END.
DISPLAY.
```

This program request asks LRF to retrieve information on all employees whose last name begins with A.



Path code:

```
SELECT USING INDEX EMP-NAME-NDX FOR FIELDNAME EMP-LAST-NAME-0415
    OBTAIN EACH EMPLOYEE USING INDEX.
```

Program request:

```
OBTAIN FIRST EMP-NAME-LR
   WHERE EMP-LAST-NAME MATCHES 'A'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT EMP-NAME-LR
          WHERE EMP-LAST-NAME MATCHES 'A'.
   END.
DISPLAY.
```

Coding a path WHERE clause

You use the path WHERE clause to specify selection criteria for the database records used to construct a logical record occurrence. Path selection criteria is evaluated *before* data is returned to program variable storage. Only those records (if any) that meet the specified selection criteria are used to construct the logical record.

Consists of comparisons

The path WHERE clause consists of one or more comparisons connected by boolean operators. To specify additional selection criteria for FIND/OBTAIN WHERE CALCKEY, FIND/OBTAIN WHERE SORTKEY, and FIND/OBTAIN WHERE DBKEY commands, you simply attach this additional criteria with an AND operator.

The following table shows the general format of the path WHERE clause. A path WHERE clause can consist of one or more comparisons connected by a boolean operator.

General format

First operand	Boolean operator	Second operand
[NOT] comparison	AND OR	[NOT] comparison

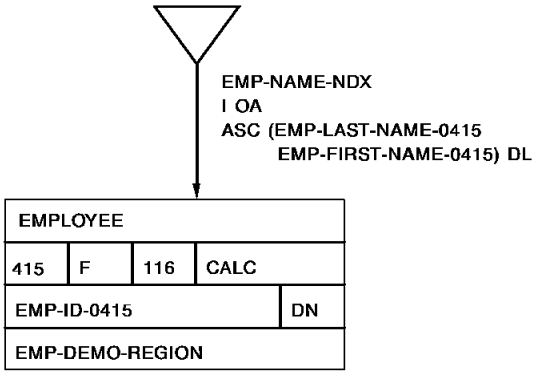
Format of a path WHERE clause comparison The following table shows the suggested format of a path WHERE clause comparison. This comparison is formed by using one first operand, one conditional operator, and one second operand.

First operand	Conditional operator	Second operand
Logical-record field name	EQ (IS) (=) NE GT (>) LT (<) GE LE	Alphanumeric or numeric literal Logical-record field name OF LR Arithmetic expression
	CONTAINS MATCHES	Alphanumeric literal Logical-record field name OF LR

For information on path WHERE clause syntax, refer to the *CA IDMS Database Administration Guide*.

Examples

This path retrieves all employees whose status code is '04'



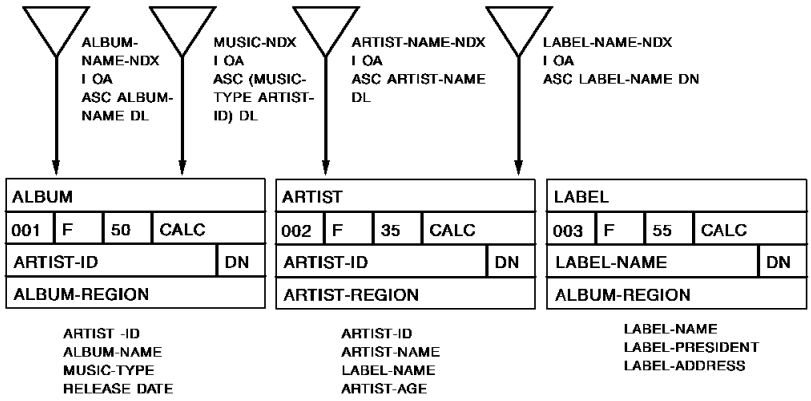
Path code:

```
SELECT USING INDEX EMP-NAME-NDX FOR KEYWORD ON-LEAVE
   OBTAIN EACH EMPLOYEE USING INDEX
      WHERE STATUS-0415 EQ '04'.
```

Program request:

```
OBTAIN FIRST EMP-INFO-LR
   WHERE ON-LEAVE.
ON LR-FOUND
    REPEAT.
       PUT DETAIL.
       OBTAIN NEXT EMP-INFO-LR
        WHERE ON-LEAVE.
   END.
DISPLAY.
```

This path retrieves all albums made by 'Baez' that were released after 12/31/84.



Path code:

```
SELECT FOR FIELDNAME-EQ ARTIST-ID OF ALBUM
    OBTAIN EACH ALBUM
        WHERE (CALCKEY EQ ARTIST-ID OF REQUEST)
            AND (RELEASE-DATE > '841231').
```

Program request:

```
OBTAIN FIRST MUSIC-LR
   WHERE ARTIST-ID OF ALBUM EQ 'BAEZ'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT MUSIC-LR
         WHERE ARTIST-ID OF ALBUM EQ 'BAEZ'.
   END.
DISPLAY.
```

Program and path WHERE clause interactions

When both the program request and the path contain a WHERE clause, LRF evaluates the WHERE clauses in the following order:

1. LRF evaluates the path WHERE clause first, before it returns a logical record to program variable storage.
2. LRF then evaluates the program WHERE clause, based on the data in program variable storage.

Allowable WHERE clause combinations

The following table summarizes the WHERE clause combinations that LRF allows. LRF allows the WHERE clause combinations described in this table. The presence of a WHERE clause is indicated by an asterisk.

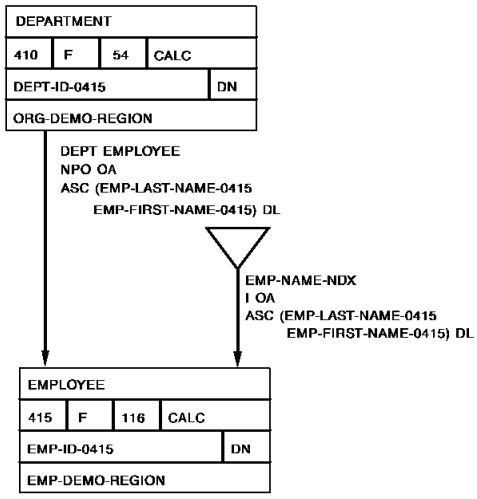
Program WHERE clause	Path WHERE clause	Action
*	*	■ Evaluates database-record occurrences according to path WHERE clause selection criteria, before data is placed in program variable storage. ■ Evaluates logical-record occurrences according to program WHERE clause selection criteria, based on the data in program variable storage.
	*	Evaluates database-record occurrences before data is placed in program variable storage.
*		Evaluates logical-record occurrences based on data in program variable storage.
		Places all data retrieved by the path into program variable storage. Does no further evaluation of this data.

Advantages Whenever possible, you should place a WHERE clause in the path, rather than the program. Placing a WHERE clause in the path offers the following advantages:

- Better efficiency in time and resources because data is only brought into program variable storage when the selection criteria is met
- Better security because programmers need not be aware of the selection criteria used
- Ease of update because the code is centralized in the path

For a discussion of these design considerations, refer to Chapter 3: (see page 21).

Examples



Example 1

This combination of path and program WHERE clauses returns employee information for each employee in the Data Processing department who started work after 12/31/80 and whose status is '01' (active).

Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410
   OBTAIN EACH DEPARTMENT
      WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
   OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      WHERE START-DATE-0415 > '801231'.
```

Program request:

```
OBTAIN FIRST DEPT-LR
    WHERE DEPT-ID-0410 EQ '5200'
        AND STATUS-0415 EQ '01'.
ON LR-FOUND
    REPEAT.
       PUT DETAIL.
       OBTAIN NEXT DEPT-LR
          WHERE DEPT-ID-0410 EQ '5200'
              AND STATUS-0415 EQ '01'.
    END.
DISPLAY.
```

Example 2

This combination of path and program WHERE clauses returns employee information for each employee in the Data Processing department who started work between 1981 and 1985 inclusive.

Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410
   OBTAIN EACH DEPARTMENT
      WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
   OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      WHERE START-DATE-0415 > '801231'.
```

Program request:

```
OBTAIN FIRST DEPT-LR
    WHERE DEPT-ID-0410 EQ '5200'
        AND START-DATE-0415 < '860101'.
ON LR-FOUND
    REPEAT.
       PUT DETAIL.
       OBTAIN NEXT DEPT-LR
          WHERE DEPT-ID-0410 EQ '5200'
              AND START-DATE-0415 < '860101'.
    END.
DISPLAY.
```

Example 3

This combination of path and program WHERE clauses returns employee information for each employee in the Data Processing department who started work during or after 1986.

Employees who started work in 1985 don't meet the selection criteria specified in the program WHERE clause.

Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
   OBTAIN EACH DEPARTMENT
      WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
   OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
      WHERE START-DATE-0415 > '851231'.
```

Program request:

```
OBTAIN FIRST EMP-INFO-LR
    WHERE DEPT-ID-0410 EQ '5200'
        AND START-DATE-0415 > '861231'.
ON LR-FOUND
    REPEAT.
       PUT DETAIL.
       OBTAIN NEXT EMP-INFO-LR
          WHERE DEPT-ID-0410 EQ '5200'
              AND START-DATE-0415 > '861231'.
    END
DISPLAY.
```

Using the EVALUATE Command

You use the EVALUATE command to determine whether a specified boolean expression is true or false, and to allow specific path logic to be performed on the basis of this evaluation.

How it differs from WHERE

The EVALUATE command differs from the path WHERE clause in the following ways:

- The selection criteria for an EVALUATE command should be applied *after* data is brought into program variable storage. This data can be returned to the program whether or not the evaluation is true.
- The EVALUATE command allows branching in the path whether or not the specified condition is met.
- One of the operands in each comparison must be specified as a logical-record field OF LR.

To EVALUATE a particular field, you should first OBTAIN the database record that contains the object field. Be sure the object field is defined in a logical-record element.

Status codes returned

LRF returns the following status codes in response to an EVALUATE command:

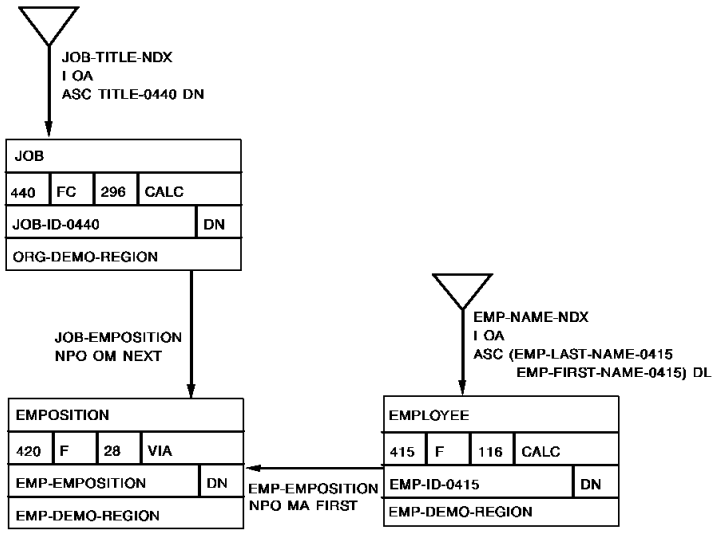
- **0000** indicates that the specified expression is true.
- **2001** indicates that the specified expression is false.

You can use an ON clause to test for a particular status code and specify what action should be taken. ON clauses are described in Chapter 13, Using Role Names.

Note: You cannot issue an EVALUATE command for a group-level element in an IDD work record. To evaluate a group-level element in an IDD work record, you must first redefine this element by using the COMPUTE statement. This procedure is described in Chapter 12, Manipulating Logical-Record Data.

Examples

This path returns employee and job information for each employee who is underpaid.



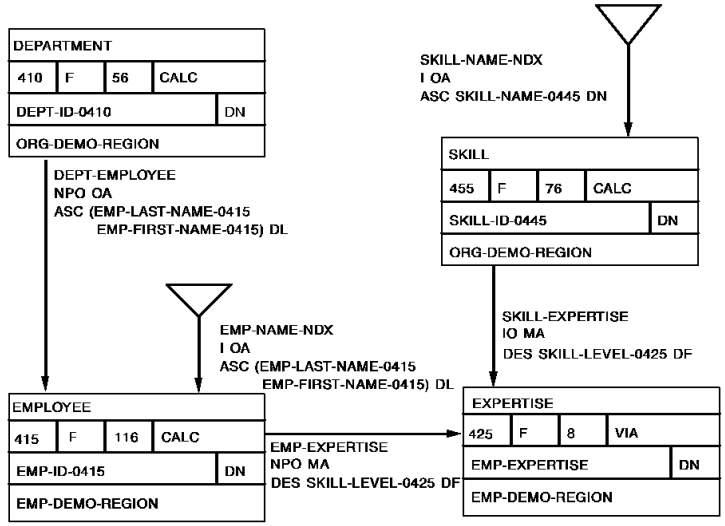
Path code:

```
SELECT FOR KEYWORD UNDERPAID
    OBTAIN EACH EMPOSITION WITHIN EMP-DEMO-REGION
       WHERE SALARY-GRADE-0420 EQ '21'
    EVALUATE SALARY-AMOUNT-0420 OF LR LT 20000
       ON 0000 DO
              OBTAIN OWNER EMPLOYEE WITHIN EMP-EMPOSITION
              FIND CURRENT EMPOSITION
              IF JOB-EMPOSITION MEMBER
                 ON 0000 NEXT
                 ON 1601 ITERATE
              OBTAIN OWNER JOB WITHIN JOB-EMPOSITION
              END
       ON 2001 ITERATE.
```

Program request:

```
OBTAIN FIRST JOB-LR
   WHERE UNDERPAID.
ON LR-FOUND
    REPEAT.
        PUT DETAIL
        OBTAIN NEXT JOB-LR
           WHERE UNDERPAID.
    END.
DISPLAY.
```

This path lists employee, expertise, and skill information for all employees in a particular department who are experts in at least one skill.



Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
    OBTAIN FIRST DEPARTMENT
       WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
    OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
    OBTAIN EACH EXPERTISE WITHIN EMP-EXPERTISE
    EVALUATE SKILL-LEVEL-0425 OF LR EQ '04'
       ON 0000 DO
              OBTAIN OWNER SKILL WITHIN SKILL-EXPERTISE
              END
       ON 2001 ITERATE.
```

Program request:

```
MOVE INPUT-DEPT-ID TO DEPT-ID-0440.
OBTAIN FIRST SKILL-LR
    WHERE DEPT-ID-0410 EQ DEPT-ID-0410 OF LR.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT SKILL-LR
         WHERE DEPT-ID-0410 EQ DEPT-ID-0410 OF LR.
   END.
DISPLAY.
```

Chapter 11: Controlling Path Execution

This section contains the following topics:

Introduction

You can control the order in which LRF executes path-DML statements by coding an ON clause. This clause checks for a particular DBMS status code and indicates what action should be taken if that status code is found. With an ON clause, you can direct LRF to do any of the following:

- Execute the next path-DML statement
- Branch within the path
- Iterate a path
- Return control to the program

Each of these options is described below, following a detailed discussion of the ON clause.

Using the ON Clause

To use the ON clause, you specify a DBMS status code and indicate what action should be taken if the DBMS returns that status code. The following table shows the format of this clause. An ON clause consists of a DBMS status code followed by a path processing request. This request is executed if the DBMS returns the indicated status code.

ON clause	Requested action
ON status-code NEXT	Tells LRF to execute the next command in the path.

ON clause	Requested action
ON status-code DO/END	Tells LRF to execute the block of path DML-commands that are nested within the ON clause. This block of commands can itself include one or more nested blocks of commands. LRF permits up to 32 levels of nested DO/END blocks.
ON status-code ITERATE	Tells LRF to reexecute the most recent, successfully executed, iterable path command. A path command is iterable if it contains an EACH option. If there is no successfully executed, iterable path command, LRF returns a path status of LR-NOT-FOUND to the program.
ON status-code [CLEAR] RETURN path-status	Tells LRF to interrupt path processing and return a particular path status to the requesting program. CLEAR directs LRF to set the contents of program variable storage to low values. If you do not specify CLEAR, the contents of program variable storage are available to the application program.

Automatically-generated ON clauses

The subschema compiler automatically generates ON clauses for all path-DML commands. You can override these ON clauses by coding ON clauses explicitly. If the DBMS returns a status code for which an ON clause is not present, LRF terminates path execution and returns a path status of LR-ERROR to the program.

The following table lists the ON clauses that the subschema compiler generates automatically for each path command.

Path command	Default ON clauses
FIND/OBTAIN WHERE DBKEY FIND/OBTAIN WHERE CALCKEY FIND/OBTAIN WITHIN SET WHERE SORTKEY FIND/OBTAIN EACH USING INDEX	ON 0000 NEXT ON 0326 ITERATE
FIND/OBTAIN WITHIN SET/AREA	ON 0000 NEXT ON 0307 ITERATE
IF SET EMPTY IF SET MEMBER	ON 0000 NEXT ON 1601 ITERATE

Path command	Default ON clauses
IF NOT EMPTY	ON 0000 ITERATE
IF NOT MEMBER **Note:** The NOT affects only the subschema compiler in generating a different flow of control.	ON 1601 NEXT
COMPUTE CONNECT DISCONNECT ERASE FIND/OBTAIN CURRENT FIND/OBTAIN OWNER WITHIN SET GET KEEP MODIFY STORE	ON 0000 NEXT
EVALUATE	ON 0000 NEXT ON 2001 ITERATE

Example

The following example shows a path whose ON clauses are generated automatically by the subschema compiler. This path obtains employee, department, and office information for a particular employee. The ON clauses are generated automatically when the path is compiled.

```
SELECT FOR FIELDNAME-EQ EMP-ID-0415
    OBTAIN FIRST EMPLOYEE
        WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
        ON 0000 NEXT
        ON 0326 ITERATE
    IF DEPT-EMPLOYEE MEMBER
        ON 0000 NEXT
        ON 1601 ITERATE
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
    FIND CURRENT EMPLOYEE
        ON 0000 NEXT
    IF OFFICE-EMPLOYEE MEMBER
        ON 0000 NEXT
        ON 1601 ITERATE
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT.
```

The following example shows the same path with some DBA-defined ON clauses. This path is similar to the one above, except that it contains some DBA-defined ON clauses. The DBA-defined ON clauses are shaded.

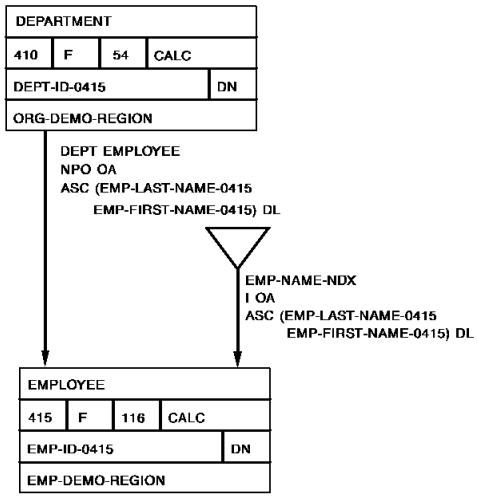
```
SELECT FOR FIELDNAME-EQ EMP-ID-0415
    OBTAIN FIRST EMPLOYEE
        WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
        ON 0000 NEXT
        ON 0326 ITERATE
    IF DEPT-EMPLOYEE MEMBER
        ON 0000 NEXT
        ON 1601 RETURN NO-DEPT
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
    FIND CURRENT EMPLOYEE
        ON 0000 NEXT
    IF OFFICE-EMPLOYEE MEMBER
        ON 0000 NEXT
        ON 1601 RETURN NO-OFFICE
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT.
```

Executing the Next Path-DML Command

What it does

The ON...NEXT clause directs LRF to process the next command in the path. If LRF encounters another ON clause, it ignores that clause and goes on to execute the next command. If LRF has successfully executed the last command in the path when it encounters an ON...NEXT clause, LRF returns a path status of LR-FOUND to the program.

The following example illustrates how to use the ON...NEXT clause. In this example, the path returns department and employee information for all employees whose status is '04' (on leave). Note that the ON clauses in this example are generated automatically by the subschema compiler.



Path code:

```
SELECT FOR KEYWORD ON-LEAVE
    OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
       WHERE STATUS-0415 EQ '04'
          ON 0000 NEXT
          ON 0307 ITERATE
    IF DEPT-EMPLOYEE MEMBER
       ON 0000 NEXT
       ON 1601 ITERATE
    OBTAIN OWNER DEPARTMENT WITHIN DEPT-EMPLOYEE
       ON 0000 NEXT.
```

Program request:

```
OBTAIN FIRST DEPT-LR
    WHERE ON-LEAVE.
ON LR-FOUND
    REPEAT.
       PUT DETAIL.
       OBTAIN NEXT DEPT-LR
          WHERE ON-LEAVE.
    END.
DISPLAY.
```

Branching Within a Path

ON...DO/END clause

You may want LRF to execute one or more path-DML commands only under certain conditions, based on the status code returned when a previous command is executed. You can do this by placing the path-DML commands within an ON...DO/END clause. If the DBMS returns the status code specified in the ON...DO/END clause, LRF processes the block of commands included in this statement.

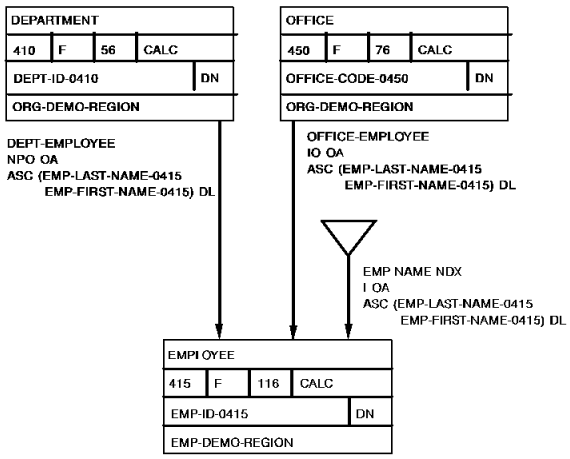
In some cases you may want to implement a multiple-alternative decision structure in a path. To implement this type of structure, you would code two or more ON...DO/END clauses and associate each of these clauses with a different status code.

Example

The example below illustrates how to use the ON...DO/END clause to branch within a path. In this example, the path returns the following information:

- For all employees assigned to a department *and* an office, the path returns employee, department, and office information.
- For all employees assigned to an office but not to a department, the path returns employee and office information.
- For all employees assigned to a department but not to an office, the path returns employee and department information.

The path returns no information for an employee who is not assigned to a department and an office.



Path code:

```
SELECT FOR KEYWORD ALL-EMP
    OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
    IF DEPT-EMPLOYEE MEMBER
        ON 0000 DO
                OBTAIN OWNER WITHIN DEPT-EMPLOYEE
                FIND CURRENT EMPLOYEE
                IF OFFICE-EMPLOYEE MEMBER
                ON 0000 DO
                        OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                        END
                ON 1601 NEXT
                END
        ON 1601 DO
                IF OFFICE-EMPLOYEE MEMBER
                ON 0000 DO
                        OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                        END
                ON 1601 ITERATE
                END.
```

Program request:

```
OBTAIN FIRST EMP-LR
   WHERE ALL-EMP.
ON LR-FOUND
    REPEAT.
        PUT DETAIL.
        OBTAIN NEXT EMP-LR
           WHERE ALL-EMP.
    END.
DISPLAY.
```

Iterating a Path

What iteration is

Path iteration is the process by which LRF reexecutes a block of code. LRF iterates a path when it performs the following activities:

- Walks a set
- Sweeps an area
- Accesses database record occurrences that contain duplicate CALC key or sort key values

When is a command iterable

A command is iterable only if it specifies the EACH option. LRF recognizes this option as the beginning of iterative code. The iterable commands are:

- FIND/OBTAIN EACH WHERE CALCKEY
- FIND/OBTAIN EACH WHERE SORTKEY
- FIND/OBTAIN EACH USING INDEX
- FIND/OBTAIN EACH WITHIN SET/AREA

There are three events that can trigger iteration within a path:

- The use of NEXT in a program request
- An unsuccessful attempt to meet program WHERE clause selection criteria
- An ON...ITERATE clause in the path

The logic of path iteration is described below, followed by separate discussions on triggering path iteration from a program and from a path.

Path iteration logic

Path iteration occurs in a cycle that involves the following steps:

1. **LRF locates the last successfully executed iterable command in the path**. This is called the **iteration point**. If no iterable command exists, LRF terminates processing and returns an LR-NOT-FOUND path status to the program.
2. **LRF passes the located command to the DBMS**.
3. **The DBMS**:
    a. **Reexecutes the command**, based on the currencies established by the previous execution of the command.
    b. **Places the new database record occurrence into the appropriate area of program variable storage** if the iterable command is an OBTAIN command. This overlays the data placed there during the previous execution of the path.
4. **LRF passes each subsequent path-DML command to the DBMS**. The DBMS executes these commands. For any OBTAIN commands, the DBMS continues to overlay data in program variable storage.

LRF continues to iterate the path by repeating steps 1 through 4 until it either:

- Finds a record occurrence that meets the selection criteria specified by the program
- Encounters an appropriate ON...RETURN clause

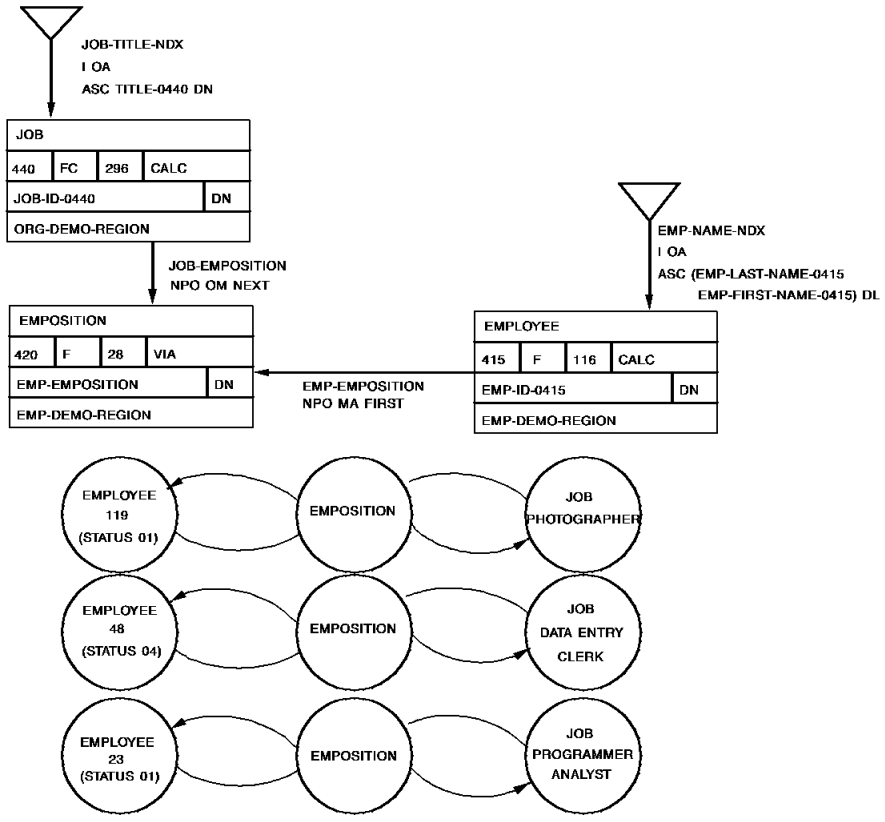
- Encounters an ON...ITERATE command, and there is nothing left to iterate
- Encounters an ON...NEXT command that's at the end of the path

LRF automatically saves appropriate db-keys after the execution of each path retrieval, update, or control command. When LRF enters an iteration cycle, it uses the appropriate db-key to restore currency for the record associated with the iterable command. For more information on currency in LRF, refer to Chapter 15, Currency Considerations.

Triggering iteration from the program

The program can trigger path iteration by specifying an OBTAIN NEXT logical-record command. This command is used when the program expects to retrieve more than one occurrence of a logical record.

Examples



Example 1

This program request asks LRF to retrieve each occurrence of the EMP-JOB-LR logical record. (This example assumes that each employee has only one job.) The information returned to the program is shown below for each path iteration.

Logical record	First iteration	Second iteration	Third iteration
EMP-JOB-LR			
EMPLOYEE			
EMP-ID-0415	0119	0048	0023
EMP-LAST-NAME-0415	BOWER	TERNER	O'HEARN
STATUS-0415	01	04	01
JOB			
JOB-ID-0440	4023	3051	3025
TITLE-0440	PHOTO-GRAPHER	DATA ENTRY CLERK	PROGRAMMER / ANALYST

Path code:

```
SELECT
    OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
    FIND FIRST EMPOSITION WITHIN EMP-EMPOSITION
    IF JOB-EMPOSITION MEMBER
        ON 0000 NEXT
        ON 1601 ITERATE
    OBTAIN OWNER JOB WITHIN JOB-EMPOSITION.
```

Program request:

```
OBTAIN FIRST EMP-JOB-LR.
ON LR-FOUND
    REPEAT.
        PUT DETAIL.
        OBTAIN NEXT-JOB-LR.
    END.
DISPLAY.
```

Example 2

This program request asks LRF to retrieve each occurrence of the EMP-JOB-LR logical record for those employees who are on leave. The information returned to the program is shown below for the first path iteration.

Logical record	First iteration
EMP-JOB-LR	
EMPLOYEE	
EMP-ID-0415	0048
EMP-LAST-NAME-0415	TERNER
STATUS-0415	04
JOB	
JOB-ID-0440	3051
TITLE-0440	DATA ENTRY CLERK

Path code:

```
SELECT
    OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
    FIND FIRST EMPOSITION WITHIN EMP-EMPOSITION
    IF JOB-EMPOSITION MEMBER
        ON 0000 NEXT
        ON 1601 ITERATE
    OBTAIN OWNER JOB WITHIN JOB-EMPOSITION.
```

Program request:

```
OBTAIN FIRST EMP-JOB-LR
    WHERE STATUS-0415 EQ '04'.
ON LR-FOUND
    REPEAT.
        PUT DETAIL.
        OBTAIN NEXT-JOB-LR
           WHERE STATUS-0415 EQ '04'.
    END.
DISPLAY.
```

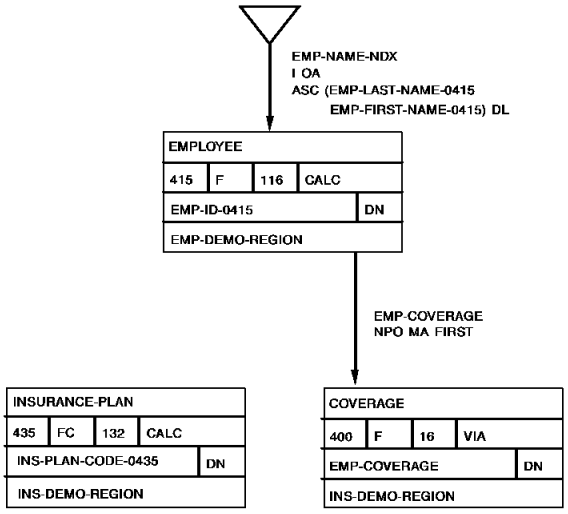
Triggering iteration from the path

You can trigger iteration from the path by using the ON...ITERATE clause. You should trigger iteration from the path if you want to:

- **Respond to an unsuccessful execution of a path-DML command** by iterating past a record. This directs LRF to continue processing without passing control back to the program.
- **Create a loop for erasing or modifying database record occurrences**. Because the program cannot initiate iteration for an update path, you must initiate iteration in the path.

Example 1

Responding to an unsuccessful path-DML command execution



This path returns insurance plan information for each employee who has family coverage. In this example, iteration is triggered from both the path and the program:

- The path-triggered iteration reexecutes path processing when the coverage type is not F (family) and when the specified insurance plan is not found.
- The program-triggered iteration reexecutes path processing so the program can receive all occurrences of the EMP-INSURANCE-LR logical record.

The ON clauses shown here are generated automatically by the subschema compiler.

Path code:

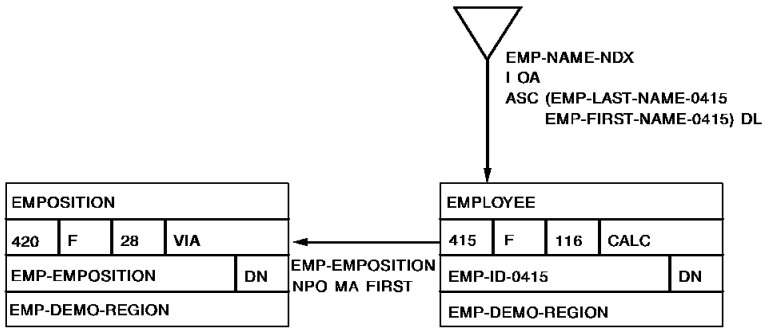
```
SELECT FOR KEYWORD FAMILY
  OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
      ON 0000 NEXT
      ON 0307 ITERATE
  OBTAIN EACH COVERAGE WITHIN EMP-COVERAGE
     WHERE TYPE-0400 EQ 'F'
         ON 0000 NEXT
         ON 0307 ITERATE
  OBTAIN FIRST INSURANCE-PLAN
     WHERE CALCKEY EQ INS-PLAN-CODE-0400 OF LR
         ON 0000 NEXT
         ON 0326 ITERATE.
```

Program request:

```
OBTAIN FIRST EMP-INSURANCE-LR WHERE FAMILY.
ON LR-FOUND
   REPEAT.
        PUT DETAIL.
        OBTAIN NEXT EMP-INSURANCE-LR WHERE FAMILY.
   END.
DISPLAY.
```

Example 2

Creating a loop to erase database record occurrences



This path erases all EMPOSITION records within a particular EMP-EMPOSITION set occurrence. The path-triggered iteration obtains the next EMPOSITION record after a successful erase.

Path code:

```
ADD
PATH-GROUP NAME IS ERASE EMP-JOB-LR
SELECT FOR KEYWORD ERASE-EMP
          FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
        OBTAIN FIRST EMPLOYEE
            WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
        FIND EACH EMPOSITION WITHIN EMP-EMPOSITION
            ON 0000 NEXT
            ON 0307 ITERATE
        ERASE EMPOSITION
            ON 0000 ITERATE.
```

Program request:

```
ERASE FIRST EMP-JOB-LR
   WHERE EMP-ID-0415 EQ '0050' AND ERASE-EMP.
```

Returning Control To the Program

LRF automatically terminates path processing and returns control to the application program when either of the following conditions is met:

- A logical-record request has executed successfully
- A logical-record request cannot be processed

System-defined path status

When LRF terminates path processing, it returns a **system-defined path status** to the program.

DBA-defined path status

You can direct LRF to return control to the application *before* path processing is complete by defining your own path statuses. Any path status that is *not* generated automatically by the subschema compiler is known as a **DBA-defined path status**.

System-defined path statuses and DBA-defined path statuses are described below, followed by a discussion of partial and complete logical records.

Using system-defined path statuses

Types of path system-defined path statuses

There are three system-defined path statuses that can be returned to the program automatically when path processing terminates:

- **LR-FOUND** is returned when the logical-record request has been processed successfully. When LR-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000. Your place in the iteration cycle is maintained.
- **LR-NOT-FOUND** is returned when the requested logical record can't be constructed for one of the following reasons:
  - There is no logical-record occurrence that satisfies the program WHERE clause
  - All occurrences of the requested logical record have already been returned

  When LR-NOT-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000. Your place in the iteration cycle is lost.
- **LR-ERROR** is returned when a logical-record request is issued incorrectly or when a processing error occurs in the path. When LR-ERROR is returned, the ERROR-STATUS field of the IDMS communications block contains one of the following:
  - A status code with a major code of 20. This usually indicates an error in the program request.
  - A status code with a major code from 00 to 19. This usually indicates an error in the path.

  Your place in the iteration cycle is lost.

The application programmer should always test for these path statuses after each logical-record request. For information on the program logic used to test path statuses, refer to the *CA IDMS Navigational DML Programming Guide*.

Using DBA-defined path statuses

You can define your own path statuses by coding an ON...RETURN clause for each appropriate DBMS status code. When LRF returns a DBA-defined path status, the ERROR-STATUS field of the IDMS communications block contains 0000. Your place in the iteration cycle is maintained.

Defining your own path statuses can be advantageous if you use the path statuses to:

- **Inform the program that LRF is returning a partial logical record**. Partial logical records are discussed later in this chapter.
- **Distinguish between the two cases of LR-NOT-FOUND**. To do this, you can:
  - Inform the program that a required database record is missing
  - Inform the program that all logical-record occurrences have already been retrieved

The use of DBA-defined LRF path statuses can result in bind errors during deadlock handling in CA ADS. If an LRF subschema path detects a deadlock condition and returns a DBA-defined path status to the CA ADS dialog which issued the LR request, the error status field is initialized to '0000'. The CA ADS deadlock handling logic issues an automatic re-bind of the run unit on the next functional DML request, only when the error status minor code equals '29'. Allowing the path status for deadlock conditions to default to LR-ERROR ensures that the error status returned will be 'XX29'.

Partial and complete logical records

LRF always tries to construct a **complete logical record** when processing a path to retrieve a logical-record occurrence. To do this, LRF returns data for all of the logical-record elements, as specified by the path. If LRF succeeds in constructing a complete logical record, it returns an LR-FOUND path status to the program.

At times, a path retrieves a logical-record occurrence for which some of the logical-record elements are not in the database. If LRF cannot construct a complete logical record, it returns a status of LR-NOT-FOUND or LR-ERROR, along with a **partial logical record**. A partial logical record contains data only for those logical-record elements that LRF was able to retrieve (if any).

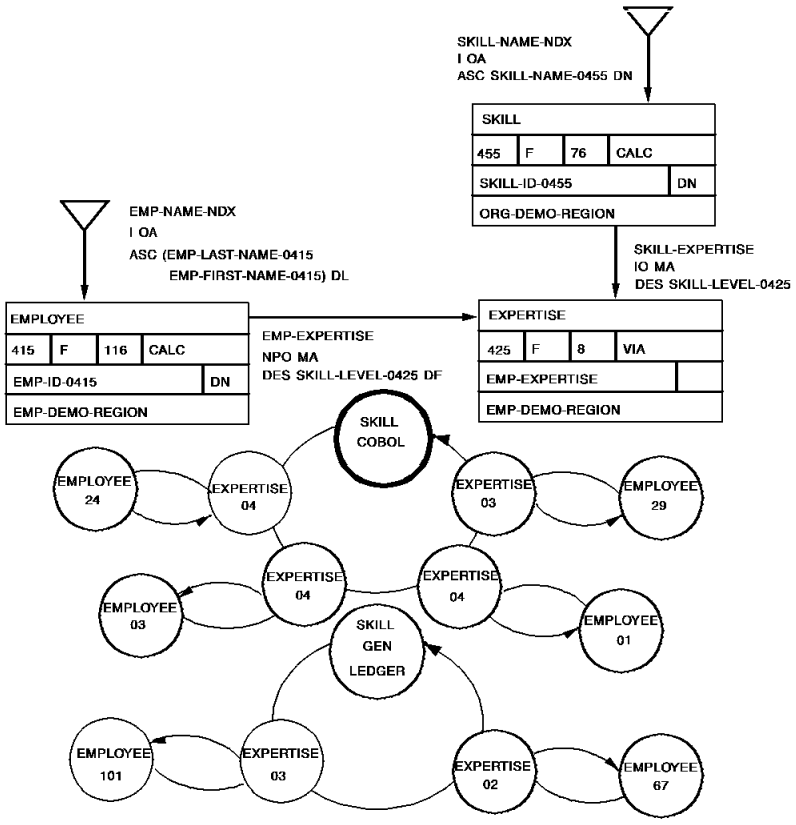
Considerations

The following considerations apply to partial and complete logical records:

- **If you want a program to access a partial logical record**, you should code your own path status. This status can perform the following functions:
  - Alert the program that a partial logical record is being returned.
  - Describe the partial logical record. This way, the program can execute different code, depending on the type of data returned.
- **If a partial logical record is returned to the program**, new values for some logical-record fields are not placed in program variable storage. As a result, these fields may contain data that is left over from a previous database record retrieval.

  There are two ways to ensure that data in the logical record area of program variable storage is accurate:
  - **If you want the programmer to have access to a partial logical record**, you can initialize the unused portions of the logical record by using the COMPUTE command. This command is described in Manipulating Logical-Record Data (see page 149).
  - **If you want the programmer to have access to a complete logical record only**, you can direct LRF to clear the logical-record area of program variable storage when a complete logical record cannot be returned. You can do this by:
    - Specifying ON LR-NOT-FOUND CLEAR and ON LR-ERROR CLEAR in the LOGICAL RECORD DDL statement, as described in Chapter 4 (see page 33).
    - Coding a DBA-defined path status that includes the CLEAR option.

Examples



Example 1

Allowing access to partial logical records

This path lists employee, skill, and expertise information for each employee who is an expert in a particular skill. If the skill does not have any experts, LRF returns a partial logical record that contains skill information only and a path status of NO-EXPERTS.

This path uses the COMPUTE command to clear the unused portions of the logical record. The contents of program variable storage are shown below for each of the following skill names:

- COBOL
- GEN LEDGER

Contents of variable storage for COBOL

Logical record	First iteration	Second iteration	Third iteration	Fourth iteration
EMP-SKILL-LR				
SKILL				
SKILL-NAME-0455	COBOL	COBOL	COBOL	COBOL
SKILL-ID-0455	2010	2010	2010	2010
EXPERTISE				
SKILL-LEVEL-0425	04	04	04	04
EXPERTISE-DATE-0425	720128	700525	611230	611230
EMPLOYEE				
EMP-ID-0415	0024	0003	0001	0001
EMP-NAME-0415	JANE DOUGH	JENNIFER GARLAND	JOHN RUPEE	JOHN RUPEE
PATH STATUS	LR-FOUND	LR-FOUND	LR-FOUND	NO-MORE-EXPERTS

Contents of variable storage for skill GEN LEDGER

Logical record	First iteration
EMP-SKILL-LR	
SKILL	
SKILL-NAME-0455	GEN LEDGER
SKILL-ID-0455	4490
EXPERTISE	
SKILL-LEVEL-0425	00
EXPERTISE-DATE-0425	000000

Logical record	First iteration
EMPLOYEE	
EMP-ID-0415	0000
EMP-NAME-0415	
PATH STATUS	NO-EXPERTS

Path code:

```
SELECT USING SKILL-NAME-NDX FOR FIELDNAME-EQ SKILL-NAME-0455 OF SKILL
    OBTAIN EACH SKILL USING INDEX
       ON 0000 NEXT
       ON 0326 ITERATE
    OBTAIN EACH EXPERTISE WITHIN SKILL-EXPERTISE
       WHERE SKILL-LEVEL-0425 EQ '04'
          ON 0000 NEXT
          ON 0307 DO
                  EVALUATE SKILL-LEVEL-0425 EQ '04'
                  ON 0000 RETURN NO-MORE-EXPERTS
                  ON 2001 DO
                          COMPUTE SKILL-LEVEL-0425 OF LR EQ '0'
                          ON 0000 NEXT
                          COMPUTE EXPERTISE-DATE-0425 OF LR EQ '0'
                          ON 0000 NEXT
                          COMPUTE EMP-ID-0415 OF LR EQ '0'
                          ON 0000 NEXT
                          COMPUTE EMP-NAME-0415 OF LR EQ ' '
                          ON 0000 RETURN NO-EXPERTS
                          END
                  END
   OBTAIN OWNER EMPLOYEE WITHIN EMP-EXPERTISE.
```

Program request:

```
MOVE INPUT SKILL-NAME TO SKILL-NAME-0425.
OBTAIN FIRST SKILL-LR
    WHERE SKILL-NAME-0455 EQ SKILL-NAME-0425 OF LR.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT SKILL-LR
         WHERE SKILL-NAME-0425 EQ SKILL-NAME-0425 OF LR.
   END.
ON NO-EXPERTS
    DISPLAY MSG TEXT IS 'NO EXPERTS FOR THIS SKILL'.
ON NO-MORE-EXPERTS
    DISPLAY MSG TEXT IS 'NO MORE EXPERTS FOR THIS SKILL'.
```

Example 2

Allowing access to complete logical records only — method 1

This path also lists employee, skill, and expertise information for each employee who is an expert in a particular skill. If the skill does not have any experts, LRF returns a DBA-defined return code of LR-NOT-FOUND but does not return any logical-record data.

This path specifies a clear option to ensure that only complete logical records are returned to the program. The contents of program variable storage are shown below for the COBOL and GEN LEDGER skill names.

Contents of variable storage for skill COBOL

Logical record	First iteration	Second iteration	Third iteration	Fourth iteration
EMP-SKILL-LR				
SKILL				
SKILL-NAME-0455	COBOL	COBOL	COBOL	
SKILL-ID-0455	2010	2010	2010	
EXPERTISE				
SKILL-LEVEL-0425	04	04	04	
EXPERTISE-DATE-0425	720128	700525	611230	
EMPLOYEE				
EMP-ID-0415	0024	0003	0001	
EMP-NAME-0415	JANE DOUGH	JENNIFER GARLAND	JOHN RUPEE	
PATH STATUS	LR-FOUND	LR-FOUND	LR-FOUND	NO-MORE-EXPERTS

Contents of variable storage for skill GEN LEDGER

Logical record	First iteration
EMP-SKILL-LR	
SKILL	

Logical record	First iteration
SKILL-NAME-0455 SKILL-ID-0455	
EXPERTISE SKILL-LEVEL-0425 EXPERTISE-DATE-0425	
EMPLOYEE EMP-ID-0415 EMP-NAME-0415	
PATH STATUS	LR-NOT-FOUND

Path code:

```
ADD
LOGICAL RECORD NAME IS EMP-SKILL-LR
    ELEMENTS ARE
        SKILL
        EXPERTISE
        EMPLOYEE
    ON LR-ERROR CLEAR
    ON LR-NOT-FOUND CLEAR
    .
    .
    .
ADD
PATH GROUP NAME IS OBTAIN EMP-SKILL-LR
    SELECT USING SKILL-NAME-NDX FOR FIELDNAME-EQ SKILL-NAME-0455
    OF SKILL
        OBTAIN EACH SKILL USING INDEX
           ON 0000 NEXT
           ON 0326 ITERATE
        OBTAIN EACH EXPERTISE WITHIN SKILL-EXPERTISE
           WHERE SKILL-LEVEL-0425 EQ '04'
              ON 0000 NEXT
              ON 0307 RETURN LR-NOT-FOUND
        OBTAIN OWNER EMPLOYEE WITHIN EMP-EXPERTISE
           ON 0000 NEXT.
```

Program request:

```
MOVE INPUT SKILL-NAME TO SKILL-NAME-0425.
OBTAIN FIRST SKILL-LR
    WHERE SKILL-NAME-0455 EQ SKILL-NAME-0425 OF LR.
ON LR-FOUND
   REPEAT.
   PUT DETAIL.
    OBTAIN NEXT SKILL-LR
        WHERE SKILL-NAME-0425 EQ SKILL-NAME-0425 OF LR.
    END.
ON LR-NOT-FOUND
    DISPLAY MSG TEXT IS 'NO MORE EXPERTS FOR THIS SKILL'.
```

Example 3

Allowing access to complete logical records only — method 2

Like the paths described above, this path lists employee, skill, and expertise information for each employee who is an expert in a particular skill. If the skill does not have any experts, LRF returns a DBA-defined return code of NO-EXPERTS but does not return any logical-record data.

This path specifies an ON..CLEAR..RETURN option to ensure that only complete logical records are returned to the program. The contents of program variable storage are shown below for the COBOL and GEN LEDGER skill names.

Contents of variable storage for skill COBOL

Logical record	First iteration	Second iteration	Third iteration	Fourth iteration
EMP-SKILL-LR				
SKILL				
SKILL-NAME-0455	COBOL	COBOL	COBOL	
SKILL-ID-0455	2010	2010	2010	
EXPERTISE				
SKILL-LEVEL-0425	04	04	04	
EXPERTISE-DATE-0425	720128	700525	611230	

Logical record	First iteration	Second iteration	Third iteration	Fourth iteration
EMPLOYEE				
EMP-ID-0415	0024	0003	0001	
EMP-NAME-0415	JANE DOUGH	JENNIFER GARLAND	JOHN RUPEE	
PATH STATUS	LR-FOUND	LR-FOUND	LR-FOUND	NO-MORE-EXPERTS

Contents of variable storage for skill GEN LEDGER

Logical record	First iteration
EMP-SKILL-LR	
SKILL	
SKILL-NAME-0455	
SKILL-ID-0455	
EXPERTISE	
SKILL-LEVEL-0425	
EXPERTISE-DATE-0425	
EMPLOYEE	
EMP-ID-0415	
EMP-NAME-0415	
PATH STATUS	NO-EXPERTS

Path code:

```
ADD
LOGICAL RECORD NAME IS EMP-SKILL-LR
    ELEMENTS ARE
        SKILL
        EXPERTISE
        EMPLOYEE
    .
    .
    .
ADD
PATH GROUP NAME IS OBTAIN EMP-SKILL-LR
    SELECT USING SKILL-NAME-NDX FOR FIELDNAME-EQ SKILL-NAME-0455
    OF SKILL
        OBTAIN EACH SKILL USING INDEX
           ON 0000 NEXT
           ON 0326 ITERATE
        OBTAIN EACH EXPERTISE WITHIN SKILL-EXPERTISE
           WHERE SKILL-LEVEL-0425 EQ '04'
              ON 0000 NEXT
              ON 0307 DO
                      EVALUATE SKILL-LEVEL-0425 EQ '04'
                         ON 0000 CLEAR RETURN NO-MORE-EXPERTS
                         ON 2001 CLEAR RETURN NO-EXPERTS
                      END
        OBTAIN OWNER EMPLOYEE WITHIN EMP-EXPERTISE
           ON 0000 NEXT.
```

Program request:

```
MOVE INPUT SKILL-NAME TO SKILL-NAME-0425.
OBTAIN FIRST SKILL-LR
    WHERE SKILL-NAME-0455 EQ SKILL-NAME-0425 OF LR.
ON LR-FOUND
   REPEAT.
   PUT DETAIL.
    OBTAIN NEXT SKILL-LR
        WHERE SKILL-NAME-0425 EQ SKILL-NAME-0425 OF LR.
    END.
ON NO-EXPERTS
    DISPLAY MSG TEXT IS 'NO EXPERTS FOR THE SKILL'.
ON NO-MORE-EXPERTS
    DISPLAY MSG TEXT IS 'NO MORE LOGICAL-RECORD OCCURRENCES'.
```

Chapter 12: Manipulating Logical-Record Data

This section contains the following topics:

The COMPUTE Command

What COMPUTE does

You can manipulate logical-record data by using the COMPUTE command. This command lets you:

- Copy data from one logical-record field to another logical-record field
- Set the value of a logical-record field equal to a literal or an arithmetic expression

The COMPUTE command is issued as a boolean expression that contains an equality operation. The following table shows the general format of this command. This command is formed by using one first operand, one conditional operator, and one second operand.

First operand	Conditional operator	Second operand
Logical-record field name OF LR	EQ (IS) (=)	Alphanumeric or numeric literal Logical-record field name OF LR Arithmetic expression

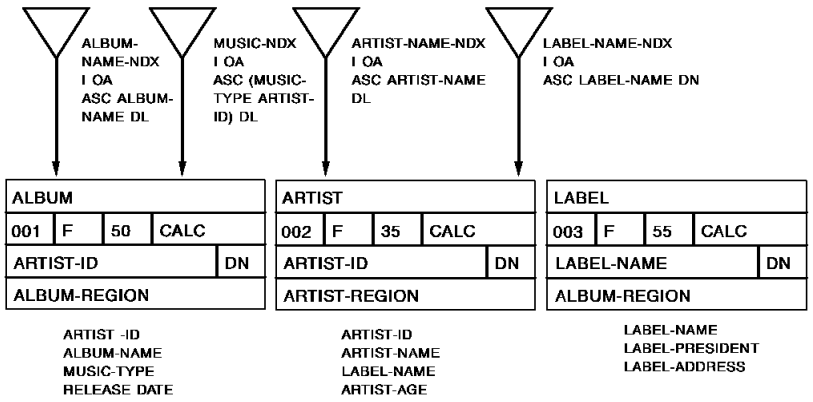
Uses of the COMPUTE command

You can use the COMPUTE command to do any of the following:

- Initialize a logical-record field that will be used as a sort key or CALC key for a subsequent FIND/OBTAIN command
- Prepare a database record occurrence to be stored or modified
- Count record occurrences and cumulate totals
- Change a group-level field into an element-level field
- Initialize unused fields when a partial logical record is returned
- Return path information to the program

Examples

Initializing a field to be used as a sort key



This path retrieves information for all jazz albums that were made by a particular musician. It retrieves the appropriate ALBUM records by using a concatenated sort key. The sort key is built in the path and contains the following information:

- A MUSIC-TYPE of JAZZ
- An ARTIST-ID that is passed through the program WHERE clause

The path constructs this key by using the COMPUTE command to move values into an IDD-defined work field. The work field is shown below:

```
01  WORK-KEY.
    02  WORK-MUSIC-TYPE    PIC X(20).
    02  WORK-ARTIST-ID     PIC 9(4).
```

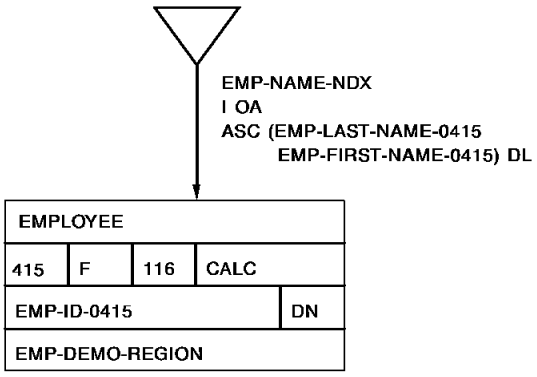
Path code:

```
SELECT FOR FIELDNAME-EQ ARTIST-ID OF ARTIST
           KEYWORD JAZZ
    COMPUTE WORK-MUSIC-TYPE OF LR EQ 'JAZZ'
    FIND FIRST ARTIST
       WHERE CALCKEY EQ ARTIST-ID OF ARTIST OF REQUEST
    COMPUTE WORK-ARTIST-ID OF LR EQ ARTIST-ID OF ARTIST OF LR
    OBTAIN EACH ALBUM
       WHERE SORTKEY EQ WORK-KEY OF LR.
```

Program request:

```
MOVE INPUT-ARTIST TO ARTIST-ID OF ARTIST.
OBTAIN FIRST MUSIC-LR
    WHERE (ARTIST-ID OF ARTIST EQ ARTIST-ID OF ARTIST OF LR)
       AND JAZZ.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT MUSIC-LR
         WHERE (ARTIST-ID OF ARTIST EQ ARTIST-ID OF ARTIST OF LR)
         AND JAZZ.
   END.
DISPLAY
```

Preparing a database record to be modified



This path modifies all EMPLOYEE records whose zip code is '01118' and who live in the city of Springfield. The path changes the zip code to '01119' for these record occurrences:

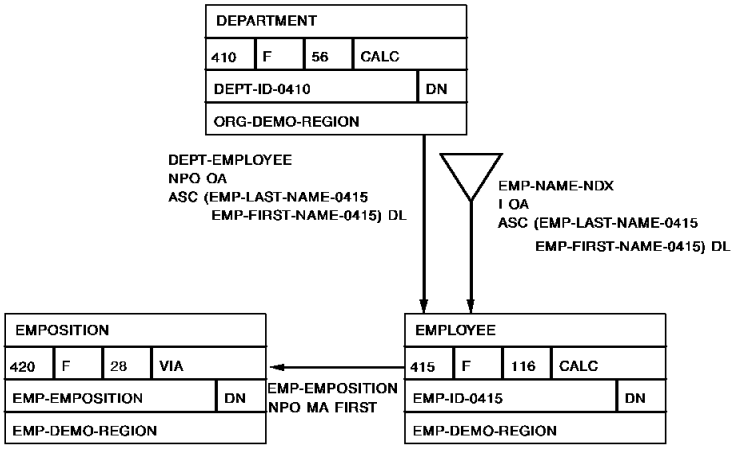
Path code:

```
ADD
PATH-GROUP NAME IS MODIFY EMP-ADDRESS-LR
   SELECT FOR KEYWORD NEW-ZIP
      OBTAIN EACH EMPLOYEE WITHIN EMP-NAME-NDX
          WHERE (EMP-ZIP-0415 EQ '01118')
          AND (EMP-CITY-0415 EQ 'SPRINGFIELD')
      COMPUTE EMP-ZIP-0415 OF LR EQ '01119'
      MODIFY EMPLOYEE
         ON 0000 ITERATE.
```

Program request:

```
MODIFY EMP-ADDRESS-LR
    WHERE NEW-ZIP.
```

Providing counter activity



This path cumulates salary totals for all employees who work in a specified department.

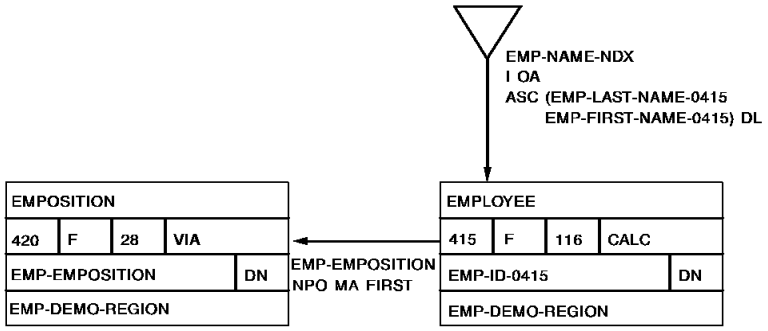
Path code:

```
SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
    COMPUTE WORK-SALARY OF LR EQ 0
    OBTAIN FIRST DEPARTMENT
        WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
    OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
    ON 0000 DO
    OBTAIN FIRST EMPOSITION WITHIN EMP-EMPOSITION
       ON 0000 DO
           COMPUTE WORK-SALARY-AMOUNT OF LR EQ
               (WORK-SALARY-AMOUNT OF LR + SALARY-AMOUNT-0420 OF LR)
           ON 0000 ITERATE
       END
       ON 0307 ITERATE
    END.
```

Program request:

```
OBTAIN FIRST TOTAL-SALARY-LR
    WHERE DEPT-ID-0410 EQ '3200'.
```

Changing group-level fields into element-level fields



This path increases the bonus percentage for each employee who has been with the company since April 31, 1985. The path uses a COMPUTE command to copy the value of START-DATE-0415 (a group-level field) into WORK-START-DATE (an element-level IDD-defined work field). The path then evaluates WORK-START-DATE to select the appropriate employees.

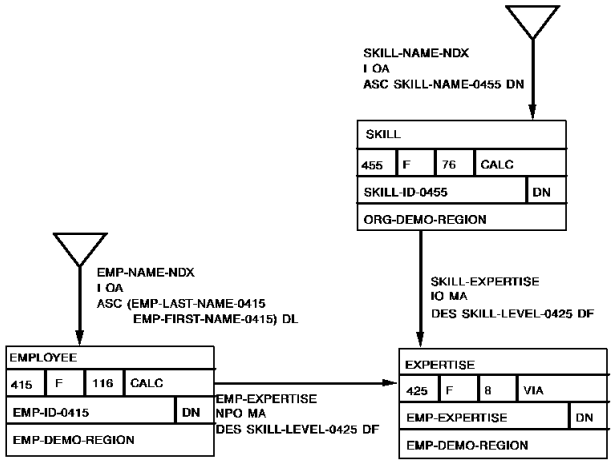
Path code:

```
SELECT FOR KEYWORD BONUS-REVIEW
    OBTAIN EACH EMPLOYEE WITHIN EMP-DEMO-REGION
    COMPUTE WORK-START-DATE OF LR EQ START-DATE-0415 OF LR
    EVALUATE WORK-START-DATE LE '850431'
    ON 0000 DO
      OBTAIN EACH EMPOSITION WITHIN EMP-EMPOSITION
         ON 0307 ITERATE.
      COMPUTE BONUS-PERCENT-0420 OF LR EQ
         (BONUS-PERCENT-0420 OF LR * 1.25)
      MODIFY EMPOSITION
         ON 0000 ITERATE.
    END
    ON 2001 ITERATE
```

Program request:

```
MODIFY EMP-LR WHERE BONUS-REVIEW.
DISPLAY.
```

Initializing unused fields



This path lists employee, skill, and expertise information for each employee who is an expert in a particular skill. If there are no 'experts' for the skill, LRF returns a partial logical record that contains skill information only.

The path uses the COMPUTE statement to initialize the fields in the EXPERTISE record before the logical record is returned to the program. For information on partial logical records, refer to Chapter 11, Controlling Path Execution.

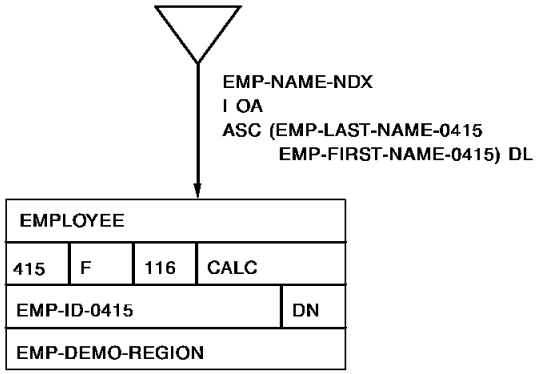
Path code:

```
SELECT USING SKILL-NAME-NDX FOR FIELDNAME-EQ SKILL-NAME-0455 OF SKILL
    OBTAIN EACH SKILL USING INDEX
    OBTAIN EACH EXPERTISE WITHIN SKILL-EXPERTISE
       WHERE SKILL-LEVEL-0425 EQ '04'
    ON 0000 DO
       OBTAIN OWNER EMPLOYEE
    END
    ON 0307 DO
            COMPUTE SKILL-LEVEL-0425 OF LR EQ '0'
            COMPUTE EXPERTISE-DATE-0425 OF LR EQ '0'
            COMPUTE EMP-ID-0415 OF LR EQ '0'
            COMPUTE EMP-NAME-0415 OF LR EQ ' '
            END
```

Program request:

```
MOVE INPUT-SKILL-NAME TO SKILL-NAME-0425.
OBTAIN FIRST SKILL-LR
    WHERE SKILL-NAME-0455 EQ 'CODING'
ON LR-FOUND
    REPEAT.
        PUT DETAIL.
        OBTAIN NEXT SKILL-LR
           WHERE SKILL-NAME-0425 EQ SKILL-NAME OF LR.
    END.
DISPLAY.
```

Returning path information to the program



This path returns information to the program that indicates which path was executed. The path returns this information through an IDD-defined work record. It uses the COMPUTE command to set the value of this work record.

Path code:

```
SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
   COMPUTE WORK-MESSAGE EQ 'EMP-ID PATH'
   OBTAIN FIRST EMPLOYEE
      WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST.
```

Program request:

```
OBTAIN FIRST EMP-LR
    WHERE EMP-ID-0415 EQ '0015'.
```

Chapter 13: Using Role Names

This section contains the following topics:

Role Names

What is a role name

You can direct LRF to return two or more occurrences of a single database record type or IDD-defined work record type simultaneously. You do this by assigning unique identifiers to the logical-record elements. These identifiers are called **role names**.

Each logical-record element can have any number of role names. LRF reserves additional space in program variable storage for each role name that you assign.

Suppose, for example, that you want to reserve space in program variable storage for two occurrences of a record. To do this, you can either:

- **Define one role name for the logical-record element**. You then refer to the element by either the role name or the record name.
- **Define two role names for the logical-record element**. You then refer to the element by its role names only.

Considerations

The following considerations apply to the use of roles names:

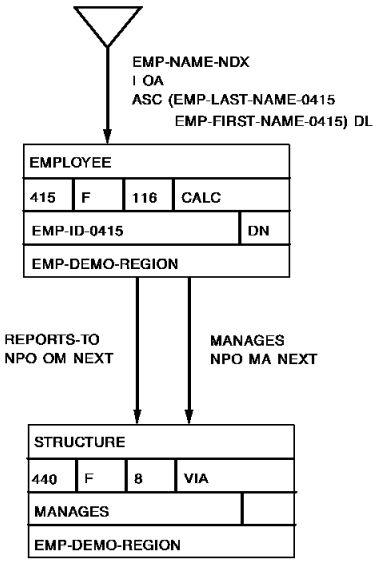
- Once you assign role names to a logical-record element, you must always qualify that element with one of its role names. This holds true for both the path code and the program request.
- A role name *cannot* be the name of:
  - A record or record synonym defined in the schema
  - A field name included in the current subschema
  - A keyword used in the current subschema
- Each role name can be assigned to only one record type per subschema. The role name can be assigned to that record type in any number of logical records within the subschema.

- LRF does not keep separate currencies for each role name specified, except during an iteration cycle. If you issue a path database update command with a role name, LRF updates the record occurrence that is current of record type.

  At the end of path execution, currency for the logical-record element will reflect the last record occurrence accessed, regardless of the role name used. For more information on currency in LRF, refer to Chapter 15, Currency Considerations.

Examples

Processing a bill of materials structure (using two role names)



This path retrieves the names and ids of all employees who work for a particular manager. The EMPLOYEE record is represented by two role names: MANAGER and WORKER. Thus, you can access the EMPLOYEE record by specifying either MANAGER or WORKER.

Because the EMP-ID-0415 field is not unique in the EMP-EMP-LR logical record, you must qualify it with the MANAGER or WORKER role names.

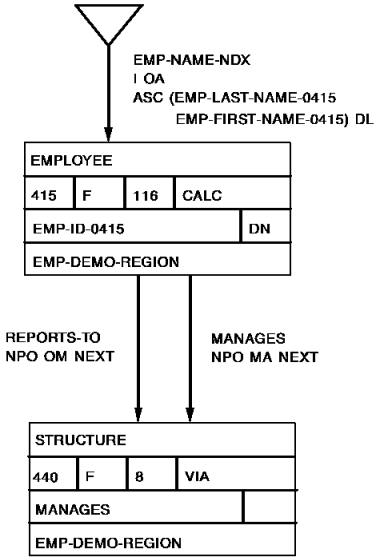
Path code:

```
ADD
LOGICAL RECORD NAME IS EMP-EMP-LR
   ELEMENTS ARE
      EMPLOYEE ROLE IS MANAGER
      EMPLOYEE ROLE IS WORKER
    .
    .
    .
ADD
PATH-GROUP NAME IS OBTAIN EMP-EMP-LR
    SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF MANAGER
        OBTAIN FIRST MANAGER
           WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
           ON 0000 NEXT
           ON 0326 ITERATE
        FIND EACH STRUCTURE WITHIN MANAGES
           ON 0000 NEXT
           ON 0307 ITERATE
        OBTAIN OWNER WORKER WITHIN REPORTS-TO
           ON 0000 NEXT.
```

Program request:

```
OBTAIN FIRST EMP-EMP-LR
   WHERE EMP-ID-0415 OF MANAGER EQ '0015'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT EMP-EMP-LR
         WHERE EMP-ID-0415 OF MANAGER EQ '0015'.
   END.
DISPLAY.
```

Processing a bill of materials structure (using one role name)



This path also retrieves the names and ids of all employees who work for a particular manager. The EMPLOYEE record has only one role name assigned to it: WORKER. Thus, you can access the EMPLOYEE record by specifying either EMPLOYEE or WORKER.

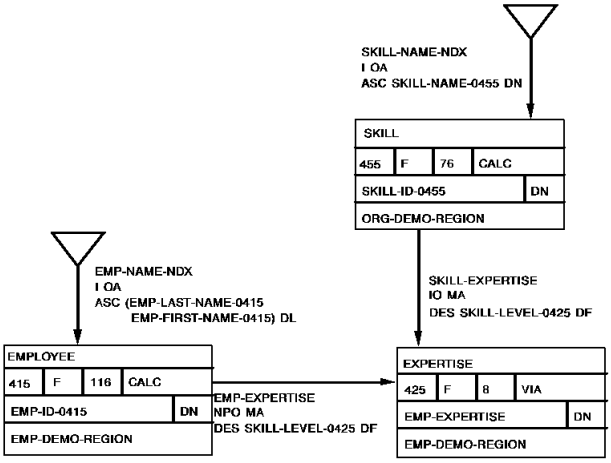
Path code:

```
ADD
LOGICAL RECORD NAME IS EMP-EMP-LR
   ELEMENTS ARE
      EMPLOYEE
      EMPLOYEE ROLE IS WORKER
    .
    .
    .
ADD
PATH-GROUP NAME IS OBTAIN EMP-EMP-LR
    SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
        OBTAIN FIRST EMPLOYEE
           WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
           ON 0000 NEXT
           ON 0326 ITERATE
        FIND EACH STRUCTURE WITHIN MANAGES
           ON 0000 NEXT
           ON 0307 ITERATE
        OBTAIN OWNER WORKER WITHIN REPORTS-TO
           ON 0000 NEXT.
```

Program request:

```
OBTAIN FIRST EMP-EMP-LR
   WHERE EMP-ID-0415 OF EMPLOYEE EQ '0015'.
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT EMP-EMP-LR
         WHERE EMP-ID-0415 OF EMPLOYEE EQ '0015'.
   END.
DISPLAY.
```

Representing multiple skills



This path retrieves skill information for each employee who is an expert in two particular skills. The logic of the path is as follows:

1. The path obtains the first SKILL record by using a value passed from the program. This SKILL record is represented by the role name SKILL1.
2. The path then finds the first EXPERTISE record in the SKILL-EXPERTISE set where the SKILL-LEVEL-0425 field has a value of 04 (expert).
3. The path obtains the owner EMPLOYEE record. This retrieves the name of the employee who has an expertise in the named skill.
4. Now, the path searches the EMP-EXPERTISE set for an EXPERTISE record that has a skill level of 04.
5. When the path locates an EXPERTISE record that has a skill level of 04, it goes on to locate the owner SKILL record. This SKILL record is represented by the role name SKILL2.

   The path obtains the SKILL record if the skill name is equal to the second skill name passed from the program. In this case, the employee is an expert in both given skills.

   If the skill name is not equal to the second skill name passed by the program, the path continues to search the EMP-EXPERTISE set for an EXPERTISE record that has an expertise level of 04 and is associated with the named skill.

Path code:

```
ADD
LOGICAL RECORD NAME IS SKILL-SKILL-LR
   ELEMENTS ARE
      SKILL ROLE IS SKILL1
      SKILL ROLE IS SKILL2
      EMPLOYEE
      EXPERTISE
   .
   .
   .
ADD
PATH-GROUP NAME IS OBTAIN SKILL-SKILL-LR
   SELECT FOR KEYWORD EXPERTS
      FIELDNAME-EQ SKILL-NAME-0455 OF SKILL1
      FIELDNAME-EQ SKILL-NAME-0455 OF SKILL2
      OBTAIN FIRST SKILL1 WITHIN SKILL-NAME-NDX
         WHERE SORTKEY EQ SKILL-NAME-0455 OF SKILL1 OF REQUEST
         ON 0000 NEXT
         ON 0326 RETURN SKILL-NOT-FOUND
      FIND EACH EXPERTISE WITHIN SKILL-EXPERTISE
         WHERE SKILL-LEVEL-0425 EQ '04'
         ON 0000 NEXT
         ON 0307 ITERATE
      OBTAIN OWNER EMPLOYEE WITHIN EMP-EXPERTISE
         ON 0000 NEXT
      FIND EACH EXPERTISE WITHIN EMP-EXPERTISE
         WHERE SKILL-LEVEL-0425 EQ '04'
         ON 0000 NEXT
         ON 0307 ITERATE
      OBTAIN OWNER SKILL2 WITHIN SKILL-EXPERTISE
         ON 0000 NEXT
         ON 0326 ITERATE.
```

Program request:

```
MOVE 'COBOL' TO SKILL-NAME-0455 OF SKILL1 OF LR
MOVE 'GEN LEDGER' TO SKILL-NAME-0455 OF SKILL2 OF LR.
OBTAIN FIRST SKILL-SKILL-LR
   WHERE EXPERTS AND SKILL-NAME-0455 OF SKILL1 EQ
         SKILL-NAME-0455 OF SKILL1 OF LR
     AND SKILL-NAME-0455 OF SKILL2 EQ
         SKILL-NAME-0455 OF SKILL2 OF LR
ON LR-FOUND
   REPEAT.
      PUT DETAIL.
      OBTAIN NEXT SKILL-SKILL-LR
         WHERE EXPERTS AND SKILL-NAME-0455 OF SKILL1 EQ
               SKILL-NAME-0455 OF SKILL1 OF LR
           AND SKILL-NAME-0455 OF SKILL2 EQ
               SKILL-NAME-0455 OF SKILL2 OF LR
   END.
DISPLAY.
```

Chapter 14: Documenting the Subschema

This section contains the following topics:

Introduction

Once you have finished defining a logical-record subschema, you are ready to document how the subschema works. At this stage of the subschema definition process, you should already have preliminary comments that list and describe the following information for each logical record in the subschema:

- The database records that the logical record will access
- The DML verbs that can be issued for the logical record
- The program selection criteria that will map to each path

For information on defining preliminary comments, refer to Chapter 4: (see page 33).

At this stage, you may also want to describe:

- The data that will be returned to the program, including the sequence of the data
- All DBA-defined path statuses and the situations under which these path statuses will be returned
- Any role names associated with the logical record

The COMMENTS clause

You document a logical record by using the COMMENTS clause of the ADD LOGICAL RECORD DDL statement. The information you provide will be copied along with the logical record to the appropriate area of program variable storage. You can retrieve this information either by displaying the logical record in the subschema compiler or by running the LRDEFS logical-record report.

The remainder of this chapter describes how to use the COMMENTS clause and how to run the LRDEFS, LRPATH, and LRACT logical-record reports.

Using the COMMENTS Clause

To use the COMMENTS clause of the ADD LOGICAL RECORD DDL statement, you enter as many lines of text as are necessary to document the logical record. The following considerations apply:

- Each line of text must start with a quote. Ending quotes are optional.
- When text extends beyond the first line of input, each subsequent line must begin with a character that indicates either continuation or concatenation:
  - **The hyphen (-)** indicates that the line is a continuation of comment text.
  - **The plus sign (+)** indicates that the line is to be appended to the previous line of comment text.

Example

The following example shows how to code comments for the sample EMP-INFO-LR logical record.

```
ADD
SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
    .
    .
    .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
    .
    .
    .
    COMMENTS
        '*****************************************************************'
  -     'THE EMP-INFO-LR LOGICAL RECORD ACCESSES INFORMATION FROM THE'
  -     'EMPLOYEE DATABASE RECORD AND ALSO ACCESSES INFORMATION'
  -     'FROM THE ASSOCIATED DEPARTMENT AND OFFICE RECORDS.'
  -     ' '
  -     'THE FOLLOWING INFORMATION IS RETURNED TO THE PROGRAM, IN'
  -     'THE ORDER SHOWN BELOW:'
  -     ' '
  -     '  EMPLOYEE RECORD   -- EMP-ID-0415, EMP-NAME-0415, START-DATE-0'
  +     '415'
  -     '                       STATUS-0415'
  -     ' '
  -     '  DEPARTMENT RECORD -- DEPT-ID-0440, DEPT-NAME-0440'
  -     ' '
  -     '  OFFICE RECORD     -- OFFICE-CODE-0450'
  -     '  PATHREC VERSION 1 -- WORK-PATH-ID'
```

```
-     ' '
-     '  NOTE THAT THE WORK-PATH-ID FIELD INDICATES WHICH PATH'
-     '  WAS EXECUTED.'
-     '*****************************************************************'
-     ' '
-     'LR VERBS ALLOWED: OBTAIN'
-     ' '
-     '*****************************************************************'
-     ' '
-     'SELECTION CRITERIA (TOTAL OF FIVE PATHS)'
-     ' '
-     '   OBTAIN PATH GROUP:'
-     ' '
-     ' '
-     '    PATH 1)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR ALL EMPLOYEES WHO ARE'
-     '             ON LEAVE.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF THE PROGRAM'
-     '             REQUEST INCLUDES THE KEYWORD ON-LEAVE.'
-     ' '
-     '             IF PATH 1 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 1".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             NO-DEPT    -- THE EMPLOYEE IS NOT ASSOCIATED WITH'
-     '                           A DEPARTMENT'
-     '             NO-OFFICE  -- THE EMPLOYEE IS NOT ASSIGNED TO AN'
-     '                           OFFICE'
-     ' '
-     ' '
-     '    PATH 2)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.'
-     '             IT USES THE EMP-ID-0415 FIELD AS A'
-     '             CALC KEY TO ACCESS EMPLOYEE INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             EMP-ID-0415 = A NUMERIC LITERAL'
-     '                           A PROGRAM VARIABLE'
-     '                           A FIELD IN THE LOGICAL-RECORD'
-     '                            AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
-     '             IF PATH 2 IS SELECTED, THE VALUE OF THE'
```

```
-     '            WORK-PATH-ID FIELD WILL BE "PATH 2".'
-     ' '
-     '            THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '            PATH STATUSES:'
-     ' '
-     '            INVALID-ID —— THE INPUT EMPLOYEE ID IS INVALID'
-     '            NO-DEPT    —— THE EMPLOYEE IS NOT ASSOCIATED WITH'
-     '                          A DEPARTMENT'
-     '            NO-OFFICE  —— THE EMPLOYEE IS NOT ASSIGNED TO AN'
-     '                          OFFICE'
-     ' '
.
.
.
```

Running the LRDEFS Report

The LRDEFS report describes each logical record in a given subschema. This report:

- Describes the logical-record comments
- Lists each database record that is part of the logical record
- Describes each field in the logical record.

You run the LRDEFS report by choosing the LRDEFS option of the IDMSRPTS utility.

For instructions on running this utility, refer to *CA IDMS Utilities Guide*.

The example below shows a sample LRDEFS report for the EMPLR35 subschema.

```
IDMSRPTS 15.0                          -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-          DATE     TIME   PAGE
LRDEFS                          FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1    04/28/99  134850     1

LOGICAL RECORD NAME EMP-INFO-LR
   COMMENTS          ************************************************************
                    THE EMP-INFO-LR LOGICAL RECORD ACCESSES INFORMATION FROM THE
                    EMPLOYEE DATABASE RECORD AND ALSO ACCESSES INFORMATION
                    FROM THE ASSOCIATED DEPARTMENT AND OFFICE RECORDS.
                     THE FOLLOWING INFORMATION IS RETURNED TO THE PROGRAM, IN
                    THE ORDER SHOWN BELOW:
                       EMPLOYEE RECORD   —— EMP-ID-0415, EMP-NAME-0415, START-DATE-0415
                                            STATUS-0415
                       DEPARTMENT RECORD —— DEPT-ID-0440, DEPT-NAME-0440
                       OFFICE RECORD     —— OFFICE-CODE-0450
                      PATHREC VERSION 1 —— WORK-PATH-ID
                       NOTE THAT THE WORK-PATH-ID FIELD INDICATES WHICH PATH
                      WAS EXECUTED.
                    ************************************************************
                     LR VERBS ALLOWED: OBTAIN
                     ************************************************************
                     SELECTION CRITERIA (TOTAL OF FIVE PATHS)
                        OBTAIN PATH GROUP:

                        PATH 1)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND
                                 OFFICE INFORMATION FOR ALL EMPLOYEES WHO ARE
                                 ON LEAVE.
                                  THE PATH WILL BE SELECTED IF THE PROGRAM
                                 REQUEST INCLUDES THE KEYWORD ON-LEAVE.
                                  IF PATH 1 IS SELECTED, THE VALUE OF THE
                                 WORK-PATH-ID FIELD WILL BE "PATH 1".
                                  THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED
                                 PATH STATUSES:
                                  NO-DEPT    —— THE EMPLOYEE IS NOT ASSOCIATED WITH
                                                A DEPARTMENT
                                 NO-OFFICE  —— THE EMPLOYEE IS NOT ASSIGNED TO AN
                                                OFFICE

                        PATH 2)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND
                                 OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.
                                 IT USES THE EMP-ID-0415 FIELD AS A
IDMSRPTS 15.0                            -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-          DATE     TIME   PAGE
 LRDEFS                             FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1    04/28/99  134850     2

                                 CALC KEY TO ACCESS EMPLOYEE INFORMATION.
                                  THE PATH WILL BE SELECTED IF ANY OF THESE
                                 COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE
                                 CLAUSE:
                                  EMP-ID-0415 = A NUMERIC LITERAL
                                                A PROGRAM VARIABLE
                                                A FIELD IN THE LOGICAL-RECORD
                                                 AREA OF PROGRAM VARIABLE STORAGE
                                  IF PATH 2 IS SELECTED, THE VALUE OF THE
                                 WORK-PATH-ID FIELD WILL BE "PATH 2".
                                  THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED
                                 PATH STATUSES:
                                  INVALID-ID —— THE INPUT EMPLOYEE ID IS INVALID
                                 NO-DEPT    —— THE EMPLOYEE IS NOT ASSOCIATED WITH
                                                A DEPARTMENT
                                 NO-OFFICE  —— THE EMPLOYEE IS NOT ASSIGNED TO AN
                                                OFFICE

                        PATH 3)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND
                                 OFFICE INFORMATION FOR EACH EMPLOYEE IN A
                                 PARTICULAR DEPARTMENT.  IT USES THE DEPT-ID-0410
                                 FIELD AS A CALC KEY TO ACCESS DEPARTMENT
                                 INFORMATION.

                                 THE PATH WILL BE SELECTED IF ANY OF THESE
                                 COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE
                                 CLAUSE:
                                  DEPT-ID-0440 =
                                                 A LITERAL
                                                A PROGRAM VARIABLE
```

```
                                        A FIELD IN THE LOGICAL-RECORD
                                         AREA OF PROGRAM VARIABLE STORAGE
                                  IF PATH 3 IS SELECTED, THE VALUE OF THE
                                 WORK-PATH-ID FIELD WILL BE "PATH 3".
                                  THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED
                                 PATH STATUSES:
                                  INVALID-DEPT —— THE INPUT DEPARTMENT ID IS INVALID
                                 NO-OFFICE    —— THE EMPLOYEE IS NOT ASSIGNED TO AN
                                                 OFFICE
IDMSRPTS 15.0                              -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-          DATE     TIME   PAGE
LRDEFS                             FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1   04/28/99  134850     3

                          PATH 4)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND
                                   OFFICE INFORMATION FOR EACH EMPLOYEE ASSIGNED TO
                                   A PARTICULAR OFFICE.
                                   IT USES THE OFFICE-CODE-0450 FIELD AS A CALC
                                   KEY TO ACCESS OFFICE INFORMATION.
                                    THE PATH WILL BE SELECTED IF ANY OF THESE
                                   COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE
                                   CLAUSE:
                                    OFFICE-CODE-0450 =
                                                 A LITERAL
                                                A PROGRAM VARIABLE
                                                A FIELD IN THE LOGICAL-RECORD
                                                 AREA OF PROGRAM VARIABLE STORAGE
                                    IF PATH 4 IS SELECTED, THE VALUE OF THE
                                   WORK-PATH-ID FIELD WILL BE "PATH 4".
                                    THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED
                                   PATH STATUSES:
                                    INVALID-OFFICE —— THE INPUT OFFICE CODE IS INVALID
                                   NO-DEPT        —— THE EMPLOYEE IS NOT ASSOCIATED
                                                     WITH A DEPARTMENT

                          PATH 5)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND
                                   OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.
                                   IT USES THE EMP-NAME-NDX TO ACCESS EMPLOYEE
                                   INFORMATION.
                                    THE PATH WILL BE SELECTED IF EITHER OF THESE
                                   FIELDS ARE REFERENCED IN THE PROGRAM WHERE
                                   CLAUSE:
                                    EMP-LAST-NAME-0415 AND EMP-FIRST-NAME-0415
                                    EMP-LAST-NAME-0415
                                    IF PATH 5 IS SELECTED, THE VALUE OF THE
                                   WORK-PATH-ID FIELD WILL BE "PATH 5".
                                    THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED
                                   PATH STATUSES:
                                    INVALID-NAME —— THE INPUT EMPLOYEE NAME IS
                                                    INVALID
                                   NO-DEPT      —— THE EMPLOYEE IS NOT ASSOCIATED
                                                   WITH A DEPARTMENT
                                   NO-OFFICE    —— THE EMPLOYEE IS NOT ASSIGNED TO
                                                   AN OFFICE
IDMSRPTS 15.0                              -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-          DATE     TIME   PAGE
LRDEFS                               FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VE=SION    1  04/28/99  134850     4


 BUILT FROM.........

 RECORD NAME........ EMPLOYEE
RECORD ID.......... 0415
RECORD VERSION..... 0100
RECORD LENGTH...... FIXED
LOCATION MODE...... CALC USING       EMP-ID-0415
WITHIN............. EMP-DEMO-REGION              FROM PAGE  4013203    THRU  4013300
ACCESS RESTRICTION ON FOR.......... ERASE
DATA ITEM.......... REDEFINES....... USAGE....... VALUE............................ PICTURE...................    STRT  LGTH
02 EMP-ID-0415                      DISPLAY                                         9(4)                           1     4
                                                   SET CONTROL ITEM FOR ————————     CALC            DUP NOT ALLOWED
02 EMP-NAME-0415                    DISPLAY                                                                        5    25
03 EMP-FIRST-NAME-0415              DISPLAY                                         X(10)                          5    10
                                                   SET CONTROL ITEM FOR ————————     DEPT-EMPLOYEE   DUP LAST
                                                   SET CONTROL ITEM FOR ————————     EMP-NAME-NDX    DUP LAST
                                                   SET CONTROL ITEM FOR ————————     OFFICE-EMPLOYEE DUP LAST
03 EMP-LAST-NAME-0415               DISPLAY                                         X(15)                         15    15
```

```
                                        SET CONTROL ITEM FOR ————————      DEPT-EMPLOYEE   DUP LAST
                                        SET CONTROL ITEM FOR ————————      EMP-NAME-NDX    DUP LAST
                                        SET CONTROL ITEM FOR ————————      OFFICE-EMPLOYEE DUP LAST
02 START-DATE-0415                  DISPLAY                                                          30     6
03 START-YEAR-0415                  DISPLAY                                9(2)                      30     2
03 START-MONTH-0415                 DISPLAY                                9(2)                      32     2
03 START-DAY-0415                   DISPLAY                                9(2)                      34     2
02 STATUS-0415                      DISPLAY                                X(2)                      36     2
88 ACTIVE-0415                      COND                                                             36
                                        '01'
88 ST-DISABIL-0415                  COND                                                             36
                                        '02'
88 LT-DISABIL-0415                  COND                                                              36
                                        '03'
88 LEAVE-OF-ABSENCE-0415            COND                                                             36
                                        '04'
88 TERMINATED-0415                  COND                                                             36
                                        '05'

RECORD NAME..... DEPARTMENT
RECORD ID....... 0410
RECORD VERSION.... 0100
RECORD LENGTH..... FIXED
LOCATION MODE..... CALC USING       DEPT-ID-0410
WITHIN.......... ORG-DEMO-REGION             FROM PAGE  4013353    THRU  4013400
IDMSRPTS 15.0                              -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-                  DATE     TIME   PAGE
LRDEFS                              FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1          04/28/99  134850     5
 DATA ITEM..... REDEFINES..... USAGE..... VALUE..... PICTURE.....    STRT  LGTH
02 DEPT-ID-0410                     DISPLAY                                9(4)                      41     4
                                        SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED
02 DEPT-NAME-0410                   DISPLAY                                X(45)                     45    45

RECORD NAME..... OFFICE
RECORD ID....... 0450
RECORD VERSION.... 0100
RECORD LENGTH..... FIXED
LOCATION MODE..... CALC USING       OFFICE-CODE-0450
WITHIN.......... ORG-DEMO-REGION             FROM PAGE  4013353    THRU  4013400
DATA ITEM..... REDEFINES..... USAGE..... VALUE..... PICTURE.....    STRT  LGTH
02 OFFICE-CODE-0450                 DISPLAY                                X(3)                      97     3
                                        SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED

RECORD NAME..... PATHREC
RECORD VERSION.... 0001
DATA ITEM..... REDEFINES..... USAGE..... VALUE..... PICTURE.....    STRT  LGTH
02 WORK-PATH-ID                     DISPLAY                                X(22)                    105    22
                                       -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS- END
IDMSRPTS 15.0                        ———— INPUT PARAMETER LISTING ————                                DATE  TIME   PAGE
                                                                                                   04/28/99  134850     1
                                          END OF INPUT PARAMETERS
```

Running the LRPATH Report

The LRPATH report lists the paths for each logical record in a given subschema.

You run the LRPATH report by choosing the LRPATH option of the IDMSRPTS utility.

For instructions on running this utility, refer to *CA IDMS Utilities Guide*.

The example below shows a sample LRPATH report for the EMPLR35 subschema.

```
IDMSRPTS 15.0                          -LOGICAL RECORD PATH DESCRIPTIONS-                      DATE     TIME   PAGE
LRPATH                          FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1      04/28/99  135407     1
 LOGICAL RECORD NAME EMP-INFO-LR
 BUILT FROM.........EMPLOYEE
                   DEPARTMENT
                   OFFICE
                   PATHREC           VERSION  0001
   OBTAIN  EMP-INFO-LR      PATH-GROUP

    SELECT FOR KEYWORD      ON-LEAVE
                    COMPUTE
                         WORK-PATH-ID OF LR EQ 'PATH 1'
                        ON 0000 NEXT
                   OBTAIN EACH  EMPLOYEE         WITHIN EMP-NAME-NDX
                         WHERE  STATUS-0415 EQ '04'
                        ON 0000 NEXT
                        ON 0307 ITERATE
                   IF DEPT-EMPLOYEE    MEMBER
                        ON 1601 RETURN          NO-DEPT
                        ON 0000 NEXT
                   OBTAIN OWNER WITHIN DEPT-EMPLOYEE
                        ON 0000 NEXT
                   FIND CURRENT EMPLOYEE
                        ON 0000 NEXT
                   IF OFFICE-EMPLOYEE  MEMBER
                        ON 1601 RETURN          NO-OFFICE
                        ON 0000 NEXT
                   OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                        ON 0000 NEXT

    SELECT FOR FIELDNAME-EQ EMPLOYEE                       EMP-ID-0415
                    COMPUTE
                         WORK-PATH-ID OF LR EQ 'PATH 2'
                        ON 0000 NEXT
                   OBTAIN FIRST EMPLOYEE
                         WHERE  CALCKEY EQ EMP-ID-0415 OF REQUEST
                        ON 0326 RETURN          INVALID-ID
                        ON 0000 NEXT
                   IF DEPT-EMPLOYEE    MEMBER
                        ON 1601 RETURN          NO-DEPT
                        ON 0000 NEXT
                   OBTAIN OWNER WITHIN DEPT-EMPLOYEE
                        ON 0000 NEXT
                   FIND CURRENT EMPLOYEE
                        ON 0000 NEXT
                   IF OFFICE-EMPLOYEE  MEMBER
                        ON 1601 RETURN          NO-OFFICE
                        ON 0000 NEXT
                   OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
IDMSRPTS 15.0                          -LOGICAL RECORD PATH DESCRIPTIONS-                      DATE     TIME   PAGE
LRPATH                          FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1      04/28/99  135407     2
                       ON 0000 NEXT

    SELECT FOR FIELDNAME-EQ DEPARTMENT                     DEPT-ID-0410
                    COMPUTE
                         WORK-PATH-ID OF LR EQ 'PATH 3'
                        ON 0000 NEXT
                   OBTAIN FIRST DEPARTMENT
                         WHERE  CALCKEY EQ DEPT-ID-0410 OF REQUEST
                        ON 0326 RETURN          INVALID-DEPT
                        ON 0000 NEXT
                   OBTAIN EACH  EMPLOYEE         WITHIN DEPT-EMPLOYEE
                        ON 0000 NEXT
                        ON 0307 ITERATE
                   IF OFFICE-EMPLOYEE  MEMBER
                        ON 1601 RETURN          NO-OFFICE
                        ON 0000 NEXT
                   OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                        ON 0000 NEXT

    SELECT FOR FIELDNAME-EQ OFFICE                         OFFICE-CODE-0450
                    COMPUTE
                         WORK-PATH-ID OF LR EQ 'PATH 4'
```

```
                    ON 0000 NEXT
                OBTAIN FIRST OFFICE
                      WHERE  CALCKEY EQ OFFICE-CODE-0450 OF REQUEST
                    ON 0326 RETURN          INVALID-OFFICE
                    ON 0000 NEXT
                OBTAIN EACH  EMPLOYEE         WITHIN OFFICE-EMPLOYEE
                    ON 0000 NEXT
                    ON 0307 ITERATE
                IF DEPT-EMPLOYEE    MEMBER
                    ON 1601 RETURN          NO-DEPT
                    ON 0000 NEXT
                OBTAIN OWNER WITHIN DEPT-EMPLOYEE
                    ON 0000 NEXT

    SELECT     USING INDEX  EMP-NAME-NDX
           FOR FIELDNAME    EMPLOYEE                        EMP-LAST-NAME-0415
               FIELDNAME    EMPLOYEE                        EMP-FIRST-NAME-0415

     SELECT     USING INDEX  EMP-NAME-NDX
           FOR FIELDNAME    EMPLOYEE                        EMP-LAST-NAME-0415
                 COMPUTE
                     WORK-PATH-ID OF LR EQ 'PATH 5'
                    ON 0000 NEXT
IDMSRPTS 15.0                          -LOGICAL RECORD PATH DESCRIPTIONS-                        DATE     TIME   PAGE
LRPATH                          FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1         04/28/99  135407     3
                 OBTAIN EACH  EMPLOYEE         USING  INDEX
                    ON 0000 NEXT
                    ON 0326 RETURN          INVALID-NAME
                IF DEPT-EMPLOYEE    MEMBER
                    ON 1601 RETURN          NO-DEPT
                    ON 0000 NEXT
                OBTAIN OWNER WITHIN DEPT-EMPLOYEE
                    ON 0000 NEXT
                FIND CURRENT EMPLOYEE
                    ON 0000 NEXT
                IF OFFICE-EMPLOYEE  MEMBER
                    ON 1601 RETURN          NO-OFFICE
                    ON 0000 NEXT
                OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                    ON 0000 NEXT
                                       -LOGICAL RECORD PATH DESCRIPTIONS-       END
IDMSRPTS 15.0                          ------- INPUT PARAMETER LISTING -------                    DATE     TIME   PAGE
                                                                                               04/28/99  135407     1
                                              END OF INPUT PARAMETERS
```

Running the LRACT Report

The LRACT report lists program activity against each logical record in a given subschema. For each logical record accessed by a program, the list includes the name of the program and the number of times the program issues each DML verb against the logical record.

You run the LRACT report by choosing the LRACT option of the IDMSRPTS utility.

For instructions on running this utility, refer to *CA IDMS Utilities Guide*.

The example below shows a sample LRACT report for the EMPLR35 subschema.

```
IDMSRPTS 15.0                          --LOGICAL RECORD ACTIVITY DESCRIPTIONS--              DATE     TIME   PAGE
LRACT                             FOR SUBSCHEMA EMPLR35  IN SCHEMA EMPSCHM  VERSION    1     04/28/99  135530     1

SCHEMA    VER SUBSCHEMA  RECORD                          PROGRAM   VER USAGE            TIMES  COMPILED  CREATED

EMPSCHM  0001 EMPLR35    EMP-INFO-LR                     DOC001    0001 OBTAIN           0003   06/11/86 06/11/86
EMPSCHM  0001 EMPLR35    EMP-INFO-LR                     DOC002    0001 OBTAIN           0003   06/11/86 06/11/86
EMPSCHM  0001 EMPLR35    EMP-INFO-LR                     DOC003    0001 OBTAIN           0003   06/11/86 06/11/86
EMPSCHM  0001 EMPLR35    EMP-INFO-LR                     DEPT-1    0001 OBTAIN           0001   07/16/86 07/16/86
                                  --LOGICAL RECORD ACTIVITY DESCRIPTIONS-- END
IDMSRPTS 15.0                     -------- INPUT PARAMETER LISTING --------                DATE     TIME   PAGE
                                                                                        04/28/99  135530     1

                                          END OF INPUT PARAMETERS
```

Chapter 15: Currency Considerations

This section contains the following topics:

Introduction

The DBMS checks and updates currency while processing path-DML statements, just as it does while processing navigational DML statements. In LRF, currency is transparent to the programmer. The programmer does not have to know the currency of records, sets, or areas to execute or reexecute a path.

Subschema currency tables

Currencies for area, set, record type, and run unit are kept in subschema currency tables, just as they are for navigational DML programs.

For more information on subschema currency tables, refer to the *CA IDMS Navigational DML Programming Guide*.

Path-group control block

After the execution of each path database retrieval, update, or control command, LRF copies the appropriate currencies from the subschema currency tables to a path-group control block. There is one path-group control block for each of the four path groups (OBTAIN, STORE, MODIFY, and ERASE). These control blocks maintain currencies by path-DML statement.

Types of currencies copied

The following table lists the types of currencies that get copied from the subschema currency tables to the path-group control block. LRF copies these currencies after the execution of each path-DML command.

Path-DML command	Types of currencies copied to a path-group control block
FIND/OBTAIN EACH WITHIN SET FIND/OBTAIN EACH USING INDEX FIND/OBTAIN EACH WHERE CALCKEY	Set currencies (current, next, prior, and owner) for the named chained, indexed, or CALC set
FIND/OBTAIN EACH WITHIN AREA	Area currencies
All other commands	Run-unit currency

The remainder of this chapter describes how LRF uses currency, how to choose a currency option for the subschema, and how currency is maintained with role names.

How LRF Uses Currency

LRF uses the currencies saved in the path-group control blocks to:

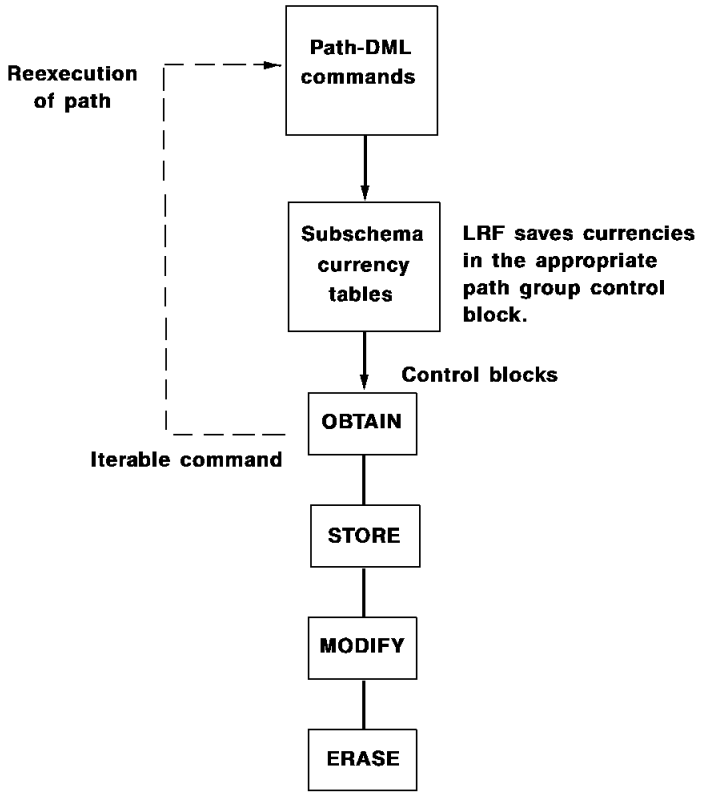
- **Protect the database record occurrences accessed by LRF from data corruption**. LRF protects these record occurrences by placing implicit locks on them. LRF keeps these locks either until the associated command is reexecuted or until the saved currencies are cleared. For more information on implicit database locks, refer to *CA IDMS Database Administration Guide*.
- **Position itself in an iteration cycle** (OBTAIN path-group currencies only). This use of currency is described below.

When a program issues an OBTAIN NEXT command (requesting path iteration), LRF copies the db-key values saved for the last iterated verb from the OBTAIN path-group control block back to the subschema currency tables. LRF then issues a FIND NEXT or OBTAIN NEXT command to retrieve the next occurrence of the target database record. The format of the FIND/OBTAIN command depends on whether the iterated verb is a FIND or an OBTAIN verb.

How LRF saves currencies

LRF saves currencies for each path group separately. This allows iteration to occur nonconsecutively as well as consecutively. For example, a programmer may want to OBTAIN a logical record, MODIFY the logical record, and then OBTAIN the next logical record. Because currencies are saved by path group, LRF can pick up the iteration cycle where it left off. For more information on iteration, refer to Chapter 11, Controlling Path Execution.

The following diagram shows how LRF uses currency. The currencies LRF copies are used to lock database record occurrences and to position LRF in an iteration cycle.



Saving currencies for an OBTAIN path

LRF saves currencies for an OBTAIN path until one of the following activities occurs:

- **The program issues an OBTAIN FIRST statement either for the same path or for a different path**. In this case, LRF clears the OBTAIN path-group control block.
- **The program issues an OBTAIN NEXT statement for the path**, directing LRF to execute the path from the last successfully executed iterable command. In this case, the existing currencies saved after the iteration point are cleared. The currencies saved before the iteration point do not change.

- **The program issues a COMMIT ALL or a ROLLBACK CONTINUE statement**, which clears all path-group currencies.
- **The run unit ends** (that is, the program issues a FINISH or ROLLBACK statement), which clears all path-group currencies.
- **The path returns LR-NOT-FOUND or LR-ERROR**, which clears all OBTAIN path-group currencies.

Saving currencies for an update path

LRF saves currencies for a STORE, MODIFY, or ERASE path until one of the following activities occurs:

- **The program issues another STORE, MODIFY, or ERASE logical record command**, respectively. In this case, LRF clears the path-group control block.
- **The program issues a COMMIT ALL or ROLLBACK CONTINUE statement**, which clears all path-group currencies.
- **The run unit ends** (that is, the program issues a FINISH or ROLLBACK) statement, which clears all path-group currencies.
- **The path returns LR-NOT-FOUND or LR-ERROR**, which clears all STORE, MODIFY, or ERASE path-group currencies, respectively.

Choosing a Currency Option

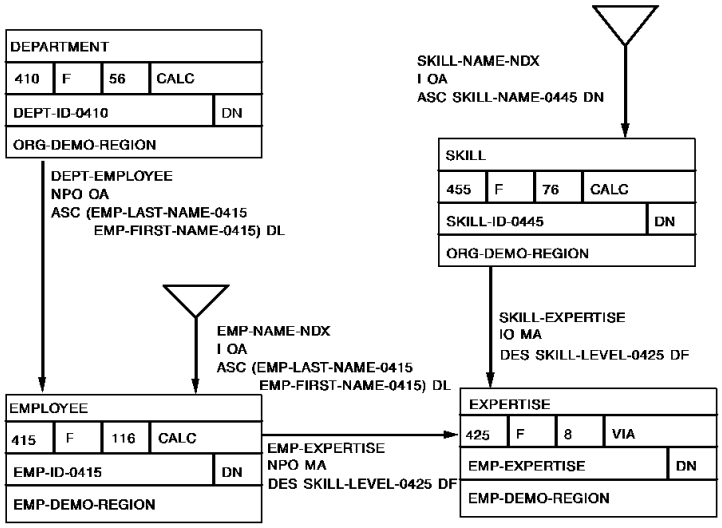
You can choose one of the following currency options when you define a logical-record subschema:

- **LR CURRENCY NO RESET** copies the db-key for the last iterated verb from the OBTAIN path-group control block to the subschema currency tables. This activity is performed when a program issues an OBTAIN NEXT logical record request (requesting path iteration).
- **LRF CURRENCY RESET (default)** issues a FIND/OBTAIN DBKEY command for each path database retrieval command issued before the iteration point. This activity refreshes program variable storage. It also reestablishes the currencies in the subschema currency tables so they reflect the currencies saved in the OBTAIN path-group control block. It is performed when a program issues an OBTAIN NEXT logical record request (requesting path iteration).

  The RESET option also directs LRF to copy the db-key for the last iterated verb from the OBTAIN path-group control block to the subschema currency tables.

Example

The following example illustrates the difference between LR CURRENCY NO RESET and LR CURRENCY RESET. The path in this example retrieves department, employee, and skill information for all employees who work in a particular department. The iteration points in the path are shaded.



Path code:

```
ADD
PATH-GROUP NAME IS OBTAIN SKILL-LR
   SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
   OBTAIN FIRST DEPARTMENT
      WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
   OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
   FIND EACH EXPERTISE WITHIN EMP-EXPERTISE
   OBTAIN OWNER SKILL WITHIN SKILL-EXPERTISE.
```

Program request:

```
OBTAIN FIRST SKILL-LR
   WHERE DEPT-ID-0410 EQ '5200'.
ON LR-FOUND
   REPEAT.
      OBTAIN NEXT SKILL-LR
         WHERE DEPT-ID-0410 EQ '5200'.
      PUT DETAIL.
   END.
DISPLAY.
```

Given the above path:

- **If the subschema currency option is NO RESET**, LRF copies the db-key of the last iterated verb from the path-group control block to the subschema currency tables, before it iterates the path. It does not reretrieve the DEPARTMENT and EMPLOYEE records.
- **If the subschema currency option is RESET**, LRF issues the commands shown below, before it iterates the path. (Assume that the last successfully executed iterable command is the FIND EACH EXPERTISE command.)

```
OBTAIN FIRST DEPARTMENT
  WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST

OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
```

  As with the NO RESET option, LRF also copies the db-key of the last iterated verb from the path group-control block to the subschema currency tables, before it iterates the path.

For a discussion of design considerations with regard to the NO RESET and RESET currency options, refer to Chapter 3, Preliminary Analysis and Design.

Currency Considerations For Role Names

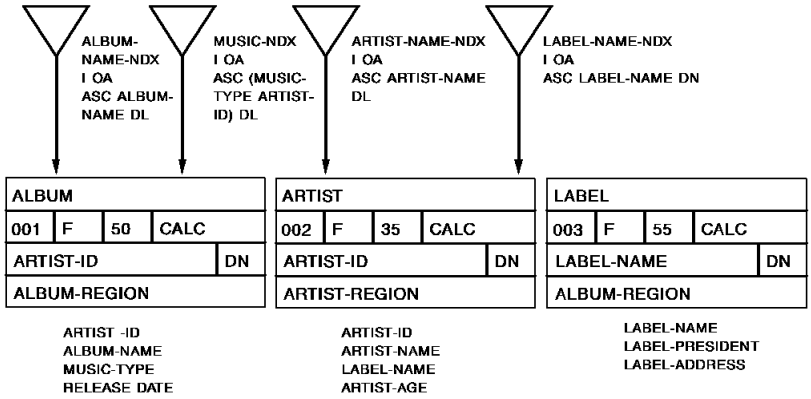
When you use role names, you should be aware of the following currency considerations:

- LRF keeps separate currencies by role name only during an iteration cycle. You must use the FIND/OBTAIN EACH *role-name* statement for this to occur.
- Do not mix iterative retrieval activity by role name with STORE, MODIFY, or ERASE activity for the same role-name occurrence. These commands will STORE, MODIFY, or ERASE whatever record occurrence is current of run unit. They do not keep track of the current role name.

Example

The example shown below illustrates how the use of role names can affect currency. The OBTAIN path in this example retrieves two occurrences of the ALBUM record. GREATEST HITS is a gold album; RUBBER SOUL is a platinum album. The MODIFY path then tries to modify both record occurrences.

When the MODIFY GOLDREC statement is issued, RUBBER SOUL is current of run unit. Thus, RUBBER SOUL is modified instead of GREATEST HITS. When the MODIFY PLATREC statement is issued, RUBBER SOUL is modified again. The GREATEST HITS album remains unchanged.



Path code:

```
ADD LOGICAL RECORD NAME IS MUSIC-LR
   ELEMENTS ARE ALBUM ROLE IS GOLDREC
                ALBUM ROLE IS PLATREC

ADD
PATH-GROUP NAME IS OBTAIN MUSIC-LR
    SELECT
       OBTAIN EACH GOLDREC WITHIN MUSIC-NDX
          WHERE FLAG OF GOLDREC EQ 'G'
       OBTAIN EACH PLATREC WITHIN MUSIC-NDX
          WHERE FLAG OF PLATREC EQ 'P'.
ADD
PATH-GROUP NAME IS MODIFY MUSIC-LR
    SELECT FOR KEYWORD MOD-REC
       FIND CURRENT GOLDREC
       MODIFY GOLDREC
       FIND CURRENT PLATREC
       MODIFY PLATREC.
```

Program request:

```
OBTAIN FIRST MUSIC-LR.
   ON LR-FOUND
      REPEAT.
         PUT DETAIL.
         MODIFY MUSIC-LR
            WHERE MOD-REC.
         OBTAIN NEXT MUSIC-LR.
END.
   DISPLAY.
```

Chapter 16: Implementing Data Integrity Rules

This section contains the following topics:

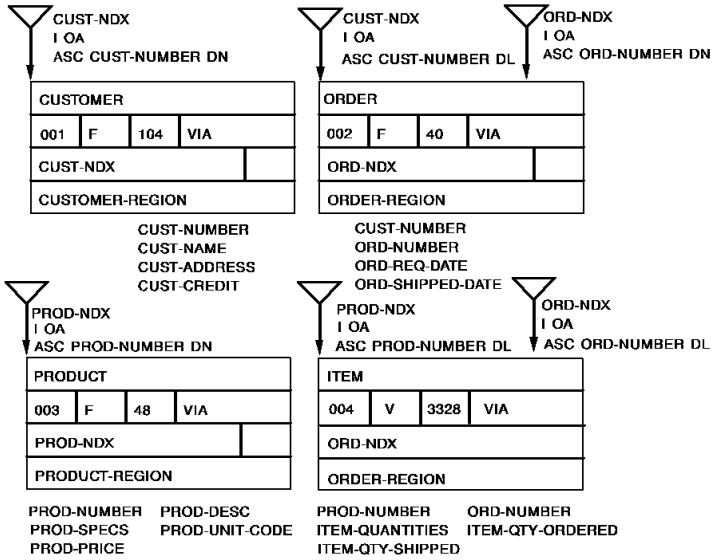
Data Integrity Rules

You can use LRF to implement integrity rules that protect the corporate data resource:

- **Referential integrity rules** protect the relationship between database records that have shared keys. For example, in the sample Customer-Order database shown below, you can prevent an application program from storing a new ORDER record for which no valid customer exists.
- **Application-dependent integrity rules** ensure that the logical relationships between data meet criteria that are established by the application. For example, you can prevent an application program from erasing an existing ORDER record if an associated item has already been shipped.

In general, you use the path WHERE clause and EVALUATE commands to implement data integrity rules through LRF. Examples that show this implementation are presented below. For more information on data integrity, refer to *CA IDMS Database Design Guide*.

Examples



Enforcing data integrity rules for STORE activity

This path ensures that the customer number is valid, before it stores a new ORDER record. If the customer number is not valid, the path returns an appropriate DBA-defined path status.

Path code:

```
ADD
PATH-GROUP NAME IS STORE CUST-ORD-LR
   SELECT FOR KEYWORD STORE-ORD
      FIND FIRST CUSTOMER WITHIN CUST-NDX
         WHERE SORTKEY EQ CUST-NUMBER OF ORDER OF LR
             ON 0000 NEXT
             ON 0326 RETURN 'INVALID-CUST'
      STORE ORDER.
```

Program request:

```
MOVE INPUT-WORK TO CUST-ORD-LR.
STORE CUST-ORD-LR
   WHERE STORE-ORD.
ON INVALID-CUST
   DISPLAY MSG TEXT IS 'CUSTOMER NUMBER IS INVALID'.
```

Enforcing data integrity rules for ERASE activity

This path erases an occurrence of the ORDER record when a customer decides to cancel an order. The path also erases all ITEM records that are associated with the order.

Before any records are erased, the path checks to see if any of the requested items have been shipped. If an item has been shipped, the path indicates that it can't erase the order.

Path code:

```
ADD
PATH-GROUP NAME IS ERASE ORDER-LR
   SELECT FOR KEYWORD ERASE-ORD
              FIELDNAME-EQ ORD-NUMBER OF ORDER
      OBTAIN FIRST ORDER WITHIN ORD-NDX
         WHERE SORTKEY EQ ORD-NUMBER OF ORDER OF REQUEST
            ON 0000 NEXT
            ON 0326 RETURN 'INVALID-ORD'
      FIND FIRST ITEM WITHIN ORD-NDX
         WHERE (SORTKEY EQ ORD-NUMBER OF ORDER OF LR)
         AND (ITEM-QTY-SHIPPED > 0)
            ON 0000 RETURN ITEM-SHIPPED
            ON 0326 NEXT
      FIND EACH ITEM WITHIN ORD-NDX
         WHERE SORTKEY EQ ORD-NUMBER OF ORDER OF LR
            ON 0000 NEXT
            ON 0326 DO
                    ERASE ORDER
                    END
      ERASE ITEM
            ON 0000 ITERATE.
```

Program request:

```
MOVE INPUT-NUMBER TO ORDER-NUMBER OF ORDER OF LR.
ERASE ORDER-LR
   WHERE (ORDER-NUMBER OF ORDER EQ ORDER-NUMBER OF ORDER OF LR)
   AND ERASE-ORD.
ON INVALID-ORD
   DISPLAY MSG TEXT IS 'ORDER NUMBER IS INVALID'.
ON ITEM-SHIPPED
   DISPLAY MSG TEXT IS 'AN ITEM HAS BEEN SHIPPED.  THE ORDER CANNOT
       BE CANCELED'.
```

Chapter 17: Using LRF with Other Facilities

This section contains the following topics:

Introduction

You can use LRF with a variety of tools. This chapter describes how to use LRF with:

- CA OLQ
- CA ADS (including CA ADS Batch)
- CA IDMS/DC Mapping Facility

For information on using LRF with CA Culprit, refer to the *CA Culprit for CA IDMS Reference Guide*.

You can also use LRF with a variety of programming languages, including COBOL, PL/I, and Assembler.

For information on using LRF with these languages, refer to the *CA IDMS Navigational DMLProgramming Guide*. and to the corresponding CA IDMS/DB or CA IDMS/DC reference.

Using LRF With CA OLQ

CA OLQ is a data retrieval system used to access information stored in an CA IDMS/DB database and to produce and format reports. You can use logical-record subschemas with both command-mode and menu-mode CA OLQ.

What you can do

By using logical-record subschemas with menu-mode CA OLQ, you can:

- **Simplify end-user queries**. If the subschema usage mode is LR, the CA OLQ Record Select screen will display only the logical-record(s) that are included in the subschema. The user will not have to choose from a variety of database records to make a specific query.
- **Ensure that CA OLQ uses the most efficient path** when it processes queries.

You can also use CA OLQ to test logical-record subschemas relatively easily and quickly. Because CA OLQ allows data retrieval only, you can't use it to test update paths.

Considerations

The following considerations apply to using LRF with CA OLQ:

- **CA OLQ's program variable storage cannot be accessed by other facilities**. This has the following implications:
  - Using the OF LR clause in a path used with CA OLQ can lead to unpredictable results. The OF LR clause directs LRF to look in program variable storage for values that may or may not be there. Where possible, use the OF REQUEST clause instead.
  - If you use the NO RESET currency option, the data above the iteration point will not be visible after the first logical-record occurrence is returned. Use the RESET currency option if you want the path to return more than one logical-record occurrence.
  - If you want the values in an IDD-work field to be displayed after the first logical record is returned, you should compute the values below the lowest iteration point. Because an IDD-defined work record has no db-key, it can't be reobtained.
- **Always use the PATHSTATUS option in CA OLQ when you execute a path that contains a DBA-defined path status**. This option allows the status to be displayed on the screen.

For more information on using CA OLQ, refer to the CA OLQ documentation set.

Using LRF With CA ADS and CA ADS Batch

The Application Development System, including CA ADS and ADS/Batch, allow users to develop and execute applications.

Considerations

The following considerations apply to using LRF with CA ADS and ADS/Batch:

- **Database considerations**:
  - **A base subschema record may not participate in more than one logical record per dialog**. Therefore, all logical records used in a dialog must have different logical-record elements, unless you assign role names to the elements.

    Note: For more information on logical-record elements, see Chapter 4 (see page 33). For more information on role names, see Using Role Names (see page 157).
  - **The OBTAIN logical-record command defaults to OBTAIN NEXT logical-record**. In certain situations, using the OBTAIN logical-record command may result in an LR-NOT-FOUND path status.

    To avoid this situation:

    Use the **OBTAIN FIRST** logical-record command when you want to retrieve the first occurrence of the named logical record.

    Use the **OBTAIN NEXT** logical-record command when you want to retrieve the next occurrence of the named logical record.
- **Flow of control considerations**:
  - **When you invoke or link to a lower-level dialog using the same logical-record subschema**, the logical-record data is available at the lower level.

    The data is *not available* when you transfer to a lower-level dialog, or invoke or link to a dialog that is not an extended run unit.
  - **To change logical-record data in a lower-level dialog and then access these changes on a higher-level dialog**, you must use the RESET subschema currency option. This option is described in Chapter 3 (see page 21) and Chapter 4 (see page 33).
- **With AUTOSTATUS, you do not get the 20xx status codes returned to your dialogs.** Because these codes usually indicate a program/path interaction problem, you might want to use the ALLOWING ERROR CODES expression to test for these codes.

For more information on CA ADS and ADS/Batch, refer to the Application Development System documentation set.

Using LRF With the CA IDMS/DC Mapping Facility

The CA IDMS/DC mapping facility is used to define the layout of maps. These maps can be associated with dialogs generated by the Application Development System and programs written in COBOL, PL/I, and Assembler.

Considerations

The following considerations apply to using logical records with the CA IDMS/DC mapping facility:

- **You must name all logical-record elements in the RECORD NAME field of the Initial Definition screen**.
- If a logical-record element is associated with a role name, **you must specify the role name in the ROLE NAME field of the Initial Definition screen**. You will need to specify both the logical-record name *and* the role name for each role name used.

For more information on the mapping facility, refer to *CA IDMS Mapping Facility Guide*.

Chapter 18: Debugging Subschema Code

This section contains the following topics:

Debugging and Testing

What to watch out for

Before you use a logical-record subschema, it is a good idea to review both the path code and the program request. Here are some things to watch out for:

- **In the path code**:
  - Missing or unresolved DO/END clauses
  - Forgetting to put ON clauses after each path-DML statement
  - Failing to establish run-unit currency prior to an IF [NOT] EMPTY command
  - Omitting qualification by role name when the named field is assigned a role name
  - Neglecting to obtain records prior to the reference of the values needed in an OF LR statement
  - Omitting EACH commands and expecting iteration to occur
  - Forgetting to specify OF LR for a logical-record field that participates in an EVALUATE or COMPUTE operation
  - Forgetting to put a version number on an IDD-defined work record when you include this record as a logical-record element
- **In the program request**:
  - Omitting the FIRST/NEXT qualifier in a logical-record request
  - Omitting qualification by role name when the named field is assigned a role name
  - Using selection fields from records that are not being obtained in the path
  - Using IDD-defined work fields as selection fields
  - Issuing a COMMIT ALL, ROLLBACK, or ROLLBACK CONTINUE command, and expecting LRF to continue an iteration cycle

Testing

After you review your subschema, you can test it by using CA OLQ, CA ADS, CA ADS Batch, or other CA IDMS facilities. If you encounter problems, you can debug your subschema code in a variety of ways. Here are some suggestions:

- **Include an IDD-defined work record** as a logical-record element and return an appropriate literal through this element. The literal can indicate which path was executed.
- **Use DBA-defined path statuses** to trace the path code.
- **Regenerate and test the basic subschema code**. If the code works, gradually add and test additional pieces of code.
- **Use the online debugger** to set breakpoints in the program and to display the contents of storage on request. For information on using the online debugger, refer to *CA IDMS Online Debugger Guide*.
- **Request a snap dump** of the contents of one or more areas in memory by using the Application Development System SNAP command (or the various DML SNAP commands). For more information on the SNAP command, refer to the *CA ADS Reference Guide*. or to the appropriate DML manual.

Chapter 19: LRF Programming Techniques

This section contains the following topics:

Introduction

This section discusses the LRF programming techniques that you use in coding CA IDMS/DB and CA IDMS/DC programs. These techniques include:

- **Using LRF documentation** —– A discussion of the LRDEFS and LRPATH reports and how to use them in designing your application.
- **Accessing logical records** —– A discussion of the DML statements available for accessing logical records and the processing that you must perform before each call to LRF. An explanation of the WHERE clause is included.
- **Testing for path status** —– A discussion on testing the path statuses that are returned after every call to LRF.

Using LRF Documentation

LRF documentation, which is available through the IDMSRPTS utility, is useful in all stages of LRF program development. These reports contain a variety of information, such as records and record elements available to your application, DBA-defined path statuses, and efficient WHERE clause arguments.

The following LRF reports are most useful to application programmers:

- **LRDEFS** lists records, record elements, and all DBA comments.
- **LRPATH** lists the records participating in the logical record and all DBA-defined path groups.

The LRDEFS report and the LRPATH report are explained below.

The LRDEFS report

When writing an LRF application, you must isolate specific information about the logical records to be accessed. This information is provided by the LRDEFS report. The LRDEFS report supplies definitions of all fields included in each logical record defined in the subschema.

DBA comments

This report also lists comments provided by the DBA on the following topics:

- Restrictions placed on the operations that the program can perform in conjunction with each logical record (that is, which of the LRF database access statements can you use).
- Selection criteria permitted in the WHERE clause that can accompany each LRF database access statement, for each logical record available to the program. The LRDEFS report provides the following information about selection criteria, as defined by the DBA:
  - Names of individual logical-record fields that can be used in WHERE clause comparisons
  - Types of comparisons that can or must be performed against the named fields
  - DBA-designated keywords

  The WHERE clause is explained in detail later in this section.
- Sequence of data returned to the program.
- IDD-defined records (if any) included in the logical record.
- Path statuses returned to the program by LRF to indicate the result of the requested operation.
- Program action to be taken following the return of DBA-defined path statuses.

Field definitions and DBA-supplied comments contained in this report provide all the information needed to access the EMP-JOB-LR logical record.

```
IDMSRPTS 15.0                              -SUBSCHEMA LOGICAL RECORD DESCRIPTIONS-                  DATE     TIME   PAGE

LRDEFS                              FOR SUBSCHEMA EMPSS19  IN SCHEMA EMPSCHM  VERSION  100         05/06/99  130256


LOGICAL RECORD NAME EMP-JOB-LR
  COMMENTS          ************************************************************
                             EMP-JOB-LR    THIS LOGICAL RECORD ACCESSES EMPLOYEE,
                                           OFFICE, DEPARTMENT, AND JOB INFORMATION.
                    ********************************************************************
                    LR VERBS  ALLOWED: OBTAIN
                    ************************************************************
                    SELECTION CRITERIA: (TOTAL OF 3 OBTAIN PATHS)

                    ▶▶▶▶▶▶▶▶▶▶▶▶▶ OBTAIN PATH 1 <<<<<<<<<<<<<<<<<<
                    THIS PATH WILL BE SELECTED IF THE FOLLOWING COMPARISON
                    IS INCLUDED IN THE PROGRAM-REQUEST
                    WHERE CLAUSE:

                    DEPT-ID-0410 = A LITERAL
                                   A PROGRAM VARIABLE THAT HAS BEEN DEFINED TO IDD
                                   A FIELD IN THE AREA RESERVED FOR THE LOGICAL
                                     RECORD IN PROGRAM VARIABLE STORAGE (OF LR)

                    THIS PATH OBTAINS AN OCCURRENCE OF THE EMP-JOB-LR
                    FOR EACH EMPLOYEE
                    IN THE SPECIFIED DEPARTMENT.

                    NOTE THAT DEPT-ID-0410 IS USED AS A CALC KEY BY THE PATH.

                    ************ PATH STATUSES ************
                     LR-NOT-FOUND —— ALL OCCURRENCES OF THE LOGICAL RECORD
                    HAVE BEEN RETRIEVED.

                     DEPT-NOT-FOUND —— THE DBMS COULD NOT FIND THE SPECIFIED
                    DEPARTMENT.
                      **** PROGRAM ACTION —— WRITE A MESSAGE INDICATING THE
                                             KEY OF THE RECORD.

                     DEPT-EMPTY —— THE DEPARTMENT HAS NO EMPLOYEES
                      **** PROGRAM ACTION —— THE DEPARTMENT DATA IS GOOD;
                                             SPACE OR BLANK OUT EVERYTHING
                                             ELSE AND WRITE A NO-EMPLOYEES
                                             MESSAGE.

                     NO-JOBS —— THE EMPLOYEE HAS NO JOB RECORDS
                      **** PROGRAM ACTION —— THE DEPARTMENT AND EMPLOYEE DATA
                                             IS GOOD; SPACE OR BLANK OUT EVERYTHING
                                             ELSE AND WRITE A NO-JOBS MESSAGE.

                     NO-OFFICE —— THE EMPLOYEE HAS NO OFFICE RECORD
                      **** PROGRAM ACTION —— EVERYTHING ELSE IS GOOD; SPACE OR
                                             BLANK OUT THE OFFICE RECORD AND
                                             WRITE A NO-OFFICE MESSAGE.
                    ▶▶▶▶▶▶▶▶▶▶▶▶▶ OBTAIN PATH 2 <<<<<<<<<<<<<<<<<<
                    THIS PATH WILL BE SELECTED IF THE FOLLOWING COMPARISON
                    IS INCLUDED IN THE PROGRAM-REQUEST
                    WHERE CLAUSE:

                    EMP-ID-0415 =  A LITERAL
                                   A PROGRAM VARIABLE THAT HAS BEEN DEFINED TO IDD
                                   A FIELD IN THE AREA RESERVED FOR THE LOGICAL
                                    RECORD IN PROGRAM VARIABLE STORAGE (OF LR)

                    THIS PATH OBTAINS AN OCCURRENCE OF THE EMP-JOB-LR
                    FOR EACH SPECIFIED EMPLOYEE

                    NOTE THAT EMP-ID-0415 IS USED AS A CALC KEY BY THE PATH.

                    ************ PATH STATUSES ************
                     LR-NOT-FOUND —— ALL OCCURRENCES OF THE LOGICAL RECORD
                    HAVE BEEN RETRIEVED.
```

```
                     EMP-NOT-FOUND —— THE DBMS COULD NOT FIND THE SPECIFIED
                    EMPLOYEE.
                      **** PROGRAM ACTION —— WRITE A MESSAGE INDICATING THE
                                             KEY OF THE RECORD.

                     NO-DEPT —— THE EMPLOYEE HAS NO DEPARTMENT RECORD
                      **** PROGRAM ACTION —— THE EMPLOYEE DATE IS GOOD; SPACE OR
                                             BLANK OUT EVERYTHING ELSE AND
                                             WRITE A NO-DEPARTMENT MESSAGE.

                     NO-OFFICE —— THE EMPLOYEE HAS NO OFFICE RECORD.
                      **** PROGRAM ACTION —— THE EMPLOYEE
                                             AND DEPARTMENT DATA IS GOOD;
                                             SPACE OR BLANK OUT EVERYTHING
                                             ELSE AND WRITE A NO-OFFICE
                                             MESSAGE.

                     NO-JOBS —— THE EMPLOYEE HAS NO JOB RECORDS
                      **** PROGRAM ACTION —— THE OFFICE, EMPLOYEE,
                                             AND DEPARTMENT DATA
                                             IS GOOD; SPACE OR BLANK OUT THE JOB
                                             RECORD AND WRITE A NO-JOBS MESSAGE.
                    ▶▶▶▶▶▶▶▶▶▶▶▶▶ OBTAIN PATH 3 <<<<<<<<<<<<<<<<<
                    THIS PATH WILL BE SELECTED FOR ALL OTHER
                    PROGRAM REQUESTS.

                    THIS PATH OBTAINS AN OCCURRENCE OF THE
                    EMP-JOB-LR LOGICAL RECORD FOR EACH EMPLOYEE.

                    THIS PATH ENTERS THE DATABASE THROUGH AN AREA SWEEP
                    OF THE EMP-DEMO-REGION.

                   ************ PATH STATUSES ************
                    LR-NOT-FOUND —— ALL OCCURRENCES OF THE LOGICAL RECORD
                    HAVE BEEN RETRIEVED.

                     NO-DEPT —— THE EMPLOYEE HAS NO DEPARTMENT RECORD
                      **** PROGRAM ACTION —— THE EMPLOYEE DATA IS GOOD; SPACE OR
                                             BLANK OUT EVERYTHING ELSE AND
                                             WRITE A NO-DEPARTMENT MESSAGE.

                     NO-OFFICE —— THE EMPLOYEE HAS NO OFFICE RECORD.
                      **** PROGRAM ACTION —— THE EMPLOYEE
                                             AND DEPARTMENT DATA IS GOOD;
                                             SPACE OR BLANK OUT EVERYTHING
                                             ELSE AND WRITE A NO-OFFICE
                                             MESSAGE.

                     NO-JOBS —— THE EMPLOYEE HAS NO JOB RECORDS
                      **** PROGRAM ACTION —— THE OFFICE, EMPLOYEE,
                                             AND DEPARTMENT DATA
                                             IS GOOD; SPACE OR BLANK OUT THE JOB
                                             RECORD AND WRITE A NO-JOBS MESSAGE.
BUILT FROM.........

RECORD NAME........ EMPLOYEE
RECORD ID.......... 0415
RECORD VERSION..... 0100
RECORD LENGTH...... FIXED
LOCATION MODE...... CALC USING       EMP-ID-0415
WITHIN............. EMP-DEMO-REGION              FROM PAGE  4013003    THRU  4013100
ACCESS RESTRICTION ON FOR.......... ERASE
DATA ITEM.......... REDEFINES....... USAGE....... VALUE............................ PICTURE...................    STRT  LGTH
02 EMP-ID-0415                      DISPLAY                                        9(4)                           1     4
                                                    SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED
02 EMP-NAME-0415                    DISPLAY                                                                       5    25
03 EMP-FIRST-NAME-0415              DISPLAY                                        X(10)                          5    10
                                                    SET CONTROL ITEM FOR ————————      DEPT-EMPLOYEE   DUP LAST
                                                    SET CONTROL ITEM FOR ————————      EMP-NAME-NDX    DUP LAST
                                                    SET CONTROL ITEM FOR ————————      OFFICE-EMPLOYEE DUP LAST
03 EMP-LAST-NAME-0415               DISPLAY                                        X(15)                         15    15
                                                    SET CONTROL ITEM FOR ————————      DEPT-EMPLOYEE   DUP LAST
                                                    SET CONTROL ITEM FOR ————————      EMP-NAME-NDX    DUP LAST
                                                    SET CONTROL ITEM FOR ————————      OFFICE-EMPLOYEE DUP LAST
```

```
02 STATUS-0415                    DISPLAY                                  X(2)                        30     2
88 ACTIVE-0415                    COND                                                                 30
                                            '01'
88 ST-DISABIL-0415                COND                                                                 30
                                            '02'
88 LT-DISABIL-0415                COND                                                                 30
                                            '03'
88 LEAVE-OF-ABSENCE-0415          COND                                                                 30
                                            '04'
88 TERMINATED-0415                COND                                                                 30
                                            '05'
02 SS-NUMBER-0415                 DISPLAY                                  9(9)                        32     9
02 START-DATE-0415                DISPLAY                                                              41     6
03 START-YEAR-0415                DISPLAY                                  9(2)                        41     2
03 START-MONTH-0415               DISPLAY                                  9(2)                        43     2
03 START-DAY-0415                 DISPLAY                                  9(2)                        45     2
RECORD NAME..... DEPARTMENT
RECORD ID.......  0410
RECORD VERSION.....  0100
RECORD LENGTH......  FIXED
LOCATION MODE......  CALC USING       DEPT-ID-0410
WITHIN.............  ORG-DEMO-REGION              FROM PAGE  4013153    THRU  4013200
DATA ITEM.........  REDEFINES.......  USAGE.......  VALUE.............................  PICTURE...................    STRT  LGTH
02 DEPT-ID-0410                   DISPLAY                                  9(4)                        49     4
                                            SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED
02 DEPT-NAME-0410                 DISPLAY                                  X(45)                       53    45
02 DEPT-HEAD-ID-0410              DISPLAY                                  9(4)                        98     4
02 FILLER                         DISPLAY                                  XXX                        102     3

RECORD NAME..... JOB
RECORD ID.......  0440
RECORD VERSION.....  0100
RECORD LENGTH......  VARIABLE
MINIMUM ROOT.......    24 CHARACTERS
MINIMUM FRAGMENT...   296 CHARACTERS
LOCATION MODE......  CALC USING       JOB-ID-0440
WITHIN.............  ORG-DEMO-REGION              FROM PAGE  4013153    THRU  4013200
CALL PROCEDURES....  NAME.... WHEN.. FUNCTION
                     IDMSCOMP BEFORE STORE
                     IDMSCOMP BEFORE MODIFY
                     IDMSDCOM AFTER  GET
DATA ITEM.........  REDEFINES.......  USAGE.......  VALUE.............................  PICTURE...................    STRT  LGTH
02 JOB-ID-0440                    DISPLAY                                  9(4)                       105     4
                                            SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED
02 TITLE-0440                     DISPLAY                                  X(20)                      109    20
                                            SET CONTROL ITEM FOR ————————      JOB-TITLE-NDX   DUP NOT ALLOWED

RECORD NAME..... OFFICE
RECORD ID.......  0450
RECORD VERSION.....  0100
RECORD LENGTH......  FIXED
LOCATION MODE......  CALC USING       OFFICE-CODE-0450
WITHIN.............  ORG-DEMO-REGION              FROM PAGE  4013153    THRU  4013200
DATA ITEM.........  REDEFINES.......  USAGE.......  VALUE.............................  PICTURE...................    STRT  LGTH
02 OFFICE-CODE-0450               DISPLAY                                  X(3)                       129     3
                                            SET CONTROL ITEM FOR ————————      CALC            DUP NOT ALLOWED
02 OFFICE-ADDRESS-0450            DISPLAY                                                             132    46
03 OFFICE-STREET-0450             DISPLAY                                  X(20)                      132    20
03 OFFICE-CITY-0450               DISPLAY                                  X(15)                      152    15
03 OFFICE-STATE-0450              DISPLAY                                  X(2)                       167     2
03 OFFICE-ZIP-0450                DISPLAY                                                             169     9
04 OFFICE-ZIP-FIRST-FIVE-0450     DISPLAY                                  X(5)                       169     5
04 OFFICE-ZIP-LAST-FOUR-0450      DISPLAY                                  X(4)                       174     4
02 OFFICE-PHONE-0450              DISPLAY      OCCURS         3            9(7)                       178    21
02 OFFICE-AREA-CODE-0450          DISPLAY                                  X(3)                       199     3
02 SPEED-DIAL-0450                DISPLAY                                  X(3)                       202     3
```

The LRPATH report

You can obtain additional logical-record information from the LRPATH report, which lists the DBA-defined path retrieval logic. You should use this report to take special note of the following significant statements and clauses in the logical-record definition:

Significant statements and clauses

- **ROLE** assigns a role name (such as MANAGER) to specified logical-record components. The DML compiler copies the data items in that record, subordinate to the role name, into program variable storage at DML compile time. If ROLE is used to assign a unique identifier to a logical-record component that occurs more than once in a single logical record, you must qualify all references to elements in occurrences of the logical-record components.

  Note: Assembler programmers: Assembler programs cannot reference roles.

  For example, if the DBA assigns the role names WORKER and MANAGER to the EMPLOYEE database record, the definitions copied into program variable storage would be as follows:

```
01  EMP-EMP-LR.
    02  MANAGER.
     03  EMP-ID-0415            PIC 9(4).
     03  EMP-NAME-0415.
      04  EMP-FIRST-NAME-0415   PIC X(10).
      04  EMP-LAST-NAME-0415    PIC X(15).
     03  STATUS-0415            PIC X(2).
     03  SS-NUMBER-0415         PIC 9(9).
     03  START-DATE-0415.
      04  START-YEAR-0415       PIC 9(2).
      04  START-MONTH-0415      PIC 9(2).
      04  START-DAY-0415        PIC 9(2).
     03  FILLER                 PIC X(2).
    02  WORKER.
     03  EMP-ID-0415            PIC 9(4).
     03  EMP-NAME-0415.
      04  EMP-FIRST-NAME-0415   PIC X(10).
      04  EMP-LAST-NAME-0415    PIC X(15).
     03  STATUS-0415            PIC X(2).
     03  SS-NUMBER-0415         PIC 9(9).
     03  START-DATE-0415.
      04  START-YEAR-0415       PIC 9(2).
      04  START-MONTH-0415      PIC 9(2).
      04  START-DAY-0415        PIC 9(2).
     03  FILLER                 PIC X(2).
```

To refer to the employee id of the MANAGER record, you would code:

```
EMP-ID-0415 OF MANAGER.
```

ROLE is typically used for reflexive joins and bill-of-materials structures. It can also be used to reduce program-DBMS communication by accessing more than one occurrence of the same record type in one request.

- **VERSION IS** identifies logical-record components that are IDD-defined work records. These records can include derived fields whose values are established through use of the COMPUTE verb.
- **COMPUTE** specifies path logic that can perform arithmetic operations. COMPUTE is typically used either to pass data within the path or to return information to the application program.
- **EVALUATE** specifies path logic that can perform field editing by determining whether a boolean expression is true or false.
- **ON** tests for a specific error status value and indicates the action the path is to take if that value is returned.
- **RETURN** returns path-status information to the application program that reports on whether path processing is interrupted or terminated. The action to be taken by the program is influenced by the CLEAR option.
- **CLEAR** clears to low values the contents of the logical record in program variable storage. If CLEAR is not specified by the RETURN command, partial logical records will be returned to the program; some fields in variable storage may contain data from the previous logical record. Invalid data can be avoided in one of the following ways:
  - The **subschema path** can use the COMPUTE verb to set invalid fields to blanks or zeros.
  - The **program** can set invalid fields to blanks or zeros after the return of a DBA-specified path status.

Sample report

Below is a sample LRPATH report. The LRPATH report provides information on ROLE names, IDD-defined records, path statuses, and partial paths that can assist in the design of your application.

```
IDMSRPTS 15.0                          -LOGICAL RECORD PATH DESCRIPTIONS-                        DATE     TIME   PAGE

LRPATH                            FOR SUBSCHEMA EMPSS71  IN SCHEMA EMPSCHM  VERSION    1          04/06/99  130523


LOGICAL RECORD NAME EMP-MATCH-LR

BUILT FROM.........DEPARTMENT
                   EMPLOYEE
                   EMPLOYEE                          ROLE  MANAGER
                   OFFICE
                   EXPERTISE
                   SKILL
                   MESSAGE-REC       VERSION  0001

  OBTAIN  EMP-MATCH-LR     PATH-GROUP


    SELECT FOR FIELDNAME-EQ DEPARTMENT                     DEPT-ID-0410

                   OBTAIN FIRST DEPARTMENT
                        WHERE  CALCKEY EQ DEPT-ID-0410 OF REQUEST
                       ON 0326 CLEAR RETURN    DEPT-NOT-FOUND
                       ON 0000 NEXT
                   IF DEPT-EMPLOYEE    IS NOT EMPTY
                       ON 0000 CLEAR RETURN    DEPT-EMPTY
                       ON 1601 NEXT
                   OBTAIN EACH  EMPLOYEE         WITHIN DEPT-EMPLOYEE
                       ON 0000 NEXT
                       ON 0307 ITERATE
                   FIND FIRST STRUCTURE        WITHIN REPORTS-TO
                       ON 0307 DO
                   COMPUTE
                        MESSAGE-3 OF LR EQ '** NO MANAGER **'
                       ON 0000 NEXT
                       END
                       ON 0000 DO
                   OBTAIN OWNER MANAGER          WITHIN MANAGES
                       ON 0000 NEXT
                   FIND FIRST STRUCTURE        WITHIN MANAGES
                       ON 0000 NEXT
                       ON 0307 ITERATE
                   FIND OWNER EMPLOYEE         WITHIN REPORTS-TO
                       ON 0000 NEXT
                       END
                   IF OFFICE-EMPLOYEE  MEMBER
                       ON 1601 DO
                   COMPUTE
                        MESSAGE-1 OF LR EQ '** NO OFFICE **'
                       ON 0000 NEXT
                   COMPUTE
                        OFFICE-CODE-0450 OF LR EQ '   '
                       ON 0000 NEXT
                   COMPUTE
                        OFFICE-STREET-0450 OF LR EQ '   '
                       ON 0000 NEXT
                  COMPUTE
                        OFFICE-CITY-0450 OF LR EQ '   '
                       ON 0000 NEXT
                   COMPUTE
                        OFFICE-STATE-0450 OF LR EQ '  '
                       ON 0000 NEXT
                   COMPUTE
                        OFFICE-ZIP-FIRST-FIVE-0450 OF LR EQ '   '
                       ON 0000 NEXT
                       END
                       ON 0000 DO
                   OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
                       ON 0000 NEXT
                       END
                   IF EMP-EXPERTISE    IS NOT EMPTY
                       ON 0000 DO
                   COMPUTE
```

```
      MESSAGE-2 OF LR EQ ' ** NO SKILL **'
    ON 0000 NEXT
COMPUTE
      SKILL-ID-0455 OF LR EQ 0
    ON 0000 NEXT
COMPUTE
      SKILL-NAME-0455 OF LR EQ '            '
    ON 0000 NEXT
COMPUTE
      SKILL-LEVEL-0425 OF LR EQ 0
    ON 0000 NEXT
    END
    ON 1601 DO
OBTAIN EACH  EXPERTISE        WITHIN EMP-EXPERTISE
    ON 0000 NEXT
    ON 0307 ITERATE
OBTAIN OWNER WITHIN SKILL-EXPERTISE
    ON 0000 NEXT
    END
```

Accessing Logical Records

The statements used to access logical records are OBTAIN, MODIFY, STORE, and ERASE. Each of these statements can include the following clauses:

- A **WHERE** clause to specify the logical-record occurrences to be accessed. For more information, see Using the WHERE clause (see page 212) later in this section.
- An **ON** clause to evaluate the result of a request and specify action based on that result. For more information, see The ON clause (see page 222) later in this section.

The OBTAIN, MODIFY, STORE, and ERASE statements are discussed below, followed by a discussion of the WHERE clause.

Retrieving logical records

To retrieve logical records, perform the following steps:

1. Initialize any appropriate symbolic- or CALC-key fields.
2. Issue the OBTAIN statement, optionally specifying selection criteria in the WHERE clause.
3. Test for all possible path statuses.
4. If LRF returns a path status of LR-ERROR, perform the IDMS-STATUS routine.

You can retrieve a single logical record, or you can code iterative logic to retrieve all logical records that meet criteria specified in the WHERE clause. Additionally, you can specify that the retrieved logical record be placed into an alternative variable-storage location.

OBTAIN FIRST and OBTAIN NEXT

If an OBTAIN FIRST statement is followed by an OBTAIN NEXT statement to retrieve a series of occurrences of the same logical record, the OBTAIN statements must direct LRF to the same path. For this reason, you must ensure that the selection criteria specified in the WHERE clauses that accompany the OBTAIN FIRST and OBTAIN NEXT statements are identical.

It is best to use the OBTAIN FIRST/OBTAIN NEXT combination.

OBTAIN NEXT without OBTAIN FIRST

If the program initially issues an OBTAIN NEXT statement without issuing an OBTAIN FIRST, or if the last path status returned for the path was LR-NOT-FOUND, LRF interprets the OBTAIN NEXT as OBTAIN FIRST.

Note: LRF does *not* interpret OBTAIN NEXT as OBTAIN FIRST in the following situations:

- After a path status of LR-FOUND
- After a path status of LR-ERROR
- After any DBA-defined path status

When retrieving logical records by specifying a symbolic- or CALC-key value, always use the OBTAIN FIRST/OBTAIN NEXT combination.

Path statuses are explained in Testing For Path Status (see page 220) later in this section.

Example

The example below shows a program that retrieves logical records. This program obtains the first EMP-INS-LR logical record, checks the path status, and obtains subsequent occurrences of the logical record by using the OBTAIN NEXT statement.

```
PROCEDURE DIVISION.
         .
         .
    OBTAIN FIRST EMP-INS-LR WHERE EMP-ID-0415 =
                                 EMP-ID-0415 OF LR.
    IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
         PERFORM A450-NO-LR
    ELSE
         PERFORM A400-GET-NEXT-EMP-INS-REC THRU A400-EXIT
            UNTIL LR-STATUS = 'LR-NOT-FOUND'.
         .
         .
         .
A400-GET-NEXT-EMP-INS-REC.
         .
         .
         .
    OBTAIN NEXT EMP-INS-LR WHERE EMP-ID-0415 =
                                 EMP-ID-0415 OF LR.
    IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
         GO TO A400-EXIT.
         .
         .
         .
A400-EXIT.
    EXIT.
```

Modifying logical records

To modify logical-record occurrences in the database, perform the following steps:

1. Either retrieve the logical record to be modified or initialize key fields, as specified by the DBA in the LRDEFS report.
2. Issue the MODIFY statement, optionally specifying selection criteria in the WHERE clause. The WHERE clause will be evaluated at the beginning and acts as a security mechanism.
3. Test for all possible path statuses.
4. If LRF returns a path status of LR-ERROR, perform the IDMS-STATUS routine.

Field values

LRF uses the field values present in the variable-storage location reserved for the logical record to update the appropriate database records. You can optionally specify an alternative variable-storage location from which the changed field values are to be taken.

The database record occurrences that are physically modified as a result of this statement are specified by the DBA in the subschema modify path group. Depending on the specifications of the DBA, database records that participate in logical records can be left as is, modified, stored, or erased. Other DBA-determined conditions related to the MODIFY statement include:

- Whether you first need to OBTAIN a logical record before issuing the MODIFY statement
- WHERE clause arguments to be used
- The number of modify paths for each logical record (for example, one MODIFY path might require a keyword)
- Path statuses to be returned to the program

Example

The example below shows a program that modifies a logical record. The program modifies the employee address and phone number fields in the logical record and issues the MODIFY statement. Note that the program checks for the DBA-defined path status INVALID-MODIFY.

```
DATA DIVISION.
FILE SECTION.
FD  NEW-EMP-ADDRESS-FILE-IN.
01  NEW-EMP-ADDRESS-REC-IN.
    02  EMP-ID-IN              PIC 9(4).
    02  NEW-ADDRESS-IN         PIC X(46).
    02  NEW-PHONE-IN           PIC 9(10).
PROCEDURE DIVISION.
         .
    READ NEW-EMP-ADDRESS-FILE-IN.
         AT END MOVE 'Y' TO EOF-SW.
    PERFORM A300-CHANGE-ADDRESS THRU A300-EXIT
                      UNTIL END-OF-FILE.
    FINISH.
    GOBACK.
A300-CHANGE-ADDRESS.
    MOVE EMP-ID-IN TO EMP-ID-0415.
*** OBTAIN SPECIFIED LOGICAL RECORD ***
    OBTAIN FIRST EMP-JOB-LR WHERE EMP-ID-0415 EQ
                                  EMP-ID-0415 OF LR.
*** CHECK FOR PATH STATUSES ***
    IF LR-STATUS = 'LR-ERROR'
       PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
         PERFORM A350-EXCP-RPT
         GO TO A300-GET-NEXT.
    PERFORM U500-WRITE-OLD-ADDRESS.
    MOVE NEW-ADDRESS-IN TO EMP-ADDRESS-0415.
    MOVE NEW-PHONE-IN   TO EMP-PHONE-0415.
*** MODIFY LOGICAL RECORD ***
    MODIFY EMP-JOB-LR.
    IF LR-STATUS = 'LR-ERROR'
       PERFORM IDMS-STATUS.
*** CHECK FOR DBA-DEFINED PATH STATUS ***
    IF LR-STATUS = 'INVALID-MODIFY'
       PERFORM A350-EXCP-RPT.
    IF LR-STATUS = 'LR-NOT-FOUND'
       PERFORM A350-EXCP-RPT.
    PERFORM U510-WRITE-NEW-ADDRESS.
A300-GET-NEXT.
    READ NEW-EMP-ADDRESS-FILE-IN
         AT END MOVE 'Y' TO EOF-SW.
A300-EXIT.
```

```
EXIT.
```

Storing logical records

To store logical-record occurrences in the database, perform the following steps:

1. Either initialize key fields or perform other processing, as specified by the DBA in the LRDEFS report.
2. Issue the STORE statement, optionally specifying selection criteria in the WHERE clause. The WHERE clause will be evaluated at the beginning and acts as a security mechanism.
3. Test for all possible path statuses.
4. If LRF returns a path status of LR-ERROR, perform the IDMS-STATUS routine.

LRF uses the field values present in the variable-storage location reserved for the logical record to update the appropriate database records. You can optionally specify an alternative variable-storage location from which the field values are to be taken.

STORE does not necessarily result in storing new occurrences of any of the database records that participate in the logical record; the path selected to service a STORE logical-record request performs whatever database-access operations the DBA has specified to service the request. Depending on the DBA's specifications, database records that participate in logical records can be left as is, modified, stored, or erased.

STORE conditions

Other DBA-determined conditions related to the STORE statement include:

- Whether you first need to OBTAIN a logical record before issuing the STORE statement
- WHERE clause arguments to be used
- The number of store paths for each logical record (for example, one STORE path might require a keyword)
- Path statuses to be returned to the program

Example

The example below shows a program that stores a logical-record occurrence. This program stores a new occurrence of the EMP-JOB-LR logical record.

```
01  NEW-EMP-REC-IN.
    02  DEPT-ID-IN              PIC 9(4).
    02  EMP-ID-IN               PIC 9(4).
    02  NAME-IN                 PIC X(25).
    02  ADDRESS-IN              PIC X(46).
    02  PHONE-IN                PIC 9(10).
    02  STATUS-IN               PIC X(2).
    02  SS-NUMBER-IN            PIC 9(9).
    02  START-DATE-IN           PIC 9(6).
    02  BIRTH-DATE-IN           PIC 9(6).
PROCEDURE DIVISION.
    READ NEW-EMP-FILE-IN.
         AT END MOVE 'Y' TO EOF-SW.
    PERFORM A300-STORE-EMP THRU A300-EXIT
                       UNTIL END-OF-FILE.
          .
A300-STORE-EMP.
    MOVE DEPT-ID-IN TO DEPT-ID-0410.
*** OBTAIN LR TO ESTABLISH CORRECT DEPARTMENT ***
    OBTAIN FIRST EMP-JOB-LR WHERE DEPT-ID-0410 EQ
                                  DEPT-ID-0410 OF LR.
    IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
       PERFORM A350-EXCP-RPT
       GO TO A300-GET-NEXT.
    PERFORM B300-INITIALIZE-EMPLOYEE.
*** STORE LOGICAL RECORD ***
    STORE EMP-JOB-JR.
    IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
       PERFORM A360-STORE-EXCP-RPT
       GO TO A300-GET-NEXT.
*** CHECK FOR DBA-DEFINED PATH STATUS ***
    IF LR-STATUS = 'INVALID-STORE'
       PERFORM A360-STORE-EXCP-RPT
       GO TO A300-GET-NEXT.
    PERFORM U500-WRITE-NEW-EMP-REPORT.
A300-GET-NEXT.
    READ NEW-EMP-FILE-IN
         AT END MOVE 'Y' TO EOF-SW.
A300-EXIT.
    EXIT.
```

Erasing logical records

To delete a logical-record occurrence from the database, perform the following steps:

1. Either retrieve the logical record to be erased or initialize key fields, as specified by the DBA in the LRDEFS report.
2. Issue the ERASE statement, optionally specifying selection criteria in the WHERE clause. The WHERE clause will be evaluated at the beginning and acts as a security mechanism.
3. Test for all possible path statuses.
4. If LRF returns a path status of LR-ERROR, perform the IDMS-STATUS routine.

Field values

LRF uses field values present in the variable-storage location reserved for the logical record to update the database. You can optionally specify an alternative variable-storage location from which the field values are to be taken (for example, if the logical record was previously obtained using the INTO option).

ERASE does not necessarily result in the deletion of any of the database records that participate in the logical record. The path selected to service an ERASE logical-record request performs whatever database access operations the DBA has specified to service the request. For example, if a DEPARTMENT loses an employee, the EMP-JOB-LR logical record that contains information about that employee would be erased; other information about the department would not be erased. That is, EMPLOYEE information would be erased, but not DEPARTMENT, JOB, or OFFICE information.

Depending on the DBA's specifications, database records that participate in logical records can be left as is, modified, stored, or erased.

ERASE conditions

Other DBA-determined conditions related to the ERASE statement include:

- Whether you first need to OBTAIN a logical record before issuing the ERASE statement
- WHERE clause arguments to be used
- The number of ERASE paths for each logical record (for example, one ERASE path might require a keyword)
- Path statuses to be returned to the program

Example

The example below illustrates a program that deletes a logical-record occurrence. This program OBTAINs the specified occurrence of the EMP-JOB-LR logical record, and issues the ERASE statement. Note that the program checks the DBA-defined path status INVALID-ERASE.

```
       DATA DIVISION.
       FILE SECTION.
       FD  EMP-ERASE-FILE-IN.
       01  EMP-ERASE-REC-IN.
           02  EMP-ID-IN              PIC 9(4).
       PROCEDURE DIVISION.
                .
           READ EMP-ERASE-FILE-IN.
                AT END MOVE 'Y' TO EOF-SW.
           PERFORM A300-ERASE-EMP THRU A300-EXIT
                           UNTIL END-OF-FILE.
           FINISH.
           GOBACK.
       A300-ERASE-EMP.
           MOVE EMP-ID-IN TO EMP-ID-0415.
      *** OBTAIN THE SPECIFIED LOGICAL RECORD ***
           OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 EQ
                                   EMP-ID-0415 OF LR.
           IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
           IF LR-STATUS = 'LR-NOT-FOUND'
              PERFORM A350-EXCP-RPT
              GO TO A300-GET-NEXT.
           PERFORM B300-ERASE-RPT.
      *** ERASE THE LOGICAL RECORD ***
           ERASE EMP-JOB-JR.
           IF LR-STATUS = 'LR-ERROR' PERFORM IDMS-STATUS.
           IF LR-STATUS = 'LR-NOT-FOUND'
              PERFORM A350-EXCP-RPT.
      *** TEST FOR DBA-DEFINED PATH STATUS ***
           IF LR-STATUS = 'INVALID-ERASE'
              PERFORM A350-EXCP-RPT.
       A300-GET-NEXT.
           READ EMP-ERASE-FILE-IN
                AT END MOVE 'Y' TO EOF-SW.
       A300-EXIT.
           EXIT.
```

Using the WHERE clause

You use the WHERE clause with any of the four logical-record database access statements. The WHERE clause allows you to:

- **Direct the program to a path predefined in the subschema by the DBA**. This process is transparent to your program. Through the path, you can access the database without issuing specific instructions for navigating the database.
- **Specify selection criteria to be applied to a logical record**. The selection criteria allow you to specify the set of logical-record occurrences to be accessed. This reduces the need for you to inspect many logical-record occurrences in order to isolate those of interest.
- The WHERE clause will be evaluated at the beginning and acts as a security mechanism.

The WHERE clause is issued in the form of an expression that consists of comparisons and keywords connected by the boolean operators (AND, OR, and NOT). If a user defined function is included in the expression, then it must not contain any external calls. Any attempt to issue a DML or DC request in the user defined function may cause a runtime abend to occur. Comparisons are presented below, followed by a discussion of keywords and LRF coding techniques and path restrictions.

Comparisons

WHERE clause comparisons perform comparison operations on operands included in the WHERE clause expression. LRF uses the results of these operations to determine the specific occurrences of a logical record to be accessed. Additionally, LRF directs each logical-record request to an appropriate path in the subschema, depending on the operands and operators included in the WHERE clause of the request.

Operands used in WHERE clause comparisons

The following table provides examples of the operands that can be used in the WHERE clause.

Operand	Example
Literal	WHERE EMP-ID-0415 = '0466'.
IDD-defined program variable field	WHERE EMP-ID-0415 = IDD-EMP-ID-IN.
Logical-record field	WHERE EMP-ID-0415 = EMP-ID-0415 OF LR.
Arithmetic expression	WHERE SALARY-AMOUNT-0420 > (AVERAGE-SAL-WK-FLD * 2).

Testing relationships

Operators included in a WHERE clause comparison can specify that the following relationships between two operands be tested:

- The value of the left operand **contains** the value of the right operand. Both operands included in the CONTAINS operator must be alphanumeric values and elementary elements.
- Each character in the left operand **matches** the corresponding character in the right operand. The right operand functions as a mask. When MATCHES is specified, LRF compares the left operand with the mask, one character at a time, moving from left to right. The result of the match is either true or false: the result is true if the end of the mask is reached before encountering a character in the left operand that does not match a corresponding character in the mask; the result is false if LRF encounters a character in the left operand that does not match a mask character.

  Three special characters can be used in the mask to perform pattern matching:

  - **@** can be matched with any alphabetic character
  - **#** can be matched with any numeric character
  - ***** can be matched with any alphabetic or numeric character; both the left operand and the mask must be alphanumeric values (in COBOL, PIC X or 9 DISPLAY) and elementary elements
- The value of the left operand is **related** to the value of the right operand in one of the following ways:
  - Equal to
  - Not equal to
  - Greater than
  - Less than
  - Greater than or equal to
  - Less than or equal to

Operations used in WHERE clause comparisons

The following table. provides examples for each of the operators used in the WHERE clause.

Operation	Example
Contains	WHERE EMP-FIRST-NAME-0415 CONTAINS 'SARA'.
Matches	WHERE EMP-ZIP-FIRST-FIVE MATCHES '021##'.
Equal to	WHERE EMP-ID-0415 EQ IDD-EMP-ID-IN.

Operation	Example
Not equal to	WHERE STATUS-0415 NE '05'.
Greater than	WHERE START-YEAR-0415 GT '75'.
Less than	WHERE START-YEAR-0415 LT '75'.
Greater than or equal to	WHERE JOB-ID-0440 GE IDD-JOB-ID-IN.
Less than or equal to	WHERE JOB-ID-0440 LE IDD-JOB-ID-IN.
AND	WHERE EMP-ID-0415 EQ IDD-EMP-ID-IN AND STATUS-0415 EQ '02'.
OR	WHERE START-YEAR-0415 LT '75' OR STATUS-0415 EQ '05'.
Arithmetic expression	WHERE SALARY-AMOUNT-0420 > (AVERAGE-SAL-WK-FLD * 2).

A WHERE clause can contain as many comparisons and keywords as required to specify the criteria to be applied to the logical record. If necessary, the value of the SIZE parameter on the COPY IDMS SUBSCHEMA-LR-CTRL statement can be increased to accommodate very large and complex WHERE clause specifications.

Evaluating the WHERE clause

LRF evaluates operations in a WHERE clause by using the standard precedence of operators in the case of arithmetic expressions. In other cases, it evaluates according to the order of appearance.

The standard precedence of operations is as follows:

1. Unary plus or minus
2. Multiplication or division
3. Addition or subtraction

Parentheses

Parentheses can be used to clarify a multiple-comparison expression or to override the precedence of operations.

Numeric and alphanumeric literals

Numeric literals coded in the WHERE clause and used by the path as CALC keys or sort keys must be enclosed in quotation marks and include leading zeros. Alphanumeric literals used as CALC keys or sort keys must be enclosed in quotation marks and padded with spaces. Numeric literals that are defined as packed (COBOL COMP-3, PL/I fixed decimal) and used as CALC keys or sort keys must be unquoted and include leading zeros.

Keywords

Keywords are names defined by the DBA to simplify processing of logical-record requests. A keyword specified in the WHERE clause routes the request to the subschema path that is associated with that keyword. The path contains logic to select the appropriate logical-record occurrences.

DBA-defined keywords are often used to force a match between a program request and an LRF path. Using keywords in this manner ensures that the program request performs the logic coded in a specific path.

Keywords can also be used in place of detailed comparisons. In this case, the LRF path performs all comparisons; you need only code a WHERE clause that uses that keyword.

Coding techniques and path restrictions

Every logical-record request must direct LRF to an appropriate path in the subschema before the request can be processed. Typically, the DBA codes a number of paths that provide efficient access to the database based on the information included in the WHERE clause. The DBA can also code paths that can be accessed regardless of the contents of the request's WHERE clause. Such paths do not even require that requests include a WHERE clause.

Note: You should not change the WHERE clause selection criteria between an OBTAIN FIRST and an OBTAIN NEXT logical-record request.

A path can require that a WHERE clause name one or more of the following:

- A particular keyword
- A particular value that is used by the path as a CALC key, sort key, or db-key
- A particular logical-record field of any kind
- Any field within a particular logical-record element (that is, within a particular database record)

When a logical-record request is issued, LRF searches through the subschema until it finds a path whose requirements are met by the WHERE clause. LRF uses the *first* such path it finds to process the request, regardless of whether the WHERE clause also meets the requirements of other paths.

Examples

The examples presented on the following pages illustrate various techniques for coding WHERE clauses to meet different path requirements. Each example begins with DBA comments (from the LRDEFS report) that describe the WHERE clause components required to access a specific path. Following these comments, a number of statements are listed, each of which would successfully access that path. (In some cases, invalid solutions are illustrated for comparison.)

Using keywords

The DBA has written the following comments about a path:

```
THIS PATH WILL BE SELECTED IF THE KEYWORD
PROGRMR-ANALYSTS IS INCLUDED IN THE PROGRAM-REQUEST
WHERE CLAUSE.

THIS PATH OBTAINS AN OCCURRENCE OF THE EMP-JOB-LR LOGICAL
RECORD FOR EACH EMPLOYEE WHO IS A PROGRAMMER-ANALYST
(THAT IS, EACH EMPLOYEE WHO HAS A JOB-ID-0440 OF 3025)
```

To access this path, you could code:

```
OBTAIN EMP-JOB-LR WHERE PROGRMR-ANALYSTS.

OBTAIN EMP-JOB-LR WHERE PROGRMR-ANALYSTS AND
                        OFFICE-CODE-0450 EQ '001' AND
                        SKILL-LEVEL-0425 EQ '04.
```

Keywords must be coded in an affirmative and logically conjunctive manner (that is, do not code NOT or OR). The requests listed below would *not* access this path:

```
OBTAIN EMP-JOB-LR WHERE NOT PROGRMR-ANALYSTS.

OBTAIN EMP-JOB-LR WHERE PROGRMR-ANALYSTS OR
                        OFFICE-CODE-0450 EQ '001'.
```

Using CALC keys

The DBA has written the following comments about a path:

```
THIS PATH WILL BE SELECTED IF THE FOLLOWING COMPARISON
IS INCLUDED IN THE PROGRAM-REQUEST WHERE CLAUSE:

EMP-ID-0415 = A LITERAL
              A PROGRAM VARIABLE THAT HAS BEEN DEFINED TO IDD
              A FIELD IN THE AREA RESERVED FOR THE LOGICAL
                RECORD IN PROGRAM VARIABLE STORAGE (OF LR)

NOTE THAT THE EMP-ID-IN VALUE INCLUDED IN THE PROGRAM REQUEST
WHERE CLAUSE IS USED AS A CALC KEY BY THE PATH.
```

To access this path, you could code:

```
OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 = '0447'.

OBTAIN EMP-JOB-LR WHERE EMP-ID-9415 = IDD-EMP-ID-IN.

OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 =
                        EMP-ID-0415 OF LR.

OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 = IDD-EMP-ID-IN
                  AND EMP-STATE-0415 = IDD-STATE-IN.
```

Logical-record fields that are used as CALC keys, sort keys, or db-keys in the WHERE clause can only be used in an equality comparison with a single value. Additionally, the WHERE clause must be logically conjunctive (for example, do not code OR). The requests listed below would *not* access this path:

```
OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 > '0447'.

OBTAIN EMP-JOB-LR WHERE EMP-ID-0415 = (IDD-EMP-ID-IN - 1).

OBTAIN EMP-JOB-LR WHERE (EMP-ID-0415 = '0466') OR
                        (EMP-LAST-NAME-0415 = 'JOHNSON        ').
```

Non-key comparisons

The DBA has written the following comments about a path:

```
THIS PATH WILL BE SELECTED IF THE FOLLOWING COMPARISON
IS INCLUDED IN THE PROGRAM-REQUEST WHERE CLAUSE:

SALARY-AMOUNT-0420 = A LITERAL
                     A PROGRAM VARIABLE THAT HAS BEEN DEFINED TO IDD
                     A FIELD IN THE AREA RESERVED FOR THE LOGICAL
                       RECORD IN PROGRAM VARIABLE STORAGE (OF LR)
                     AN ARITHMETIC EXPRESSION

ANY COMPARISON OPERATION CAN BE USED.
```

All comparisons are allowed for logical-record fields that are not used as CALC keys, sort keys, or db-keys. To access this path, you could code:

```
OBTAIN EMP-JOB-LR WHERE SALARY-AMOUNT-0420 LT
        (IDD-IN-AMT + IDD-ADJUST-VALUE).

OBTAIN EMP-JOB-LR WHERE
        (SALARY-AMOUNT-0420 GT SALARY-AMOUNT-0420 OF LR)
         OR (DESCRIPTION-0440 CONTAINS 'TOP SECRET').

OBTAIN EMP-JOB-LR WHERE
        (JOB-ID-0440 MATCHES 'A###') AND NOT
        (SALARY-AMOUNT-0420 LT (IDD-IN-AMT + IDD-ADJUST-VALUE)).
```

Non-key comparisons

The DBA has written the following comments about a path:

```
THIS PATH WILL BE SELECTED IF THE FOLLOWING COMPARISON
IS INCLUDED IN THE PROGRAM-REQUEST WHERE CLAUSE:

ANY FIELD FROM
THE EMPLOYEE RECORD
OF EMP-JOB-LR = A LITERAL
                A PROGRAM VARIABLE THAT HAS BEEN DEFINED TO IDD
                A FIELD IN THE AREA RESERVED FOR THE LOGICAL
                  RECORD IN PROGRAM VARIABLE STORAGE (OF LR)
                AN ARITHMETIC EXPRESSION

ANY COMPARISON OPERATION CAN BE USED.
```

All comparisons are allowed for database records that participate as elements in logical records. To access this path, you could code:

```
OBTAIN EMP-JOB-LR WHERE EMP-LAST-NAME-0415 =
                        EMP-LAST-NAME-0415 OF LR.

OBTAIN EMP-JOB-LR WHERE EMP-ZIP-FIRST-FIVE EQ '58201'.

OBTAIN EMP-JOB-LR WHERE EMP-LAST-NAME-0415 MATCHES 'C@@@@@@@@@@@@@@'.
```

Generic retrieval

The DBA has written the following comments about a path:

```
THIS PATH WILL BE SELECTED FOR ALL OTHER PROGRAM REQUESTS.
```

Any valid logical-record retrieval request can access this path.

A request that contains no WHERE clause can be processed only by this type of path. To access this path, you could code:

```
OBTAIN EMP-JOB-LR.

OBTAIN EMP-JOB-LR WHERE DESCRIPTION-0440 CONTAINS 'WEATHER'.
```

Testing For Path Status

LRF returns a specific path status to the LR-STATUS field of the program's LRC block to indicate the result of each logical-record request. You can examine this information and use it to determine further processing.

1- to 16-character string

Each path status is 1- to 16-character string; it can be defined by either the system or the DBA. You should check the path status after every call to LRF by using either the host-language IF statement or the ON clause (explained later in this section). Additionally, a path status may indicate that LRF has returned a partial logical record as defined by the DBA.

System-defined path statuses, DBA-defined path statuses, the ON clause, and partial logical records are discussed below, followed by examples of path status testing.

System-defined path statuses

There are three system-defined path statuses:

- **LR-FOUND** is returned when the logical-record request has been executed successfully. This status can be returned in response to any of the four LRF DML statements. When LR-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000.
- **LR-NOT-FOUND** is returned when the logical record specified cannot be found, either because no such record exists or because all logical-record occurrences have already been retrieved. LR-NOT-FOUND can be returned in response to any of the four LRF DML statements, provided that the path to which LRF is directed includes retrieval logic. When LR-NOT-FOUND is returned, the ERROR-STATUS field of the IDMS communications block contains 0000.

  You should always check for LR-NOT-FOUND; if this path status is returned, you should take appropriate action based on the fact that LRF could not find the requested logical record.

  Note: A successful STORE, MODIFY, or ERASE can return a status of LR-NOT-FOUND if its WHERE clause does not evaluate at the start.

- **LR-ERROR** is returned when a logical-record request is issued incorrectly or when an error occurs in the processing of the path selected to service the request. When LR-ERROR is returned, the type of error-status code returned to the program in the ERROR-STATUS field of the IDMS communications block differs according to the type of error:
  - When the error occurs in the **logical-record request**, the ERROR-STATUS field contains an error-status code issued by LRF (major code of 20).
  - When an error occurs in the **logical-record path processing**, the ERROR-STATUS field contains an error-status code issued by the DBMS (major code from 00 to 19).

  Note: When a logical record detects a deadlock condition and returns a DBA-defined path status, the error status field contains '0000' (not 'XX29') and bind errors occur during CA ADS deadlock handling. Allow the path status to default to LR-ERROR to ensure that error status is 'XX29' which will be recognized by CA ADS as well as the dialog which issued the request.

  You should always check for LR-ERROR; if LR-ERROR is returned, you should perform an error routine, such as IDMS-STATUS, before initiating error recovery or aborting the program.

DBA-defined path statuses

The DBA can define additional path statuses in the subschema path group. Typically, these statuses are documented in the subschema comments, which can be retrieved by using the LRDEFS parameter of the IDMSRPTS utility. The LRDEFS report should indicate:

- All DBA-defined path statuses
- Conditions under which DBA-defined path statuses will be returned
- Program action required for each DBA-defined path status

Unless the DBA has specified otherwise, you should test for LR-ERROR before testing for any DBA-defined path statuses.

Note: Programs that issue OBTAIN NEXT requests (allowing OBTAIN NEXT to default to OBTAIN FIRST) should always be sure that LR-NOT-FOUND has been returned for a logical record before OBTAIN NEXT is issued for a different logical record. This also applies to DBA-defined path statuses.

The ON clause

Every LRF DML statement can include an ON clause. The ON clause can be defined to test for a specific path status following an LRF request.

The ON clause of a logical-record request consists of:

- A system- or DBA-defined path status
- An imperative statement directing the program to execute some function based on the result of the test

If LRF returns the path status specified in the ON clause, the imperative statement is executed. If the specified path status is not returned, control is passed as follows:

- In **COBOL and PL/I**, the program performs the IDMS-STATUS routine; this routine will abort the program if a logical-record error (LR-ERROR) has occurred.
- In **Assembler**, control is passed to the next statement in the program.

Partial logical records

LRF is sometimes unable to retrieve all data required to provide the requested logical record. In such cases the DBA can specify that LRF should return that portion of the logical record that it *was* able to obtain. That portion that is returned is referred to as a partial logical record. Typically, you are advised of this condition in a DBA-defined path status.

Keep in mind that those fields in program variable storage for which LRF was unable to provide data can contain values placed there by a previous logical-record request. It is your responsibility to recognize this condition by testing explicitly for a path status that indicates the return of a partial logical record and proceed accordingly.

The example below shows one possible response to partial logical records. Following the LRF request, the program tests for the system-defined path statuses LR-ERROR and LR-NOT-FOUND; it then tests for the DBA-defined path statuses EMP-NOT-FOUND and PARTIAL-LR. IF PARTIAL-LR is returned, the program performs a routine to clear variable-storage fields of any data that remains there from a previous logical-record request.

```
OBTAIN NEXT EMP-JOB-LR
    WHERE EMP-ID-0415 = IDD-EMP-ID-IN.
IF LR-STATUS = 'LR-ERROR'
    PERFORM IDMS-STATUS.
IF LR-STATUS = 'LR-NOT-FOUND'
    MOVE 'N' TO LRF-SW
    GO TO A200-EXIT.
IF LR-STATUS = 'EMP-NOT-FOUND'
    PERFORM A200-EMP-NOT-FOUND
    GO TO A200-EXIT.
IF LR-STATUS = 'PARTIAL-LR'
    PERFORM A200-CLEAR-OLD-DATA.
```

Path status examples

The following examples test for either LR-ERROR or LR-FOUND, LR-NOT-FOUND (where appropriate), and one or more DBA-defined path statuses.

OBTAIN path statuses

This example retrieves logical-record occurrences for a specified employee id. After the OBTAIN statement, the program checks for:

- LR-FOUND
- LR-ERROR
- LR-NOT-FOUND
- EMP-NOT-FOUND
- NO-DEPT
- NO-OFFICE
- NO-JOBS

```
A200-GET-ALL.
    OBTAIN NEXT EMP-JOB-LR
        WHERE EMP-ID-0415 = IDD-EMP-ID-IN.
    IF LR-STATUS = 'LR-FOUND'
        NEXT SENTENCE
    ELSE
        PERFORM A220-CHECK-PATH-STATUS THRU A220-EXIT.
    IF LRF-SW = 'N' GO TO A200-EXIT.
    PERFORM U000-FORMAT.
    PERFORM U100-WRITE-LINE.
A200-EXIT.
    EXIT.
     .
     .
     .
A220-CHECK-PATH-STATUS.
    IF LR-STATUS = 'LR-ERROR'
        PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
        MOVE 'N' TO LRF-SW
        GO TO A220-EXIT.
    IF LR-STATUS = 'EMP-NOT-FOUND'
        PERFORM A220-EMP-NOT-FOUND
        GO TO A220-EXIT.
    IF LR-STATUS = 'NO-DEPT'
        PERFORM A220-NO-DEPT.
        GO TO A220-EXIT.
    IF LR-STATUS = 'NO-OFFICE'
        PERFORM A220-NO-OFFICE.
```

```
            GO TO A220-EXIT.
        IF LR-STATUS = 'NO-JOBS'
            PERFORM A220-NO-JOBS.
            GO TO A220-EXIT.
    A220-EXIT.
        EXIT.
```

MODIFY path statuses

This example requires the keyword MODIFY-EMP to access the correct MODIFY path. After the MODIFY statement, the program checks for:

- LR-ERROR
- LR-NOT-FOUND
- NOT-CURRENT
- NO-KEY-MOD

```
    A200-MOD-EMP.
        OBTAIN FIRST EMP-JOB-LR
            WHERE EMP-ID-0415 = IDD-MOD-EMP-ID-IN.
        IF LR-STATUS = 'LR-FOUND'
            NEXT SENTENCE
        ELSE
            PERFORM A220-CHECK-PATH-STATUS THRU A220-EXIT.
        MOVE NEW-ADDRESS-IN TO EMP-ADDRESS-0415.
        MOVE NEW-PHONE-IN   TO EMP-PHONE-0415.
        MODIFY EMP-JOB-LR WHERE MODIFY-EMP.
        IF LR-STATUS = 'LR-ERROR'
            PERFORM IDMS-STATUS.
        IF LR-STATUS = 'LR-NOT-FOUND'
            PERFORM A220-NOT-FOUND
            GO TO A200-EXIT.
        IF LR-STATUS = 'NOT-CURRENT'
            PERFORM A220-NOT-CURRENT
            GO TO A200-EXIT.
        IF LR-STATUS = 'NO-KEY-MOD'
            PERFORM A220-NO-KEY-MOD
            GO TO A200-EXIT.
        PERFORM U000-FORMAT-MOD-RPT.
        PERFORM U100-WRITE-LINE.
    A200-EXIT.
        EXIT.
```

STORE path statuses

This example requires the program to pass DEPT-ID-0410 in the WHERE clause; it does not require a previous OBTAIN statement. After the STORE statement, the program checks for:

- LR-ERROR
- LR-NOT-FOUND
- DUP-EMP-ID
- INVALID-DEPT-ID

```
A200-STORE-EMP.
    PERFORM A210-INITIALIZE-EMP.
    MOVE DEPT-ID-IN TO DEPT-ID-0410.
    STORE EMP-JOB-LR
        WHERE DEPT-ID-0410 = DEPT-ID-0410 OF LR.
    IF LR-STATUS = 'LR-ERROR'
        PERFORM IDMS-STATUS.
    IF LR-STATUS = 'LR-NOT-FOUND'
        PERFORM A220-LR-NOT-FOUND
        GO TO A200-EXIT.
    IF LR-STATUS = 'DUP-EMP-ID'
        PERFORM A220-DUP-EMP-ID
        GO TO A200-EXIT.
    IF LR-STATUS = 'INVALID-DEPT-ID'
        PERFORM A220-INVALID-DEPT-ID
        GO TO A200-EXIT.
    PERFORM U000-FORMAT-STORE-RPT.
    PERFORM U100-WRITE-LINE.
A200-EXIT.
    EXIT.
```

ERASE path statuses

This example requires the program to OBTAIN the requested occurrence of the EMP-JOB-LR logical record before executing the ERASE statement. After the ERASE statement, the program checks for:

- LR-ERROR
- INVALID-ERASE

```
A200-STORE-EMP.
    OBTAIN FIRST EMP-JOB-LR
        WHERE EMP-ID-0415 = IDD-DEL-EMP-ID-IN.
    IF LR-STATUS = 'LR-FOUND'
        NEXT SENTENCE
    ELSE
        PERFORM A220-CHECK-PATH-STATUS THRU A220-EXIT.
    PERFORM U000-FORMAT-ERASE-RPT.
    ERASE EMP-JOB-LR.
    IF LR-STATUS = 'LR-ERROR'
        PERFORM IDMS-STATUS.
    IF LR-STATUS = 'INVALID-ERASE'
        PERFORM A220-INVALID-ERASE
        GO TO A200-EXIT.
    PERFORM U100-WRITE-LINE.
A200-EXIT.
    EXIT.
```

Appendix A: Sample Subschema EMPLR35

Overview

The EMPLR35 subschema defines the logical record EMP-INFO-LR, which can be accessed through an OBTAIN path group.

Subschema listing

Below is the subschema compiler listing for this subschema.

```
     ADD
     SUBSCHEMA NAME IS EMPLR35 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
*+       DATE CREATED IS      07/17/91
*+       TIME CREATED IS      09010303
*+       DATE LAST UPDATED IS 08/27/91
*+       TIME LAST UPDATED IS 17063809
*+       PREPARED BY DEH
*+       REVISED  BY DEH
         DESCRIPTION IS 'SAMPLE SUBSCHEMA FOR LRF MANUAL'
         PUBLIC ACCESS IS ALLOWED FOR ALL
         USAGE IS LR
         LR CURRENCY NO RESET
         .
     ADD
     AREA NAME IS EMP-DEMO-REGION
         .
     ADD
     AREA NAME IS ORG-DEMO-REGION
         .
     ADD
     RECORD NAME IS EMPLOYEE
         ELEMENTS ARE EMP-ID-0415 EMP-NAME-0415 START-DATE-0415 STATUS-0415
         .
     ADD
     RECORD NAME IS DEPARTMENT
         ELEMENTS ARE DEPT-ID-0410 DEPT-NAME-0410
         .
     ADD
     RECORD NAME IS OFFICE
         ELEMENTS ARE OFFICE-CODE-0450
         .
     ADD
     SET NAME IS EMP-NAME-NDX
         .
```

```
ADD
SET NAME IS DEPT-EMPLOYEE
    .
ADD
SET NAME IS OFFICE-EMPLOYEE
    .
ADD
LOGICAL RECORD NAME IS EMP-INFO-LR
    ON LR-ERROR CLEAR
    ON LR-NOT-FOUND CLEAR
    ELEMENTS ARE
        EMPLOYEE
        DEPARTMENT
        OFFICE
        PATHREC VERSION 1
    COMMENTS
        '****************************************************************'
  -     'THE EMP-INFO-LR LOGICAL RECORD ACCESSES INFORMATION FROM THE'
  -     'EMPLOYEE DATABASE RECORD AND ALSO ACCESSES INFORMATION'
  -     'FROM THE ASSOCIATED DEPARTMENT AND OFFICE RECORDS.'
  -     ' '
  -     'THE FOLLOWING INFORMATION IS RETURNED TO THE PROGRAM, IN'
  -     'THE ORDER SHOWN BELOW:'
  -     ' '
  -     '  EMPLOYEE RECORD   -- EMP-ID-0415, EMP-NAME-0415, START-DATE-0'
  +     '415'
  -     '                       STATUS-0415'
  -     ' '
  -     '  DEPARTMENT RECORD -- DEPT-ID-0440, DEPT-NAME-0440'
  -     ' '
  -     '  OFFICE RECORD     -- OFFICE-CODE-0450'
  -     '  PATHREC VERSION 1 -- WORK-PATH-ID'
  -     ' '
  -     '  NOTE THAT THE WORK-PATH-ID FIELD INDICATES WHICH PATH'
  -     '  WAS EXECUTED.'
  -     '****************************************************************'
  -     ' '
  -     'LR VERBS ALLOWED: OBTAIN'
  -     ' '
  -     '****************************************************************'
```

```
-     ' '
-     'SELECTION CRITERIA (TOTAL OF FIVE PATHS)'
-     ' '
-     '   OBTAIN PATH GROUP:'
-     ' '
-     ' '
-     '    PATH 1)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR ALL EMPLOYEES WHO ARE'
-     '             ON LEAVE.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF THE PROGRAM'
-     '             REQUEST INCLUDES THE KEYWORD ON-LEAVE.'
-     ' '
-     '             IF PATH 1 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 1".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             NO-DEPT    -- THE EMPLOYEE IS NOT ASSOCIATED WITH'
-     '                           A DEPARTMENT'
-     '             NO-OFFICE  -- THE EMPLOYEE IS NOT ASSIGNED TO AN'
-     '                           OFFICE'
-     ' '
-     ' '
-     '    PATH 2)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.'
-     '             IT USES THE EMP-ID-0415 FIELD AS A'
-     '             CALC KEY TO ACCESS EMPLOYEE INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             EMP-ID-0415 = A NUMERIC LITERAL'
-     '                           A PROGRAM VARIABLE'
-     '                           A FIELD IN THE LOGICAL-RECORD'
-     '                            AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
-     '             IF PATH 2 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 2".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             INVALID-ID -- THE INPUT EMPLOYEE ID IS INVALID'
-     '             NO-DEPT    -- THE EMPLOYEE IS NOT ASSOCIATED WITH'
-     '                           A DEPARTMENT'
```

```
-     '            NO-OFFICE  -- THE EMPLOYEE IS NOT ASSIGNED TO AN'
-     '                       OFFICE'
-     ' '
-     ' '
-     '    PATH 3)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR EACH EMPLOYEE IN A'
-     '             PARTICULAR DEPARTMENT.  IT USES THE DEPT-ID-0410'
-     '             FIELD AS A CALC KEY TO ACCESS DEPARTMENT'
-     '             INFORMATION.'
-     ' '
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             DEPT-ID-0440 ='
-     ' '
-     '                          A LITERAL'
-     '                          A PROGRAM VARIABLE'
-     '                          A FIELD IN THE LOGICAL-RECORD'
-     '                           AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
-     '             IF PATH 3 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 3".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             INVALID-DEPT -- THE INPUT DEPARTMENT ID IS INVALID'
-     '             NO-OFFICE    -- THE EMPLOYEE IS NOT ASSIGNED TO AN'
-     '                             OFFICE'
-     ' '
-     ' '
-     '    PATH 4)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR EACH EMPLOYEE ASSIGNED TO'
-     '             A PARTICULAR OFFICE.'
-     '             IT USES THE OFFICE-CODE-0450 FIELD AS A CALC'
-     '             KEY TO ACCESS OFFICE INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             OFFICE-CODE-0450 ='
-     ' '
-     '                          A LITERAL'
-     '                          A PROGRAM VARIABLE'
-     '                          A FIELD IN THE LOGICAL-RECORD'
-     '                           AREA OF PROGRAM VARIABLE STORAGE'
```

```
-     ' '
-     '             IF PATH 4 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 4".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             INVALID-OFFICE -- THE INPUT OFFICE CODE IS INVALID'
-     '             NO-DEPT        -- THE EMPLOYEE IS NOT ASSOCIATED'
-     '                               WITH A DEPARTMENT'
-     ' '
-     ' '
-     '    PATH 5)  THIS PATH RETRIEVES EMPLOYEE, DEPARTMENT, AND'
-     '             OFFICE INFORMATION FOR A PARTICULAR EMPLOYEE.'
-     '             IT USES THE EMP-NAME-NDX TO ACCESS EMPLOYEE'
-     '             INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF EITHER OF THESE'
-     '             FIELDS ARE REFERENCED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             EMP-LAST-NAME-0415 AND EMP-FIRST-NAME-0415'
-     ' '
-     '             EMP-LAST-NAME-0415'
-     ' '
-     '             IF PATH 5 IS SELECTED, THE VALUE OF THE'
-     '             WORK-PATH-ID FIELD WILL BE "PATH 5".'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '
-     '             INVALID-NAME -- THE INPUT EMPLOYEE NAME IS'
-     '                             INVALID'
-     '             NO-DEPT      -- THE EMPLOYEE IS NOT ASSOCIATED'
-     '                             WITH A DEPARTMENT'
-     '             NO-OFFICE    -- THE EMPLOYEE IS NOT ASSIGNED TO'
-     '                             AN OFFICE'

-     ' '
-     ' '
   .
ADD
PATH-GROUP NAME IS OBTAIN EMP-INFO-LR
    SELECT FOR KEYWORD ON-LEAVE
        COMPUTE WORK-PATH-ID OF LR EQ 'PATH 1'
            ON 0000 NEXT
        OBTAIN EACH EMPLOYEE WITHIN EMP-NAME-NDX
            WHERE STATUS-0415 EQ '04'
            ON 0000 NEXT
            ON 0307 ITERATE
```

```
    IF DEPT-EMPLOYEE MEMBER
        ON 1601 RETURN NO-DEPT
        ON 0000 NEXT
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
    FIND CURRENT EMPLOYEE
        ON 0000 NEXT
    IF OFFICE-EMPLOYEE MEMBER
        ON 1601 RETURN NO-OFFICE
        ON 0000 NEXT
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT
SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
    COMPUTE WORK-PATH-ID OF LR EQ 'PATH 2'
        ON 0000 NEXT
    OBTAIN FIRST EMPLOYEE
        WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
        ON 0326 RETURN INVALID-ID
        ON 0000 NEXT
    IF DEPT-EMPLOYEE MEMBER
        ON 1601 RETURN NO-DEPT
        ON 0000 NEXT
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
    FIND CURRENT EMPLOYEE
        ON 0000 NEXT
    IF OFFICE-EMPLOYEE MEMBER
        ON 1601 RETURN NO-OFFICE
        ON 0000 NEXT
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT
SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT
    COMPUTE WORK-PATH-ID OF LR EQ 'PATH 3'
        ON 0000 NEXT
    OBTAIN FIRST DEPARTMENT
        WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
        ON 0326 RETURN INVALID-DEPT
        ON 0000 NEXT

    OBTAIN EACH EMPLOYEE WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
        ON 0307 ITERATE
    IF OFFICE-EMPLOYEE MEMBER
        ON 1601 RETURN NO-OFFICE
        ON 0000 NEXT
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT
SELECT FOR FIELDNAME-EQ OFFICE-CODE-0450 OF OFFICE
    COMPUTE WORK-PATH-ID OF LR EQ 'PATH 4'
        ON 0000 NEXT
```

```
    OBTAIN FIRST OFFICE
        WHERE CALCKEY EQ OFFICE-CODE-0450 OF REQUEST
        ON 0326 RETURN INVALID-OFFICE
        ON 0000 NEXT
    OBTAIN EACH EMPLOYEE WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT
        ON 0307 ITERATE
    IF DEPT-EMPLOYEE MEMBER
        ON 1601 RETURN NO-DEPT
        ON 0000 NEXT
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
SELECT USING INDEX EMP-NAME-NDX
    FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE FIELDNAME
          EMP-FIRST-NAME-0415 OF EMPLOYEE
SELECT USING INDEX EMP-NAME-NDX
    FOR FIELDNAME EMP-LAST-NAME-0415 OF EMPLOYEE
    COMPUTE WORK-PATH-ID OF LR EQ 'PATH 5'
        ON 0000 NEXT
    OBTAIN EACH EMPLOYEE USING INDEX
        ON 0000 NEXT
        ON 0326 RETURN INVALID-NAME
    IF DEPT-EMPLOYEE MEMBER
        ON 1601 RETURN NO-DEPT
        ON 0000 NEXT
    OBTAIN OWNER WITHIN DEPT-EMPLOYEE
        ON 0000 NEXT
    FIND CURRENT EMPLOYEE
        ON 0000 NEXT
    IF OFFICE-EMPLOYEE MEMBER
        ON 1601 RETURN NO-OFFICE
        ON 0000 NEXT
    OBTAIN OWNER WITHIN OFFICE-EMPLOYEE
        ON 0000 NEXT
.
                * * *   END OF DATA   * * *
```

Appendix B: Sample Subschema EMPLR40

Overview

The EMPLR40 subschema defines the logical records EMPMOD-LR and EMPTRANS-LR. Both of these logical records can be accessed through OBTAIN and MODIFY path groups.

Subschema listing

Below is the subschema compiler listing for this subschema.

```
      ADD
      SUBSCHEMA NAME IS EMPLR40 OF SCHEMA NAME IS EMPSCHM VERSION IS 1
 *+       DATE CREATED IS      07/18/91
 *+       TIME CREATED IS      11555265
 *+       DATE LAST UPDATED IS 07/23/91
 *+       TIME LAST UPDATED IS 16260845
 *+       PREPARED BY DEH
 *+       REVISED  BY DEH
          DESCRIPTION IS 'SAMPLE SUBSCHEMA FOR LRF MANUAL'
          PUBLIC ACCESS IS ALLOWED FOR ALL
          USAGE IS LR
          LR CURRENCY NO RESET
 *+       USED BY: PROGRAM IS MOD-D VERSION IS 1
 *+       USED BY: PROGRAM IS STORE-D VERSION IS 1
 *+       USED BY: PROGRAM IS TRANS-D VERSION IS 1
          .
      ADD
      AREA NAME IS EMP-DEMO-REGION
          .
      ADD
      AREA NAME IS ORG-DEMO-REGION
          .
      ADD
      RECORD NAME IS DEPARTMENT
          ELEMENTS ARE DEPT-ID-0410 DEPT-NAME-0410
          .
      ADD
      RECORD NAME IS OFFICE
          ELEMENTS ARE OFFICE-CODE-0450
          .
```

```
ADD
RECORD NAME IS EMPLOYEE
    .
ADD
SET NAME IS DEPT-EMPLOYEE
    .
ADD
SET NAME IS OFFICE-EMPLOYEE
    .
ADD
SET NAME IS EMP-NAME-NDX
    .
ADD
LOGICAL RECORD NAME IS EMP-LR
    ON LR-ERROR CLEAR
    ON LR-NOT-FOUND CLEAR
    ELEMENTS ARE
        EMPLOYEE
        DEPARTMENT
        OFFICE

    COMMENTS
        '*************************************************************'
  -     'THE EMP-LR LOGICAL RECORD LETS YOU STORE A NEW EMPLOYEE RECORD'
  -     'AND MODIFY AN EXISTING EMPLOYEE RECORD.'
  -     ' '
  -     'TO STORE A NEW EMPLOYEE RECORD:'
  -     ' '
  -     '    1) EXECUTE OBTAIN PATH 1, USING A VALID DEPARTMENT ID'
  -     '       AND OFFICE CODE.  THIS PATH SETS CURRENCY ON THE'
  -     '       APPROPRIATE DEPARTMENT AND OFFICE RECORDS.'
  -     ' '
  -     '    2) EXECUTE STORE PATH 1.  THIS PATH USES THE CURRENCIES'
  -     '       ESTABLISHED PREVIOUSLY TO STORE THE NEW EMPLOYEE RECORD.'
  -     '       BE SURE THE NEW EMPLOYEE INFORMATION IS IN THE'
  -     '       LOGICAL-RECORD AREA OF PROGRAM VARIABLE STORAGE.'
  -     ' '
  -     'TO MODIFY AN EXISTING EMPLOYEE RECORD:'
  -     ' '
  -     '    1) EXECUTE OBTAIN PATH 2, USING A VALID EMPLOYEE ID.'
  -     '       THIS PATH RETRIEVES ALL EMPLOYEE INFORMATION.'
  -     ' '
  -     '    2) MAKE THE NECESSARY CHANGES TO THE EMPLOYEE INFORMATION.'
  -     ' '
  -     '    3) EXECUTE MODIFY PATH 1.  THIS PATH MODIFIES THE'
  -     '       EMPLOYEE RECORD AS SPECIFIED.'
  -     ' '
  -     ' '
  -     '*************************************************************'
  -     ' '
```

```
-     'LR VERBS ALLOWED: OBTAIN, STORE, MODIFY'
-     ' '
-     '*************************************************************'
-     ' '
-     '   OBTAIN PATH GROUP:'
-     ' '
-     '   SELECTION CRITERIA (TOTAL OF TWO PATHS)'
-     ' '
-     ' '
-     '    PATH 1)  THIS PATH SETS CURRENCY ON THE APPROPRIATE'
-     '             DEPARTMENT AND OFFICE RECORDS BEFORE A NEW'
-     '             EMPLOYEE RECORD IS STORED.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
-     ' '
-     '             DEPT-ID-0410 AND OFFICE=CODE-0450 ='
-     ' '
-     '                          A LITERAL'
-     '                          A PROGRAM VARIABLE'
-     '                          A FIELD IN THE LOGICAL-RECORD'
-     '                           AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUSES:'
-     ' '

-     '             INVALID-DEPT   -- THE INPUT DEPARTMENT ID IS'
-     '                               INVALID'
-     '             INVALID-OFFICE -- THE INPUT OFFICE CODE IS'
-     '                               INVALID'
-     ' '
-     ' '
-     '    PATH 2)  THIS PATH RETRIEVES ALL EMPLOYEE INFORMATION'
-     '             FOR A PARTICULAR EMPLOYEE.  ONCE THIS'
-     '             INFORMATION IS RETRIEVED, IT CAN BE'
-     '             MODIFIED.  NOTE THAT THE PATH USES THE'
-     '             EMP-ID-0415 FIELD AS A CALC KEY TO ACCESS'
-     '             THIS INFORMATION.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF ANY OF THESE'
-     '             COMPARISONS ARE INCLUDED IN THE PROGRAM WHERE'
-     '             CLAUSE:'
```

```
-     ' '
-     '             EMP-ID-0410 ='
-     ' '
-     '                          A LITERAL'
-     '                          A PROGRAM VARIABLE'
-     '                          A FIELD IN THE LOGICAL-RECORD'
-     '                           AREA OF PROGRAM VARIABLE STORAGE'
-     ' '
-     '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
-     '             PATH STATUS:'
-     ' '
-     '             NO-EMP -- THE REQUESTED EMPLOYEE CANNOT'
-     '                       BE FOUND'
-     ' '
-     ' '
-     '   STORE PATH GROUP:'
-     ' '
-     '   SELECTION CRITERIA (TOTAL OF ONE PATH)'
-     ' '
-     ' '

-     '             THIS PATH STORES A NEW EMPLOYEE RECORD.'
-     '             TO STORE AN EMPLOYEE, YOU MUST FIRST'
-     '             SELECT OBTAIN PATH 1.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF THE PROGRAM'
-     '             REQUEST INCLUDES THE KEYWORD STORE-EMP.'
-     ' '
-     '             IT DOES NOT RETURN ANY DBA-DEFINED'
-     '             PATH STATUSES.'
-     ' '
-     ' '
-     '   MODIFY PATH GROUP:'
-     ' '
-     '   SELECTION CRITERIA (TOTAL OF ONE PATH)'
-     ' '
-     ' '
-     '             THIS PATH MODIFIES AN EXISTING EMPLOYEE RECORD'
-     '             AS SPECIFIED BY THE APPLICATION PROGRAM.'
-     ' '
-     '             THE PATH WILL BE SELECTED IF THE PROGRAM'
-     '             REQUEST INCLUDES THE KEYWORD MOD-EMP.'
-     ' '
-     '             IT DOES NOT RETURN ANY DBA-DEFINED'
-     '             PATH STATUSES.'
-     ' '
-     ' '
   .
ADD
```

```
LOGICAL RECORD NAME IS EMP-TRANS-LR
    ON LR-ERROR CLEAR
    ON LR-NOT-FOUND CLEAR
    ELEMENTS ARE
        EMPLOYEE
        DEPARTMENT ROLE IS OLD-DEPT
        DEPARTMENT ROLE IS NEW-DEPT
    COMMENTS
        '*****************************************************************'
-       'THE EMP-TRANS-LR LOGICAL RECORD MODIFIES THE RELATIONSHIP'
-       'BETWEEN AN EXISTING EMPLOYEE AND HIS OR HER DEPARTMENT.'
-       'IT IS USED TO PROCESS AN EMPLOYEE TRANSFER FROM ONE'
-       'DEPARTMENT TO ANOTHER.'
-       ' '
-       ' '
-       '*****************************************************************'
-       ' '
-       'LR VERBS ALLOWED: MODIFY'
-       ' '
-       '*****************************************************************'
-       ' '
-       '   MODIFY PATH GROUP:'
-       ' '
-       '   SELECTION CRITERIA (TOTAL OF ONE PATH)'
-       ' '
-       ' '
-       '             THIS PATH PROCESSES THE TRANSFER OF AN EMPLOYEE'
-       '             FROM ONE DEPARTMENT TO ANOTHER.  THE PATH'
-       '             DOES THE FOLLOWING:'
-       ' '
-       '               1) CHECKS THAT THE OLD DEPARTMENT ID, NEW'
-       '                  DEPARTMENT ID, AND EMPLOYEE ID ARE VALID'
-       ' '
-       '               2) DISCONNECTS THE EMPLOYEE RECORD FROM'
-       '                  THE OLD DEPARTMENT'
-       ' '
-       '               3) CONNECTS THE EMPLOYEE RECORD TO THE NEW'
-       '                  DEPARTMENT'
-       ' '
-       '             THE PATH USES THE FOLLOWING ROLE NAMES:'
-       ' '
-       '                OLD-DEPT REPRESENTS THE DEPARTMENT THAT THE'
-       '                EMPLOYEE IS CURRENTLY A MEMBER OF'
-       ' '
-       '                NEW-DEPT REPRESENTS THE DEPARTMENT THAT THE'
-       '                EMPLOYEE WILL TRANSFER TO'
-       ' '
-       '           THE PATH WILL BE SELECTED IF A VALUE FOR DEPT-ID-0415'
        '           OF OLD-DEPT IS PASSED THROUGH PROGRAM VARIABLE STORAGE'
```

```
           '           AND IF THESE COMPARISONS ARE INCLUDED IN THE PROGRAM'
      -    '           WHERE CLAUSE:'
      -    ' '
      -    '             EMP-ID-0415 AND DEPT-ID-0415 OF NEW-DEPT ='
      -    ' '
      -    '                          A LITERAL'
      -    '                          A PROGRAM VARIABLE'
      -    '                          A FIELD IN THE LOGICAL-RECORD'
      -    '                           AREA OF PROGRAM VARIABLE STORAGE'
      -    ' '
      -    '             THE PATH CAN RETURN THE FOLLOWING DBA-DEFINED'
      -    '             PATH STATUSES:'
      -    ' '
      -    '             INVALID-ID        -- THE INPUT EMPLOYEE ID'
      -    '                                  IS INVALID'
      -    '             INVALID-NEW-DEPT  -- THE INPUT NEW DEPARTMENT'
      -    '                                  ID IS INVALID'
      -    '             INVALID-OLD-DEPT  -- THE INPUT OLD DEPARMENT'
      -    '                                  ID IS INVALID'
      -    '             NO-DEPT           -- THE REQUESTED EMPLOYEE'
      -    '                                  IS NOT PART OF THE'
      -    '                                  INDICATED DEPARTMENT'
      -    ' '
      -    ' '

    .
ADD
PATH-GROUP NAME IS MODIFY EMP-LR
    SELECT FOR KEYWORD MOD-EMP
        FIND CURRENT EMPLOYEE
            ON 0000 NEXT
        MODIFY EMPLOYEE
            ON 0000 NEXT
    .
ADD
PATH-GROUP NAME IS OBTAIN EMP-LR
    SELECT FOR FIELDNAME-EQ DEPT-ID-0410 OF DEPARTMENT FIELDNAME-EQ
          OFFICE-CODE-0450 OF OFFICE
        OBTAIN FIRST DEPARTMENT
            WHERE CALCKEY EQ DEPT-ID-0410 OF REQUEST
            ON 0000 NEXT
            ON 0326 RETURN INVALID-DEPT
        OBTAIN FIRST OFFICE
            WHERE CALCKEY EQ OFFICE-CODE-0450 OF REQUEST
            ON 0000 NEXT
            ON 0326 RETURN INVALID-OFFICE
    SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
        OBTAIN FIRST EMPLOYEE
            WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
            ON 0000 NEXT
```

```
            ON 0326 RETURN NO-EMP
    .
ADD
PATH-GROUP NAME IS STORE EMP-LR
    SELECT FOR KEYWORD STORE-EMP
        FIND CURRENT DEPARTMENT
            ON 0000 NEXT
        FIND CURRENT OFFICE
            ON 0000 NEXT
        STORE EMPLOYEE
            ON 0000 NEXT
    .
ADD
PATH-GROUP NAME IS MODIFY EMP-TRANS-LR
    SELECT FOR FIELDNAME-EQ EMP-ID-0415 OF EMPLOYEE
        FIELDNAME-EQ DEPT-ID-0410 OF NEW-DEPT
        FIND FIRST NEW-DEPT
            WHERE CALCKEY EQ DEPT-ID-0410 OF NEW-DEPT OF REQUEST
            ON 0000 NEXT
            ON 0326 RETURN INVALID-NEW-DEPT
        OBTAIN FIRST EMPLOYEE
            WHERE CALCKEY EQ EMP-ID-0415 OF REQUEST
            ON 0000 NEXT
            ON 0326 RETURN INVALID-ID
        IF DEPT-EMPLOYEE MEMBER

            ON 0000 NEXT
            ON 1601 RETURN NO-DEPT
        OBTAIN OWNER OLD-DEPT WITHIN DEPT-EMPLOYEE
            WHERE DEPT-ID-0410 OF OLD-DEPT EQ DEPT-ID-0410 OF OLD-DEPT OF
              LR
            ON 0000 NEXT
            ON 0326 RETURN INVALID-OLD-DEPT
        DISCONNECT EMPLOYEE FROM DEPT-EMPLOYEE
            ON 0000 NEXT
        OBTAIN FIRST NEW-DEPT
            WHERE CALCKEY EQ DEPT-ID-0410 OF NEW-DEPT OF REQUEST
            ON 0000 NEXT
            ON 0326 ITERATE
        CONNECT EMPLOYEE TO DEPT-EMPLOYEE
            ON 0000 NEXT
    .
```

Appendix C: Sample Subschema EMPSCHM

Overview

The EMPSCHM schema defines defines the sample employee database.

The data structure diagram for this database is found in Chapter 3, Preliminary Analysis and Design.

Schema listing

Below is the schema compiler listing for this schema.

```
       ADD
       SCHEMA NAME IS EMPSCHM VERSION IS 1
*+         DATE CREATED IS      12/17/90
*+         TIME CREATED IS      14065127
*+         DATE LAST UPDATED IS 12/30/90
*+         TIME LAST UPDATED IS 14035955
*+         PREPARED BY TDB
*+         REVISED  BY TDB
           ASSIGN RECORD IDS FROM 1001
           PUBLIC ACCESS IS ALLOWED FOR ALL
*+         SUBSCHEMA IS EMPLR35
*+         SUBSCHEMA IS EMPLR40
           .
       ADD
       AREA NAME IS EMP-DEMO-REGION
           .
       ADD
       AREA NAME IS INS-DEMO-REGION
           .
       ADD
       AREA NAME IS ORG-DEMO-REGION
           .
```

```
        ADD
        RECORD NAME IS COVERAGE
*+          USES STRUCTURE OF RECORD COVERAGE VERSION 100
            RECORD ID IS 400
            LOCATION MODE IS VIA EMP-COVERAGE SET
            RECORD SYNONYM NAME FOR ASSEMBLER IS COVERGE
            .
        02 SELECTION-DATE-0400
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS COVSELDT
            .
            03 SELECTION-YEAR-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVSELYR
                .
            03 SELECTION-MONTH-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVSELMO
                .
            03 SELECTION-DAY-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVSELDA
                .
        02 TERMINATION-DATE-0400
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS COVTRMDT
            .
            03 TERMINATION-YEAR-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVTRMYR
                .
            03 TERMINATION-MONTH-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVTRMMO
                .
            03 TERMINATION-DAY-0400
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS COVTRMDA
                .
        02 TYPE-0400
            PICTURE IS  X
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS COVTYPE
```

```
            .
                        88 MASTER-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS 'M'
                        SYNONYM NAME FOR ASSEMBLER IS COVMASTR
                        .
                        88 FAMILY-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS 'F'
                        SYNONYM NAME FOR ASSEMBLER IS COVFAMLY
                        .
                        88 DEPENDENT-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS 'D'
                        SYNONYM NAME FOR ASSEMBLER IS COVDPNDT
                        .
        02 INS-PLAN-CODE-0400
            PICTURE IS  X(3)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS COVPLNCD
            .
                        88 GROUP-LIFE-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS '001'
                        .
                        88 HMO-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS '002'
                        .
                        88 GROUP-HEALTH-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS '003'
                        .
                        88 GROUP-DENTAL-0400
                        USAGE IS CONDITION-NAME
                        VALUE IS '004'
                        .
        ADD
        RECORD NAME IS DENTAL-CLAIM
*+          USES STRUCTURE OF RECORD DENTAL-CLAIM VERSION 100
            RECORD ID IS 405
            LOCATION MODE IS VIA COVERAGE-CLAIMS SET
            MINIMUM ROOT LENGTH IS 132 CHARACTERS
            MINIMUM FRAGMENT LENGTH IS 930 CHARACTERS
            RECORD SYNONYM NAME FOR ASSEMBLER IS DENTCLM
            .
        02 CLAIM-DATE-0405
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS DCCLMDT
```

```
    .
    03 CLAIM-YEAR-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCCLMYR
        .
    03 CLAIM-MONTH-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCCLMMO
        .
    03 CLAIM-DAY-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCCLMDA
        .
02 PATIENT-NAME-0405
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCPNAME
    .
    03 PATIENT-FIRST-NAME-0405
        PICTURE IS  X(10)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCPFNAME
        .
    03 PATIENT-LAST-NAME-0405
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCPLNAME
        .
02 PATIENT-BIRTH-DATE-0405
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCPBIRDT
    .
    03 PATIENT-BIRTH-YEAR-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCPBIRYR
        .
    03 PATIENT-BIRTH-MONTH-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCPBIRMO
        .
    03 PATIENT-BIRTH-DAY-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCPBIRDA
        .
```

```
02 PATIENT-SEX-0405
    PICTURE IS  X
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCPSEX
    .
02 RELATION-TO-EMPLOYEE-0405
    PICTURE IS  X(10)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCRELEMP
    .
02 DENTIST-NAME-0405
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCDNNAME
    .
    03 DENTIST-FIRST-NAME-0405
        PICTURE IS  X(10)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNFNAM
        .
    03 DENTIST-LAST-NAME-0405
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNLNAM
        .
02 DENTIST-ADDRESS-0405
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCDNADDR
    .
    03 DENTIST-STREET-0405
        PICTURE IS  X(20)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNSTR
        .
    03 DENTIST-CITY-0405
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNCITY
        .
    03 DENTIST-STATE-0405
        PICTURE IS  X(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNSTAT
        .
    03 DENTIST-ZIP-0405
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCDNZIP
        .
        04 DENTIST-ZIP-FIRST-FIVE-0405
            PICTURE IS  X(5)
```

```
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS DCDNZPF5
            .
        04 DENTIST-ZIP-LAST-FOUR-0405
            PICTURE IS  X(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS DCDNZPL4
            .
02 DENTIST-LICENSE-NUMBER-0405
    PICTURE IS  9(6)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS DCDNLICN
    .
02 NUMBER-OF-PROCEDURES-0405
    PICTURE IS  9(2)
    USAGE IS COMP
    SYNONYM NAME FOR ASSEMBLER IS DCNOPROC
    .
02 FILLER
    PICTURE IS  XXX
    USAGE IS DISPLAY
    .
02 DENTIST-CHARGES-0405
    USAGE IS DISPLAY
    OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER-OF-PROCEDURES-0405
    SYNONYM NAME FOR ASSEMBLER IS DCDNCHGS
    .
    03 TOOTH-NUMBER-0405
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCTOTHNO
        .
    03 SERVICE-DATE-0405
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS DCSERVDT
        .
        04 SERVICE-YEAR-0405
            PICTURE IS  9(2)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS DCSERVYR
            .
        04 SERVICE-MONTH-0405
            PICTURE IS  9(2)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS DCSERVMO
            .
        04 SERVICE-DAY-0405
            PICTURE IS  9(2)
            USAGE IS DISPLAY
```

```
                    SYNONYM NAME FOR ASSEMBLER IS DCSERVDA
                    .
                03 PROCEDURE-CODE-0405
                    PICTURE IS  9(4)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS DCPROCCD
                    .
                03 DESCRIPTION-OF-SERVICE-0405
                    PICTURE IS  X(60)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS DCDESCSV
                    .
                03 FEE-0405
                    PICTURE IS  S9(7)V99
                    USAGE IS COMP-3
                    SYNONYM NAME FOR ASSEMBLER IS DCFEE
                    .
                03 FILLER
                    PICTURE IS  XXX
                    USAGE IS DISPLAY
                    .
            ADD
            RECORD NAME IS DEPARTMENT
*+              USES STRUCTURE OF RECORD DEPARTMENT VERSION 100
                RECORD ID IS 410
                LOCATION MODE IS CALC USING ( DEPT-ID-0410 ) DUPLICATES ARE
                      NOT ALLOWED
                RECORD SYNONYM NAME FOR ASSEMBLER IS DEPARTMT
                .
            02 DEPT-ID-0410
                PICTURE IS  9(4)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS DEPTID
                .
            02 DEPT-NAME-0410
                PICTURE IS  X(45)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS DEPTNAME
                .
            02 DEPT-HEAD-ID-0410
                PICTURE IS  9(4)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS DEPTHDID
                .
            02 FILLER
                PICTURE IS  XXX
                USAGE IS DISPLAY

                .
            ADD
```

```
       RECORD NAME IS EMPLOYEE
*+         USES STRUCTURE OF RECORD EMPLOYEE VERSION 100
           RECORD ID IS 415
           LOCATION MODE IS CALC USING ( EMP-ID-0415 ) DUPLICATES ARE
                 NOT ALLOWED
           RECORD SYNONYM NAME FOR ASSEMBLER IS EMPLOYE
           .
       02 EMP-ID-0415
           PICTURE IS  9(4)
           USAGE IS DISPLAY
           SYNONYM NAME FOR ASSEMBLER IS EMPID
           .
       02 EMP-NAME-0415
           USAGE IS DISPLAY
           SYNONYM NAME FOR ASSEMBLER IS EMPNAME
           .
           03 EMP-FIRST-NAME-0415
               PICTURE IS  X(10)
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPFNAME
               .
           03 EMP-LAST-NAME-0415
               PICTURE IS  X(15)
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPLNAME
               .
       02 EMP-ADDRESS-0415
           USAGE IS DISPLAY
           SYNONYM NAME FOR ASSEMBLER IS EMPADDR
           .
           03 EMP-STREET-0415
               PICTURE IS  X(20)
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPSTRET
               .
           03 EMP-CITY-0415
               PICTURE IS  X(15)
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPCITY
               .
           03 EMP-STATE-0415
               PICTURE IS  X(2)
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPSTATE
               .
           03 EMP-ZIP-0415
               USAGE IS DISPLAY
               SYNONYM NAME FOR ASSEMBLER IS EMPZIP
               .
```

```
        04 EMP-ZIP-FIRST-FIVE-0415
            PICTURE IS  X(5)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EMPZIPF5
            .
        04 EMP-ZIP-LAST-FOUR-0415
            PICTURE IS  X(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EMPZIPL4
            .
02 EMP-PHONE-0415
    PICTURE IS  9(10)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPPHONE
    .
02 STATUS-0415
    PICTURE IS  X(2)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPSTATU
    .
                88 ACTIVE-0415
                USAGE IS CONDITION-NAME
                VALUE IS '01'
                .
                88 ST-DISABIL-0415
                USAGE IS CONDITION-NAME
                VALUE IS '02'
                SYNONYM NAME FOR ASSEMBLER IS STDSBL
                .
                88 LT-DISABIL-0415
                USAGE IS CONDITION-NAME
                VALUE IS '03'
                SYNONYM NAME FOR ASSEMBLER IS LTDSBL
                .
                88 LEAVE-OF-ABSENCE-0415
                USAGE IS CONDITION-NAME
                VALUE IS '04'
                SYNONYM NAME FOR ASSEMBLER IS LVOFAB
                .
                88 TERMINATED-0415
                USAGE IS CONDITION-NAME
                VALUE IS '05'
                SYNONYM NAME FOR ASSEMBLER IS TRMINATD
                .
02 SS-NUMBER-0415
    PICTURE IS  9(9)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPSSNUM
    .
```

```
02 START-DATE-0415
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPSTDT
    .
    03 START-YEAR-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPSTYR
        .
    03 START-MONTH-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPSTMO
        .
    03 START-DAY-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPSTDA
        .
02 TERMINATION-DATE-0415
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPTRMDT
    .
    03 TERMINATION-YEAR-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPTRMYR
        .
    03 TERMINATION-MONTH-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPTRMMO
        .
    03 TERMINATION-DAY-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPTRMDA
        .
02 BIRTH-DATE-0415
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS EMPBIRDT
    .
    03 BIRTH-YEAR-0415
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS EMPBIRYR
        .
    03 BIRTH-MONTH-0415
        PICTURE IS  9(2)
```

```
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EMPBIRMO
                .
            03 BIRTH-DAY-0415
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EMPBIRDA
                .
        02 FILLER
            PICTURE IS  XX
            USAGE IS DISPLAY
            .
        ADD
        RECORD NAME IS EMPOSITION
*+          USES STRUCTURE OF RECORD EMPOSITION VERSION 100
            RECORD ID IS 420
            LOCATION MODE IS VIA EMP-EMPOSITION SET
            RECORD SYNONYM NAME FOR ASSEMBLER IS EMPOSITN
            .
        02 START-DATE-0420
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EPSTRTDT
            .
            03 START-YEAR-0420
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPSTRTYR
                .
            03 START-MONTH-0420
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPSTRTMO
                .
            03 START-DAY-0420
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPSTRTDA
                .
        02 FINISH-DATE-0420
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EPFINIDT
            .
            03 FINISH-YEAR-0420
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPFINIYR
                .
            03 FINISH-MONTH-0420
                PICTURE IS  9(2)
```

```
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPFINIMO
                .
            03 FINISH-DAY-0420
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS EPFINIDA
                .
        02 SALARY-GRADE-0420
            PICTURE IS  9(2)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EPSALGRD
            .
        02 SALARY-AMOUNT-0420
            PICTURE IS  S9(7)V99
            USAGE IS COMP-3
            SYNONYM NAME FOR ASSEMBLER IS EPSALAMT
            .
        02 BONUS-PERCENT-0420
            PICTURE IS  SV999
            USAGE IS COMP-3
            SYNONYM NAME FOR ASSEMBLER IS EPBONPCT
            .
        02 COMMISSION-PERCENT-0420
            PICTURE IS  SV999
            USAGE IS COMP-3
            SYNONYM NAME FOR ASSEMBLER IS EPCMMPCT
            .
        02 OVERTIME-RATE-0420
            PICTURE IS  S9V99
            USAGE IS COMP-3
            SYNONYM NAME FOR ASSEMBLER IS EPOTRATE
            .
        02 FILLER
            PICTURE IS  XXX
            USAGE IS DISPLAY
            .
        ADD
        RECORD NAME IS EXPERTISE
*+          USES STRUCTURE OF RECORD EXPERTISE VERSION 100
            RECORD ID IS 425
            LOCATION MODE IS VIA EMP-EXPERTISE SET
            RECORD SYNONYM NAME FOR ASSEMBLER IS EXPRTISE
            .
        02 SKILL-LEVEL-0425
            PICTURE IS  XX
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS EXPSKLVL
            .
```

```
                        88 EXPERT-0425
                        USAGE IS CONDITION-NAME
                        VALUE IS '04'

                        .
                        88 PROFICIENT-0425
                        USAGE IS CONDITION-NAME
                        VALUE IS '03'
                        SYNONYM NAME FOR ASSEMBLER IS PROFICNT
                        .
                        88 COMPETENT-0425
                        USAGE IS CONDITION-NAME
                        VALUE IS '02'
                        SYNONYM NAME FOR ASSEMBLER IS COMPETNT
                        .
                        88 ELEMENTARY-0425
                        USAGE IS CONDITION-NAME
                        VALUE IS '01'
                        SYNONYM NAME FOR ASSEMBLER IS ELEMNTRY
                        .
                02 EXPERTISE-DATE-0425
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS EXPDATE
                    .
                    03 EXPERTISE-YEAR-0425
                        PICTURE IS  9(2)
                        USAGE IS DISPLAY
                        SYNONYM NAME FOR ASSEMBLER IS EXPYEAR
                        .
                    03 EXPERTISE-MONTH-0425
                        PICTURE IS  9(2)
                        USAGE IS DISPLAY
                        SYNONYM NAME FOR ASSEMBLER IS EXPMONTH
                        .
                    03 EXPERTISE-DAY-0425
                        PICTURE IS  9(2)
                        USAGE IS DISPLAY
                        SYNONYM NAME FOR ASSEMBLER IS EXPDAY
                        .
                ADD
                RECORD NAME IS HOSPITAL-CLAIM
            *+      USES STRUCTURE OF RECORD HOSPITAL-CLAIM VERSION 100
                    RECORD ID IS 430
                    LOCATION MODE IS VIA COVERAGE-CLAIMS SET
                    RECORD SYNONYM NAME FOR ASSEMBLER IS HOSPCLM
                    .
                02 CLAIM-DATE-0430
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS HCCLMDT
                    .
```

```
    03 CLAIM-YEAR-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCCLMYR
        .
    03 CLAIM-MONTH-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCCLMMO
        .
    03 CLAIM-DAY-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCCLMDAY
        .
02 PATIENT-NAME-0430
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCPTNAME
    .
    03 PATIENT-FIRST-NAME-0430
        PICTURE IS  X(10)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCPTFNAM
        .
    03 PATIENT-LAST-NAME-0430
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCPTLNAM
        .
02 PATIENT-BIRTH-DATE-0430
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCPTBDAT
    .
    03 PATIENT-BIRTH-YEAR-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCPTBYR
        .
    03 PATIENT-BIRTH-MONTH-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCPTBMO
        .
    03 PATIENT-BIRTH-DAY-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCPTBDA
        .
02 PATIENT-SEX-0430
```

```
    PICTURE IS  X
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCPTSEX
    .
02 RELATION-TO-EMPLOYEE-0430
    PICTURE IS  X(10)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCRELEMP
    .
02 HOSPITAL-NAME-0430
    PICTURE IS  X(25)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCHSPNAM
    .
02 HOSP-ADDRESS-0430
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCHSPADD
    .
    03 HOSP-STREET-0430
        PICTURE IS  X(20)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCHSPSTR
        .
    03 HOSP-CITY-0430
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCHSPCTY
        .
    03 HOSP-STATE-0430
        PICTURE IS  X(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCHSPSTA
        .
    03 HOSP-ZIP-0430
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCHSPZIP
        SYNONYM NAME FOR FORTRAN IS HCHZIP
        .
        04 HOSP-ZIP-FIRST-FIVE-0430
            PICTURE IS  X(5)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS HCHSPZF5
            .
        04 HOSP-ZIP-LAST-FOUR-0430
            PICTURE IS  X(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS HCHSPZL4
            .
02 ADMIT-DATE-0430
```

```
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCADMTDT
    .
    03 ADMIT-YEAR-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCADMTYR
        .
    03 ADMIT-MONTH-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCADMTMO
        .
    03 ADMIT-DAY-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCADMTDA
        .
02 DISCHARGE-DATE-0430
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCDSCGDT
    .
    03 DISCHARGE-YEAR-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCDSCGYR
        .
    03 DISCHARGE-MONTH-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCDSCGMO
        .
    03 DISCHARGE-DAY-0430
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCDSCGDA
        .
02 DIAGNOSIS-0430
    PICTURE IS  X(60)
    USAGE IS DISPLAY
    OCCURS 2 TIMES
    SYNONYM NAME FOR ASSEMBLER IS HCDIAGN
    .
02 HOSPITAL-CHARGES-0430
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS HCHSPCHG
    .
    03 ROOM-AND-BOARD-0430
        USAGE IS DISPLAY
```

```
        SYNONYM NAME FOR ASSEMBLER IS HCRMBRD
        .
        04 WARD-0430
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS HCWARD
            .
            05 WARD-DAYS-0430
                PICTURE IS  S9(5)
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCWDDAYS
                .
            05 WARD-RATE-0430
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCWDRATE
                .
            05 WARD-TOTAL-0430
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCWDTOTL
                .
        04 SEMI-PRIVATE-0430
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS HCSPRIV
            .
            05 SEMI-DAYS-0430
                PICTURE IS  S9(5)
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCSDAYS
                .
            05 SEMI-RATE-0430
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCSRATE
                .
            05 SEMI-TOTAL-0430
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS HCSTOTAL
                .
    03 OTHER-CHARGES-0430
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS HCOTHCHG
        .
        04 DELIVERY-COST-0430
            PICTURE IS  S9(7)V99
            USAGE IS COMP-3
            SYNONYM NAME FOR ASSEMBLER IS HCDELVCH
            .
```

```
                04 ANESTHESIA-COST-0430
                    PICTURE IS  S9(7)V99
                    USAGE IS COMP-3
                    SYNONYM NAME FOR ASSEMBLER IS HCANSTHC
                    .
                04 LAB-COST-0430
                    PICTURE IS  S9(7)V99
                    USAGE IS COMP-3
                    SYNONYM NAME FOR ASSEMBLER IS HCLABCST
                    .
        ADD
        RECORD NAME IS INSURANCE-PLAN
*+          USES STRUCTURE OF RECORD INSURANCE-PLAN VERSION 100
            RECORD ID IS 435
            LOCATION MODE IS CALC USING ( INS-PLAN-CODE-0435 )
                  DUPLICATES ARE NOT ALLOWED
            RECORD SYNONYM NAME FOR ASSEMBLER IS INSPLAN
            .
        02 INS-PLAN-CODE-0435
            PICTURE IS  X(3)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS INPCODE
            .
                        88 GROUP-LIFE-0435
                        USAGE IS CONDITION-NAME
                        VALUE IS '001'
                        SYNONYM NAME FOR ASSEMBLER IS GROUPLIF
                        .
                        88 HMO-0435
                        USAGE IS CONDITION-NAME
                        VALUE IS '002'
                        .
                        88 GROUP-HEALTH-0435
                        USAGE IS CONDITION-NAME
                        VALUE IS '003'
                        SYNONYM NAME FOR ASSEMBLER IS GRPHLTH
                        .
                        88 GROUP-DENTAL-0435
                        USAGE IS CONDITION-NAME
                        VALUE IS '004'
                        SYNONYM NAME FOR ASSEMBLER IS GROUPDNT
                        .
        02 INS-CO-NAME-0435
            PICTURE IS  X(45)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS INPCNAME
            .
        02 INS-CO-ADDRESS-0435
            USAGE IS DISPLAY
```

```
    SYNONYM NAME FOR ASSEMBLER IS INPCADDR
    .
    03 INS-CO-STREET-0435
        PICTURE IS  X(20)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS INPCSTRT
        .
    03 INS-CO-CITY-0435
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS INPCCITY
        .
    03 INS-CO-STATE-0435
        PICTURE IS  X(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS INPCSTAT
        .
    03 INS-CO-ZIP-0435
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS INPCZIP
        .
        04 INS-CO-ZIP-FIRST-FIVE-0435
            PICTURE IS  X(5)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS INPCZPF5
            .
        04 INS-CO-ZIP-LAST-FOUR-0435
            PICTURE IS  X(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS INPCZPL4
            .
02 INS-CO-PHONE-0435
    PICTURE IS  9(10)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS INPCPHON
    .
02 GROUP-NUMBER-0435
    PICTURE IS  9(6)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS INPGRPNO
    .
02 PLAN-DESCRIPTION-0435
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS INPDESCR
    .
    03 DEDUCT-0435
        PICTURE IS  S9(7)V99
        USAGE IS COMP-3
        SYNONYM NAME FOR ASSEMBLER IS INPDEDCT
```

```
                .
            03 MAXIMUM-LIFE-COST-0435
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS INPMXLIF
                .
            03 FAMILY-COST-0435
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS INPFAMCS
                .
            03 DEP-COST-0435
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS INPDEPCS
                .
        02 FILLER
            PICTURE IS  XX
            USAGE IS DISPLAY
            .
        ADD
        RECORD NAME IS JOB
*+          USES STRUCTURE OF RECORD JOB VERSION 100
            RECORD ID IS 440
            LOCATION MODE IS CALC USING ( JOB-ID-0440 ) DUPLICATES ARE
                  NOT ALLOWED
            MINIMUM ROOT LENGTH IS 24 CHARACTERS
            MINIMUM FRAGMENT LENGTH IS 296 CHARACTERS
            CALL IDMSCOMP BEFORE STORE
            CALL IDMSCOMP BEFORE MODIFY
            CALL IDMSDCOM AFTER GET
            RECORD SYNONYM NAME FOR ASSEMBLER IS JOBA
            .
        02 JOB-ID-0440
            PICTURE IS  9(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS JOBID
            .
        02 TITLE-0440
            PICTURE IS  X(20)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS JOBTITLE
            .
        02 DESCRIPTION-0440
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS JOBDESCR
            .
            03 DESCRIPTION-LINE-0440
                PICTURE IS  X(60)
```

```
                USAGE IS DISPLAY
                OCCURS 2 TIMES
                SYNONYM NAME FOR ASSEMBLER IS JOBDSCLN
                .
            02 REQUIREMENTS-0440
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS JOBRQMNT
                .
                03 REQUIREMENT-LINE-0440
                    PICTURE IS  X(60)
                    USAGE IS DISPLAY
                    OCCURS 2 TIMES
                    SYNONYM NAME FOR ASSEMBLER IS JOBREQLN
                    .
            02 MINIMUM-SALARY-0440
                PICTURE IS  S9(6)V99
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS JOBMNSAL
                .
            02 MAXIMUM-SALARY-0440
                PICTURE IS  S9(6)V99
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS JOBMXSAL
                .
            02 SALARY-GRADES-0440
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                OCCURS 4 TIMES
                SYNONYM NAME FOR ASSEMBLER IS JOBSALGR
                .
            02 NUMBER-OF-POSITIONS-0440
                PICTURE IS  9(3)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS JOBNMPOS
                .
            02 NUMBER-OPEN-0440
                PICTURE IS  9(3)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS JOBNMOPN
                .
            02 FILLER
                PICTURE IS  XX
                USAGE IS DISPLAY
                .
            ADD
            RECORD NAME IS NON-HOSP-CLAIM
*+              USES STRUCTURE OF RECORD NON-HOSP-CLAIM VERSION 100
                RECORD ID IS 445
                LOCATION MODE IS VIA COVERAGE-CLAIMS SET
```

```
    MINIMUM ROOT LENGTH IS 248 CHARACTERS
    MINIMUM FRAGMENT LENGTH IS 1008 CHARACTERS
    RECORD SYNONYM NAME FOR ASSEMBLER IS NONHSPCL
    .
02 CLAIM-DATE-0445
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHCLMDT
    .
    03 CLAIM-YEAR-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHCLMYR
        .
    03 CLAIM-MONTH-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHCLMMO
        .
    03 CLAIM-DAY-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHCLMDAY
        .
02 PATIENT-NAME-0445
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHPTNAME
    .
    03 PATIENT-FIRST-NAME-0445
        PICTURE IS  X(10)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPTFNAM
        .
    03 PATIENT-LAST-NAME-0445
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPTLNAM
        .
02 PATIENT-BIRTH-DATE-0445
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHPTBDAT
    .
    03 PATIENT-BIRTH-YEAR-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPTBYR
        .
    03 PATIENT-BIRTH-MONTH-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
```

```
        SYNONYM NAME FOR ASSEMBLER IS NHPTBMO
        .
    03 PATIENT-BIRTH-DAY-0445
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPTBDA
        .
02 PATIENT-SEX-0445
    PICTURE IS  X
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHPTSEX
    .
02 RELATION-TO-EMPLOYEE-0445
    PICTURE IS  X(10)
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHRELEMP
    .
02 PHYSICIAN-NAME-0445
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHPHYNAM
    .
    03 PHYSICIAN-FIRST-NAME-0445
        PICTURE IS  X(10)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYFNM
        .
    03 PHYSICIAN-LAST-NAME-0445
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYLNM
        .
02 PHYSICIAN-ADDRESS-0445
    USAGE IS DISPLAY
    SYNONYM NAME FOR ASSEMBLER IS NHPHYADD
    .
    03 PHYSICIAN-STREET-0445
        PICTURE IS  X(20)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYSTR
        .
    03 PHYSICIAN-CITY-0445
        PICTURE IS  X(15)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYCTY
        .
    03 PHYSICIAN-STATE-0445
        PICTURE IS  X(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYSTA
```

```
            .
        03 PHYSICIAN-ZIP-0445
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS NHPHYZIP
            .
            04 PHYSICIAN-ZIP-FIRST-FIVE-0445
                PICTURE IS  X(5)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS NHPHYZ5
                .
            04 PHYSICIAN-ZIP-LAST-FOUR-0445
                PICTURE IS  X(4)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS NHPHYZ4
                .
    02 PHYSICIAN-ID-0445
        PICTURE IS  9(6)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS NHPHYSID
        .
    02 DIAGNOSIS-0445
        PICTURE IS  X(60)
        USAGE IS DISPLAY
        OCCURS 2 TIMES
        SYNONYM NAME FOR ASSEMBLER IS NHDIAGN
        .
    02 NUMBER-OF-PROCEDURES-0445
        PICTURE IS  9(2)
        USAGE IS COMP
        SYNONYM NAME FOR ASSEMBLER IS NHNOPROC
        .
    02 FILLER
        PICTURE IS  X
        USAGE IS DISPLAY
        .
    02 PHYSICIAN-CHARGES-0445
        USAGE IS DISPLAY
        OCCURS 0 TO 10 TIMES DEPENDING ON NUMBER-OF-PROCEDURES-0445
        SYNONYM NAME FOR ASSEMBLER IS NHPHYCHG
        .
        03 SERVICE-DATE-0445
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS NHSERVDT
            .
            04 SERVICE-YEAR-0445
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS NHSERVYR
                .
```

```
                04 SERVICE-MONTH-0445
                    PICTURE IS  9(2)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS NHSERVMO
                    .
                04 SERVICE-DAY-0445
                    PICTURE IS  9(2)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS NHSERVDA
                    .
            03 PROCEDURE-CODE-0445
                PICTURE IS  9(4)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS NHPROCCD
                .
            03 DESCRIPTION-OF-SERVICE-0445
                PICTURE IS  X(60)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS NHDESCSV
                .
            03 FEE-0445
                PICTURE IS  S9(7)V99
                USAGE IS COMP-3
                SYNONYM NAME FOR ASSEMBLER IS NHFEE
                .
            03 FILLER
                PICTURE IS  X
                USAGE IS DISPLAY
                .
        ADD
        RECORD NAME IS OFFICE
*+          USES STRUCTURE OF RECORD OFFICE VERSION 100
            RECORD ID IS 450
            LOCATION MODE IS CALC USING ( OFFICE-CODE-0450 )
                  DUPLICATES ARE NOT ALLOWED
            RECORD SYNONYM NAME FOR ASSEMBLER IS OFFIC
            .
        02 OFFICE-CODE-0450
            PICTURE IS  X(3)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS OFFCODE
            .
        02 OFFICE-ADDRESS-0450
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS OFFADDR
            .
            03 OFFICE-STREET-0450
                PICTURE IS  X(20)
                USAGE IS DISPLAY
```

```
                    SYNONYM NAME FOR ASSEMBLER IS OFFSTRT
                    .
                03 OFFICE-CITY-0450
                    PICTURE IS  X(15)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS OFFCITY
                    .
                03 OFFICE-STATE-0450
                    PICTURE IS  X(2)
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS OFFSTATE
                    .
                03 OFFICE-ZIP-0450
                    USAGE IS DISPLAY
                    SYNONYM NAME FOR ASSEMBLER IS OFFZIP
                    .
                    04 OFFICE-ZIP-FIRST-FIVE-0450
                        PICTURE IS  X(5)
                        USAGE IS DISPLAY
                        SYNONYM NAME FOR ASSEMBLER IS OFFZIPF5
                        .
                    04 OFFICE-ZIP-LAST-FOUR-0450
                        PICTURE IS  X(4)
                        USAGE IS DISPLAY
                        SYNONYM NAME FOR ASSEMBLER IS OFFZIPL4
                        .
            02 OFFICE-PHONE-0450
                PICTURE IS  9(7)
                USAGE IS DISPLAY
                OCCURS 3 TIMES
                SYNONYM NAME FOR ASSEMBLER IS OFFPHONE
                .
            02 OFFICE-AREA-CODE-0450
                PICTURE IS  X(3)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS OFFAREA
                .
            02 SPEED-DIAL-0450
                PICTURE IS  X(3)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS OFFSPEED
                .
            ADD
            RECORD NAME IS SKILL
*+              USES STRUCTURE OF RECORD SKILL VERSION 100
                RECORD ID IS 455
                LOCATION MODE IS CALC USING ( SKILL-ID-0455 ) DUPLICATES ARE
                      NOT ALLOWED
                RECORD SYNONYM NAME FOR ASSEMBLER IS SKILLA
```

```
            .
        02 SKILL-ID-0455
            PICTURE IS  9(4)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS SKILID
            .
        02 SKILL-NAME-0455
            PICTURE IS  X(12)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS SKILNAME
            .
        02 SKILL-DESCRIPTION-0455
            PICTURE IS  X(60)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS SKILDESC
            .
        ADD
        RECORD NAME IS STRUCTURE
*+          USES STRUCTURE OF RECORD STRUCTURE VERSION 100
            RECORD ID IS 460
            LOCATION MODE IS VIA MANAGES SET
            RECORD SYNONYM NAME FOR ASSEMBLER IS STRUCTUR
            .
        02 STRUCTURE-CODE-0460
            PICTURE IS  X(2)
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS STRCODE
            .
                        88 ADMIN-0460
                        USAGE IS CONDITION-NAME
                        VALUE IS 'A'
                        .
                        88 PROJECT-0460
                        USAGE IS CONDITION-NAME
                        VALUE IS 'P1' THRU 'P9'
                        SYNONYM NAME FOR FORTRAN IS PROJCT

                        .
        02 STRUCTURE-DATE-0460
            USAGE IS DISPLAY
            SYNONYM NAME FOR ASSEMBLER IS STRDATE
            .
            03 STRUCTURE-YEAR-0460
                PICTURE IS  9(2)
                USAGE IS DISPLAY
                SYNONYM NAME FOR ASSEMBLER IS STRYEAR
                .
            03 STRUCTURE-MONTH-0460
                PICTURE IS  9(2)
                USAGE IS DISPLAY
```

```
        SYNONYM NAME FOR ASSEMBLER IS STRMONTH
        .
    03 STRUCTURE-DAY-0460
        PICTURE IS  9(2)
        USAGE IS DISPLAY
        SYNONYM NAME FOR ASSEMBLER IS STRDAY
        .
ADD
SET NAME IS COVERAGE-CLAIMS
    ORDER IS LAST
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS COVERAGE
        NEXT DBKEY POSITION IS 4
        PRIOR DBKEY POSITION IS 5
    MEMBER IS HOSPITAL-CLAIM
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        MANDATORY AUTOMATIC
    MEMBER IS NON-HOSP-CLAIM
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        MANDATORY AUTOMATIC
    MEMBER IS DENTAL-CLAIM
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        MANDATORY AUTOMATIC

    .
ADD
SET NAME IS DEPT-EMPLOYEE
    ORDER IS SORTED
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS DEPARTMENT
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
    MEMBER IS EMPLOYEE
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 3
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( EMP-LAST-NAME-0415 EMP-FIRST-NAME-0415
              )
            DUPLICATES ARE LAST
    .
ADD
SET NAME IS EMP-COVERAGE
    ORDER IS FIRST
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
```

```
        NEXT DBKEY POSITION IS 7
        PRIOR DBKEY POSITION IS 8
    MEMBER IS COVERAGE
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 3
        MANDATORY AUTOMATIC
    .
ADD
SET NAME IS EMP-EMPOSITION
    ORDER IS FIRST
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
        NEXT DBKEY POSITION IS 9
        PRIOR DBKEY POSITION IS 10
    MEMBER IS EMPOSITION
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 3
        MANDATORY AUTOMATIC

    .
ADD
SET NAME IS EMP-EXPERTISE
    ORDER IS SORTED
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
        NEXT DBKEY POSITION IS 11
        PRIOR DBKEY POSITION IS 12
    MEMBER IS EXPERTISE
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 3
        MANDATORY AUTOMATIC
        DESCENDING KEY IS ( SKILL-LEVEL-0425 )
            DUPLICATES ARE FIRST
    .
ADD
SET NAME IS EMP-NAME-NDX
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 40 KEYS
    OWNER IS SYSTEM
    MEMBER IS EMPLOYEE
        INDEX DBKEY POSITION IS 4
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( EMP-LAST-NAME-0415 EMP-FIRST-NAME-0415
              ) COMPRESSED
```

```
            DUPLICATES ARE LAST
    .
ADD
SET NAME IS JOB-EMPOSITION
    ORDER IS NEXT
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS JOB
        NEXT DBKEY POSITION IS 2
        PRIOR DBKEY POSITION IS 3
    MEMBER IS EMPOSITION
        NEXT DBKEY POSITION IS 4
        PRIOR DBKEY POSITION IS 5
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 6
        OPTIONAL MANUAL
    .
ADD
SET NAME IS JOB-TITLE-NDX
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS SYSTEM
    MEMBER IS JOB
        INDEX DBKEY POSITION IS 1
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( TITLE-0440 ) UNCOMPRESSED
            DUPLICATES ARE NOT ALLOWED

    .
ADD
SET NAME IS MANAGES
    ORDER IS NEXT
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
        NEXT DBKEY POSITION IS 13
        PRIOR DBKEY POSITION IS 14
    MEMBER IS STRUCTURE
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 3
        MANDATORY AUTOMATIC
    .
ADD
SET NAME IS OFFICE-EMPLOYEE
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS OFFICE
        NEXT DBKEY POSITION IS 1
        PRIOR DBKEY POSITION IS 2
    MEMBER IS EMPLOYEE
```

```
        INDEX DBKEY POSITION IS 5
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 6
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( EMP-LAST-NAME-0415 EMP-FIRST-NAME-0415
              ) COMPRESSED
            DUPLICATES ARE LAST
    .
ADD
SET NAME IS REPORTS-TO
    ORDER IS NEXT
    MODE IS CHAIN LINKED TO PRIOR
    OWNER IS EMPLOYEE
        NEXT DBKEY POSITION IS 15
        PRIOR DBKEY POSITION IS 16
    MEMBER IS STRUCTURE
        NEXT DBKEY POSITION IS 4
        PRIOR DBKEY POSITION IS 5
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 6
        OPTIONAL MANUAL

    .
ADD
SET NAME IS SKILL-EXPERTISE
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS SKILL
        NEXT DBKEY POSITION IS 2
        PRIOR DBKEY POSITION IS 3
    MEMBER IS EXPERTISE
        INDEX DBKEY POSITION IS 4
        LINKED TO OWNER
            OWNER DBKEY POSITION IS 5
        MANDATORY AUTOMATIC
        DESCENDING KEY IS ( SKILL-LEVEL-0425 ) UNCOMPRESSED
            DUPLICATES ARE FIRST
    .
ADD
SET NAME IS SKILL-NAME-NDX
    ORDER IS SORTED
    MODE IS INDEX BLOCK CONTAINS 30 KEYS
    OWNER IS SYSTEM
    MEMBER IS SKILL
        INDEX DBKEY POSITION IS 1
        OPTIONAL AUTOMATIC
        ASCENDING KEY IS ( SKILL-NAME-0455 ) UNCOMPRESSED
            DUPLICATES ARE NOT ALLOWED
    .
```

Index

F

I

K

L

M

N

O

P

R

S

U

V

W