Publication Number
SH20-6603-01

IMS Application Development Facility II
Version 2 Release 2

DATABASE 2 Application Specification Guide

IBM

IMS Application Development Facility II Version 2 Release 2

DATABASE 2 Application Specification Guide

Publication Number
SH20-6603-01

File Number
S/370/4300-32

Program Number
5665-348

Second Edition (June 1986)

This edition applies to Version 2, Release 2 of the program product IMS Application Development Facility II (5665-348), and to all subsequent releases and modifications unless otherwise indicated in new editions or Technical Newsletters.

Information in this publication is subject to change. Changes will be published in new editions or technical newsletters. Before using this publication, consult either your _IBM System/370 and 4300 Processors Bibliography_ (GC20-0001) or _IBM System/370 and 4300 Processors Bibliography of Industry Systems Systems and Application Programs_ (GC20-0370) to learn which editions and technical newsletters are current and applicable.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available outside the United States. Any reference to an IBM program product in this document is not intended to state or imply that only IBM's program products may be used. Any functionally equivalent program products may be used instead.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments has been provided at the back of this publication. If this form has been removed, address comments to:

IBM Corporation
Information Processing
Department 6DD
220 Las Colinas Blvd.
Irving, Texas 75039-5513

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

(c) Copyright International Business Machines Corporation 1985, 1986

PREFACE

This publication is a guide to developing IMSADF II applications that access IBM DATABASE 2 (DB2) data bases.

This publication consists of five chapters and six appendixes.

- **Chapter 1, "An Overview of Application Design"** gives an overview of application design. It describes the DB2 Relational Data Base support, the IMSADF II Application Design and Performance considerations.
- **Chapter 2, "User Specification"** describes how to define the environment and functions required by an IMSADF II transactions accessing DB2 data.
- **Chapter 3, "RGLGEN Utility"** describes how to extract Rules Generator source from the DB2 catalog.
- **Chapter 4, "Execution Processing"** describes the IMSADF II functions that are available to IMSADF II DB2 transactions at execution.
- **Chapter 5, "Installation"** describes one optional IMSADF II installation step, and the IMSADF II DB2 sample problem.
- The appendixes include:

  Appendix A, "Sample Rules Generator Source and Output"

  Appendix B, "Specification Example"

  Appendix C, "Sample Problem"

  Appendix D, "BTS in an IMS/VS - DB2 Environment"

  Appendix E, "IMSADF II Trace Facility"

  Appendix F, "RGLGEN Utility Link-edit Plan"

RELATED PUBLICATIONS

IMSADF II PUBLICATIONS

- IMS Application Development Facility II Version 2 Release 2 General Information, GH20-6591.
- IMS Application Development Facility II Version 2 Release 2 User Reference, SH20-6592.
- IMS Application Development Facility II Version 2 Release 2 Installation Guide, SH20-6593.
- IMS Application Development Facility II Version 2 Release 2 Application Development Reference, SH20-6594.
- IMS Application Development Facility II Version 2 Release 2 Application Development Guide, SH20-6595.
- IMS Application Development Facility II Version 2 Release 2 Rules Documentation User's Guide, SH20-6596.
- IMS Application Development Facility II Version 2 Release 2 Data Dictionary Extension User's Guide, SH20-6597.
- IMS Application Development Facility II Version 2 Release 2 Master Index, SH20-6599.
- IMS Application Development Facility II Version 2 Release 2 Introduction to Using the Interactive ADF, SH20-6601.
- IMS Application Development Facility II Version 2 Release 2 Interactive ADF Administration Guide, SH20-6602.
- IMS Application Development Facility II Version 2 Release 2 DATABASE 2 Application Specification Guide, SH20-6603.
- IMS Application Development Facility II Version 2 Release 2 Diagnosis Guide, LY20-6401.

OTHER PUBLICATIONS

- IBM DATABASE 2 Reference, SC26-4078
- IBM DATABASE 2 Application Programming Guide for IMS/VS Version 1 Users, SC26-4079
- IBM DATABASE 2 Application Programming Guide for CICS/OS/VS Users, SC26-4080
- IBM DATABASE 2 Application Programming Guide for TSO Users, SC26-4081
- IBM DATABASE 2 System Planning and Administration Guide, SC26-4085
- IBM DATABASE 2 Messages and Codes, SC26-4113
- IMS/VS BTS Program Reference and Operations Manual, SH20-5523

CONTENTS

FIGURES

CHAPTER 1. AN OVERVIEW OF APPLICATION DESIGN

DB2 RELATIONAL DATA BASE SUPPORT

DESCRIPTION

The IMS Application Development Facility II supports IBM's relational data base management system DATABASE 2 (DB2). A DB2 table/view is defined to the IMSADF II Rules Generator by TABLE and COLUMN statements (or alternatively, by SEGMENT and FIELD statements.) These statements can be user specified or can be produced using the IMSADF II RGLGEN Utility. The Rules Generator builds a Table Layout Rule to define the table/view layout and a Table Handler Rule to define SQL access. The Table Layout Rule defines the table layout to the execution time transaction drivers. The Table Handler Rule is a generated Assembler program that issues static SQL statements for the IMSADF II transaction. The SQL WHERE clauses include standard generated predicates (that is, columns marked as KEY=YES, connected with the = relational operator) and user-supplied predicates. Host variables, representing column I/O areas for SELECT, UPDATE, DELETE, or INSERT calls are defined as DSECTs on the IMSADF II SPA workarea. The Table Handler program is created by the Rules Generator, which dynamically invokes the DB2 pre-compiler, assembler and linkage editor.

Standard IMSADF II functions process one row from a DB2 table per transaction iteration. Each DB2 row processed should be defined with unique column values to satisfy IMSADF II key requirements. Multiple DB2 tables can be processed simultaneously. Processing multiple rows from a DB2 table in a single iteration of an IMSADF II transaction requires additional audit or special processing routine logic.

At execution, an IMSADF II transaction driver - conversational, nonconversational, or batch (BMP) - issues SQL calls to access and update a Table/View, as appropriate. Applications can be defined, with additional SQL WHERE clauses, that are still controlled by the IMSADF II transaction driver, but are triggered with application logic through an Audit operation or with a SQLHNDLR Call in an Audit Exit or Special Processing Routine. Additionally, an Audit Exit or Special Processing Routine can issue native SQL calls (static or dynamic).

IMSADF II APPLICATION DESIGN CONSIDERATIONS

TABLE/VIEW

- A Table/View is defined to the IMSADF II Rules Generator via TABLE and COLUMN statements. IMSADF II does not distinguish between a Table or a View. This is analogous to DL/I support, in that IMSADF II does not distinguish between logical and physical data bases.

  When a SQL statement is defined to IMSADF II in a Table Handler Rule, <u>all</u> DB2 Columns defined in the corresponding Table Layout Rule are named and SELECTed for that Table/View. In the UPDATE and INSERT statements, the SET and VALUES clauses include all columns specifying the Rules Generator operands, SQLUPD=YES and SQLISRT=YES.

- The Rules Generator accepts TABLE and COLUMN names of up to 50 characters, including SQL built-in functions and arithmetic expressions. IMSADF II uses these names to communicate with DB2. IMSADF II Table and Column IDs must be associated with every DB2 name defined. This is the same technique used to relate DL/I segment and field names to the IMSADF II. The dual naming convention is required by IMSADF II for inter-rule relationships.

- All DB2 data types are supported by IMSADF II. However, DB2 data types VARCHAR and FLOAT have limited support.

VARCHAR IMSADF II support is limited to a 253-byte varying character string. (DB2 allows 32K.)

A data field TYPE=VARCHAR is modified when:

Entered from a screen or batch input.

IMSADF II recalculates the current data length, based on the last non-blank character.

Modified through audits, or modified from an exit with the MAPPER function.

The length of the source field(s) is used to calculate the current length of a target VARCHAR field. Trailing blanks are significant. IMSADF II does <u>NOT</u> recalculate the current data length based on the last non-blank character.

Exits that modify a field with TYPE=VARCHAR within their own data area must also set the current data length attribute unless the field is mapped to the SPA Table I/O area with the MAPPER function. Recalculation of the current data length does not occur when a changed VARCHAR field is moved to the SPA Table I/O area with the COPYSEG function.

When defining VARCHAR Columns to the Rules Generator, only specify the maximum length of the data area. The Rules Generator reserves two additional bytes for the halfword length attribute.

FLOAT Support is limited to display and data entry. Data conversions of FLOAT to display and display to FLOAT are handled automatically. Data manipulation functions are not provided, except for float to float, float to alphanumeric, and alphanumeric to float. Arithmetic operations in the Auditor and other data conversions are not supported.

Data types FLOAT and VARCHAR are also available to all other IMSADF II segment types: DL/I data base, pseudo, map, and out.

- IMSADF II supports DB2 indicator variables by means of an indicator structure. The purpose of the indicator variable is to allow for the null value. When none of the DB2 Columns are eligible for null processing, the indicator structure is omitted from the Table I/O area. When one or more DB2 Columns are eligible for null processing, (SQLNULL=YES), an indicator structure containing an indicator variable for every Column is defined at the end of the Table I/O area.

- DB2 Tables are not supported in IMSADF II Text Utility transactions.

SQL LANGUAGE USAGE

- For standard access to DB2 tables (that is, no procedural coding), unique column values (or keys) should be specified. The user should CREATE INDEX with the UNIQUE attribute for Tables that are accessed using standard IMSADF II functions, (that is, key selection browse, automatic data base update).

- The standard key selection browse function has an ORDER BY clause associated with the SQL CURSOR SELECT statement. This ensures that the rows displayed on the key selection browse screen are in an ordered sequence. Even though a UNIQUE INDEX is defined for a TABLE, the DB2 BIND process may choose to ignore it and define a search strategy using a TABLE scan technique. This causes the rows to be displayed in an arbitrary order. Specifying an ORDER BY clause may influence DB2 to use the INDEX to avoid the overhead of the sort associated with using an ORDER BY clause.

- When DB2 utilizes the ORDER BY clause, a sort may be performed for the entire set of rows satisfying the WHERE clause at CURSOR OPEN. Terminal I/O causes the cursor to be closed. Therefore, if the terminal operator requests the next screen of key selection, the repositioning process will require the re-execution of the SQL call and another sort of the selected set of rows. Use of the key selection browse function with DB2 tables should be reviewed carefully, due to the potential adverse impact on performance.

- Arithmetic expressions are not allowed in the SET and VALUES clauses of the UPDATE and INSERT calls. These functions can be accomplished by current operations in the Audit Language.

- Static SQL is supported through the generation of a Table Handler Rule. Dynamic SQL is not supported in the Table Handler Rule, though an Audit Exit or Special Processing Routine may issue native SQL calls to PREPARE, DESCRIBE, and EXECUTE a dynamic SQL request and map the data back to the SPA workarea for display.

DB2 SCHEDULING

Since one Application Plan (same name as PSB and mini-driver name) may include many Table Handler Rules with intent to UPDATE, it is recommended that page locking be used. Additionally, the BIND input for the Application Plan should specify cursor stability. This allows more optimal DB2 processing and reduces lock conflicts between applications.

Table locking and access path are determined by DB2 during the BIND process. Even if PAGE locks are specified, the DB2 BIND may choose a TABLE SPACE lock.

One IMS/VS PSB can cluster many IMSADF II transactions. This implies that many IMSADF II transactions can also be defined by one DB2 Application Plan. However, large DB2 Application Plans are not as efficient as smaller ones.

The IBM DATABASE 2 System Planning and Administration Guide contains additional information on the DB2 BIND and Application Plan process and the Concurrency Control Mechanisms (Locking).

DB2 FUNCTIONS SUPPORTED INDIRECTLY

The following DB2 functions are supported indirectly via the DB2 View mechanism:

- Joins of Tables where multiple Table/View name combinations exceed the 50 character IMSADF II Rules Generator SQLNAME operand limit.

- DB2 Column name expressions that exceed the 50 character IMSADF II Rules Generator SQLNAME operand limit.

  - SQL built-in functions (such as DISTINCT, COUNT)
  - Arithmetic expressions
  - Multiple DB2 Column names

DB2 SERVICES NOT SUPPORTED USING STANDARD IMSADF II FUNCTIONS

The following DB2 categories of SQL statements are not supported using standard IMSADF II functions. They can be invoked by defining native SQL statements in an Audit Exit or a Special Processing Routine. The last three can also be implemented with IMSADF II-defined USER SQL statements.

- SQL Data Definition Statements

- Dynamic SQL Statements

- SQL Control Statements (that is, LOCK)

- SQL Authorization Statements

- SQL Statements with UNION
- SQL clauses GROUP BY and HAVING
- SQL subselects

STANDARD IMSADF II FUNCTIONS NOT AVAILABLE FOR DB2 TABLES

- DB2 Tables are not supported in Text Utility transactions.
- Standard TWIN processing function (DL/I only)
- 'N' OPTION
- DL/I EXIT

CHAPTER 2. USER SPECIFICATION

Note: Refer to the following appendixes while reading this chapter:

- Appendix A, "Sample Rules Generator Source and Output," for Rules Generator input and output
- Appendix B, "Specification Example," for a complete specification example
- Appendix C, "Sample Problem," for Screen flow for two sample DB2 transactions

RULES GENERATOR

Four Rules Generator statements are used to describe a DB2 application to IMSADF II:

SYSTEM	Specifies setup information
TABLE	Defines Table/View information, (synonym - SEGMENT)
COLUMN	Defines Table/View layout, (synonym - FIELD)
GENERATE	Requests the application rule and screen generations

SYSTEM STATEMENT

The SYSTEM statement defines the major application system identification and setup information. The SYSTEM statement must precede any table/view or segment definitions. See the IMS Application Development Facility II Version 2 Release 2 Application Development Reference for a description of the SYSTEM statement and its operands. The only SYSTEM operand not applicable to applications processing DB2 Tables/Views is PCBNO, which is a DL/I parameter.

TABLE (SEGMENT) STATEMENT

The TABLE (synonym SEGMENT) statement describes a DB2 Table/View. See the IMS Application Development Facility II Version 2 Release 2 Application Development Reference for a description of the SEGMENT statement and its operands.

The following TABLE statement operands are used exclusively to define DB2 Tables to IMSADF II.

TYPE=TBL

The value TBL indicates a DB2 Table/View is being defined. The TBL value indicates to the IMSADF II transaction drivers that data base I/O is performed through SQL functions in the associated Table Handler Rule. Those SQL functions include SELECT, UPDATE, INSERT, DELETE, or user supplied functions.

If TABLE is used to identify the TABLE statement (rather than SEGMENT), the default value for TYPE is TBL.

```
SQLDIST=NO
        YES
```

SQLDIST=YES indicates that the SQL DISTINCT keyword is used on all SELECT SQL statements defined in the Table Handler Rule. Both standard and user-specified SQL SELECT statements are defined with the DISTINCT keyword.

```
SQLIND=NO
       YES
```

SQLIND=YES indicates that Columns in the Table/View may have indicator variables. Indicator variables are available to receive the SQL indication that a Column is NULL or truncated, or to set a column NULL in UPDATE or INSERT SQL statements.

Indicator variables, if present, are used by screen handling for NULL display, and can be tested or set (NULL) in audits.

If no Columns are defined with SQLNULL=YES, indicator variables are not defined for the Table (no additional Table I/O area reserved).

When IMSADF II generates indicator variables, (at least 1 Column defined with SQLNULL=YES), one is assigned to each Column in the Table even though only those Columns specifying SQLNULL=YES will be eligible for indicator variable processing (SQL statements and audits).

Each indicator variable is a binary halfword. The Table I/O area length reflects the lengths of all Columns and all indicator variables.

The setting of SQLIND on the TABLE statement cannot be changed on a TABLE OVERRIDE statement.

Format of the Table I/O area as defined by the Table Layout Rule follows:

```
 Table Definition Information
------------------------------
    Data Elements (Columns)
       COL1 . . . . . . . .
       COL2 . . . . . . . .
       COLn . . . . . . . .
------------------------------
    Indicator Variables
       IND1 . . . . . . . .
       IND2 . . . . . . . .
       INDn . . . . . . . .
```

SQLISRT=YES
NO

SQLISRT=YES indicates that all Columns in the Table are eligible for insert. SQLISRT=NO indicates that no Columns are eligible for insert.

Specifying SQLISRT=YES

- Causes SQLISRT=YES to be specified for each Column entry.
- Includes the Column in the VALUES clause of the SQL INSERT statement in the Table Handler Rule.

Examples of columns that cannot be inserted are those derived from built-in functions or arithmetic expressions. Other columns not eligible for insert are application dependent.

Specification of SQLISRT on the COLUMN statement overrides the SQLISRT operand on the TABLE statement.

The setting of SQLISRT on the TABLE statement cannot be changed on a TABLE OVERRIDE statement.

SQLNAME='1 to 50 characters'

The SQLNAME operand is required for all DB2 Tables/Views defined to IMSADF II. The SQLNAME operand on a Table statement can contain DB2 Table names in four forms:

Unqualified name Table name, a long identifier, one to 18 characters.

Qualified name 'Authorization ID.Table name'

Authorization ID, a short identifier, one to eight characters.

Table name, a long identifier, one to 18 characters.

The Unqualified and Qualified Table names are valid for all SQL statements that can be included in a Table Handler Rule.

Join 'table1, table2, table3'

Defining multiple names is used to define a view that is a join of several DB2 Tables. When this type of name is specified the normal DB2 restrictions for views are enforced. Namely, the View is not eligible for Update, Insert, or Delete.

Correlation-name 'dsn8.templ x'

If a correlation-name is specified, it is defined as part of the Table name on all SQL functions except INSERT. The correlation-name is ignored by IMSADF II and SQL, except where referenced.

When specifying a Qualified Table name, a period separates the authorization ID and the Table name, and the entire name must be enclosed in quotes.

The SQLNAME operand is required to build the Table Handler Rule and to define SQL access to DB2.

The SQLNAME operand must be specified on the initial TABLE statement. It cannot be overridden on a Table/Segment OVERRIDE statement.

```
SQLUPD=YES
       NO
```

SQLUPD=YES indicates that all Columns in the Table are eligible for update. SQLUPD=NO indicates there are no Columns eligible for update.

Specifying SQLUPD=YES:

- Causes SQLUPD=YES to be specified for each Column entry, (except KEY=YES Columns).
- Includes the Column in the SET clause of the SQL UPDATE statement in the Table Handler Rule.

Examples of columns that cannot be updated are those derived from built-in functions or arithmetic expressions. Other columns not eligible for update are application-dependent.

Specification of SQLUPD on the COLUMN statement overrides the SQLUPD operand on the TABLE statement.

The setting of SQLUPD on the TABLE statement cannot be changed on a TABLE OVERRIDE statement.

```
SKSEGS=0
       n
```

SKSEGS=0, the default, indicates that Secondary Key Selection browsing is not available for this DB2 Table/View.

SKSEGS=n specifies the number of row occurrences to be displayed. It indicates that this DB2 Table/View is eligible to be processed by Secondary Key Selection Browse functions under the following conditions:

- The Table is defined in the DBPATH operand of the conversational Input Transaction Rule GENERATE (CVALL).

- A generic search is requested ('>', '<', '%', '_') by the terminal operator, or an incorrect key, or no key is entered by the terminal operator.
- A user-defined Secondary Key Selection browse function is requested through a Primary Key audit.

DBPATH defines the Tables and Segments to be automatically retrieved by the IMSADF II transaction drivers. The keys to retrieve these Tables and/or Segments are supplied by the terminal operator or through audits.

COLUMN (FIELD) STATEMENT

The COLUMN (synonym FIELD) statement describes a DB2 Table/View column. IMSADF II supports a maximum of 512 columns per table/view. See the IMS Application Development Facility II Version 2 Release 2 Application Development Reference for a description of the FIELD statement and its operands.

The following COLUMN statement operands are used exclusively to define DB2 Tables:

KEY=NO
YES

The KEY=YES operand specifies Columns to be used in the SQL WHERE clause of the Table Handler Rule. The relational operator between the DB2 Column name and the IMSADF II host variable is '=' for the standard SELECT, DELETE, and UPDATE statements. The standard Secondary Key Selection browse SELECT uses '>=' or '<=' or the DB2 LIKE predicate for its relational operator.

Standard processing functions assume that the key values uniquely identify a row. It is recommended that KEY=YES columns be defined in a Table UNIQUE INDEX for better DB2 performance in processing the SQL WHERE clause. It is NOT recommended for tables with multiple unique indexes to be processed under IMSADF II. If multiple unique indexes are defined, it would be possible for an end user to specify values which are a unique combination of all the KEY columns, but are not unique within each index. This would result in a DB2 error condition (SQLCODE -803).

If more than one column is defined as KEY=YES and a DB2 composite INDEX exists, the order of the columns specified to the Rules Generator should match the order specified in the INDEX.

Every Table must have at least one Column specifying KEY=YES to uniquely identify a row. This requirement exists for all data base segments (DL/I and DB2) that are defined to IMSADF II. If the Table is accessed through one of the standard SQL calls, the Table's KEY=YES Columns are included in the WHERE clause.

Notes:

1. DB2 Columns defined as KEY Columns are not eligible for NULL processing.
2. TYPE=FLOAT and TYPE=BIT Columns/Fields cannot be defined as keys.

SQLISRT=YES
NO

SQLISRT=NO identifies a Column that is not eligible for insert. This Column is not included in the VALUES clause of the SQL INSERT statement in the Table Handler Rule.

Arithmetic expressions are not allowed in the SET and VALUES clauses of the UPDATE and INSERT calls. These functions can be accomplished prior to issuing the SQL call with currently available Audit operations.

SQLNAME='1 to 50 characters'

The SQLNAME operand identifies the DB2 Column name as it is specified in the DB2 catalog. An unqualified SQLNAME is one to 18 characters long. If qualified, a period separates the qualifier and the unqualified Column name.

Quotes are required for all forms of SQLNAME, except the unqualified Column name.

Every Column in a Table speci-ication must have a SQLNAME operand as DB2 addressability is to the column level.

The SQLNAME operand must be specified on each COLUMN statement that represents a DB2 Column. It cannot be changed by a COLUMN MERGE, or OVERRIDE statement.

The SQLNAME operand may also identify a column name with a built-in function or arithmetic expression. The Rules Generator uses the value between the single quotes when building the column list in the SQL SELECT statement.

If a DB2 built-in function or arithmetic expression is used the Column should be marked as SQLUPD=NO and SQLISRT=NO.

Every DB2 Table (TYPE=TBL) must have at least one Column defined with a SQLNAME operand.

SQLNULL=<u>NO</u>
YES

Identifies those Columns that can be NULL. NO indicates that this Column is defined to DB2 as NOT NULL. If the value YES is specified, the column can be NULL. SQLNULL=YES implies SQLIND=YES at the Table level has been specified.

A Column can be marked NULL in an UPDATE or INSERT function through the setting of the indicator variable. The indicator variable is set through a screen input convention or through Audit operations.

DB2 Columns defined to IMSADF II as KEY Columns are not eligible for NULL processing.

SQLORD=<u>ASC</u>
DESC

SQLORD indicates the order of column values for those columns marked KEY=YES. ASC means ascending values and DESC means descending values. SQLORD is used to define the following Secondary Key Selection browse functions:

- Define the WHERE clause relational operator, '>=' or '<='.
- Define the ORDER BY clause, ASC or DESC.
- Initialize keys, HI or LO values.

SQLUPD=<u>YES</u>
NO

SQLUPD=NO identifies a Column that is not eligible for update. This Column is not included in the SET clause of the SQL UPDATE statement in the Table Handler Rule.

For example, if a View Column is a virtual column (that is, derived from a built-in function), it cannot be updated in the SQL UPDATE call.

Default is YES, except for KEY=YES columns, which default to NO and cannot be changed to YES. KEY=YES columns are never eligible for update.

Arithmetic expressions are not allowed in the SET and VALUES clauses of the UPDATE and INSERT calls. These functions can be accomplished prior to issuing the SQL call with currently available Audit operations.

TYPE=ALPHA
ALPHANUM
BIN
BIT
DATE
DBCS
DEC
FLOAT
HEX
MIXED
NUM
PD
VARCHAR

The TYPE identifies the data format for the Column. The following table shows DB2 data types and the corresponding IMSADF II data types to which the data is converted.

DB2 Data Types	IMSADF II Data Types
INTEGER -- 32 bits	BINARY---- 4 bytes
SMALLINT - 15 bits	BINARY---- 2 bytes
DECIMAL -- precision - 15 digits	PDEC ----- 8 bytes - 15 digits
-- scale - 15 digits	-- scale - 13 digits
CHAR -----254 characters	ALPHANUM--255 bytes
VARCHAR --32k characters	VARCHAR --253 bytes
FLOAT ---- 64 bits	FLOAT ---- 8 bytes

Figure 2-1. DATA TYPES

Because IMSADF II uses the LENGTH operand to determine packed decimal precision, the DB2 column definition for decimal fields should be defined with a precision that will match the length defined in the IMSADF II rules.

For example, a DB2 column precision of (4,2) will not match an IMSADF II length of 3 and will have to be specified as (5,2).

Data conversion for field type FLOAT is supported only for display to FLOAT and FLOAT to display. Audit move operations support FLOAT to FLOAT, FLOAT to Alphanumeric, and Alphanumeric to FLOAT. Other audit move and arithmetic operations are not allowed on Columns/Fields defined as TYPE=FLOAT. FLOAT represents a long floating point number with approximate range of 5.4E-79 to 7.2E+75.

Data type VARCHAR has the following format:

2 bytes length	data area (1 to 253 characters)

LENGTH=n

LENGTH specifies in bytes the data storage occupied by this column.

An exception is TYPE=VARCHAR, where LENGTH is the maximum length of the data area and does not include the halfword length attribute. This halfword precedes the data area, dynamically specifying the current data length.

When defining VARCHAR Columns to the Rules Generator only specify the maximum length of the data area. The Rules Generator reserves two additional bytes for the halfword length attribute.

When a data field TYPE=VARCHAR is modified:

Entered from a screen or batch BMP input - IMSADF II recalculates the current data length, based on the last non-blank character.

Modified through audit operations, or modified from an Audit Exit or Special Processing Routine with the MAPPER function - the length of the source field(s) is used to calculate the current length of a target VARCHAR field. Trailing blanks are significant. IMSADF II does NOT recalculate the current data length, based on the last non-blank character.

Audit Exits and Special Processing Routines that modify a TYPE=VARCHAR field in their own data area must also set the current data length attribute unless the field is mapped to the SPA Table I/O area with the MAPPER function.

Recalculation of the current data length does not occur when a changed VARCHAR field is moved to the SPA Table I/O area with the COPYSEG function.

POSITION=n

DB2 Column definitions should **not** include the POSITION/START operand. If the RGLGEN Utility is used to extract definitions from the DB2 catalog the POSITION operand is not included. The Rules Generator automatically calculates position in the Table I/O area based on offset and length of preceding Columns.

When adding non-DB2 Column redefinitions to the DB2 COLUMN statements, the Rules Generator operands REDEFINE and OFFSET should be used to define the correct layout.

GENERATE STATEMENT

The GENERATE statement is used to request generation of

- static rules
- transaction drivers
- screen source

See the IMS Application Development Facility II Version 2 Release 2 Application Development Reference for a description of the GENERATE statement and its operands. The GENERATE operands that follow are used exclusively to generate rules for processing DB2 Tables/Views.

Table Layout Rule

OPTIONS=TABL
SEGL

Format of the GENERATE statement is

GENERATE OPTIONS=TABL,TABLES=(id,id,...id)

A Table Layout Rule is required for each Table/View to be processed by IMSADF II. The GENERATE option TABL/SEGL builds the Table Layout Rule control block describing the data type, length, offset, etc. of each Column in the DB2 Table/View. This information is utilized at execution time by the transaction drivers to interpret the data returned and updated through DB2 SQL.

DB2 addresses data at the Column level. Therefore, the Table Layout Rule must match the Assembler DSECTs generated in the Table Handler Rule.

The Rules Generator doesn't do any boundary alignment for DB2 SMALLINT (halfword), INTEGER (fullword), DECIMAL (doubleword), FLOAT (doubleword) data types. The generated Assembler DSECT in the Table Handler Rule is assembled using the Assembler (NOALIGN) parameter to ensure that it is consistent with the Table Layout Rule definition.

TABLES=(id,id,...id)

TABLES specifies the two-character ID (or IDs) identifying the DB2 Table/View requiring a Table Layout Rule generation.

Table Handler Rule

OPTIONS=TABH

The format of the GENERATE statement is:

```
GENERATE  OPTIONS=TABH,TABLES=(id,id,...id),SQLCALL=,SQLUSER=
* 1-8     10-17    19-71
label    sqlfunc  WHERE clause
*              16 to 71 continuation
&SQLENDS
```

A Table Handler Rule is required for each Table/View ID to be processed by the IMSADF II transaction drivers through standard processing functions, Audit operations or SQLHNDLR Calls from an Audit Exit or Special Processing Routine.

The GENERATE option TABH builds an Assembler program, containing standard static SQL calls and USER SQL calls.

The SQL functions supported are:

- CURSOR SELECT
- CURSOR UPDATE
- CURSOR DELETE
- CURSOR SELECT for multiple key selection row browse
- SELECT for single row inquiry
- UPDATE
- INSERT
- DELETE

All standard IMSADF II functions, (data base retrieval, Key Selection browse, and data base update) are performed using <u>only</u> the following SQL calls:

- CURSOR SELECT
- CURSOR UPDATE
- CURSOR DELETE
- INSERT

The Table Handler Rule, an Assembler language program is created by the Rules Generator. This program contains static SQL statements that execute the specified SQL calls. The Rules Generator dynamically invokes the DB2 pre-compiler which replaces the static SQL statements with additional Assembler code. The Rules Generator also dynamically invokes the Assembler, and the linkage editor. The Assembler is invoked with the Assembler (NOALIGN) parameter in effect to ensure that the Assembler DSECTs in the Table Handler Rule are assembled with no boundary alignment. This ensures that the Assembler DSECTs match the Table Layout Rule.

When the Rules Generator dynamically invokes the DB2 pre-compiler, a DBRM member is created and placed in a DBRM library. The DB2 BIND process must be invoked to include this DBRM in a DB2 Application Plan prior to executing an IMSADF II transaction that accesses the DB2 Table.

Note: Refer to "DB2 Specifications" on page 2-38 and "BIND Process" on page 2-38 for additional details.

TABLES=(id,id,...id)

TABLE specifies the two-character ID (or IDs) identifying this DB2 Table/View. Only **one id should** be specified if SQLCALL requirements vary between Tables. Only **one id can** be specified if SQLUSER=YES.

```
SQLCALL=(CSELECT,INSERT,CUPDATE,CDELETE,KSELECT1)
         SELECT,UPDATE,DELETE,KSELECT2,DSQLCALL
         NONE
```

The SQLCALL operand specifies the SQL calls to be included in the Table Handler Rule.

- If SQLCALL is not specified, the five defaults CSELECT, INSERT, CUPDATE, CDELETE, KSELECT1 are taken.
- If SQLCALL is specified, all keywords including the defaults must be specified.
- Defaults are entered individually, or by specifying the DSQLCALL keyword to include all five defaults.
- If SQLCALL=NONE is specified, SQLUSER=YES must be specified and no other keywords are allowed.
- If SQLCALL=KSELECT2 is specified, the key Column in the WHERE clause that is manipulated by the DB2 LIKE relational operator must be the first key Column defined with an alphanumeric data type.

Internal IMSADF II Naming Conventions:

The following internal naming conventions are used by the Rules Generator when building DSECTs and SQL statements in the Table Handler Rule.

They should be used as a reference while reading the following sections defining IMSADF II SQLCALL functions.

The only time application developers need concern themselves with the names defined in the Table Handler Rule is when defining Host Variables in USER SQL statements. Refer to "USER SQL Statements" on page 2-18.

- column   - Column name as known to DB2
- tablename - Table/View name as known to DB2
- :Axxffffn, :Axxffff$, :Axxffff@ - Host Variables where -

```
    :    - constant, Host Variable indicator
    A    - constant, Assembler requirement
    xx   - Table ID
    ffff - Column ID
    n    - Key Column Host Variables
           1,2 Standard key select functions
               KSELECT1 and KSELECT2
           3-9 USER defined key select functions
               KSELECTn
           A-Z USER defined SQL functions
    $    - constant, Column Host Variable
    @    - constant, Column Indicator Variable
```

SQLCALL Functions

DB2 SQL statements that are defined in IMSADF II Table Handler Rules are divided into three categories:

Standard SQL statements that IMSADF II uses to perform its standard data base calls.

The standard IMSADF II SQL functions are:

- CSELECT - CURSOR SELECT for single row inquiry
- CUPDATE - CURSOR UPDATE for single row
- CDELETE - CURSOR DELETE for single row
- INSERT - INSERT of a single row
- KSELECT1 - SELECT for Secondary Key Selection Browse
- KSELECT2 - SELECT for Secondary Key Selection Browse

Standard IMSADF II data base calls are used for:

- Primary Key Selection - CSELECT
- Key Selection browse - KSELECT1, KSELECT2
- Data Base Update - CUPDATE, CDELETE, INSERT
- Data Compare - CUPDATE, CDELETE

Note: IMSADF II standard processing functions process a single row at a time using the DB2 cursor technique.

This technique is consistent with IMSADF II DL/I support, in that IMSADF II only processes the first occurrence of a non-uniquely keyed segment type.

Non-standard SQL statements that IMSADF II <u>only</u> processes when they are:

1. Defined by the application developer using the SQLCALL operand.
2. Invoked using Audit operations or with a SQLHNDLR call in an Audit Exit or Special Processing Routine.

The **non**-standard IMSADF II SQL functions are:

- SELECT - SELECT of a single row
- UPDATE - UPDATE of a single row
- DELETE - DELETE of a single row

USER USER SQL statements include additional Secondary Key Selection browse functions and SQL statements that require more complex WHERE clauses.

USER SQL statements are invoked invoked using Audit operations or with a SQLHNDLR call in an Audit Exit or Special Processing Routine.

Note: Standard IMSADF II SQL functions can also be invoked using Audit operations or with a SQLHNDLR call in an Audit Exit or Special Processing Routine.

The WHERE clauses for the DB2 SQL statements defined in the Table Handler Rule are built by the Rules Generator, except for USER SQL statements in which the WHERE clauses are user defined.

The WHERE clause contains those Column names specifying KEY=YES. If more than one Column indicates KEY=YES, the predicates are connected with the 'AND' relational operator. Each KEY=YES Column is connected to a host variable, defining the Column I/O area, with the = relational operator. Secondary Key Selection browse functions use the >=, <=, or the DB2 LIKE relational operator.

The SELECT statements contained in the SQLCALL functions select all DB2 Columns defined to the Rules Generator for the Table/View.

If SQLDIST=YES is specified on the Rules Generator Table definition, then the DB2 DISTINCT keyword is specified on all SELECT statements in the Table Handler Rule.

CSELECT - CURSOR SELECT for Single Row Inquiry

CURSOR SELECT is the standard IMSADF II SQL function used to retrieve a single row from DBPATH DB2 Tables. This technique is used in order to protect against non-unique rows. The FETCH retrieves only one row into the Table I/O area.

```
FUNCTION      GENERATED SQL STATEMENT

CSELECTO -    DECLARE CSELECT CURSOR FOR
              SELECT column1,column2, ... column(n)
              FROM   tablename
              WHERE  column1=:Axxffff1

              OPEN CSELECT

              FETCH CSELECT
              INTO   :Axxffff$,:Axxffff$:Axxffff@,...

CSELECTC      CLOSE CSELECT
```

Figure 2-2. CURSOR SELECT, Standard IMSADF II Function

The CSELECT statement declares a CURSOR to SELECT a row, OPENs the CURSOR, fetches the row into the IMSADF II work area, and CLOSEs the CURSOR. The DECLARE CURSOR, OPEN, and FETCH are invoked through the CSELECTO function. The CLOSE CURSOR function is invoked with the CSELECTC function.

At execution time the CSELECT function can be used as follows:

- Standard Primary Key Selection if the Table ID is specified in DBPATH to indicate transaction driver retrieval.
- High Level Audit Language 'CSELECT', 'CSELECTO', 'CSELECTC'
- SQLHNDLR Call function 'CSELECT', 'CSELECTO', 'CSELECTC'

INSERT - INSERT of a Single Row

INSERT is the standard IMSADF II SQL function used to INSERT a single row into the DB2 Table or VIEW if allowed. All Columns that specify SQLISRT=YES are be included in the VALUES clause.

If SQLIND=YES is specified, each Column host variable that is defined as eligible for the null value, (SQLNULL=YES) is followed by an indicator variable. If the Column is null the indicator variable is set to -1 according to DB2 protocol.

```
FUNCTION    GENERATED SQL STATEMENT

INSERT   -  INSERT INTO tablename
            (column1,column2,.....columnn)
            VALUES (:Axxffff$:Axxffff@,:Axxffff$:Axxffff@,...)
```

Figure 2-3. INSERT, Standard IMSADF II Function

At execution time the INSERT function can be used as follows:

- Standard data base insert function
- High Level Audit Language 'INSERT'
- SQLHNDLR Call function 'INSERT'

CUPDATE - CURSOR UPDATE for Single Row

CURSOR UPDATE is the standard IMSADF II SQL function used to update a single row in a DB2 Table or View if allowed.

The CURSOR UPDATE function updates a row that has been previously SELECTed and modified. The CURSOR UPDATE function is used to FETCH a new copy of the row and hold the cursor open. If the IMSADF II data compare (DATACOMP) function has been specified, the new copy of the row is compared with an initial copy of the row saved prior to user modifications. Upon verification of the row the UPDATE is issued. If data compare is not specified the UPDATE is issued without data comparison.

Note: Columns defined as KEY=YES are not eligible for update and do not appear in the SET clause.

```
FUNCTION      GENERATED SQL STATEMENT

CUPDATEO -    DECLARE CUPDATE CURSOR FOR
              SELECT column1,column2, ... column(n)
              FROM   tablename
              WHERE  column1=:Axxffff1
              FOR UPDATE OF column2,...column(n)

              OPEN CUPDATE

              FETCH CUPDATE
              INTO   :Axxffff$,:Axxffff$:Axxffff@,...

CUPDATEU      UPDATE tablename
              SET    column2=:Axxffff$:Axxffff@,...,
                     columnn=:Axxffff$:Axxffff@
              WHERE  CURRENT OF CUPDATE

CUPDATEC      CLOSE CUPDATE
```

Figure 2-4. CURSOR UPDATE, Standard IMSADF II Function

The CUPDATE statement declares a CURSOR to SELECT a row, OPENs the CURSOR, fetches the row into an IMSADF II work area, updates the row in the Table (or VIEW if allowed), and CLOSEs the CURSOR. The DECLARE CURSOR, OPEN and FETCH are invoked through the CUPDATEO function. The UPDATE, where current of cursor, is executed with the CUPDATEU function. The close CURSOR is executed with the CUPDATEC function.

At execution time the CUPDATE function can be used as follows:

- Standard data base update function
- High Level Audit Language 'CUPDATE','CUPDATEO','CUPDATEU','CUPDATEC'
- SQLHNDLR Call function 'CUPDATE','CUPDATEO','CUPDATEU','CUPDATEC'

CDELETE - CURSOR DELETE for Single Row

CURSOR DELETE is the standard IMSADF II SQL function used to delete a single row from a DB2 Table.

The CURSOR DELETE function deletes a row that has been previously SELECTed. The CURSOR DELETE function is used to FETCH a new copy of the row and hold the cursor open. If the IMSADF II data compare (DATACOMP) function has been specified, the new copy of the row is compared with an initial copy of the row that was previously SELECTed. Upon verification of the row the DELETE is issued. If data compare is not specified the DELETE is issued without data comparison.

```
FUNCTION     GENERATED SQL STATEMENT

CDELETEO -   DECLARE CDELETE CURSOR FOR
             SELECT column1,column2, ... column(n)
             FROM   tablename
             WHERE  column1=:Axxffff1

             OPEN CDELETE

             FETCH CDELETE
             INTO   :Axxffff$,:Axxffff$:Axxffff@,...

CDELETED     DELETE FROM tablename
             WHERE  CURRENT OF CDELETE

CDELETEC     CLOSE CDELETE
```

Figure 2-5. CURSOR DELETE, Standard IMSADF II Function

The CDELETE statement declares a CURSOR to SELECT a row, OPENs the CURSOR, fetches the row into the IMSADF II work area, deletes the row in the Table (or VIEW if allowed), and CLOSEs the CURSOR. The DECLARE CURSOR, OPEN and FETCH are invoked through the CDELETEO function. The DELETE, where current of cursor, is executed with the CDELETED function. The close cursor is executed with the CDELETEC function.

At execution time the CDELETE function can be used as follows:

- Standard data base delete function
- High Level Audit Language 'CDELETE','CDELETEO','CDELETED','CDELETEC'
- SQLHNDLR Call function 'CDELETE','CDELETEO','CDELETED','CDELETEC'

SELECT - SELECT of a Single Row

SELECT is a non-standard IMSADF II SQL function that is used to retrieve a single row from a DB2 Table.

This statement can only be used successfully when the result Table contains a single row (that is, Unique key columns).

When this SELECT statement is executed, and the result Table contains more than one row, DB2 issues a SQL return code of -811 and the transaction is terminated.

```
FUNCTION     GENERATED SQL STATEMENT

SELECT   -   SELECT column1,column2, ... column(n)
             INTO   :Axxffff$,:Axxffff$:Axxffff@,...
             FROM   tablename
             WHERE  column1=:Axxffff1
```

Figure 2-6. SELECT, Non-Standard IMSADF II Function

At execution time the SELECT function can be used as follows:

- High Level Audit Language 'SELECT'
- SQLHNDLR Call function 'SELECT'

UPDATE - UPDATE of a Single Row

UPDATE is a non-standard IMSADF II SQL function that is used to update a one or more rows of a DB2 Table.

If the key Columns in the WHERE clause define a unique key value then only one row in the DB2 Table is updated.

If the key Columns in the WHERE clause do not define a unique search condition then all rows for which the search condition is true are updated.

The UPDATE statement updates all columns defined in the Table/View that are eligible for update (SQLUPD=YES).

DB2 columns specified as KEY=YES are not eligible for update and do not appear in the SET clause.

```
FUNCTION    GENERATED SQL STATEMENT

UPDATE   -  UPDATE tablename
            SET    column2=:Axxffff$:Axxffff@,...,
                   columnn=:Axxffff$:Axxffff@
            WHERE  column1=:Axxffff1
```

Figure 2-7. UPDATE, Non-Standard IMSADF II Function

At execution time the UPDATE function can be used as follows:

- High Level Audit Language 'UPDATE'
- SQLHNDLR Call function 'UPDATE'

DELETE - DELETE of a Single Row

DELETE is a non-standard IMSADF II SQL function that is used to delete one or more rows of a DB2 Table.

If the key Columns in the WHERE clause define a unique key value then only one row in the DB2 Table is deleted.

If the key Columns in the WHERE clause do not define a unique search condition then all rows for which the search condition is true are deleted.

```
FUNCTION    GENERATED SQL STATEMENT

DELETE   -  DELETE FROM tablename
            WHERE  column1=:Axxffff1
```

Figure 2-8. DELETE, Non-Standard IMSADF II Function

At execution time the DELETE function can be used as follows:

- High Level Audit Language 'DELETE'
- SQLHNDLR Call function 'DELETE'

KSELECT - SELECT for Secondary Key Selection Browse

The KSELECT1 and KSELECT2 functions are special cases of the standard IMSADF II CURSOR SELECT function. The Key Columns in the Where clauses are defined with '>=', '<=', or the DB2 LIKE relational operator.

These functions are used during Secondary Key Selection browse for accessing DB2 Tables/Views.

The KSELECT1 function is invoked if the terminal operator:

        enters a partial key with a > or <
    or
        enters an incorrect key
    or
        enters no key
    and
        the Table is eligible for secondary key selection (SKSEGS>0), and the KSELECT1 function was defined for the Table Handler Rule.
    and
        SPASQLKS was not set to a user defined function KSELECTn by a primary key audit.

The KSELECT2 function is invoked if the terminal operator:

        enters a partial key with a % or _
    and
        the Table is eligible for secondary key selection (SKSEGS>0), and the KSELECT2 function was defined for the Table Handler Rule.
    and
        SPASQLKS was not set to a user defined function KSELECTn by a primary key audit.

The Columns in the WHERE clause should be defined as Columns in a UNIQUE INDEX. The host variables in the WHERE clause may contain an initialized value, a partial value or full value depending on the terminal operator input.

The secondary Key Selection browse functions have an ORDER BY clause associated with their SQL SELECT statement. This ensures that the rows displayed on the key selection browse screen are in an ordered sequence. Even though a UNIQUE INDEX is defined for a Table the DB2 BIND process may choose to ignore it and search the Table using a Table scan technique. This causes the rows to be displayed in an arbitrary order. Specifying an ORDER BY clause may influence DB2 to use the INDEX to avoid the overhead of the sort associated with using an ORDER BY clause.

KSELECT2 is included in the key selection SQL calls only if SQLCALL=KSELECT2 is specified, and if the Table keys include at least one Column with an alphanumeric data type. Additionally, the LIKE relational operator is only available for the first KEY Column with an alphanumeric data type. IMSADF II alphanumeric data types are alphanumeric, alpha, numeric, and varchar. Normally DB2 does not use the index when processing WHERE clauses containing the LIKE relational operator. However, if a UNIQUE INDEX is defined for the Column being operated on by the LIKE relational operator, then DB2 may use it.

Use of the key select browse function should be reviewed for performance considerations, that is, number of rows in the Table/View satisfying the WHERE condition and potential number of terminal operator iterations of the browse function.

Each screen iteration during the key selection browse function is a separate transaction iteration. The result Table for the rows being browsed are retrieved for each screen iteration. This implies that the WHERE clause should be defined so that each subsequent key selection browse iteration be positioned correctly into the Table, beyond the rows previously retrieved.

The key selection browse function for large Tables and/or a large number of KEY Columns may be expensive in terms of CPU time and resources.

```
FUNCTION     GENERATED SQL STATEMENT

KSELEC10   - DECLARE KSELECT1 CURSOR FOR
             SELECT column1,column2, ... column(n)
             FROM   tablename
             WHERE  column1>=:Axxffff1
             ORDER BY COLUMN1 ASC
*
* :Axxffff1 Holds initial key value entered on the first iteration.
*            On subsequent iterations it holds the last key displayed
*            on the Secondary Key Selection Browse Screen
*
*            >= is used when SQLORD=ASC and <= when SQLORD=DESC
*
             OPEN KSELECT1

KSELEC1F     FETCH KSELECT1
             INTO  :Axxffff$,:Axxffff$:Axxffff@,...

KSELEC1C     CLOSE KSELECT1
----------------------------------------------------------------------
KSELEC20   - DECLARE KSELECT2 CURSOR FOR
             SELECT column1,column2, ... column(n)
             FROM   tablename
             WHERE  column1 LIKE :Axxffff2
             AND column1 >=:Axxffff1
             ORDER BY COLUMN1 ASC
*
* :Axxffff1 Holds initial key value entered on the first iteration.
*            On subsequent iterations it holds the last key displayed
*            on the Secondary Key Selection Browse Screen
* :Axxffff2 Holds search value with the LIKE Predicate (% or _).
*
*            The LIKE Predicate is ONLY available for the first
*            character KEY Column.
*

             OPEN KSELECT2

KSELEC2F     FETCH KSELECT2
             INTO  :Axxffff$,:Axxffff$:Axxffff@,...

KSELEC2C     CLOSE KSELECT2
```

Figure 2-9. KSELECT, Secondary Key Selection Browse Function

The KSELECT statement declares a CURSOR to SELECT a set of rows, OPENs the CURSOR, fetches one or more rows into an IMSADF II work area, and CLOSEs the CURSOR. The DECLARE CURSOR and OPEN are invoked through the KSELECn0 function. The KSELECnF function fetches the next row in the table and the KSELECnC function closes the cursor.

The following defines the WHERE clauses required for the Secondary Key Selection Browse functions with ONE and TWO key Columns:

- KSELECT1 - ONE KEY

```
WHERE COLUMNA >= CURRKEY1
```

- KSELECT2 - ONE KEY

```
WHERE COLUMNA LIKE INITKEY1 and COLUMNA >= CURRKEY1
```

- KSELECT1 - TWO KEYS

```
WHERE (COLUMNA >= CURRKEY1 AND COLUMNB >= CURRKEY2)
    OR
      (COLUMNA > CURRKEY1)
```

- KSELECT2 - TWO KEYS

```
        WHERE (COLUMNA LIKE INITKEY1 AND
               COLUMNA >= CURRKEY1 AND COLUMNB >= CURRKEY2)
            OR
              (COLUMNA LIKE INITKEY1 AND
               COLUMNA > CURRKEY1 AND KEY2 >= INITKEY2)
```

INIT	Initial KEY value: Specified by the user or HI or LO values assigned by IMSADF II.
CURR	Current KEY value: Contains the initial key value on the first iteration. On subsequent iterations it contains the key of the last row displayed on the Secondary Key Selection Browse Screen.

Notes:

1. The LIKE Predicate is ONLY available for the first character KEY Column.

2. Three or more KEY Columns follow the same pattern of increased complexity.

3. Invocation of Secondary Key Selection browse is controlled by the SKSEGS operand on the TABLE statement.

At execution time the KSELECT1 and KSELECT2 functions can be used as follows:

- Secondary Key Selection Browse (Conversational only)

- High Level Audit Language 'KSELEC1O','KSELEC1F','KSELEC1C',
  'KSELEC2O','KSELEC2F','KSELEC2C'

- SQLHNDLR Call function 'KSELEC1O','KSELEC1F','KSELEC1C',
  'KSELEC2O','KSELEC2F','KSELEC2C'

USER SQL Statements

User SQL calls may also be included in the Table Handler Rule for execution in the High Level Audit Language or in the SQLHNDLR Call function.

SQLUSER=<u>NO</u>
YES

SQLUSER=YES indicates that USER SQL statements with user specified WHERE clauses are to be included in this Table Handler Rule. Default value is NO. If YES, the user supplies a label, the SQLFUNC, and a WHERE clause. User SQL statements are specified immediately following the GENERATE OPTIONS=TABH SQLUSER=YES statement.

USER SQL statements can be executed through Audit operations or with the SQLHNDLR Call function in an Audit Exit or Special Processing Routine.

A maximum of 26 SQLUSER WHERE clauses can be specified for a Table Handler Rule.

User SQL statements have the following format:

```
GENERATE OPTIONS=TABH,TABLES=xx,SQLUSER=YES
label   sqlfunc    where clause
&SQLENDS
```

Note: The &SQLENDS label must be specified in columns 1 to 8 to terminate USER SQL statement definitions.

LABEL	LABEL is a one- to eight-character name, where the first character must be alphabetic. The LABEL must be specified starting in column one.

USER specified LABELs cannot match any of the labels used to define the Standard and Non-Standard SQL statements. The functions defined in the previous section, (Figure 2-2 through Figure 2-9) represent the reserved LABELs.

LABEL defines the type of SQL function in the High Level Audit Language SQL call operation and the function name specified in a SQLHNDLR Call.

Another SQLUSER option is to augment the Secondary Key Selection browse function. In this case, LABEL has the format KSELECTn, where n is 3 through 9, (n=1 and 2 are reserved for standard Secondary Key Selection browse functions). Refer to "USER SQL - Key Selection Browse" on page 2-30 for additional details.

SQLFUNC

SQLFUNC defines the type of SQL statement to be executed. It must be specified starting in column 10.

The valid SQL statements are:

SELECT

UPDATE

DELETE

CURSOR

OPEN

FETCH

CLOSE

Note: The SQL INSERT statement is not valid as a USER SQL statement. It's definition is the same as the standard INSERT function. If INSERT is required for Table processing use the standard INSERT function.

WHERE CLAUSE

WHERE CLAUSE defines the DB2 operands required for the specified SQLFUNC. It must be defined in columns 19 through 71. If more than one line is required, then enter a continuation character in column 72 and start the continuation in column 16 of the next line.

WHERE CLAUSE contains the WHERE clause for a SELECT, UPDATE, DELETE, and CURSOR statement. It contains only the CURSOR label in the OPEN, FETCH and CLOSE statements.

Notes:

1. USER SQL statements cannot override the Columns to be SELECTed or the Columns to be UPDATEd. The Rules Generator uses the associated Table Layout Rule for this information.

2. When a SQL statement is defined to IMSADF II in a Table Handler Rule, all DB2 Columns defined in the corresponding Table Layout Rule are named and SELECTed for that Table/View. In the UPDATE and INSERT statements, the SET and VALUES clauses include all Columns specifying the Rules Generator operands, SQLUPD=YES and SQLISRT=YES.

The WHERE clause may contain any of the DB2 comparison operators, predicate relational operators and calculated values. The host variables in the WHERE clause are represented through the IMSADF II Column names.

The format of a host variable is - :ffff.xx

COLON	- required host variable delimiter
ffff	- one- to four-character Column ID
PERIOD	- required
xx	- two-character Table ID

Note: Table ID and Column ID must be previously defined in this Rules Generator input stream.

At execution time, the host variables are passed by the Auditor or the SQLHNDLR Call in a contiguous string and in the order specified in the WHERE clause. Reference the High Level Audit Language and Exit Functions sections for additional details on the SQLHNDLR Call.

The WHERE clause is included, as is, in the Table Handler Rule, with only the substitutions of host variables. The :ffff.xx is converted to :Axxffffn where n is a sequential letter A to Z uniquely identifying the appropriate USER SQL statement variables.

For the OPEN, FETCH, and CLOSE statements, the WHERE Clause contains the LABEL name of the applicable CURSOR.

The selected columns, and the DB2 SET and VALUES clauses are built by IMSADF II according to the Rules Generator Table/Column definitions, the SQLIND, SQLNULL, SQLUPD, and SQLISRT operands.

If the SQLDIST=YES is specified on the Rules Generator TABLE statement then all SELECT statements in the Table Handler Rule are defined with the DB2 DISTINCT keyword.

If 'FOR UPDATE OF' is specified in the WHERE Clause of a CURSOR statement, then 'FOR UPDATE OF' is included in the SELECT statement declared for the cursor.

For USER SQL key selection, specify (LABEL - KSELECTn, SQLFUNC - SELECT). The WHERE clause can contain the Key column host variables, non-key host variables and pseudo segment field host variables. If the host variables are not keys, the High Level Audit Language function SPAWHERE must be executed during a Primary Key Audit. SPAWHERE must reflect the host variables in the defined data format, length, and order. Additionally, the Primary Key Audit should specify which USER SQL key selection function to execute, (that is, KSELECT3, or KSELECT4) by also setting SPASQL in the Primary Key Audit. Refer to the High Level Audit Language section for details on defining SPAWHERE and SPASQL.

User specified secondary key selection WHERE clauses should be defined to allow for repositioning into the table for each subsequent iteration, when the terminal operator requests the next page of selections. This can be done by updating the host variables to reflect the last row on the page. A Secondary Key Audit routine called after each row is FETCHed can perform this function.

If a row is selected for display from the Secondary Key Selection browse screen, it is re-fetched using the standard IMSADF II CSELECT call. The host variables used for this fetch may be different from the values used for the USER SQL key selection function if non-keyed host variables are used for the secondary key selection browse.

When specifying KSELECTn (n=3-9) functions, only define the SELECT statement and the associated WHERE clause. Logic is included in the Table Handler Rule to DECLARE, OPEN, FETCH, and CLOSE the KSELECTn CURSOR. These functions are manipulated by IMSADF II secondary key selection modules when KSELECTn is specified.

Figure 2-10 contains examples of USER SQL statements.

```
GENERATE OPTIONS=TABH,TABLES=XX,SQLUSER=YES

* LABEL   SQLFUNC  WHERE CLAUSE
*    columns
* 1 to 8 10 to 17 19  to  71
*              16 continuation (16 TO 71)
*-------- -------- ---------------------------------------------------
XXSELECT SELECT   WHERE TABCOLUMN1 BETWEEN :ffff.xx AND :ffff.xx
               AND TABCOLUMN2 > :ffff.xx
XXUPDATE UPDATE   WHERE TABCOLUMN1 IN (:ffff.xx,:ffff.xx)
XXDELETE DELETE   WHERE TABCOLUMN1 <= :ffff.xx AND TABCOLUMN1 ¬= 0
XXCURS1  CURSOR   WHERE TABCOLUMN1 LIKE :ffff.xx FOR UPDATE OF
XXCURS1O OPEN     XXCURS1
XXCURS1F FETCH    XXCURS1
XXCURS1U UPDATE   WHERE CURRENT OF XXCURS1
XXCURS1D DELETE   WHERE CURRENT OF XXCURS1
XXCURS1C CLOSE    XXCURS1
KSELECT3 SELECT   WHERE TABCOLUMN1 BETWEEN :ffff.xx AND :ffff.xx
&SQLENDS
```

Figure 2-10. USER SQL Statement Formats

If additional DB2 function is required, native SQL calls may be issued in an Audit Exit or Special Processing Routine and mapped back to the SPA work area (Table ID or pseudo segment ID) via the MAPPER or COPYSEG function.

Only those SQL calls, that are executed by the IMSADF II transaction drivers, should be included in the Table Handler Rule. The number of SQL calls determine the size of an DB2 Application Plan and the size of an DB2 Application Plan is an important consideration in the performance of the DB2 system.

Summary for Table Handler Rule

In summary, all Column names in a Table are specified in the SELECT list and Indicator Variables if specified (SQLIND=YES on the TABLE statement) are associated with each Column. However, only those Columns that have SQLNULL=YES specified are eligible for Indicator Variable processing.

The Host variables, specified in the INTO clause, define the receiving I/O area for each Column. This I/O area may be in the IMSADF II SPA work area, an IMSADF II internal buffer area for secondary key selection browse or Data Compare, or in an exits work area.

The WHERE clause specifies the Column names, comparison operators '=', '>=', '<=', or the DB2 LIKE relational operator, and host variables representing the KEY columns.

IMSADF II support allows up to 512 Columns per Table. The maximum Table I/O area including Indicator Variables is 6000 bytes.

The DB2 pre-compiler issues a Return Code 4 when processing the generated Table Handler Rule. This is due to the absence of a SQL TABLE DECLARATION statement. For COBOL and PL/I, the DB2 DCLGEN function builds a data structure and SQL DECLARE. The pre-compiler matches the SQL statement usage against SQL DECLARE for consistency. Since the Rules Generator created Table Handler Rule is an Assembler program and DB2 does not provide DCLGEN support for Assembler language programs this SQL DECLARE is omitted. It provides no additional checking for IMSADF II.

Summary of Rules Generator Operands for the Table Handler Rule

The Rules Generator operands that determine the content of the Table Handler Rule follow:

TABLE statement

ID	Naming convention for Table Handler Rule (ssssSid)
SKSEGS	If greater than zero then eligible for SKS calls
SQLIND	Specifies indicator variables are to be generated
SQLDIST	Specifies the presence of DISTINCT in SELECT statements.
SQLNAME	Table name in FROM and INTO clause
TYPE	Indicates these statements define a DB2 Table

COLUMN statement

KEY	Indicates search columns in WHERE clause
TYPE	Indicates data type for the column host variable
ID	Naming convention for host variables where xx is Table ID and ffff is Column ID. Host Variables - :Axxffff$ Indicator Variables - :Axxffff@ KEY Columns - :Axxffffn (n is 1-9 or a-z)
LENGTH	Indicates length of the column
SQLISRT	Indicates whether column exists in VALUES clause
SQLNAME	Column name
SQLNULL	Indicates whether column may be NULL
SQLORD	Indicates order of key values (ASC, DESC)
SQLUPD	Indicates whether column exists in SET clause

GENERATE statement

SQLCALL	Specifies the SQL statements to be built
SQLUSER	Specifies user SQL statements and WHERE clauses to be built

Figure 2-11. Rules Generator Operands Applicable to Table Handler Rule

Other GENERATE Statement Operands

OPTIONS=SGALL

Format of the SGALL GENERATE statement is:

GENERATE OPTIONS=SGALL,SEGMENTS=(id,id,...id)

The SGALL operand is a Rules Generator operand that is compatible with DB2 table definitions and DL/I segment definitions.

The GENERATE option SGALL produces a Table Layout Rule for each DB2 Table ID and a Segment Layout and Segment Handler Rule for each DL/I segment specified in the SEGMENTS operands. If the SEGMENTS operand is not specified, a Table Layout Rule is generated for each DB2 Table defined in this Rules Generator input stream. For each DL/I segment defined in this Rules Generator input stream, both a Segment Layout and Segment Handler Rule are generated.

Table Handler Rules must be generated using the GENERATE OPTIONS=TABH statement.

Input Transaction Rules

The Input Transaction Rules will support the inclusion of both DL/I segments and DB2 Tables in the same rule. Both segments and tables will be displayed on the same screen and updated in the appropriate data base management system.

The following existing GENERATE statement operands apply additionally to DB2 tables:

DBPATH DBPATH list those DB2 Tables to be SELECTed by the transaction drivers and for which the terminal operator is prompted for keys in the conversational environment. Table ID order within the DBPATH operand becomes the processing order at execution time. If the DL/I segment IDs are intermixed, an entire DL/I path is still processed together.

TSEGS TSEGS list those DB2 Tables to be allocated space in the SPA work area. Access to TSEGS is provided with SQL statements invoked with Audit operations or the SQLHNDLR Call function.

DLET DLET eligibility indicates that the Table row may be deleted through an Audit operation or through the SQLHNDLR call function. This is the analogous function to DL/I segments.

ISRT If the transaction mode is Update or Delete and this Table row is not found, the row is INSERTed if the USERID has ADD authority and data was entered for the row. This is the analogous function to DL/I segments.

Because each DB2 table specified is considered by IMSADF II to be a root-only data base, each table will have insert elibibility independent of all other tables specified in the DBPATH.

For example, a user could specify:

```
DBPATH=(T1,T2,T3),ISRT=(T3)
```

In this case, data for table T3 may be inserted even though no rows exist for tables T1 and T2.

DATACOMP The DATACOMP operand specifies Tables/Segments which are compared for change prior to update or delete. This operand is valid only for conversational processing and should be specified for any Table/Segment that could have simultaneous updates by different users. Before an UPDATE or DELETE is performed, the Table row is SELECTed and compared with the copy saved at the first screen display. The CUPDATE and CDELETE Standard SQL statements are used for the data base update and data base delete functions.

Other existing operands specifying Input Transaction Rule generation requirements follow:

- Request for Generation

  OPTIONS=CVALL, TPALL, BAIT

- Transaction Identification

  TRXID, DBPATH, TSEGS, AGROUP, LRULE, LINKLIB, ASMREQ

- Transaction Exit Specifications

  SPECIAL, LANGUAGE, BYPASS, STX, DLIEXIT (only data base DL/I calls)

- Delete and Insert Eligibility

  DLET, ISRT

- Conversatioral and Nonconversational Only

  CURSOR, DEVNAME, DEVTYPE, IMAGE, PFKDATA, PFKLIT, PFKNUMB

- Conversational Only

  SPOS, TRXNAME, DISPNAME, DATACOMP, DAMSG, ADDMODE, COMMLEN, KEYSEL, MAXKEY, PGROUP, SFORMAT, SHEADING, SOMTX, DKEY, DTRAN

- Nonconversational Only

  MODNAME, ORID

- Batch Only

  CNT, EOF

Note: The standard processing TWIN operands do not apply to DB2 Tables. They are only available for DL/I processing.

Preload, Composite and Driver Link-Edits

The Rules Generator PRELOAD, COMPOSITE and DRIVER LINK-EDITS statements have not changed for DB2 transactions. Only DB2 transactions must be linked to an IMSADF II driver, as all IMSADF II transactions make use of the facility data bases.

The GENERATE statements incorporating SHTABLE (the list of Segment Handler Rules), also incorporates Table Handler Rules. The GENERATE OPTIONS applying to the expanded use of the SHTABLE operand include:

- PREL Preload Table
- CTLE,NCLE Composite Load Modules
- STLE,SPLE Conversational Link-edits
- NCLE,TPLE Nonconversational Link-edits
- BDLE Batch Driver Link-edit

USING THE RULES GENERATOR EXECUTION PROCEDURE

A catalog procedure, ????G (???? - Substitute installed ADFID) is provided to invoke the Rules Generator. Reference the IMS Application Development Facility II Version 2 Release 2 Application Development Reference Rules Generator chapter, and the IMS Application Development Facility II Version 2 Release 2 Installation Guide for complete details.

A new PARM operand and several new DDNAME statements have been added to this procedure for the DB2 support.

#COLS A PARM operand specified to indicate the maximum number of DB2 columns for one SYSTEM statement in this execution of the Rules Generator. #cols is used in the GETMAIN size calculation.

Following is the format of the PARM operand specified on the Rules Generator JCL execute statement.

```
PARM=(#fields,#screens,#cols,sysid)
```

The #cols is the third positional parameter. SYSID remains an optional parameter. If the SYSID is not specified the PARM field would be specified as:

```
PARM=(#fields,#screens,#cols)
```

#cols is required for DB2 tables, since there is a significant size difference in the NAME and SQLNAME operands.

The default is 100. This includes Table and Column statements that contain the SQLNAME operand.

ADFSQLHO Input to DB2 pre-complier. Contains Rules Generator created Assembler language source for a Table Handler Rule.

ADFSQLHW Work file used by the Rules Generator during Table Handler Rule generation.

DBRMLIB The DB2 pre-compilier stores a DBRM member for each Table Handler Rule created into this cataloged MVS partitioned data set.

DB2 requires that the member name be included in the JCL. This implies that only one DBRM member can be defined per invocation of the DB2 pre-compiler.

However, the Rules Generator can create many Table Handler Rules (and DBRMs) during one execution. Therefore, the hard coded member name in the DBRMLIB DD JCL statement is not used. It must be there, but it is treated as a dummy name.

In order for the Rules Generator to process the DBRMLIB DD JCL statement, the DISP parameter must be specified as OLD or SHR.

The Rules Generator builds a member name for each Table Handler Rule and DBRM created in the following format ssssSid where:

```
ssss - Rules Generator SYSTEM statement SYSID operand
   S - Constant
  id - TABLE ID being processed
```

Note: This naming convention is the same as the naming convention used when the Rules Generator creates Segment Handler Rules.

DB2PRINT Used by DB2 pre-compilier as SYSPRINT file.

DB2TERM Used by DB2 pre-compilier as SYSTERM file.

HIGH LEVEL AUDIT LANGUAGE

- All Audit Phases

  DB2 Table Columns (fields) are eligible for all three phases of auditing:

  - KEY - pre SQL call
  - PRELIM - before transaction screen display
  - PROCESS - after transaction screen input

  Additionally, all three legs of auditing apply.

  - P0 - Automatic Field Assignment
  - P1 - Field Audit
  - P2 - Message sending

- All audit operations, except DL/I related are available for DB2 Table processing.

  All field types are supported except TYPE=FLOAT Columns which are limited to data moves of FLOAT to FLOAT, FLOAT to ALPHANUMERIC, and ALPHANUMERIC to FLOAT. Arithmetic operations and other data conversions are not supported.

- All current audit routines, including arithmetic, data compares and moves, encode-decode, subroutine branching, message sending, dynamic screen attribute modification, transaction switching, are available to DB2 Table Columns.

- Refer to the Audit Language, and the Data Base Rules Specification chapters of IMS Application Development Facility II Version 2 Release 2 Application Development Reference for additional details on all audit operations.

DB2 RELATED AUDIT OPERATIONS

- CONCAT and SUBSTR operations are available for character string manipulation for field types ALPHA, ALPHANUM, NUM and VARCHAR.
- IMMEDIATE DB2 SQL call operation
- Interrogate DB2 SQLCODE and SQLWARN(0-7)
- USER defined Secondary Key Selection Browse
- DB2 Column Null value (Set and Test)
- DB2 Column Truncation (Test)

CONCAT

The CONCAT operation concatenates two source fields into one target field. The entirety of each source field is used in the concatenation. For VARCHAR fields this is determined by the current field length. If one of the source fields is null, the target is set to null.

field = CONCAT field field

The fields used can be audited or related fields.

Example:

```
    STR1 = CONCAT BAPSSTR2 BAPSSTR3

    - will generate -

        01GOBAPSSTR10202
        0001(BAPSSTR2,BAPSSTR3)
```

SUBSTR

There are two forms of the SUBSTR audit operation. In one the substringing is performed on the source field, while on the other, it is performed on the target field. For both operations, the target field name is in the related field area of the operation descriptor, the source field name is in the first data descriptor and the starting position and length of the substring are in the second data descriptor. The auditor uses the operation descriptor code to tell whether to perform the substring on the target or the source field. The two forms of the SUBSTR are as follows:

tgtfield = SUBSTR srcfield starting-position : length

SUBSTR tgtfield starting-position : length = srcfield

Any field name used may be an audited or a related field. Both starting position and length must be numeric constants or both must be fields containing numeric constants. The numeric constants must be 1 to 255.

Example: Source field substringing

```
    STR1 = SUBSTR BAPSSTR2 7 : 3

           OR

    STR1 = SUBSTR BAPSSTR2 BAPSSTR3 : BAPSSTR4

    - will generate -

        01G1BAPSSTR10202                  01G1BAPSSTR10202
           0001(BAPSSTR2)         OR         0001(BAPSSTR2)
           0002(7,3)                         0002(BAPSSTR3,BAPSSTR4)
```

Example: Target field substringing

```
    SUBSTR BAPSSTR1 7 : 3 = STR4

           OR

    SUBSTR BAPSSTR1 BAPSSTR2 : BAPSSTR3 = BAPSSTR4

    - will generate -

        01G2BAPSSTR10202                  01G2BAPSSTR10202
           0001(BAPSSTR4)         OR         0001(BAPSSTR4)
           0002(7,3)                         0002(BAPSSTR2,BAPSSTR3)
```

IMMEDIATE SQL Call

An operation, comparable to the IMMEDIATE DL/I call operation, is available to execute SQL calls for a Table/View.

SQL call expressions are coded in the following format:

IF SQL label   field   <'>tableid<'> <NOT> OK

SQL Identifies the statement as a DB2 SQL function.

label Defines the type of SQL function, standard, non-standard, or USER SQL statement.

Following are the valid SQL functions that may be specified as labels.

```
SELECT, UPDATE, DELETE, INSERT
CSELECT, CUPDATE, CDELETE
KSELECnO, KSELECnF, KSELECnC
CSELECTO, CSELECTC
CUPDATEO, CUPDATEU, CUPDATEC
```

CDELETEO, CDELETED, CDELETEC
USER SUPPLIED - label is one to eight characters (first character alphabetic) and is a label that was specified in the SQLUSER LABEL section when the Rules Generator was creating the Table Handler Rule for the specified Table ID.

Notes:

1. All DB2 SQL calls are immediate. If non-immediate DB2 SQL calls are required, (that is, Table updates do not occur until transaction driver data base updates, after successful PROCESS Audits), the SETFLAG operation should be used.
2. The IMSADF II data compare DATACOMP function only applies to transaction driver updates or a SEGUPDTE Call function.

field Name of the field containing the search values. This field must represent a field containing the concatenation of all host variables in the WHERE clause.

For the standard SQL functions, the host variables are in the order in which the KEY columns are specified to the Rules Generator.

A keyword 'KEYFIELD' specifies that the current key associated with the Table will be used.

For USER SQL functions, the field must represent a concatenation of all host variables in the WHERE clause, in the order specified in the WHERE clause.

tableid Specifies the two-character Table ID of the DB2 Table to be accessed.

OK Indicates that SQLCODE is zero and SQLWARN0 is not 'W'

The SQL call expression performs both a SQL call and a check of the SQL status code returned from the call. If the SQLCODE is zero and SQLWARN0 is blank, the NEXT TRUE branch is taken. To determine the specific setting of SQLCODE and SQLWARN returned, a system expression using SQLCODE and SQLWARN(0-7) keywords can be used.

Example:

```
      IF SQL DEPTSELC  SAP2DEPT ES NOT OK
        ERRORMSG = 0805
```

- will generate -

```
    01G8SAP2DEPT02  0805   If DEPTSELC function on table ES
       0001(ES,DEPTSELC)   using the key from related
                           field SAP2DEPT is NOT OK, issue
                           error msg 0805 and terminate audit
```

Interrogate SQL SQLCODE and SQLWARN(0-7)

To interrogate the DB2 SQL return code and warning conditions associated with a SQL call, the following system field keywords have been defined.

SQLCODE A fullword integer in the SQL Communication Area that contains a return code pertaining to the most recently executed SQL statement. Some of the return code values you might be interested in are:

Zero The SQL statement executed successfully.

-n A negative integer value, DB2 error.

+n A positive integer value, the SQL statement executed successfully, and an exceptional condition has occurred.

	+100	No data exists to process.
SQLWARN	Eight single character variables that denote a warning, of which only the first six are used by DB2.	
	SQLWARN0	If blank, all other SQLWARNn variables are blank. Otherwise the value 'W' and at least one other SQLWARNn variable is also 'W'.
	SQLWARN1	If 'W', at least one column's value was truncated when it was stored into a host variable.
	SQLWARN2	If 'W', at least one null value was eliminated from the argument set of a function.
	SQLWARN3	If 'W', the number of host variables specified in the SQL statement is unequal to the number of columns in the table or view being operated on by the statement.
	SQLWARN4	If 'W', a prepared (that is, dynamic SQL) UPDATE or DELETE statement does not include a WHERE clause.
	SQLWARN5	If 'W', your program tried to execute a statement that applies only to SQL/DS.
	SQLWARN6	Reserved.
	SQLWARN7	Reserved.

The format of the SQLCODE system expressions is:

SQLCODE comparison-operator numeric-constant(s)

Where:

comparison-operator:	=, ¬=, <, >, <=, >=, EQ,NE,LT,GT,LE,GE
numeric-constant:	list of one or more numeric constants

Use of SQLCODE causes a compare between the corresponding system field and a list of one or more numeric constants. If any one compare is true, the result of the compare is true. It is recommended that only one constant be coded for a compare using GT, LT, GE and LE.

The format of the SQLWARNn system expressions is:

SQLWARNn comparison-operator literal

Where:

comparison-operator:	=, ¬=, EQ ,NE
literal:	1 character warning value 'W' or ' '

Use of one of the SQLWARNn keywords causes the corresponding system field to be checked for a value of 'W' or ' '.

Following is a scenario of a check after a SQL call:

```
    IF SQLCODE < 0
      ERRORMSG = 0001
    ELSE
      IF SQLCODE = 100
        ERRORMSG = 0100
      ELSE
        IF SQLWARN0 = 'W'
          WARNMSG = 1000
        ENDIF
      ENDIF
    ENDIF
```

- will generate -

```
  02H1SPASQLCD  030001   If SPASQLCD is less than 0, put out
      0001(0)              error msg 0001 and terminate audit
  03G9SPASQLCD  040100   Else If SPASQLCD is 100, put out
      0001(100)            error msg 0100 and terminate audit
  04G9SPAWARN000001000   Else IF SPAWARN0 is 'W' put out
      0001(W)              error msg 1000 and terminate audit
```

During execution of a SQL call, the IMSADF II SPA (Scratch Pad Area) fields SPASQLCD and SPAWARN are updated with the DB2 SQL return code and warning conditions. The values in these SPA fields are the values interrogated when the SQLCODE and SQLWARNn system expressions are coded.

The Auditor maintains additional error information if a SQL call results in an error or warning condition. The DB2 SQL Communication Area information (SQLCODE, and SQLWARN) is maintained in the IMSADF II SPA for each audited field flagged in error. The information can be mapped into user error and warning messages using VARLIST6, and VARLIST7.

VARLIST6	4 bytes - SQLCODE
VARLIST7	8 bytes - SQLWARN(0-7)

USER SQL - Key Selection Browse

If a '>' or '<' is entered in a Key Column for a DBPATH table, the standard SQL statement KSELECT1 is executed.

If %string% or __string (DB2 LIKE relational operator) is entered, the standard SQL statement KSELECT2 is executed.

If a USER SQL statement is required for the Key Selection browse function, then USER SQL statement must be defined to the key selection process in a Primary Key Audit.

Note: The overall invocation control for the Secondary Key Selection browse function is the Rules Generator SKSEGS operand.

Two audit operations are available to aid in the Secondary Key Selection browsing of DB2 Tables.

SPASQL Sets a system field in the IMSADF II SPA (SPASQLKS).

The number of the user defined key selection function to be executed is moved to this field by setting SPASQL in a Primary Key Audit.

Note: The value coded is of the form KSELECTn. Only the n is stored in SPASQLKS.

The USER SQL Key Selection browse function is invoked instead of standard Secondary Key Selection browse whenever the terminal operator:

```
        enters a partial key with a >, <, %, or _
    or
        enters an incorrect key
    or
        enters no key
    and
        the Table is eligible for secondary key selection
        (SKSEGS>0), and the specified KSELECTn function
        was defined in the Table Handler Rule.
    and
        SPASQLKS was set to a user defined function
        KSELECTn (n=3-9) by a primary key audit.
```

Syntax for SPASQL statement is:

SPASQL = {field,KSELECTn,n}

Where:

field	a related or audited field that contains an 8 character name of the format KSELECTn, and n is a digit 3 to 9.
KSELECTn	a constant, and n is a digit 3 to 9.
n	a digit 3 to 9.

Example of 3 forms of SPASQL statements:

```
SPASQL = RELFIELD
SPASQL = KSELECT3
SPASQL = 3
```

- will generate -

```
01G3RELFIELD0202   Related field - KSELECTn, n is 3-9
                   SPASQLKS is set to n
02G3        0303   SPASQLKS is set to 3
   0001(3)
03G3        0000   SPASQLKS is set to 3
   0001(3)
```

SPAWHERE Sets a system field in the IMSADF II SPA (SPAWHERE).

It contains the name of the field holding the secondary key selection WHERE clause host variables. SPAWHERE should be set in conjunction with the SPASQL secondary key selection function if the WHERE clause requires other than the Table Key Column values.

The field named in SPAWHERE should be updated, with a Secondary Key Audit after each FETCH. This is to accommodate repositioning for subsequent secondary key selection iterations.

The value coded is an eight-character IMSADF II field name. At execution time the field name is converted into a two-byte offset into the Table I/O area. The two-byte offset is the value stored in SPAWHERE.

Syntax for SPAWHERE statement is:

SPAWHERE = {field,KEYFIELD}

Where:

field	a related or audited field.
KEYFIELD	a constant, that specifies that the key area for the Table contains the correct keys.

Example of SPAWHERE statement:

```
    SPAWHERE = AUDFIELD
    SPAWHERE = RELFIELD
    SPAWHERE = KEYFIELD

 - will generate -

    01G4        0202   Audited field
    02G4RELFIELD0303   Related field
    03G4KEYFIELD0000   Table key area
```

DB2 Column Null and Truncation (Test and Set)

- Test a DB2 Column for the null value.

  A test is made on the Columns indicator variable. A value of less than zero indicates the null value.

  Syntax for NULL test:

  IF <field> NULL comparison-operator {ON,OFF}

  Where:

field	a related or audited field to be tested for null value.
comparison-operator:	=, ¬=, EQ, NE
(ON,OFF)	ON - null, OFF - not NULL

  Example of NULL test:

```
        IF NULL = ON
        IF AUDFIELD NULL = ON
        IF RELFIELD NULL = ON

     - will generate -

        01G5        0200   test audited field
        02G5        0300   test audited field
        03G5RELFIELD0400   test related field
```

- Set a DB2 Column to the null value.

  The Columns indicator variable is set to a negative number.

  Note: This operation cannot be used to set a Column to not null. In order to change a Column from null to not null, data must be moved into the Column. Part of the logic associated with the data move is to set the indicator variable to zero.

  Syntax for Set to NULL:

  field NULL = ON

  Where:

field	a related or audited field to be SET to NULL.

  Example of SET to NULL:

```
        AUDFIELD NULL = ON
        RELFIELD NULL = ON

     - will generate -

        01G7        0203   set audited field to NULL
        01G7RELFIELD0203   set related field to NULL
```

- Test a DB2 Column for truncation.

A test is made on the Columns indicator variable. A value greater than zero indicates truncation occurred when DB2 moved data into the host variable. The positive value in the indicator variable is the length of the source Column in the DB2 data base.

Syntax for Truncation test:

IF <field> TRUNC comparison-operator {ON,OFF}

Where:

field	a related or audited field to be tested for truncation.
comparison-operator:	=, ¬=, EQ, NE
(ON,OFF)	ON - truncated, OFF - not truncated

Example of Truncation test:

```
IF TRUNC = ON
IF AUDFIELD TRUNC = ON
IF RELFIELD TRUNC = ON
```

- will generate -

```
01G6        0200   test audited field
02G6        0300   test audited field
03G6RELFIELD0400   test related field
```

MESSAGE GENERATION

To aid in the mapping of diagnostic information on SQL calls issued by the IMSADF II transaction drivers or an audit operation, VARLIST6 and VARLIST7 have been added to the message mapping process.

VARLIST provides a technique for mapping data other than Table/Column data into the message text.

Currently VARLIST1 can be used when Segment Handler Rule errors occur, to map the two-character DL/I status code into the message. VARLIST6 can be used in a similar manner to map the four-character DB2 SQL return code, (SQLCODE) into a message.

Messages generated by the Auditor can map both VARLIST6 and VARLIST7 data into the message text. The Auditor maintains an error table in the SPA for each audited field flagged in error. Fields that are set in error using the SETERROR function are not included in this table.

VARLIST6	4 bytes - SQLCODE
VARLIST7	8 bytes - SQLWARN(0-7)

EXIT FUNCTIONS

SQLHNDLR CALL

A SQLHNDLR function is available to Audit Exits and Special Processing Routines to retrieve or update DB2 data base Tables. The SQLHNDLR call, in conjunction with the appropriate Table Handler Rule, executes IMSADF II Standard, Non-Standard, and USER SQL statements to provide all the functions required to SELECT, INSERT, UPDATE, and DELETE a row in a DB2 Table/View.

The use of the Table Handler Rule eliminates the need for programming detailed DB2 SQL statements. It also eliminates the requirement that Audit Exits and Special Processing Routines must be processed by the DB2 pre-compiler, and be included in the DB2 application plan. Each Table/View that is accessed with a SQLHNDLR call must have a Table Handler Rule that contains the desired SQL statements.

When the transaction logic is controlled with a Special Processing Routine, it is also responsible for all data base updates (UPDATE, INSERT, DELETE). This can be accomplished either by individual SQLHNDLR calls or by using the automatic data base update routine, SEGUPDTE. A SEGUPDTE call causes all Tables/Segments in the IMSADF II SPA work area to be scanned to see if they should be INSERTed, UPDATEd, or DELETEd. Only the SEGUPDTE facility verifies that a Table row to be updated has not changed since originally fetched, if the IMSADF II data compare (DATACOMP) function was specified for this Table.

The logic of a Special Processing Routine should determine when and how a Table should be updated and issue the appropriate SQLHNDLR Calls or SEGUPDTE Call. In addition, any retrieval of Tables other than the Table rows SELECTed by the transaction driver, DBPATH, must be handled by the Special Processing Routine.

The application developer/programmer can perform data base I/O either into an area in the exit or into a Table I/O area reserved in the IMSADF II SPA work area. If the IMSADF II SPA work area is used, the Tables accessed must be described in the Input Transaction Rule.

I/O performed in the IMSADF II SPA work area has the following advantages:

- SQLHNDLR Call parameters are simplified.
- IMSADF II transaction drivers maintains the current key and updates it upon a successful FETCH.
- The Auditor can be invoked to validate Columns in the IMSADF II SPA work area.
- Input data is mapped to the appropriate Table Column(s) by the transaction driver.
- Table Columns in the IMSADF II SPA work area can be formatted and displayed by the transaction driver.
- The programmer can retrieve specific Columns or the entire row by calling the Data MAPPER or COPYSEG function.

SQLHNDLR Call Format

The format of the SQLHNDLR call is:

```
  In COBOL  WORKING STORAGE SECTION.
            77 ID      PICTURE XX.
            77 FUNC    PICTURE XXXXXXXX.
            77 KEY     PICTURE X(n).  NOTE n is defined as necessary.
            77 AREA    PICTURE X(n).  NOTE n is defined as necessary.
            77 TLR     PICTURE X(n).  NOTE n is defined as necessary.
            EXEC SQL
              INCLUDE SQLCA
            END-EXEC

            CALL 'SQLHNDLR' USING ID, FUNC, {KEY, AREA, TLR, SQLCA}.

  In PL/I
            DCL (SQLHNDLR) ENTRY OPTIONS(ASSEMBLER,INTER);
            DCL  ID    CHAR(2),
                 FUNC  CHAR(8),
                 KEY   CHAR(n),    /* n is defined as necessary  */
                 AREA  CHAR(n),    /* n is defined as necessary  */
                 TLR   CHAR(n);    /* n is defined as necessary  */
            EXEC SQL INCLUDE SQLCA ;

            CALL  SQLHNDLR (ID,FUNC,{KEY,AREA,TLR,SQLCA});
```

Note: The EXEC SQL INCLUDE SQLCA is <u>only</u> required if the optional parameter SQLCA is included in the call list. This implies that the common SQL Communication Area maintained by IMSADF II is not used. The Audit Exit or Special Processing Routine contains its own SQL Communication Area.

SQLHNDLR Call Parameters

The first two parameters of the SQLHNDLR Call are required. The remaining three are optional depending on the access requirements.

ID Two-character target Table/View ID. This ID is used to identify the appropriate Table Handler Rule and to determine whether the Table I/O area in the IMSADF II SPA work area is required.

FUNC A one- to eight-character label that defines the Standard, Non-Standard, or USER SQL statement to be executed.

Valid SQL statement labels are:

SELECT, UPDATE, DELETE, INSERT
CSELECT, CUPDATE, CDELETE
KSELECnO, KSELECnF, KSELECnC
CSELECTO, CSELECTC
CUPDATEO, CUPDATEU, CUPDATEC
CDELETEO, CDELETED, CDELETEC
USER SQL - label is one to eight characters (first character alphabetic) and is a label that was specified in the SQLUSER LABEL section when the Rules Generator was creating the Table Handler Rule for the specified Table ID.

KEY Optional Parameter

If specified, key defines an I/O area in the Audit Exit or Special Processing Routine that contains the WHERE clause host variables for the SQL function being executed.

If key is not specified, the transaction driver uses the current key for the Table which is kept in a reserved area in the IMSADF II SPA work area. This key is located using the ID parameter. Whichever key area is used, must contain the complete key required by the SQL function being executed.

If the SQLHNDLR Call is made using the Table I/O area in the IMSADF II SPA work area and if the SQL statement executed results in I/O, (that is, FETCH, UPDATE) then the current key area associated with the Table is updated.

No attempt is made to parallel the current IMSADF II SEGHNDLR call DL/I support that always updates the key area. DB2 does not provide a key feedback area.

The application programmer must ensure that the key area is correctly defined to match the host variables in the SQL statement being executed.

AREA Optional Parameter

If specified defines the I/O area in the Audit Exit or Special Processing Routine in which data base I/O is to be performed.

If AREA is not specified, the transaction driver uses the Table I/O area in the IMSADF II SPA work area for data base I/O.

TLR Optional Parameter

If specified defines the Table Layout Rule associated with the Table ID.

If TLR is not specified, the transaction driver uses the Table Layout Rule in the IMSADF II SPA work area if available. Otherwise, the parameter is initialized to zero.

SQLCA Optional Parameter

If the SQLCA parameter is specified DB2 maps the status of the SQL statement being executed into the SQL Communication Area defined in the Audit Exit or Special Processing Routine.

If specified it requires that the DB2 SQL Communication Area be defined in the Audit Exit or Special Processing Routine.

To define the SQL Communication Area requires coding the following DB2 statement in the Audit Exit or Special Processing Routine.

```
EXEC SQL INCLUDE SQLCA
```

It also requires that the Audit Exit or Special Processing Routine be processed by the DB2 pre-compiler prior to compile, assemble and link-edit.

If the SQLCA parameter is not specified the copy of the SQL Communication Area contained in the IMSADF II transaction driver is used. DB2 maps the status of the SQL statement being executed into this copy.

This eliminates the need to define a separate copy of SQL Communication Area for each Audit Exit and Special Processing Routine.

The address of the IMSADF II SQL Communication Area is passed to Audit Exits and Special Processing Routines.

When the executed SQL function involves data base I/O the DB2 SQL return code (SQLCODE) and warning conditions (SQLWARN) are also mapped to the IMSADF II SPA work area fields, (SPASQLCD, and SPAWARN). Audit Exits and Special Processing Routines have addressability to these SPA fields.

Note: If native SQL statements are coded, then the SQL Communication Area must be defined in the Audit Exit or Special Processing Routine.

When any of the optional parameters are specified with the SQLHNDLR Call, all parameters that precede it must also be specified. If a Table is not described in the Input Transaction Rule (DBPATH, or TSEGS), then all data base I/O must be performed in the Audit Exit or Special Processing Routine I/O area. When data base I/O is performed in the Audit Exit or Special Processing Routine I/O area the first four parameters are required.

The SQLHNDLR Call returns the following Return Codes in SPARTNCD.

0 SQL statement was executed.

Check SQLCODE, and SQLWARN (SPASQLCD, SPAWARN) for completion status.

4 SQL statement was executed.

DB2 SQL return code is less than zero, or equals one hundred, (SQLCODE<0, or SQLCODE=100). Check SQLCODE (SPASQLCD).

8 SQL statement was not executed.

The SQL function specified by the LABEL parameter is not in the Table Handler Rule for Table ID specified.

EXIT PARAMETER LISTS

- Audit Exit

  The following parameters are passed to an audit exit routine:

  audit field, field definition, op code, audit pcb, comopt, true/false, function, spa, pcb address list, cokey, related field, related field definition, data descriptor, **SQLCA**, reserved, reserved, pcb count, user pcb1, user pcb2, ...

- Special Processing Routine

  The following parameters are passed to a special processing routine:

  spa, comopt, audit pcb, message pcb, user pcb1, user pcb2, ..., iopcb, alternate iopcb, express iopcb, **SQLCA**

RULES DOCUMENTATION

The following RDOC reports will reflect DB2 pertinent data. DB2 Table layout TYPE=TBL and DB2 Column data is reported. The DB2 Table name and Column names are not reported. These can be referenced by their IMSADF II Table and Column ID's.

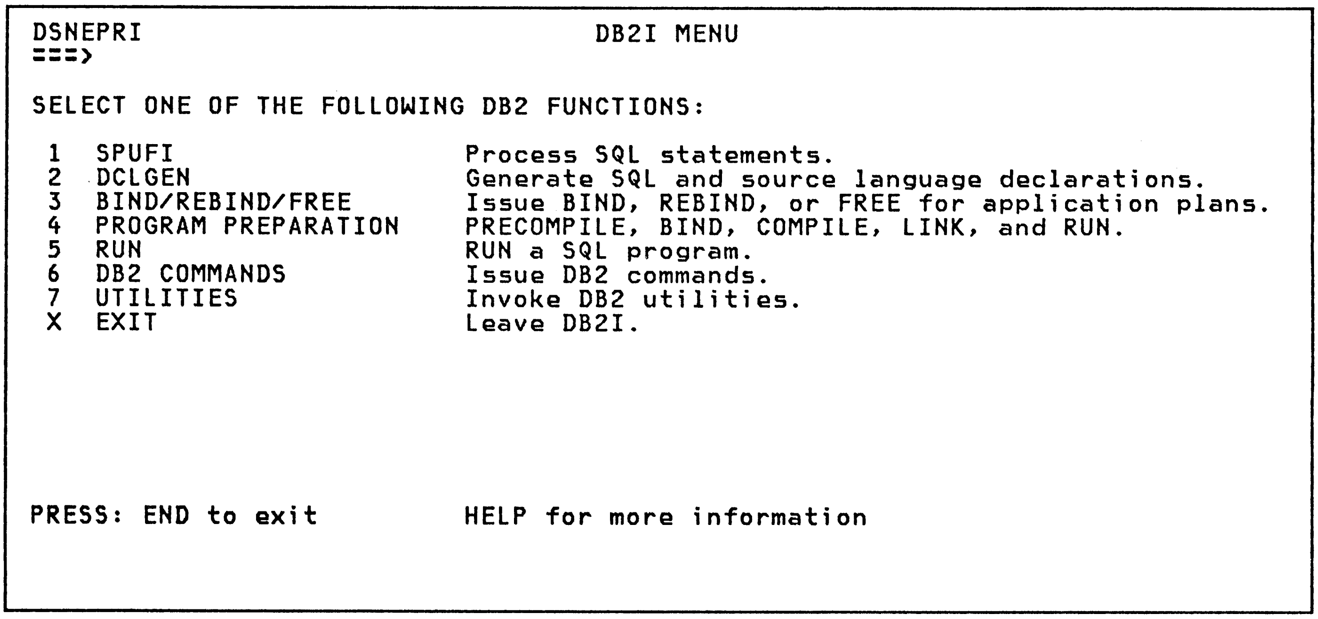
The Table Layout and Table Handler Rules are treated the same as Segment Layout and Segment Handler Rules.

- SR01 Static Rules Summary
- SR01 Segment Layout Summary
- SR02 Static Rules Summary
- SR02 Segment Layout Summary
- SR02 Input Transaction Rule Details
- SR03 Segment Where Used Report
- SR04 Segment Where Used Report
- SR04 Segment Layout Details
- SR05 Segment Where Used Report
- SR05 Segment Layout Details
- SR06 Static Rules Summary
- SR06 Input Transaction Rule Details
- SR07 Composite Load Module Details by System

BIND PROCESS

In the DB2 environment the 'Application Plan' is used for scheduling and authorization checking. The plan is built by a DB2 BIND subcommand issued in the TSO environment to incorporate all related modules, issuing SQL calls. The Application Plan name in the IMS/VS environment is the PSB name. For an IMSADF II application, all Table Handler Rules and Audit Exits and Special Processing Routines with native SQL calls must be included in the BIND process for successful execution.

From the ISPF-MVS primary option menu, the DB2I option is selected. The DB2I MENU panel is then displayed listing DB2 functions.



```
DSNEPRI                                   DB2I MENU
===>

SELECT ONE OF THE FOLLOWING DB2 FUNCTIONS:

 1  SPUFI                      Process SQL statements.
 2  DCLGEN                     Generate SQL and source language declarations.
 3  BIND/REBIND/FREE           Issue BIND, REBIND, or FREE for application plans.
 4  PROGRAM PREPARATION        PRECOMPILE, BIND, COMPILE, LINK, and RUN.
 5  RUN                        RUN a SQL program.
 6  DB2 COMMANDS               Issue DB2 commands.
 7  UTILITIES                  Invoke DB2 utilities.
 X  EXIT                       Leave DB2I.




PRESS: END to exit             HELP for more information
```

Figure 2-12. DB2I MENU Panel

The function BIND/REBIND/FREE is selected and the BIND/REBIND/FREE menu panel is displayed.

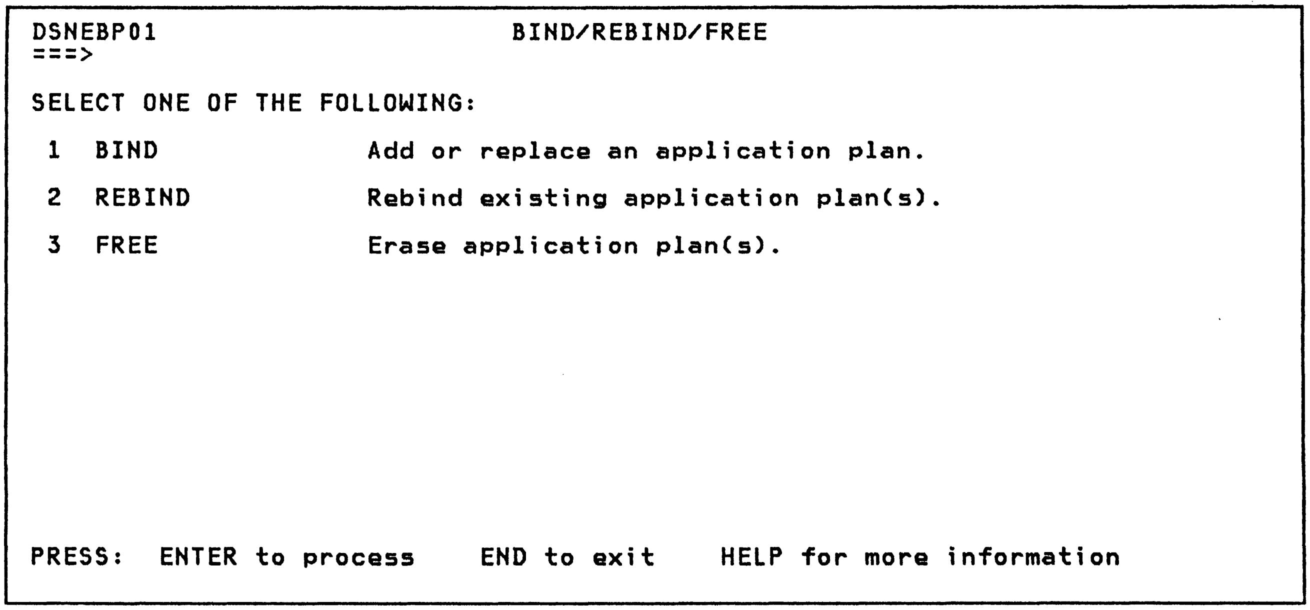

```
DSNEBP01                          BIND/REBIND/FREE
===>

SELECT ONE OF THE FOLLOWING:

 1  BIND             Add or replace an application plan.

 2  REBIND           Rebind existing application plan(s).

 3  FREE             Erase application plan(s).









PRESS:  ENTER to process    END to exit    HELP for more information
```

Figure 2-13. DB2I BIND/REBIND/FREE MENU Panel

When the BIND option is selected, the BIND panel is displayed.

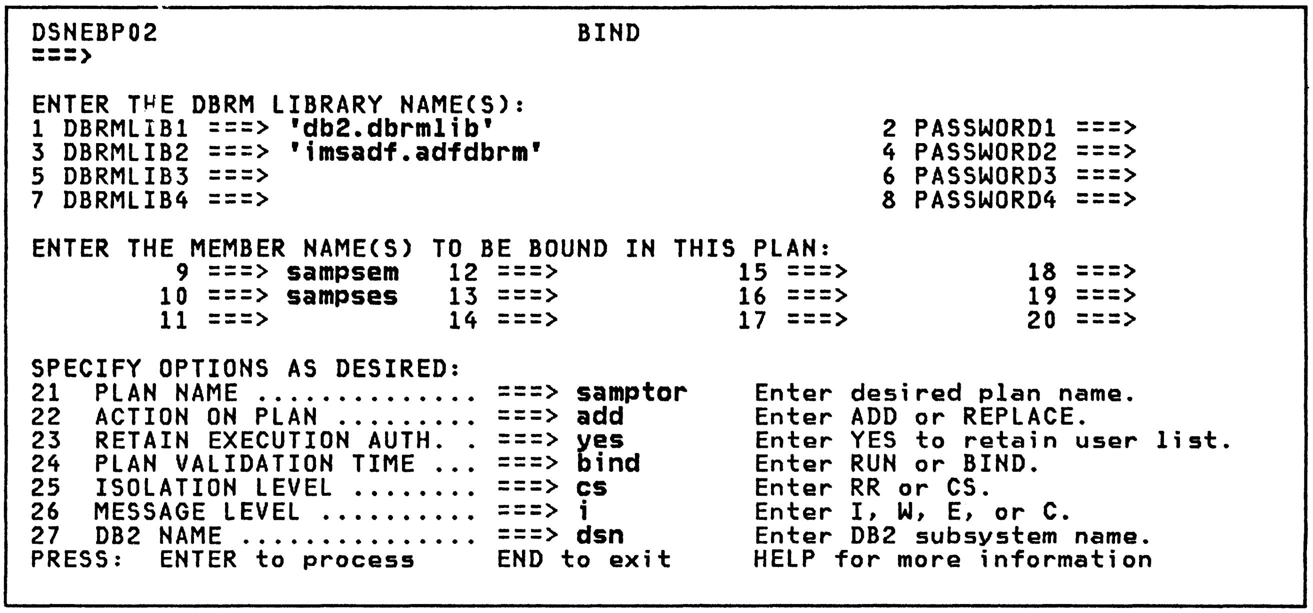

```
DSNEBP02                              BIND
===>

ENTER THE DBRM LIBRARY NAME(S):
1 DBRMLIB1 ===> 'db2.dbrmlib'                           2 PASSWORD1 ===>
3 DBRMLIB2 ===> 'imsadf.adfdbrm'                        4 PASSWORD2 ===>
5 DBRMLIB3 ===>                                         6 PASSWORD3 ===>
7 DBRMLIB4 ===>                                         8 PASSWORD4 ===>

ENTER THE MEMBER NAME(S) TO BE BOUND IN THIS PLAN:
         9 ===> sampsem   12 ===>            15 ===>            18 ===>
        10 ===> sampses   13 ===>            16 ===>            19 ===>
        11 ===>           14 ===>            17 ===>            20 ===>

SPECIFY OPTIONS AS DESIRED:
21  PLAN NAME .............. ===> samptor    Enter desired plan name.
22  ACTION ON PLAN ......... ===> add        Enter ADD or REPLACE.
23  RETAIN EXECUTION AUTH. . ===> yes        Enter YES to retain user list.
24  PLAN VALIDATION TIME ... ===> bind       Enter RUN or BIND.
25  ISOLATION LEVEL ........ ===> cs         Enter RR or CS.
26  MESSAGE LEVEL .......... ===> i          Enter I, W, E, or C.
27  DB2 NAME ............... ===> dsn        Enter DB2 subsystem name.
PRESS:  ENTER to process     END to exit     HELP for more information
```

Figure 2-14. BIND Panel

On this panel enter the required DB2 BIND information:

- Libraries containing the DBRM entries
- DBRM member names to be included in the application plan
- application plan name
- other BIND options

DBRM entries include Table Handler rules, and Audit Exits and Special Processing Routines issuing native SQL calls.

On entry, the DB2 BIND process is executed in TSO foreground.

BIND options are described in the IBM DATABASE 2 Reference under the BIND subcommand.

IMSADF II - DB2 NAMING CONVENTIONS

IMSADF II transactions execute in the IMS/VS online and batch environments, and support conversational and nonconversational processing. To uniquely define IMSADF II applications to IMS/VS, it is necessary to employ a naming convention that uses the IMSADF II major application system identification and the processing type as keys. These keys are established when link-editing the generalized IMSADF II application programs and defining the IMS/VS transaction names.

This naming convention where the IMSADF II transaction driver name is the same as the IMS/VS transaction name and PSB name has been extended to include the DB2 Application Plan name.

The following description provides a summary of the IMSADF II naming conventions:

- IMSADF II member names to be bound in a DB2 Application Plan include:
  - Table Handler Rules - ssssSxx
  - Special Processing Routine (ssssUxx) with native SQL calls
  - audit exits (naming convention is user specified) with native SQL calls

  Note:

ssss	= IMSADF II major application system identification
S	= Table Handler Rule
U	= Special Processing Routine
xx	= Table ID, or Transaction ID

- The DB2 Application Plan name is the same as the IMSADF II transaction driver and IMS/VS PSB name under which this IMSADF II transaction ID is executing.

CHAPTER 3. RGLGEN UTILITY

DESCRIPTION

The IMSADF II utility (RGLGEN), a DB2 TSO application program, is similar to the DB2 DCLGEN in function. It extracts Table and Column definitions from the DB2 catalog in the form of Rules Generator TABLE and COLUMN source statements.

The RGLGEN Utility can be invoked using Interactive Application Development Facility (IADF) panels as a TSO foreground/background program, or by submitting JCL to invoke a TSO background job.

The RGLGEN Utility executes in the TSO foreground or background under the control of the TSO Terminal Monitor Program (TMP). Required runtime parameters are entered on the IADF RGLGEN panel or are defined with the TSO DSN and RUN commands.

Multiple DB2 Table definitions can be processed with a single invocation. Each Table to be processed is defined as a record in a sequential input file. Output for each Table processed is routed to a member of a partitioned data set. Output is also routed to the submitting terminal or SYSPRINT, depending on foreground or background processing. Error, Warning, and Informational messages are also routed to the submitting terminal or SYSPRINT.

DEPENDENCIES

- DB2 and its TSO attachment must be installed and available at execution.
- The DB2 Table (SYSADF.ADFCOLUMNID) must be defined.
- The DB2 BIND process must be invoked to create the RGLGEN application plan.
- DB2 authorization must be granted to:

  GRANT EXECUTE authority for the DB2 application plan.

  GRANT SELECT authority for the SYSIBM.SYSCOLUMNS catalog Table.

  GRANT SELECT authority for the SYSIBM.SYSINDEXES catalog Table.

  GRANT SELECT authority for the SYSIBM.SYSKEYS catalog Table.

  GRANT SELECT, UPDATE and INSERT authority for the SYSADF.ADFCOLUMNID Table.
- When invoked using IADF, a 4096K TSO region may be required.

SYSIBM.SYSCOLUMNS TABLE

The RGLGEN Utility extracts Table and Column definitions from the DB2 SYSIBM.SYSCOLUMNS Catalog Table. The SYSIBM.SYSCOLUMNS Table contains one row for every column of each table and view (including the columns of the DB2 catalog tables), defined to the DB2 system.

The following figure defines the view of the SYSIBM.SYSCOLUMNS Table that is interrogated by the RGLGEN Utility.

Column Name	**Data Type**	**Description**
NAME	VARCHAR(18)	Name of the column.
TBNAME	VARCHAR(18)	Name of the table or view which contains the column.
TBCREATOR	CHAR(8)	Authorization ID of the creator of the table or view.
COLNO	SMALLINT	Ordinal number of the column in the table or view.
COLTYPE	CHAR(8)	Type of column: 'INTEGER'=large integer 'SMALLINT'=small integer 'FLOAT'=floating-point 'CHAR'=fixed length character string 'VARCHAR'=varying length character string 'LONGVAR'=varying length character string 'DECIMAL'=decimal
LENGTH	SMALLINT	The length attribute of the column; or, in the case of a decimal column, its precision. The number does not include the internal prefixes used to record actual length and null state where applicable. 'INTEGER'=4 'SMALLINT'=2 'FLOAT'=8 'CHAR'=length of string 'VARCHAR'=maximum length of string 'LONGVAR'=maximum length of string 'DECIMAL'=precision of number
SCALE	SMALLINT	Scale of decimal data.
NULLS	CHAR(1)	Indicates whether the column can contain null values. 'N'=no 'Y'=yes
UPDATES	CHAR(1)	Indicates whether the column can be updated for reasons other than being the column of a read-only view. 'N'=no 'Y'=yes

Figure 3-1. SYSIBM.SYSCOLUMNS TABLE

SYSIBM.SYSINDEXES TABLE

The RGLGEN Utility also accesses the SYSIBM.SYSINDEXES Table to obtain information regarding the unique indexes which have been created for the DB2 table being processed. This information, in combination with the information obtained from the SYSIBM.SYSKEYS Table, is used to determine which columns should have the KEY=YES parameter in the RGLGENO Rules Generator source produced by the RGLGEN Utility.

The following figure defines the view of the SYSIBM.SYSINDEXES Table that is interrogated by the RGLGEN Utility.

Column Name	Data Type	Description
NAME	VARCHAR(18)	Name of the index.
CREATOR	CHAR(8)	Authorization ID of the creator of the index.
TBNAME	VARCHAR(18)	Name of the table on which the index is defined.
TBCREATOR	CHAR(8)	Authorization ID of the creator of the table.
UNIQUERULE	CHAR(1)	Whether the index is unique (Duplicates allowed, U=Unique)
COLCOUNT	SMALLINT	The number of columns in the key.

Figure 3-2. SYSIBM.SYSINDEXES TABLE

SYSIBM.SYSKEYS TABLE

The SYSIBM.SYSINDEXES Table is used to obtain the name of the unique indexes which have been created for the DB2 table being processed. The SYSIBM.SYSKEYS Table is used to obtain the names of the key columns within the index. All key columns within all unique indexes will be given the KEY=YES parameter in the RGLGENO Rules Generator source produced by the RGLGEN Utility.

The following figure defines the view of the SYSIBM.SYSKEYS Table that is interrogated by the RGLGEN Utility.

Column Name	Data Type	Description
IXNAME	VARCHAR(18)	Name of the index.
IXCREATOR	CHAR(8)	Authorization ID of the creator of the index.
COLNAME	VARCHAR(18)	Name of the column of the key.
COLNO	SMALLINT	Numerical position of the column in the row.

Figure 3-3. SYSIBM.SYSKEYS TABLE

SYSADF.ADFCOLUMNID TABLE

When the RGLGEN Utility is invoked an optional DB2 Table, SYSADF.ADFCOLUMNID, can be used to define IMSADF II Column IDs and to control the relationship between the IMSADF II Column ID and the DB2 Column names.

The RGLGEN Utility contains SQL statements that reference the SYSADF.ADFCOLUMNID. Therefore, even if this Table is not used at execution, it must be defined to the DB2 system before a valid DB2 Application Plan can be created.

The Column definitions of the SYSADF.ADFCOLUMNID Table are:

Column Name	**Data Type**	**Description**
COLNAME	VARCHAR(18)	Name of the column.
TBNAME	VARCHAR(18)	Name of the table or view which contains the column.
TBCREATOR	CHAR(8)	Authorization ID of the creator of the table or view.
ADFCOLID	CHAR(4)	IMSADF II Column ID.

Figure 3-4. SYSADF.ADFCOLUMNID TABLE

When Rules Generator source is extracted from the DB2 catalog by the RGLGEN Utility the four-character IMSADF II Column ID is defined for each Column name.

There are two methods available to RGLGEN Utility for defining the Column ID.

1. The RGLGEN Utility maintains the SYSADF.ADFCOLUMNID Table. For each DB2 Column processed the RGLGEN Utility will interrogate the SYSADF.ADFCOLUMNID Table to determine if this DB2 Column has been previously defined.

   If the DB2 Column is found then the associated IMSADF II Column ID is used.

   If the DB2 Column is not found then the following occurs:

   - A master row contained in the Table is SELECTed. This row contains the last used Column ID.
   - The value of this last used Column ID is incremented by one.
   - The Table is interrogated to determine that this Column ID is unique.

     If the Column ID is unique it is associated with the Column name.

     If the Column ID is not unique it is incremented by one until a unique Column ID is obtained.
   - Once a unique Column ID is obtained the Rules Generator source is created for the DB2 Column. A row is INSERTed into the Table recording the relationship between the DB2 Column name and the IMSADF II Column ID.

     Note: The master row is updated after all requested DB2 Tables have been processed.

     If the master row is not found one is INSERTed. The first IMSADF II Column ID used is 0001.

IMSADF II Column ID's are defined in the following order:

All numeric:	0001-9999
All alphabetic:	AAAA-ZZZZ and @@@@, ####, $$$$
Alphanumeric:	unique values not previously used

2. The second method available to the RGLGEN Utility does not use the SYSADF.ADFCOLUMNID Table.

   Each time the RGLGEN Utility is invoked, assign 0001 as the first Column ID, and increment each subsequent Column ID by one.

The advantages of the first method are:

- Eliminates the need for the user to define their own Column ID's.
- Defines unique IMSADF II Column ID - DB2 Column name relationships within a DB2 system.
- If a DB2 Table is reorganized, or if a View is defined, the Column ID's associated with the Column names do not change. This is especially important because it implies that IMSADF II audits, specified by Column ID, do not have to be respecified to reflect the new Column ID's. Also, previously defined Rules Generator Column merge statements still reflect the same IMSADF II Column ID/DB2 Column name.

IMSADF II ADMINISTRATOR CONTROL

The SYSADF.ADFCOLUMNID Table can be accessed outside the control of the RGLGEN Utility, (for example, SPUFI).

This allows an IMSADF II administrator to:

- INSERT new rows.

  Unique combinations of Column name, Table name and Table creator can be added. The associated ADFCOLID is user defined.

  Subsequent invocations of the RGLGEN Utility that process this row will use the user specified ADFCOLID.

  When the RGLGEN Utility INSERTs new rows into the Table it only enters data into the Column name and ADFCOLID. The Table name and Table creator Columns are left blank (that is, the Column name is unqualified). These two Columns can be used by the IMSADF II administrator to further qualify Column names that appear in multiple Tables/Views. The Rules Generator SQLNAME operand in the output is qualified or unqualified based on the contents of the Table name Column.

- UPDATE existing rows.

  Change the RGLGEN Utility defined ADFCOLID to a user specified ID.

  Qualify the Column name by defining the Table name and/or Table creator Columns.

- DELETE existing rows.

- UPDATE or INSERT the master row.

  The master row contains the last used ADFCOLID. If the IMSADF II administrator changes the last used ADFCOLID, then subsequent invocations of the RGLGEN Utility will define ADFCOLIDs from this new starting position. The key for the master row is COLNAME='##################'.

PROGRAM PREPARATION

Before executing the RGLGEN Utility the following steps MUST be completed:

- DB2 and IMSADF II installation
- Install SYSADF.ADFCOLUMNID Table
- BIND process

DB2 AND IMSADF II INSTALLATION

- DB2 installation is defined in the IBM DATABASE 2 Installation Guide.
- Installing the RGLGEN Utility is an optional part of the standard IMSADF II installation process.

  Note: Refer to the IMS Application Development Facility II Version 2 Release 2 Installation Guide for details on installing and customizing IMSADF II.

  The RGLGEN Utility is supplied in load module format. It contains static SQL statements that have been processed by the DB2 precompiler. The corresponding DBRM is also supplied.

  A copy of the DB2 TSO LANGUAGE interface module (DSNELI) must be available for the RGLGEN Utility link-edit.

  Prior to executing the IMSADF II installation link-edit step the library referenced by the ALOAD DD statement may have to be changed. This library must reference the SMP DLIB library where the DB2 TSO language interface module (DSNELI) resides.

  Whether you install IMSADF II using the IADF installation dialogs, or the Batch method, an SMP system library (IMSADF.ADFDBRM) is created when the product function is applied.

  This library contains the supplied DBRM for the RGLGEN Utility (MFC1Y25D), and can also be used to hold DBRMs associated with IMSADF II DB2 transactions.

INSTALL SYSADF.ADFCOLUMNID TABLE

This Table MUST be defined to your DB2 system if the RGLGEN Utility is used.

The IMSADF II distributed library (IMSADF.ADFMAC) contains a member (ADFDB2TC) that contains Table CREATE and Index CREATE source statements for the SYSADF.ADFCOLUMNID Table.

The following is a listing of the DB2 Table definition statements for the SYSADF.ADFCOLUMNID Table.

```
  CREATE TABLE SYSADF.ADFCOLUMNID
            (COLNAME                      VARCHAR(18) NOT NULL,
             TBNAME                       VARCHAR(18) NOT NULL,
             TBCREATOR                    CHAR(8)     NOT NULL,
             ADFCOLID                     CHAR(4)     NOT NULL)
         IN DATABASE XXXXXXXX;

  CREATE UNIQUE INDEX SYSADF.ADFCOL1
         ON SYSADF.ADFCOLUMNID
            (COLNAME,TBNAME,TBCREATOR);
```

This source must be modified and processed by the DB2 SPUFI function.

Copy this member to your own library before making any modifications.

The only modification required is to define a valid DB2 Data Base or Table Space, (Replace the XXXXXXXX).

The modified source should then be used as input to the DB2 SPUFI function.

BIND PROCESS

The DB2 Application Plan associated with the RGLGEN Utility must be built.

The Application Plan name is user specified. However, the DBRM member name specified is always MFC1Y25D.

The following is an example of what should be specified on the DB2I BIND Panel for the RGLGEN Utility Application Plan. The name of the Application Plan in this illustration is RGLGEN.

```
DSNEBP02                          BIND
===>

ENTER THE DBRM LIBRARY NAME(S):
1 DBRMLIB1 ===> 'imsadf.adfbrm'                          2 PASSWORD1 ===>
3 DBRMLIB2 ===>                                          4 PASSWORD2 ===>
5 DBRMLIB3 ===>                                          6 PASSWORD3 ===>
7 DBRMLIB4 ===>                                          8 PASSWORD4 ===>

ENTER THE MEMBER NAME(S) TO BE BOUND IN THIS PLAN:
        9 ===> mfc1y25d  12 ===>        15 ===>              18 ===>
       10 ===>           13 ===>        16 ===>              19 ===>
       11 ===>           14 ===>        17 ===>              20 ===>

SPECIFY OPTIONS AS DESIRED:
21  PLAN NAME ............. ===> rglgen    Enter desired plan name.
22  ACTION ON PLAN ......... ===> add      Enter ADD or REPLACE.
23  RETAIN EXECUTION AUTH. . ===> yes      Enter YES to retain user list.
24  PLAN VALIDATION TIME ... ===> bind     Enter RUN or BIND.
25  ISOLATION LEVEL ........ ===> cs       Enter RR or CS.
26  MESSAGE LEVEL .......... ===> i        Enter I, W, E, or C.
27  DB2 NAME ............... ===> dsn      Enter DB2 subsystem name.
PRESS:  ENTER to process     END to exit   HELP for more information
```

Figure 3-5. DB2I BIND Panel, RGLGEN Utility

TSO PROGRAM PARAMETERS

In order to execute the RGLGEN Utility in the TSO environment, data sets must be defined/allocated and TSO commands must be defined.

If the IADF RGLGEN panels are used to invoke the RGLGEN Utility all of the JCL and TSO commands required for execution as a TSO foreground/background job are predefined.

The following example defines the JCL required to invoke the RGLGEN Utility as a TSO background (batch) job.

```
//          JOB
//RGLGEN EXEC PGM=IKJEFT01
//STEPLIB   DD DSN=IMSADF.INSTALL.ADFLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//RGLGENI   DD DSN=TSOUSER.ADFDB2.RGLGENIN,DISP=SHR
//RGLGENO   DD DSN=TSOUSER.ADFDB2.TABH,DISP=SHR
//RGLGENT   DD SYSOUT=(X,,SYS)
//SYSPRINT DD SYSOUT=(X,,SYS),DCB=(LRECL=133,BLKSIZE=133,RECFM=FB)
//SYSUDUMP DD SYSOUT=*
//ADFDUMP   DD SYSOUT=(X,,SYS),DCB=(RECFM=FBA,LRECL=121,BLKSIZE=121,
//             BUFNO=1),OUTLIM=10000
//SYSOUT    DD SYSOUT=(X,,SYS)
//REPORT    DD SYSOUT=*
//SYSTSIN   DD *
 DSN SYSTEM(DSN)
 RUN PROGRAM (MFC1Y25) -
     PLAN (RGLGEN) -
     LIB ('IMSADF.INSTALL.ADFLOAD') -
     PARMS ('B,Y')
 END
/*
```

The required TSO program parameters are:

IKJEFT01 Execute the TSO Terminal Monitor Program (TMP).

STEPLIB Optional DD statement.

This DD statement defines the library that contains the IMSADF II trace modules, MFC1FLLM and MFC1V40.

If you wish to invoke the IMSADF II trace facility while executing the RGLGEN Utility this DD must be specified. When this DD statement is not specified IMSADF II tracing is disabled.

RGLGENI This DD statement defines the sequential card image (80-character) input file. Refer to "Parameters" on page 3-10.

RGLGENO This DD statement defines the output partitioned data set. The Rules Generator source created for each Table processed by the RGLGEN Utility is stored in a separate member of this data set.

The member name for each Table is passed to the RGLGEN Utility as an input parameter. Refer to "Parameters" on page 3-10.

RGLGENT This DD statement defines output routed to the terminal that invoked the RGLGEN Utility in the TSO foreground. The output contains the Rules Generator source statements and all RGLGEN Utility messages.

SYSPRINT This DD statement defines output routed to the printer. This DD is used if the RGLGEN Utility is invoked in the TSO background. The output contains the Rules Generator source statements and all RGLGEN Utility messages.

ADFDUMP Optional DD statement. This DD statement defines the IMSADF II trace facility output. This DD is only required when IMSADF II trace is requested.

SYSTSIN This DD statement defines the TSO TMP input.

DSN The TSO command that connects the TSO job to the DB2 SYSTEM.

RUN The DSN subcommand used to invoke a TSO application program containing SQL statements.

PARMS defines the runtime parameters passed to the TSO application program.

Two runtime parameters are passed to the RGLGEN Utility.

1. Foreground/background indicator - (F,B) a one character parameter defining the TSO application program output routing.

2. SYSADF.ADFCOLUMNID Table indicator - (Y,N) a one-character parameter defining the use of the optional SYSADF.ADFCOLUMNID Table, (Yes,No).

These two parameters are separated by a comma, and enclosed in quotes.

END The DSN subcommand used to disconnect the TSO job from DB2

INPUT

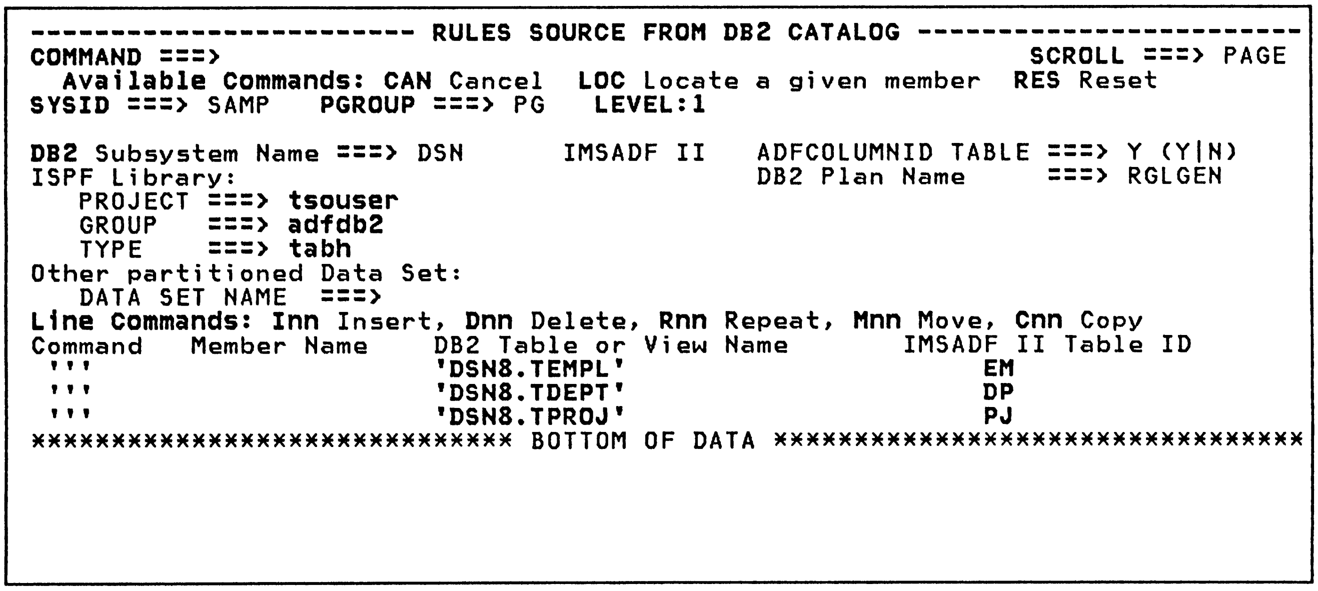
The input parameters required at execution are passed to the RGLGEN Utility as 80-character card image records defined in the sequential input file referenced by the RGLGENI DD statement.

If the RGLGEN Utility is invoked by submitting a TSO background job the RGLGENI DD statement can reference an existing sequential data set, or it can be processed as an inline data file.

If the IADF RGLGEN panels are used to invoke the RGLGEN Utility then the input parameters are defined on the RGLGEN GENERATION panel.

The following is an example of what should be specified on the IADF RGLGEN GENERATION panel.

In this example three Tables are defined. The output will be written to three members in the 'TSOUSER.ADFDB2.TABH' partitioned data set. The SYSADF.ADFCOLUMNID Table is used to create IMSADF II Column ID's.



```
------------------------ RULES SOURCE FROM DB2 CATALOG ------------------------
COMMAND ===>                                                   SCROLL ===> PAGE
  Available Commands: CAN Cancel  LOC Locate a given member  RES Reset
SYSID ===> SAMP   PGROUP ===> PG   LEVEL:1

DB2 Subsystem Name ===> DSN      IMSADF II   ADFCOLUMNID TABLE ===> Y (Y|N)
ISPF Library:                                DB2 Plan Name     ===> RGLGEN
   PROJECT ===> tsouser
   GROUP   ===> adfdb2
   TYPE    ===> tabh
Other partitioned Data Set:
   DATA SET NAME  ===>
Line Commands: Inn Insert, Dnn Delete, Rnn Repeat, Mnn Move, Cnn Copy
Command   Member Name    DB2 Table or View Name       IMSADF II Table ID
 '''                     'DSN8.TEMPL'                           EM
 '''                     'DSN8.TDEPT'                           DP
 '''                     'DSN8.TPROJ'                           PJ
****************************** BOTTOM OF DATA *********************************
```

Figure 3-6. Rules Source from DB2 Catalog Panel

The Rules Source from DB2 Catalog panel parameters are:

SYSID	The four-character IMSADF II System ID.
PGROUP	The two-character IMSADF II Project/Group.
DB2 SUBSYSTEM NAME	The DB2 Subsystem to which the TSO job should be connected.

DB2 PLAN NAME

The DB2 Application Plan Name specified for the BIND of the RGLGEN Utility for this DB2 sub system using the IMSADF II supplied DBRM (MFC1Y25D).

A default DB2 Application Plan Name of RGLGEN is displayed.

ADFCOLUMNID TABLE

Y - process with the SYSADF.ADFCOLUMNID Table.

N - do not use the SYSADF.ADFCOLUMNID Table.

ISPF Library

Output partitioned data set (RGLGENO DD).

Command

Available line commands.

Member Name

The Rules Generator source statements for this Table are routed to this member.

When using the IADF panels this input parameter is normally left blank. It is only required for non-standard member names.

Standard member names have the following format, and are built automatically by IADF when the Member Name is blank.

ssssTBxx

Where:

```
ssss = Current IMSADF II system ID
  TB = constant
  xx = Current Table ID
```

Table Name

The qualified name of the DB2 Table/View to be processed.

If the Table Name is in quotes, IADF assumes it is qualified and passes it as it is.

If the Table Name is not in quotes, IADF assumes it is not qualified, appends the current TSO Userid and puts the entire name in quotes.

If the Table Name input parameter is not qualified, the RGLGEN Utility will terminate processing with an error message.

Note: Refer to the IBM DATABASE 2 Reference manual for details on DB2 naming conventions.

Table ID

The IMSADF II Table ID to be associated with this DB2 Table/View.

PARAMETERS

The RGLGEN Utility treats all input parameters as keywords. Each keyword must be followed by an equal sign and its value. A comma is the only valid delimiter between keywords. Blanks are not valid, except after the last parameter value. There are no allowable abbreviations for the keywords.

The input parameters are as follows:

MEMBER

Table/View Column definitions written to this output member name.

One to eight characters in length. The first character must be alphabetic.

NAME

The qualified name of the DB2 Table/View to be processed.

A qualified DB2 Table name has the following format:

```
authorization-id.long-identifier
```

The maximum length of the qualified Table name is 27 characters.

The qualified Table name must be enclosed in single quotes.

Note: Refer to the IBM DATABASE 2 Reference manual for details on DB2 naming conventions.

ID The two-character IMSADF II Table ID to be associated with this DB2 Table/View.

Whether the input parameters are defined through IADF or built by the user, the input records should have the following format:

```
MEMBER=SAMPTBEM,NAME='DSN8.TEMPL',ID=EM
MEMBER=SAMPTBDP,NAME='DSN8.TDEPT',ID=DP
MEMBER=SAMPTBPJ,NAME='DSN8.TPROJ',ID=PJ
```

Note: These input records define the three Tables requested on the previous RGLGEN GENERATION panel.

OUTPUT

The RGLGEN Utility queries the DB2 catalog for the specified Table/View Column definitions and creates the appropriate Rules Generator operands to define the Table layout.

The Rules Generator source statements are:

```
TABLE   ID=,TYPE=TBL,SQLNAME=,SQLIND=
COLUMN  ID=0001,SQLNAME=,SNAME=,
        TYPE=,LENGTH=,DEC=,SQLNULL=,SQLUPD=
```

The SQLNAME operand on the TABLE statement contains the qualified Table name.

The SQLNAME operand on the COLUMN statement contains the qualified form of the Column name if the optional SYSADF.ADFCOLUMNID Table was used, and the Table name Column was not blank. Otherwise, the SQLNAME operand is unqualified.

Note: Refer to "Rules Generator" on page 2-1 for details on the Rules Generator operands.

The output for each Table/View processed by the RGLGEN Utility is written to a separate member of the partitioned data set referenced by the RGLGENO DD statement. Eighty-character card image records are written to this data set. If the MEMBER exists it is replaced. If the member does not exist it is added.

If the RGLGEN Utility is executing in the TSO foreground then a copy of the Rules Generator source statements as well as all messages are routed back to the submitting terminal. This output routing is controlled by the RGLGENT DD statement.

If the RGLGEN Utility is executing in the TSO background then a copy of the Rules Generator source statements as well as all messages are routed to the printer. This output routing is controlled by the SYSPRINT DD statement.

MESSAGES

The RGLGEN Utility generates Informational, Warning, and Error messages.

Informational	- For each DB2 table successfully processed
Warning	- Processing continues
Error	- Processing terminates for current table

If the RGLGEN Utility is executing in the TSO foreground, messages are routed to the submitting terminal.

If the RGLGEN Utility is executing in the TSO background, messages are routed to the SYSPRINT DD.

RETURN CODES

The RGLGEN Utility issues the following Return codes:

0	Processing successful for all table(s), informational message(s).
4	Processing continues for current table, warning message(s).
8	Processing terminated for current table, error message(s).

ABNORMAL TERMINATION CODES

The RGLGEN Utility does not generate any abnormal termination codes.

However, the TSO Terminal Monitor Program invokes the RGLGEN Utility, and controls the access to DB2. Therefore, the RGLGEN Utility is subject to TSO and DB2 Abnormal Termination Codes.

ERROR, WARNING, AND INFORMATIONAL MESSAGES

This section lists all messages generated by the RGLGEN Utility. Associated with each message is a more detailed explanation. If applicable, there is a brief description of the system action, and a suggested user response.

Each message generated by the RGLGEN Utility is preceded by a message identification header of the following format:

ADFY9nn t

ADF	Distinguishes this message as an IMSADF II message.
Y	The IMSADF II component code. Y - Utilities message.
9	Range, of all RGLGEN Utility messages is from 900 to 999.
nn	Message sequence number.
t	Identifies the type of message, as follows:

I	Information message. (Return Code=0)
W	Warning message. Execution continues. (Return code=4)
E	Error message. Processing terminated for current Table. (Return code=8)

Each invocation of the RGLGEN Utility may generate multiple messages. However, the return code always reflects the severest message issued.

MESSAGE PARAMETERS

RGLGEN Utility input parameters, DB2 status codes, and other dynamic information, are embedded in the message text.

The message text shown here displays these dynamic message parameters in lower case. During execution, the RGLGEN Utility substitutes current values into the message text.

The dynamic message parameters are:

- table.name DB2 qualified Table name.
- table.id IMSADF II Table ID.

•	membername	Output Member Name.
•	sqlcode	DB2 return code.
•	sqlwarn	DB2 warning indicators.
•	invalid.parm	Invalid parameter.
•	ddname	RGLGEN Utility DD statement.
•	function	Function being invoked.
•	offset	Starting position of the invalid data.
•	string	Ten bytes of invalid data from the offset.
•	column.name	DB2 unqualified Column name.
•	creator	DB2 authorization-id.
•	adfcolumnid	IMSADF II Column ID.

MESSAGE TEXT

ADFY900 I IMSADF II RGLGEN UTILITY SUCCESSFUL EXECUTION, TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: RGLGEN Utility processing was successfully completed with a return code of zero for this Table.

System Action: None

Operator Response: None

ADFY901 E SQL PROCESSING: SQLCODE=sqlcode, SQLWARN(0-7)=sqlwarn, TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: A non-zero return code was returned by DB2.

If SQLCODE is less than zero this is an error message.

If SQLCODE is greater than zero this is a warning message.

System Action: If the SQLCODE is less than zero the RGLGEN Utility terminates processing for this Table, otherwise processing continues.

Operator Response: Look up the SQLCODE in the _IBM DATABASE 2 Messages and Codes_ manual and take the appropriate corrective action. If additional information is required to resolve the error (that is, full SQL Communication Area), invoke the IMSADF II Trace facility for the RGLGEN Utility.

ADFY902 E SQL PROCESSING THE OPTIONAL SYSADF.ADFCOLUMNID TABLE: SQLCODE=sqlcode, SQLWARN(0-7)=sqlwarn, TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: A non-zero return code was returned by DB2.

If SQLCODE is less than zero this is an error message.

If SQLCODE is greater than zero this is a warning message.

System Action: If the SQLCODE is less than zero the RGLGEN Utility terminates processing for this Table, otherwise processing continues.

Operator Response: Look up the SQLCODE in the _IBM DATABASE 2 Messages and Codes_ manual and take the appropriate corrective action. If additional information is required to resolve the error (that is, full SQL Communication Area), invoke the IMSADF II Trace facility for the RGLGEN Utility.

ADFY903 E INVALID TSO FOREGROUND/BACKGROUND PARAMETER - invalid.parm, VALID VALUES ARE F OR B

Explanation: The TSO foreground/background runtime parameter is invalid.

System Action: The RGLGEN Utility terminates processing.

Operator Response: The TSO FOREGROUND/BACKGROUND parameter is a runtime parameter that is specified in the DSN subcommand RUN. It is passed to the RGLGEN Utility when the TSO Terminal Monitor Program passes it control.

If the RGLGEN Utility was invoked using IADF this parameter is built by IADF. If this is the case and no modifications have been made to IADF, and the error persist, notify your IBM representative.

If the RGLGEN Utility was invoked using your own batch JCL stream, or you changed the IADF output, correct the PARMS keyword on the TSO DSN subcommand RUN and resubmit the RGLGEN Utility.

Refer to the TSO PROGRAM PARAMETERS section in this guide for additional information on runtime PARAMETERS.

ADFY904 E INVALID OPTIONAL SYSADF.ADFCOLUMNID TABLE PARAMETER _ invalid.parm, VALID VALUES ARE Y OR N

Explanation: The optional SYSADF.ADFCOLUMNID Table parameter is invalid.

System Action: The RGLGEN Utility terminates processing.

Operator Response: The optional SYSADF.ADFCOLUMNID Table parameter is a runtime parameter that is specified in the DSN subcommand RUN. It is passed to the RGLGEN Utility when the TSO Terminal Monitor Program passes it control.

If the RGLGEN Utility was invoked using IADF this parameter is verified by IADF. If this is the case and no modifications have been made to IADF, and the error persist, notify your IBM representative.

If the RGLGEN Utility was invoked using your own batch JCL stream, or you changed the IADF output, correct the PARMS keyword on the TSO DSN subcommand RUN and resubmit the RGLGEN Utility.

Refer to the TSO PROGRAM PARAMETERS section in this guide for additional information on runtime PARAMETERS.

ADFY905 E ddname DD IS NOT IN JOBSTEP OR IS DEFINED AS DD DUMMY

Explanation: One or more of the required RGLGEN Utility DD statements is missing or incorrectly defined, (RGLGENI, RGLGENT, SYSPRINT).

System Action: The RGLGEN Utility terminates processing.

Operator Response: If the RGLGEN Utility was invoked using IADF, define the required DD statements to IADF and reinvoke the RGLGEN Utility.

If the RGLGEN Utility was invoked using your own batch JCL stream, add the required DD statements to the JCL and resubmit the RGLGEN Utility.

Refer to the TSO PROGRAM PARAMETERS section in this guide for additional information on required DD statements.

ADFY906 E ddname DD function ERROR ENCOUNTERED BY MFC1V48. NOTIFY IMSADF II ADMINISTRATOR. TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The IMSADF II module MFC1V48 is used to control output being written to members in the RGLGENO DD statement.

System Action: The RGLGEN Utility terminates processing.

Operator Response: The MFC1V48 module controls four functions associated with the RGLGENO DD statement.

DEFINE	Verify that the data set specified by the RGLGENO DD statement is a valid partitioned data set.
OPEN	Open the current member name as output for this Table.
WRITE	Write the DB2 Table/View Column definitions.
CLOSE	Close the current member name and add it to the partitioned data set specified by the RGLGENO DD statement.

If the RGLGEN Utility was invoked using IADF, define or correct the RGLGENO DD statement to IADF and reinvoke the RGLGEN Utility.

If the RGLGEN Utility was invoked using your own batch JCL stream, add a valid RGLGENO DD statement and resubmit the RGLGEN Utility.

If the error persist notify your IBM representative.

Refer to the TSO PROGRAM PARAMETERS section in this guide for additional information on the RGLGENO DD statement.

ADFY907 E INVALID OR MISSING INPUT PARAMETER(S) AT OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: One or more of the three input parameters required by the RGLGEN Utility is invalid or missing.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Review the required input parameters, correct the error(s), and resubmit the RGLGEN Utility.

Refer to the PARAMETERS section in this guide for additional information on input parameters.

ADFY908 E INVALID PARAMETER KEYWORD AT OFFSET-offset, STRING VALUE: string: VALID KEYWORDS ARE: NAME=, ID=, MEMBER= CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: One or more of the three input parameter keywords required by the RGLGEN Utility is invalid.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Review the required input parameter keywords, correct the error(s), and resubmit the RGLGEN Utility.

Refer to the PARAMETERS section in this guide for additional information on input parameter keywords.

ADFY909 E DUPLICATE invalid.parm INPUT PARAMETER AT OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The specified input parameter is a duplicate.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Eliminate the duplicate input parameter.

ADFY910 E invalid.parm INPUT PARAMETER TRUNCATED OR TOO LONG AT OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The specified input parameter value exceeds its maximum allowable length or the value has been truncated.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Correct the input parameter value.

ADFY911 E invalid.parm INPUT PARAMETER IS TOO LONG OR TOO SHORT AT OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The specified input parameter value exceeds its maximum allowable length or is less than minimum allowable length.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Correct the input parameter value.

ADFY912 E invalid.parm PORTION OF THE TABLE NAME INPUT PARAMETER IS MISSING OR TOO LONG AT OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: Either the authorization-id or the unqualified Table name portion of the qualified Table name is missing or exceeds the maximum allowable length.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Correct the Table Name input parameter value.

ADFY913 E INVALID TABLE ID INPUT PARAMETER AT OFFSET-offset, STRING VALUE: string: VALID TABLE ID CHARACTERS ARE ALPHABETIC, NUMERIC, OR @, $, #

Explanation: The specified two character IMSADF II Table ID is not alphanumeric.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Correct the Table ID input parameter value.

ADFY914 E COMMA MUST BE FOLLOWED BY A NON-BLANK CHARACTER, OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: A blank character was encountered after a valid delimiter (comma).

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Remove blank characters from the input parameters. Do not put a comma after the last input parameter.

ADFY915 E IMBEDDED BLANKS NOT ALLOWED IN INPUT PARAMETERS, OFFSET-offset, STRING VALUE: string: CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: An imbedded blank was encountered in the input parameters.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Remove blank characters from the input parameters.

ADFY916 E invalid.parm IS A REQUIRED INPUT PARAMETER. INPUT KEYWORDS ARE NAME=, ID=, MEMBER=. CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The specified input parameter was not found.

System Action: The RGLGEN Utility terminates processing for this input record.

Operator Response: Define all required input parameters.

ADFY917 E NO COLUMNS FOUND FOR SPECIFIED TABLE. SQLCODE=sqlcode, SQLWARN(0-7)=sqlwarn, CURRENT VALUES: TABLE NAME=table.name, TABLE ID=table.id, MEMBER=membername

Explanation: The DB2 return code is +100. There were no rows found in the DB2 SYSIBM.SYSCOLUMNS Table representing Columns in the specified Table.

System Action: The RGLGEN Utility terminates processing for this Table.

Operator Response: Verify that the qualified Table name is valid and that it is defined in the DB2 catalog SYSIBM.SYSCOLUMNS Table for the current DB2 sub-system.

ADFY918 E INVALID COLUMN DATA TYPE: function. THE IMSADF II COLUMN TYPE AND LENGTH OPERANDS HAVE BEEN SET TO QUESTION MARKS.

Explanation: A DB2 data type not recognized by the RGLGEN Utility has been encountered.

System Action: The RGLGEN Utility continues processing for this Table. However, the Rules Generator TYPE and LENGTH operands have been set to question marks.

Operator Response: Verify that the DB2 data type is valid. If it is and this error persist contact your IBM representative.

ADFY919 E INVALID COLUMN LENGTH: function. THE IMSADF II COLUMN LENGTH OPERAND HAS BEEN SET TO QUESTION MARKS.

Explanation: A DB2 data type has been specified with a data length attribute that exceeds the allowable IMSADF II maximum length.

System Action: The RGLGEN Utility continues processing for this Table. However, the Rules Generator LENGTH operand has been set to question marks.

Operator Response: Verify that the DB2 data length is valid and that it is within the allowable IMSADF II maximum length for that data type.

If the DB2 length exceeds the allowable IMSADF II maximum then this Column cannot be processed directly by standard IMSADF II functions.

ADFY920 E INVALID COLUMN DECIMAL SCALE: function. THE IMSADF II COLUMN DECIMAL OPERAND HAS BEEN SET TO QUESTION MARKS.

Explanation: A DB2 data type decimal has been specified that has a SCALE value greater than the IMSADF II allowable maximum of 13.

System Action: The RGLGEN Utility continues processing for this Table. However, the Rules Generator DECIMAL operand has been set to question marks.

Operator Response: Verify that the DB2 decimal scale value is valid and that it is within the allowable IMSADF II maximum of 13.

If the DB2 decimal scale exceeds the allowable IMSADF II maximum then this Column cannot be processed directly by standard IMSADF II functions.

ADFY921 W INVALID COLUMN DATA TYPE: LONG VARCHAR. THE IMSADF II DOES NOT SUPPORT THIS DATA TYPE. THE DATA TYPE HAS BEEN CONVERTED TO VARCHAR.

Explanation: A DB2 data type of LONG VARCHAR has been encountered. IMSADF II does not support this data type. The data type is converted to SHORT VARCHAR which is supported by IMSADF II.

System Action: The RGLGEN Utility continues processing for this Table.

Operator Response: Verify that the DB2 data type is valid.

ADFY922 E NO INPUT RECORDS FOUND. THE SEQUENTIAL INPUT DATA SET REFERENCED BY RGLGENI DD IS EMPTY.

Explanation: The sequential input data set referenced by the RGLGENI DD statement is empty. There are no input records for the RGLGEN Utility to process.

System Action: The RGLGEN Utility terminates processing.

Operator Response: If the RGLGEN Utility was invoked using the IADF RGLGEN GENERATE panel, the input parameters specified are built into input records by IADF.

If the RGLGEN Utility was invoked as a TSO batch job then the RGLGENI DD data set must be defined, either as instream records or as a valid sequential data set.

In either case valid input records must be defined.

ADFY923 I OPTIONAL SYSADF.ADFCOLUMNID TABLE USED. THE ROW SELECTED WAS: COLUMN NAME: column.name, TABLE NAME: table.name, TABLE CREATOR: creator, IMSADF II COLUMN ID: adfcolumnid

Explanation: This informational message is generated every time a previously defined IMSADF II Column ID is used. It implies that this DB2 Column has been previously defined to IMSADF II.

System Action: None

Operator Response: None

ADFY924 I OPTIONAL SYSADF.ADFCOLUMNID TABLE USED. THE FOLLOWING ROW WAS A MATCH BUT WAS NOT SELECTED BECAUSE ANOTHER ROW CONTAINED A BETTER MATCH. COLUMN NAME: column.name, TABLE NAME: table.name, TABLE CREATOR: creator, IMSADF II COLUMN ID: adfcolumnid

Explanation: This informational message is generated when a DB2 Column is defined in the SYSADF.ADFCOLUMNID Table multiple times. This can happen if the COLUMN name is defined once with a blank Table name and creator, and one or more additional times with non-blank Table names and creator.

System Action: None

Operator Response: None

ADFY925 I OPTIONAL SYSADF.ADFCOLUMNID TABLE USED. THE MASTER ROW CONTAINING THE LAST USED IMSADF II COLUMN ID WAS NOT FOUND. A MASTER ROW WAS INSERTED WITH COLUMN NAME: column.name, and IMSADF II COLUMN ID: adfcolumnid

Explanation: This informational message is generated when the first DB2 Column is defined in the SYSADF.ADFCOLUMNID Table. A master row containing the last used IMSADF II Column ID must be INSERTed the first time the SYSADF.ADFCOLUMNID Table is used. After that the master row is UPDATEd.

System Action: None

Operator Response: None

ADFY926 E OPTIONAL SYSADF.ADFCOLUMNID TABLE USED. THE LAST USED IMSADF II COLUMN ID IN THE MASTER ROW CONTAINS INVALID CHARACTERS OR IMBEDDED BLANKS. NOTIFY YOUR IMSADF II ADMINISTRATOR. COLUMN NAME: column.name, IMSADF II COLUMN ID: adfcolumnid

Explanation: The last used IMSADF II Column ID is invalid. This should only happen if some one has modified the last used row outside of the control of the RGLGEN Utility.

System Action: The RGLGEN Utility terminates processing.

Operator Response: Modify the master row in the SYSADF.ADFCOLUMNID Table so that it contains a valid last used IMSADF II Column ID, or DELETE the master row and INSERT a new row with a valid last used ID.

ADFY927 I OPTIONAL SYSADF.ADFCOLUMNID TABLE USED. IMSADF II COLUMN ID VALUES ARE BEING SWITCHED FROM ALL NUMERIC TO ALL ALPHABETIC, OR FROM ALL ALPHABETIC TO ALPHANUMERIC. STARTING WITH IMSADF II COLUMN ID: adfcolumnid

Explanation: This informational message is generated when IMSADF II Column ID, 9999 is the last used ID and another Column ID is required. Once the RGLGEN Utility has exhausted all numeric Column ID's it automatically switches to generating all alphabetic IMSADF II Column ID's.

System Action: None

Operator Response: None

ADFY928 W TABLE DOES NOT HAVE A UNIQUE INDEX DEFINED. NO COLUMNS IN THIS TABLE HAVE THE KEY=YES PARAMETER SPECIFIED. RULEGEN ERROR WILL RESULT. TABLE NAME: table.name, TABLE CREATOR: creator

Explanation: The table does not have a unique index defined. IMS Application Development Facility II requires one or more columns in each DB2 table to be key columns.

System Action: None

Operator Response: Use the SQL CREATE INDEX command to define a unique index. Then, either re-run the RGLGEN job to obtain the KEY=YES parm(s), or add the KEY=YES parm(s) manually in the Rules Generator source in the RGLGEN0 member. This should be done prior to using the RGLGEN0 member for input to the Rules Generator or an error condition will result.

ADFY929 W TABLE HAS MULTIPLE UNIQUE INDEXES DEFINED. THIS MAY RESULT IN PROCESSING ERROR(S) AT IMSADF II EXECUTION TIME. TABLE NAME: table.name, TABLE CREATOR: creator

Explanation: The table has multiple unique indexes defined. All the key columns in the unique indexes have the KEY=YES parm specified in the Rules Generator source created by RGLGEN. When the end user is performing an insert under online IMSADF II, IMSADF II will enforce uniqueness over the combination of values in all the KEY columns. DB2 enforces uniqueness over the combination of values in the KEY columns of EACH unique index. It would be possible for an end user to specify values which are a unique combination of all the KEY columns, but are not unique within each index. This will result in a DB2 error condition (SQLCODE -803).

System Action: None

Operator Response: The unique indexes defined for the table should be re-examined and verified. It is not recommended for tables with multiple unique indexes to be processed under IMSADF II.

CHAPTER 4. EXECUTION PROCESSING

At execution:

- All DB2 SQL calls issued by IMSADF II are processed by the Table Handler Rule defined for the DB2 Table/View being processed. The Table Handler Rules are the only standard IMSADF II modules that contain SQL statements. Table Handler Rules must be defined in a DB2 Application Plan prior to execution.

  Special Processing Routines and Audit Exits that process IMSADF II SQLHNDLR calls use the IMSADF II Table Handler Rule interface.

  Native static and dynamic SQL statements contained in Special Processing Routines and Audit Exits do not use the IMSADF II Table Handler Rule interface. These exits must be defined in a DB2 Application Plan prior to execution.

- The DB2 SQL return code, SQLCODE and SQLWARN, are mapped into IMSADF II SPA fields, SPASQLCD and SPAWARN for DB2 SQL calls issued by an IMSADF II Table Handler Rule, that result in data I/O or result in a SQLCODE of less than zero, or equal to one hundred.

  The status, DB2 SQL return code, of DECLARE CURSOR, and CLOSE CURSOR SQL calls is not saved unless the call results in a SQLCODE less than zero.

SIGNON AND MENUS

The signon process, and Primary Option and Secondary Option menu display apply for DB2 applications. An IMSADF II USERID must be authorized to execute an IMSADF II transaction at a specified level of intent (that is, retrieve, update, insert, delete). Additionally, if the DB2 authorization is violated by an IMSADF II user during a SQL Call, IMSADF II issues an error message with the appropriate DB2 SQL return code. Reference "Error Processing" on page 4-6 for more information.

KEY SELECTION

- Primary Key Selection

  DB2 Table search values IMSADF II key Columns should uniquely identify a row and should represent the Table Index for optimum performance.

  A row from each Table that the user specifies in the Rules Generator DBPATH operand is automatically fetched by IMSADF II at execution time.

  At execution the key Columns are analyzed.

  - If a >, <, %, _ is found the secondary key selection browse function is invoked.
  - Else, the standard SQL statement CSELECT is used to FETCH a row from the Table based on the key entered.

    The DB2 SQL return codes associated with the CSELECT call are as follows:

    SQLCODE=0 Display the fetched row on the transaction display screen.

    SQLCODE<0 DB2 error.

    Display the PROCESSING ERROR message with the SQLCODE mapped into the message. Redisplay the primary option menu screen.

SQLCODE=100	Invalid key. Invoke the secondary key selection browse function.
SQLCODE>0	and ¬= 100 Processing continues.
	Display the fetched row on the transaction display screen.

The DB2 CURSOR SELECT SQL statements are used by IMSADF II to ensure that only one row is fetched for each Table.

Primary Key Selection DECLAREs a CURSOR, OPENs the CURSOR, FETCHes a single row into the IMSADF II SPA workarea, and CLOSEs the CURSOR.

Note: If the WHERE clause used by the SELECT statement does not define a unique key, the IMSADF II standard functions only process the first row with that key value.

Secondary Key Selection Browse

DL/I secondary key selection allows a browse of all segments, a selected set controlled through secondary key audits, or a generic browse using a partial key and >.

The Secondary Key Selection browse function available for DB2 Tables allows for all of the DL/I capability as well as additional function only available to DB2, (that is, additional generic search arguments, and user defined arguments).

In order for a Table to be eligible it must be specified in the Rules Generator DBPATH operand, and SKSEGS > 0 must be specified on the Rules Generator TABLE statement.

At execution time the Secondary Key Selection browse functions are invoked as follows:

- The KSELECT1 function is invoked if the terminal operator:

  enters a partial key with a > or <
  or
  enters an incorrect key
  or
  enters no key
  and
  the Table is eligible for secondary key selection (SKSEGS>0), and the KSELECT1 function was defined for the Table Handler Rule.
  and
  SPASQLKS was not set to a user defined KSELECTn function in a primary key audit.

- The KSELECT2 function is invoked if the terminal operator:

  enters a partial key with a % or _
  and
  the Table is eligible for secondary key selection (SKSEGS>0), and the KSELECT2 function was defined for the Table Handler Rule.
  and
  SPASQLKS was not set to a user defined KSELECTn function in a primary key audit.

  When the KSELECT2 function, DB2 LIKE predicate relational operator, is invoked, the entire key column must be filled in. DB2 treats trailing blanks as significant characters.

  For example:

  — A DB2 Column is defined to the IMSADF II Rules Generator as follows:

```
COLUMN ID=0001,TYPE=C,LENGTH=006,SQLNAME=EMPNO,KEY=YES
```

- The Terminal operator enters: '%000% ' in the EMPNO display area on the primary key selection screen.

  Request for the KSELECT2 function to display all rows where the EMPNO Column contains the character string '000'.

- DB2 searches for EMPNO LIKE '%000% '. Note the trailing blank.

  DB2 SQL return code is SQLCODE=100.

- IMSADF II redisplays the primary key selection screen with message: ADFE109 REQUESTED DATA NOT FOUND FOR GIVEN KEYS

- If the terminal operator enters: '%000%%' in the EMPNO display area on the primary key selection screen.

- DB2 searches for EMPNO LIKE '%000%%'. No trailing blanks.

  DB2 return code is SQLCODE=0.

- IMSADF II displays the rows on the secondary key selection screen.

- A USER SQL secondary key selection browse function is invoked instead of standard secondary key selection browse whenever the terminal operator:

  enters a partial key with a >, <, %, or _
  or
  enters an incorrect key
  or
  enters no key
  and
  the Table is eligible for secondary key selection (SKSEGS>0), and the specified KSELECTn function was defined in the Table Handler Rule.
  and
  SPASQLKS was set to a user defined KSELECTn function in a primary key audit.

The setting of SPASQLKS in a primary key audit allows any of the KSELECTn functions defined in the Table Handler Rule to be invoked, including the standard KSELECT1, KSELECT2, and the USER SQL KSELECTn. However, normally this technique is used to invoke only the USER SQL KSELECTn functions.

Note: If a valid key is entered, the secondary key selection browse function is not invoked even if a primary key audit has set SPASQLKS. To ensure that a user defined secondary key selection browse function is executed the primary key audit should also invalidate the key.

If the WHERE clause includes other than the key columns, the primary key audit must also set SPAWHERE to the name of the field holding the host variables for the WHERE clause. The Secondary Key Selection modules pass these host variables to the Table Handler Rule rather than the Table key columns.

Each screen iteration during the secondary key selection browse process is a separate transaction iteration. The selected rows of the Table being browsed are retrieved for each screen iteration. This implies that the WHERE clause should be defined so that each subsequent iteration is positioned correctly into the Table, beyond the row previously retrieved. A secondary key audit can be used to update the fields containing the host variables in the WHERE clause so that subsequent iterations are correct.

The standard secondary key selection browse functions KSELECT1 and KSELECT2 have an ORDER BY clause associated with their SQL SELECT statement. This ensures that the rows are displayed in an ordered sequence.

All secondary key selection browse functions are variations of the standard SQL statement CSELECT. The standard functions KSELECT1 and KSELECT2 have >, <, or LIKE as the relational operator in the WHERE clause and additional host variables for repositioning. The USER SQL statements KSELECTn (n=3-9), have user defined WHERE clauses. At execution the secondary key selection browse function DECLAREs a CURSOR, OPENs the CURSOR, FETCHes rows until the secondary key selection page is filled, and CLOSEs the CURSOR.

- Keyareas

  Keyareas represent the data base form of the keys and are maintained for DB2 tables to reflect the current values of Key columns.

  Note: If the key input uses the IMSADF II COFIELD function with a character data type, the terminal input is retained in the COFIELD area of the SPA work area.

  Key Columns can not be defined as null Columns. This implies that indicator variables are not supported as host variables in WHERE clauses.

SCREEN HANDLING

The input and display of DB2 Columns is handled in the same manner as DL/I Fields. Key columns for DB2 Tables named in the DBPATH of an Input Transaction Rule, are processed in the same manner as DL/I DBPATH keys. If a key is changed on the transaction display screen, the transaction is processed as a new transaction, that is, a request for a new set of Table/Segment data.

The screen handler accepts a NULL value for input and output. NULL Columns are displayed as all '-' (Hyphens). This is the same display character used by DB2 SPUFI.

- When a user enters two or more consecutive hyphens into a Columns display area and the Column is defined as allowing NULLs (SQLNULL=YES,SQLIND=YES) then IMSADF II inserts a negative value into the Columns associated Indicator Variable. Only one hyphen is required for a single character Column display area.

- When a user enters data into a Column display area that contained all hyphens (NULL representation) IMSADF II stores the data into the Columns I/O area and sets the Columns associated Indicator Variable to zero. On a subsequent UPDATE the new Column value is changed in the DB2 data base if the Column has been defined to IMSADF II as eligible for update (SQLUPD=YES).

IMSADF II displays DB2 Columns on a screen with initialized values, (that is, Columns in Tables defined as TSEGS, or Columns in Tables defined in DBPATH for an INSERT transaction), as follows:

• Eligible for NULL Column:	All '-' (Hyphens)
• TYPE=VARCHAR Columns:	All ' ' blanks
• All other Column types:	Underscores, blanks, or zero

  When IMSADF II displays a fetched row all Columns on the screen are displayed with their data base value.

DB2 data types of FLOAT and VARCHAR are processed for input and display.

  FLOAT is displayed in scientific notation.

  VARCHAR is allocated its maximum length on the screen and displayed according to the current length.

- The 'N' OPTION is not processed for DB2 Tables.

- The standard twin processing support is not available for DB2 Tables.

AUDITOR

- All Audit Phases are available for DB2 Table processing.

  DB2 Table Columns are eligible for all three phases of auditing (KEY-pre SQL call, PRELIM-before screen display, and PROCESS-after screen input) as well as the three legs of auditing (Automatic Field Assignment, Field Audit, and Messages).

- All audit operations are available, except DL/I related operations.

  All current audit operations, including arithmetic, data compares and moves, encode-decode, subroutine branching, message sending, dynamic screen attribute modification, transaction switching, are available to DB2 Tables.

- DB2 related audit operations

  - The CONCAT and SUBSTR operations manipulating strings for field types ALPHA, NUM, ALPHANUM and VARCHAR.

    The CONCAT operation allows concatenation of two source fields into a target field.

    The SUBSTR operation operation can be performed two ways. Both are move operations, with one performing the substringing on the source field and the other performing the substringing on the target field.

  - The IMMEDIATE SQL Call is an audit operation comparable to the IMMEDIATE DL/I Call audit operation. It can execute all SQL calls specified in a Table Handler Rule.

    After the Table Handler Rule executes the specified SQL function the DB2 SQL return code, SQLCODE and SQLWARN, is tested. If the SQLCODE is zero and SQLWARN0 is blank, the NEXT TRUE branch is taken.

    Two additional operations are available to test the results of the SQL call executed by the Table Handler Rule. These operations allow the SQLCODE to be compared with a list of one or more numeric constants, and SQLWARN to be compared with either a blank or 'W' constant. SQLCODE = 0 and SQLWARN0 = blank implies no errors or warnings. Otherwise, error or warning conditions exist.

  - Two operations are available to customize the secondary key selection browse function. They are only valid during primary key audit and are used by secondary key selection browse.

    The SPASQL operation sets the IMSADF II SPA fields, SPASQLKS, to the number of the standard or user defined secondary key selection browse function to be invoked.

    The SPAWHERE operation sets the IMSADF II SPA field, SPAWHERE to the name of the field that contains the host variables to be used by the secondary key selection browse function being invoked.

    The field named in SPAWHERE should be updated, with a secondary key audit after each FETCH. This is to accommodate repositioning for subsequent secondary key selection iterations.

  - Operations are available to test a DB2 Column for NULL or Truncation status, or to set a field to NULL.

    To test a column for NULL or Truncation or set a column NULL, indicator variables must have been generated for the Table, SQLIND=YES, and SQLNULL=YES must have been specified for the Column.

    Note: No operation has been defined to set a Column to not NULL. In order to change a Column from NULL to not NULL, data must be moved into the Column. Part of the logic associated with data move operations is to set the indicator variable to zero.

MESSAGE HANDLING

ERROR AND WARNING MESSAGES

The Auditor function of flagging a field in error or for warning message applies to DB2 Columns. Appropriate error or warning messages are displayed from the IMSADF II Message Data Base.

For DB2 SQL calls issued by the auditor that result in a field being flagged in error, the Auditor maintains additional error information. The DB2 SQL return code, SQLCODE and SQLWARN are saved. This information can be used during message generation by specifying VARLIST6 and VARLIST7 for mapping in a user message.

ERROR PROCESSING

On return from a SQL call issued by a Table Handler Rule, the DB2 SQL return code, SQLCODE, in the SQL Communications Area (SQLCA) is tested.

If the SQLCODE is 100, the basic DL/I logic for NOT FOUND is followed. This implies a REQUESTED DATA NOT FOUND message, invocation of the Secondary Key Selection browse function listing available rows, or an IMS/VS ROLL call if the SQLCODE=100 is encountered during data base update logic and the updates have been partially completed.

If the SQLCODE is < 0 is received, the current DL/I logic for PROCESSING LOGIC error is followed. The current IMSADF II transaction is terminated, an error screen is displayed, and control is returned to the Primary Option Menu screen.

If the SQLCODE is >= 0 (except 100), processing continues.

If SQLWARN0 and SQLWARN1 are set to 'W', indicating truncation, processing continues. It is the IMSADF II application developers responsibility to test for truncation.

Figure 4-1 describes the IMSADF II action and SQLCODE value.

```
SQLCODE          ACTION

>=0 & ¬= 100   • Processing continues.

=100           • Conversational Environment
                  -REQUESTED DATA NOT FOUND message and keys in
                   error highlighted
                  -Secondary Key Selection Browse for list of rows
                   for selection
                  -Terminate Secondary Key Selection Browse,
                   display last Secondary Key Selection screen
                  -ROLL call and error screen (using Express IOPCB)
                   if 100 occurred and updates were in progress
               • Nonconversational Environment
                  -REQUESTED DATA NOT FOUND message on Transaction
                   display screen
               • Batch Environment
                  -REQUESTED DATA NOT FOUND message on Transaction
                   register and ERRMSG data set.
                   Transaction in error is written to the
                   ERRTRX data set.
<0             • Conversational Environment
                  -PROCESSING ERROR message with SQLCA appropriate
                   values and return to Primary Option Menu screen
               • Nonconversational Environment
                  -PROCESSING ERROR message with SQLCA appropriate
                   values to a stand alone error screen.  That is,
                   the segment display screen is not chained.
               • Batch Environment
                  -PROCESSING ERROR message with SQLCA appropriate
                   values to transaction register and ERRMSG data
                   set.  Transaction in error is written to the
                   ERRTRX data set.
               • ALL Environments
                  -ROLL CALL if successful previous updates
```

Figure 4-1. SQLCODE and IMSADF II ACTION

```
 OPTION:              ERROR MESSAGES

ADFE225 No action - Processing error (DB2 STATUS=    )
```

Figure 4-2. IMSADF II SQL Error Message (SQLCODE < 0)

Figure 4-2 shows the content of the error message generated when a SQL call issued by a transaction driver, (not including SQL calls issued through audit operations or the SQLHNDLR Call function) that results in a DB2 SQL return code of less than zero, (that is, SQLCODE < 0). This message is displayed on the error screen in conversational or nonconversational environments. A similar error message is written to the batch ERRTRX data set in the batch environment.

DATA BASE HANDLING

The IMSADF II standard data base update functions are performed automatically after the PROCESS audit phase has been successfully completed. Updates of DB2 Tables are performed with the IMSADF II standard SQL statements CUPDATE, CDELETE, and INSERT.

CUPDATE - CURSOR UPDATE FOR SINGLE ROW

CURSOR UPDATE is the standard IMSADF II SQL statement used to update a single row in a DB2 Table or View if allowed.

The CURSOR UPDATE function updates a row that has been previously SELECTed and modified.

The CURSOR UPDATE function is used to FETCH a new copy of the row and hold the cursor open. The SQL FOR UPDATE OF clause is specified on the DECLARE CURSOR statement to allow the CURSOR to be referenced on the subsequent UPDATE statement.

The row to be updated is FETCHed into an IMSADF II I/O area.

If the IMSADF II data compare (DATACOMP) function is specified for the Table, the row fetched into the IMSADF II I/O area is compared with a copy of the row saved in the SPA from when the row was initially fetched. If data compare is only specified for a single Column (DCFIELD), then only that Column was saved, and only that Column is compared.

If data compare fails and no previous data base updates, (DL/I or DB2), have occurred within this sync point, the transaction terminates with an error message. A ROLL CALL is issued if previous data base updates have occurred.

If data compare is successful, or the data compare function is not specified for this Table, IMSADF II issues the UPDATE SQL call using the WHERE CURRENT OF CURSOR clause, pointing at the modified row in the SPA.

Note: All DB2 Columns defined to IMSADF II as eligible for update (SQLUPD=YES) are included in the SET clause of the SQL UPDATE statement. Key columns are not eligible for Update.

The DB2 SQL return codes associated with the CUPDATE call are as follows:

SQLCODE=0	Processing continues. Redisplay the transaction display screen with the DATA MODIFIED SUCCESSFULLY completion message.
SQLCODE<0	Processing terminates - DB2 error. Display the PROCESSING ERROR message with the SQLCODE mapped into the message. Redisplay the primary option menu screen if no previous data base updates. Issue a ROLL CALL and display error screen if previous data base updates.
SQLCODE=100	Processing terminates - data base error. Issue a ROLL CALL and display error screen.
SQLCODE>0	and ¬= 100 Processing continues - DB2 warning. Redisplay the transaction display screen with the DATA MODIFIED SUCCESSFULLY completion message.

CDELETE - CURSOR DELETE FOR SINGLE ROW

CURSOR DELETE is the standard IMSADF II SQL statement used to delete a single row from a DB2 Table.

The CURSOR DELETE function deletes a row that has been previously SELECTed.

The row to be deleted is fetched into an IMSADF II I/O area.

If the IMSADF II data compare (DATACOMP) function is specified for the Table, the row fetched into the IMSADF II I/O area is compared with a copy of the row saved in the SPA when the row is initially fetched. If data compare is only specified for a single Column (DCFIELD), then only that Column is saved in the SPA when the row is initially fetched and only that Column is compared.

If data compare fails and no previous data base updates, (DL/I or DB2), have occurred within this sync point, the transaction terminates with an error message. A ROLL CALL is issued if previous data base updates have occurred.

If data compare is successful, or the data compare function is not specified for this Table, IMSADF II issues the DELETE SQL call using the WHERE CURRENT OF CURSOR clause, pointing at the row in the SPA.

The DB2 SQL return codes associated with the CDELETE call are as follows:

SQLCODE=0 Processing continues.

Redisplay the transaction display screen with the DATA DELETED SUCCESSFULLY completion message.

SQLCODE<0 Processing terminates - DB2 error.

Display the PROCESSING ERROR message with the SQLCODE mapped into the message. Redisplay the primary option menu screen, if no previous data base updates. Issue an IMS/VS ROLL CALL if previous data base updates have occurred.

SQLCODE=100 Processing terminates - data base error.

Issue a ROLL CALL and display error screen.

SQLCODE>0 and ¬= 100 Processing continues - DB2 warning.

Redisplay the transaction display screen with the DATA DELETED SUCCESSFULLY completion message.

INSERT - INSERT OF A SINGLE ROW

INSERT is the standard IMSADF II SQL statement used to insert a single row into a DB2 Table.

All Columns defined as eligible for insert (SQLISRT=YES) are included in the VALUES clause of the SQL INSERT statement.

The DB2 SQL return codes associated with the INSERT call are as follows:

SQLCODE=0 Processing continues.

Redisplay the transaction display screen with the DATA ADDED SUCCESSFULLY completion message.

SQLCODE<0 Processing terminates - DB2 error.

Display the PROCESSING ERROR message with the SQLCODE mapped into the message. Redisplay the primary option menu screen, if no previous data base updates. If previous data base updates, issue an IMS/VS ROLL CALL.

SQLCODE>0 and ¬= 100 Processing continues - DB2 warning.

Redisplay the transaction display screen with the DATA ADDED SUCCESSFULLY completion message.

Notes:

1. An IMS/VS ROLL CALL is issued if a DL/I or SQL data base update call fails and previous updates have occurred within this sync point.

2. If SELECTed Columns are truncated, SQLWARN0 and SQLWARN1 set to 'W' by DB2, IMSADF II continues processing. The truncated data is displayed as returned. It is the user's responsibility to test for truncation and to determine if an error condition exists. Standard Audit operations have been provided to test SQLWARN and to test

individual Columns for truncation. If truncation does occur the user must determine if transaction logic (for example, suppress data base updates) should be altered.

3. If the IMSADF II data compare (DATACOMP) function is specified, and only a single Column (DCFIELD) is being compared, that Column should not be defined as being eligible for NULL value. When the DCFIELD is compared only the data area is compared. The associated indicator variable is not compared.

4. Native SQL (static and dynamic) calls can also be issued by an Audit Exit or Special Processing Routine.

RULE AND SPA WORKAREA HANDLING

The Table Handler Rule is loaded and invoked in the same manner as the current DL/I Segment Handler Rules and is eligible for the PRELOAD rule and Composite rules load module.

The requesting transaction driver passes the function, host variables for the INTO and WHERE clauses, indicator variables, the SQL communication area (SQLCA), and the address of DSNHLI, the language interface entry point. In turn, the Table Handler Rule issues the appropriate SQL calls and indicates the result.

The DB2 Tables/Views, defined to an IMSADF II transaction, are allocated space in the SPA workarea in the same manner as current DL/I and pseudo segments.

EXIT PROCESSING

A SQLHNDLR Call function is available to Audit exits and Special Processing routines, similar in structure to the SEGHNDLR function available for DL/I segments. A SQLHNDLR Call issued by the COBOL, PL/I or Assembler exit passes the following parameters to the Table Handler Rule:

- Table ID
- Label of function to Execute
- Host Variables to describe the search values - optional
- I/O area - optional
- SQLCA - optional

In PL/I:

```
CALL SQLHNDLR(ID,LABEL,{KEY,AREA,SQLCA});
```

In COBOL:

```
CALL 'SQLHNDLR' USING ID, LABEL, {KEY, AREA, SQLCA}.
```

Note:

1. The IMSADF II MAPPER function does not map indicator variables into a user's I/O area. The IMSADF II COPYSEG function maps the entire row, including indicator variables into a user's I/O area.
2. The MAPPER function converts TYPE=FLOAT Columns to alphanumeric and alphanumeric to FLOAT. The MAPPER function calculates the current length of a TYPE=VARCHAR Column based on the length of the source field when modified data is mapped into the SPA.
3. Audit exits or Special processing routines can also issue native SQL calls. Optionally, the exit can then map the resulting row(s) into the SPA workarea, using the MAPPER or COPYSEG function. The row is then accessible for other IMSADF II functions.
4. The DL/I Exit available before and after each DL/I call is not available for SQL calls.

CHAPTER 5. INSTALLATION

There is one optional step in the IMSADF II installation process dealing with additional DB2 support. This optional installation step must be executed, if you choose to use the IMSADF II supplied RGLGEN Utility to extract Rules Generator source from the DB2 catalog.

The RGLGEN Utility is linked with the DB2 TSO Language Interface Module, DSNELI by this step.

Refer to the RGLGEN Utility chapter for details on installing and using this utility.

SAMPLE PROBLEM

After the IMSADF II installation is completed, the following steps must also be completed before the IMSADF II DB2 sample problem can be executed.

The sample problem is composed of three IMSADF II transactions that are defined in detail in appendixes B and C.

1. **Create the IMSADF II Static Rules.**

   The Rules Generator source statements required to create all the IMSADF II static rules are contained in the IMSADF.RULES.SOURCE library, member RGLDB2S. The three required screen image source members are contained in the IMSADF.ADFIMG library, members EM, EX, and ES.

   Note:

   - The DB2 Table processed by the three sample transactions is the DB2 sample Employee Table, 'DSN8.TEMPL'. In order to access a private copy of this table, change the SQLNAME operand on the two Rules Generator TABLE statements to the new DB2 Table name.

   - The Rules Generator Table and Column source statements for the TABLE ID=EM can also be extracted from the DB2 Catalog. The RGLGEN Utility should be executed and the created Table and Column statements should be compared with the Table and Column statements in source member RGLDB2S.

   - The information contained on the SYSTEM statement in source member RGLDB2S establishes that the three DB2 sample transactions are clustered under IMS/VS transaction SAMPTOR. The associated IMSADF II PROJECT/GROUP is ZZ. This is the same IMS/VS transaction and PROJECT/GROUP that the standard IMSADF II sample problem uses.

   - The last two GENERATE statements in source member RGLDB2S can be commented out if the standard IMSADF II sample problem has been defined.

2. **Create the IMSADF II Dynamic Rules.**

   - Define additional transaction logic required to successfully execute the IMSADF II DB2 sample transactions. Appendix B contains samples of the required High Level Audit Language code. This additional transaction logic should be added to the Audit data base. No source has been provided.

   - Optionally define transaction Help information. Transaction help can be made available for all conversational transactions. Appendix B contains a sample of transaction Help for the 'EM' transaction. No source is provided.

- Define additional error messages. The sample High Level Audit Language for transaction 'ES' references error message 1000. This message should be added to the Message data base.

- The three sample DB2 transactions 'EM', 'EX', and 'ES' must be added to the ZZ Project/Group profile in the SIGNON/PROFILE data base.

3. **Create the DB2 Application Plan.**

   Part of the output created by the Rules Generator are two DB2 DBRMs. The DB2 BIND process is invoked to combine these DBRMs into a DB2 Application Plan. The name of the Application Plan must be the same as the IMS/VS transaction and PSB, (that is, SAMPTOR). The names of the DBRMs that must be specified when the BIND process is invoked are SAMPSEM, and SAMPSES.

Once these additional steps have been completed the IMSADF II DB2 sample problem becomes a part of the standard IMSADF II sample problem.

DB2 APPLICATION PLAN RESTRICTION

IMSADF II Version 2 Release 2, in the CICS/OS/VS environment, has the following restriction - all DB2 transactions for an IMSADF II SYSID must be in the same application plan. This is accomplished by specifying the same cluster code (i.e., PGMID= and SOMTX= operands) to the Rules Generator.

If DB2 Release 1 is used, the number of DB2 transactions per SYSID will be limited by the total application plan size. This is because plans are stored and loaded as a single unit of actual code. There is code in the plan for each SQL statement, and plan size is dependent on the number of tables referenced and the complexity of the user SQL statements.

However, in DB2 Release 2, because of changes in the locking mechanism and application plan segmentation, the above IMSADF II CICS/OS/VS restriction will still apply, but more DB2 transactions per SYSID may be used. Specifically, plans consist of more compact structures. Each SQL statement, or set of cursor-related statements, creates a separate control structure. Thus, virtual storage demands are relieved because each control structure is smaller than the DB2 Release 1 application plan. These control structures are loaded into virtual storage when the program first executes the SQL statement. In addition, the control structures are released from the Environmental Descriptor Manager (EDM) pool based on the RELEASE value specified during the BIND. This decreases demand for real storage and paging I/O related to EDM pool size.

APPENDIX A. SAMPLE RULES GENERATOR SOURCE AND OUTPUT

This appendix contains Rules Generator source statements that define two IMSADF II transactions that access both DL/I and DB2 data bases. It also contains the Assembler Source statements output for the Table Handler Rule created by the Rules Generator. These source statements are used as input to the DB2 pre-compiler. The Rules Generator dynamically invokes the DB2 pre-compiler prior to invoking the Assembler and the Linkage Editor.

RULES GENERATOR SOURCE STATEMENTS

```
***********************************************************************
*                                                                     *
*   APPLICATION DEFINITION - DLI PARTS DATA BASE - PA SEGMENT         *
*                            DB2 DSN8  DATA BASE - EM TABLE           *
*                                                                     *
***********************************************************************
*
SYSTEM   SYSID=SAMP,DBID=PA,                             RULE ID CHARS
         USRLANG=E,                           FORCE TO ENGLISH
         SOMTX=OR,                            DEFAULT SECONDARY OPTION CODE
         PCBNO=1,                             PCB NUMBER FOR DATA BASE
         SHEADING='S A M P L E   P R O B L E M',         GENERAL HEADING
         SFORMAT=DASH,                                   SCREEN FORMAT
         PGROUP=ZZ                                       PROJECT GROUP
*
***********************************************************************
*                                                                     *
*   APPLICATION DEFINITION INPUT FOR PA ROOT SEGMENT                  *
*                                                                     *
***********************************************************************
*
SEGMENT ID=PA,PARENT=0,NAME=PARTROOT,LENGTH=50,SKSEG=18
  FIELD ID=KEY,LENGTH=17,KEY=YES,NAME=PARTKEY,SNAME='PART NUMBER'
  FIELD ID=DESC,LENGTH=20,POS=27,SNAME='DESCRIPTION',REL=YES
*
***********************************************************************
*                                                                     *
*    TABLE:    ID=EM      DATE:    12/25/84     TIME:    00:04:29     *
*            NAME=DSN8.TEMPL                     MEMBER=SAMPTBEM      *
*                                                                     *
***********************************************************************
*
TABLE  ID=EM,TYPE=TBL,SQLNAME='DSN8.TEMPL',SQLIND=YES
COLUMN ID=0001,SQLNAME='EMPNO',SNAME='EMPNO',
       TYPE=C,LENGTH=006,SQLUPD=NO
COLUMN ID=0002,SQLNAME='FIRSTNME',SNAME='FIRSTNME',
       TYPE=V,LENGTH=012
COLUMN ID=0003,SQLNAME='MIDINIT',SNAME='MIDINIT',
       TYPE=C,LENGTH=001
COLUMN ID=0004,SQLNAME='LASTNAME',SNAME='LASTNAME',
       TYPE=V,LENGTH=015
COLUMN ID=0005,SQLNAME='WORKDEPT',SNAME='WORKDEPT',
       TYPE=C,LENGTH=003
COLUMN ID=0006,SQLNAME='PHONENO',SNAME='PHONENO',
       TYPE=C,LENGTH=004,SQLNULL=YES
COLUMN ID=0007,SQLNAME='HIREDATE',SNAME='HIREDATE',
       TYPE=P,LENGTH=004,SQLNULL=YES
COLUMN ID=0008,SQLNAME='JOBCODE',SNAME='JOBCODE',
       TYPE=P,LENGTH=002,SQLNULL=YES
COLUMN ID=0009,SQLNAME='EDUCLVL',SNAME='EDUCLVL',
       TYPE=I,LENGTH=002,SQLNULL=YES
COLUMN ID=0010,SQLNAME='SEX',SNAME='SEX',
       TYPE=C,LENGTH=001,SQLNULL=YES
COLUMN ID=0011,SQLNAME='BRTHDATE',SNAME='BRTHDATE',
       TYPE=P,LENGTH=004,SQLNULL=YES
COLUMN ID=0012,SQLNAME='SALARY',SNAME='SALARY',
       TYPE=P,LENGTH=005,DEC=02,SQLNULL=YES
*
* FIELD MERGE INFORMATION FOR 'EM' KEY SELECTION AND AUDITS
*
COLUMN  ID=0001,KEY=YES,SNAME='EMPLOYEE SEARCH DATA',COL=01
COLUMN  ID=0004,RELATED=YES,COL=16
COLUMN  ID=0005,RELATED=YES,COL=33
COLUMN  ID=0012,RELATED=YES,COL=43
*
* SEGMENT OVERRIDE INFORMATION FOR 'EM' KEY SELECTION
*
TABLE   OVERRIDE=ID,ID=EM,SKSEGS=15,
        SKLEFT='EMPLOYEE         LAST          WORK',
        SKLEFT='NUMBER           NAME          DEPT',
        SKRIGHT='       SALARY'
*
***********************************************************************
*                                                                     *
* PSEUDO SEGMENT TO HOLD 'EM' KSELECT3 WHERE CLAUSE HOST VARIABLE     *
*                                                                     *
***********************************************************************
*
SEGMENT ID=P1,TYPE=PS
  FIELD ID=KS3,TYPE=C,LENGTH=09
  FIELD ID=EMP#,TYPE=C,LENGTH=06,REDEF=KS3
  FIELD ID=DEPT,TYPE=C,LENGTH=03,REDEF=KS3,OFFSET=6
*
***********************************************************************
*                                                                     *
*     GENERATE SEGMENT LAYOUT AND HANDLER RULES FOR SEGMENT - PA      *
*     GENERATE SEGMENT LAYOUT RULE FOR PSEUDO SEGMENT - P1            *
*     GENERATE TABLE LAYOUT RULE FOR DB2 TABLE - EM                   *
*                                                                     *
***********************************************************************
*
GENERATE OPTIONS=(SEGL,SEGH),SEGMENT=PA
GENERATE OPTIONS=SEGL,SEGMENT=P1
GENERATE OPTIONS=TABL,TABLE=EM
*
***********************************************************************
*                                                                     *
*     GENERATE A TABLE HANDLER RULE FOR TABLE - EM                    *
*                                                                     *
***********************************************************************
*
GENERATE OPTIONS=TABH,TABLES=EM,SQLCALL=(DSQLCALL,KSELECT2),
         SQLUSER=YES,ASMREQ=YES
KSELECT3 SELECT   WHERE EMPNO >= :EMP#.P1 AND WORKDEPT = :DEPT.P1
               ORDER BY EMPNO ASC, WORKDEPT ASC
&SQLENDS
*
***********************************************************************
*                                                                     *
*     GENERATE CONVERSATIONAL INPUT TRANSACTION RULE AND SCREENS      *
*                                                                     *
***********************************************************************
*
GENERATE TRXID=EX,OPTIONS=CVALL,DBPATH=(PA,EM),TSEGS=(P1),
         TRXNAME='DLI-PART / DB2-EMPLOYEE',DATACOMP=(EM),
         DEVNAME=(2,A3,A4),
         DEVCHRS=(0,0,0),
         DEVTYPE=(2,7,4),SPOS=SIMAGE
               'D B 2   'S A M P L E   'P R O B L E M
&=1
 &MODE                     TRANSACTION: 'DLI PART / 'DB2 EMPLOYEE
 OPTION:&OPT TRX:&TRAN KEY:&KEY
    &SYSMSG
&=1
   DLI SEGMENT      PART NUMBER-----------&5KEY.PA
                    DESCRIPTION-----------&5DESC.PA
&=2
   DB2 TABLE        EMPLOYEE NUMBER-------&50001.EM
                    FIRST NAME------------&50002.EM
                    MIDDLE INITIAL--------&50003.EM
                    LAST NAME-------------&50004.EM
                    DEPARTMENT NUMBER-----&50005.EM
                    PHONE EXTENSION-------&50006.EM
                    DATE HIRED------------&50007.EM
                    JOB CODE--------------&50008.EM
                    EDUCATION LEVEL-------&50009.EM
                    SEX-------------------&50010.EM
                    BIRTH DATE------------&50011.EM
                    SALARY----------------&50012.EM
&*         1     1     2      3      3       4    5   5
&*---------2-----8-----4------1------8-------6----1---5
&*   ID   *SLEN *VROW *VCOL  *VMODE *ASTATUS*KSEL*CLR*XHL
&:KEY.PA                                          P
&:DESC.PA                                         G
&:0001.EM                                     KA  P
&:0002.EM                                         Y
&:0003.EM                                         Y
&:0004.EM                                         Y
&:0005.EM                                         Y
&:0006.EM                                         Y
&:0007.EM   6                                     Y
&:0008.EM   3                                     Y
&:0009.EM   5                                     Y
&:0010.EM                                         Y
&:0011.EM   6                                     Y
&:0012.EM   10                                    Y
&ENDS
*
************************************************************************
*                                                                      *
*     GENERATE NONCONVERSATIONAL INPUT TRANSACTION RULE AND SCREENS    *
*                                                                      *
************************************************************************
*
GENERATE TRXID=NX,OPTIONS=TPALL,DBPATH=(PA,EM),TSEGS=(P1),
         TRXNAME='DB2 EMPLOYEE TABLE',MODNAME=SAMPNM,
         DEVNAME=(2,A3,A4),
         DEVCHRS=(0,0,0),
         DEVTYPE=(2,7,4),SPOS=SIMAGE
               'D B 2   'S A M P L E   'P R O B L E M
&=1
 &MODE                      TRANSACTION: 'DLI PART / 'DB2 EMPLOYEE
 ACTION:&ACTION     TRX:&X
    &SYSMSG
&=1
   DLI SEGMENT      PART NUMBER-----------&5KEY.PA
                    DESCRIPTION-----------&5DESC.PA
&=2
   DB2 TABLE        EMPLOYEE NUMBER-------&50001.EM
                    FIRST NAME------------&50002.EM
                    MIDDLE INITIAL--------&50003.EM
                    LAST NAME-------------&50004.EM
                    DEPARTMENT NUMBER-----&50005.EM
                    PHONE EXTENSION-------&50006.EM
                    DATE HIRED------------&50007.EM
                    JOB CODE--------------&50008.EM
                    EDUCATION LEVEL-------&50009.EM
                    SEX-------------------&50010.EM
                    BIRTH DATE------------&50011.EM
                    SALARY----------------&50012.EM
&*         1     1     2      3      3       4    5   5
&*---------2-----8-----4------1------8-------6----1---5
&*   ID   *SLEN *VROW *VCOL  *VMODE *ASTATUS*KSEL*CLR*XHL
&:KEY.PA                                          P
&:DESC.PA                                         G
&:0001.EM                                         P
&:0002.EM                                         Y
&:0003.EM                                         Y
&:0004.EM                                         Y
&:0005.EM                                         Y
&:0006.EM                                         Y
&:0007.EM   6                                     Y
&:0008.EM   3                                     Y
&:0009.EM   5                                     Y
```

```
&:0010.EM                                         Y
&:0011.EM   6                                     Y
&:0012.EM   10                                    Y
&ENDS
*
***********************************************************************
*                                                                     *
*     GENERATE SECONDARY OPTION MENU                                  *
*                                                                     *
***********************************************************************
*
GENERATE OPTIONS=SOM
*
***********************************************************************
*                                                                     *
*     LINK CONVERSATIONAL AND NONCONVERSATIONAL DRIVERS               *
*                                                                     *
***********************************************************************
*
GENERATE OPTION=STLE,PGMID=OR
GENERATE OPTION=NCLE,PGMID=OR
*** END OF DATA ***
```

TABLE HANDLER GENERATED ASSEMBLER SOURCE STATEMENTS

```
SEM      TITLE 'TABLE HANDLER FOR IMSADF II TABLE ID -EM, SYSID -SAMP'
SAMPSEM  CSECT
         SPACE 2
***********************************************************************
*                                                                     *
* MODULE NAME: SAMPSEM                                                *
*                                                                     *
* DESCRIPTIVE NAME: TABLE HANDLER FOR TABLE ID -EM, SYSID -SAMP       *
*                                                                     *
* STATUS: GENERATED BY IMSADF II VERSION 2 RELEASE 1                  *
*                                                                     *
* FUNCTION: TO ISSUE SQL CALLS FOR DB2 TABLES DEFINED TO IMSADF       *
*                                                                     *
* MODULE TYPE:   CSECT                                                *
*   PROCESSOR:   DB2 PREPROCESSOR AND ASSEMBLER                       *
*   ATTRIBUTES:  SERIALLY REUSABLE                                    *
*                                                                     *
* INPUT: QCONTROL TABLE                                               *
*              ------------------                                     *
*             | SQLFUNC  ADDRESS |                                    *
*             | IOAREA   ADDRESS |                                    *
*             | KEYS     ADDRESS |                                    *
*             | SQLCA    ADDRESS |                                    *
*             | SQLCA-WS ADDRESS |                                    *
*             | DSNHLI   ADDRESS |                                    *
*              ------------------                                     *
* OUTPUT:                                                             *
*                                                                     *
*     COMPLETED SQL CALL                                              *
*     RETURN CODE (REGISTER 15)                                       *
*                                                                     *
* EXTERNAL REFERENCES:                                                *
*                                                                     *
*   ROUTINES:       DSNHLI                                            *
*                                                                     *
*   DATA AREAS:     TABLE I/O AREA  (DATA AND INDICATOR VARIABLES)    *
*                   SEARCH VARIABLES (KEYS)                           *
*                                                                     *
*   CONTROL BLOCKS: SQLCA                                             *
*                                                                     *
***********************************************************************
         SPACE 2
R0       EQU   0
R1       EQU   1
R2       EQU   2                      SQL FUNCTION TO BE PERFORMED
R3       EQU   3                      TABLE IOAREA
R4       EQU   4                      TABLE IOAREA PLUS 4096
R5       EQU   5                      SEARCH VARIABLES (KEYS)
```

```
R6       EQU   6                    SQLCA ADDRESS
R7       EQU   7                    SQLDA ADDRESS
R8       EQU   8                    DSNHLI ADDRESS
R9       EQU   9                    BASE
R10      EQU   10                   BASE
R11      EQU   11                   BASE
R12      EQU   12                   BASE
R13      EQU   13                   SAVE AREA
R14      EQU   14                   RETURN ADDRESS
R15      EQU   15
WARNING  EQU   C'W'
RC00     EQU   0
RC08     EQU   8
RC21     EQU   21
RC22     EQU   22
RC23     EQU   23
RC24     EQU   24
         EJECT
* TABLE EM IOAREA DSEC FOR HOST VARIABLES
SAMPEMO  DSECT ,
AEM0001$ DS    CL006
AEM0002$ DS    H,CL012
AEM0003$ DS    CL001
AEM0004$ DS    H,CL015
AEM0005$ DS    CL003
AEM0006$ DS    CL004
AEM0007$ DC    PL004'9999999'
AEM0008$ DC    PL002'999'
AEM0009$ DS    H
AEM0010$ DS    CL001
AEM0011$ DC    PL004'9999999'
AEM0012$ DC    PL005'9999999.99'
* TABLE EM INDICATOR VARIABLES FOR HOST VARIABLES
         DS    H
         DS    H
         DS    H
         DS    H
         DS    H
AEM0006@ DS    H
AEM0007@ DS    H
AEM0008@ DS    H
AEM0009@ DS    H
AEM0010@ DS    H
AEM0011@ DS    H
AEM0012@ DS    H
* SEARCH VALUES FOR TABLE EM
SAMPEMK  DSECT ,
AEM00011 DS    CL006              CURRENT VALUE KEY 001
AEM00012 DS    CL006              BASE VALUE KEY 001 FOR REPOSITIONING
* USER SEARCH VALUES FOR USER LABEL - KSELECT3
SAMPEMK3 DSECT ,
AP1EMP#3 DS    CL006
AP1DEPT3 DS    CL003
         EJECT
SAMPSEM  CSECT
         USING SAMPSEM,R15          TEMP BASE IN R15.
         B     SQLSTART             BRANCH AROUND ID.
         DC    AL1(SQLSTART-*-1),C'SAMPEM &SYSDATE &SYSTIME'
SQLSTART DS    0H
         STM   R14,R12,12(R13)      SAVE CALLERS REGS.
         LA    R14,SQLSAVE          PICK UP SAVE AREA ADDR.
         ST    R14,8(,R13)          SET FWD PTR.
         ST    R13,4(,R14)          SET BKWD PTR.
         LR    R13,R14              R13 TO SAVE AREA.
         LR    R12,R15
         DROP  R15
         SPACE
         USING SAMPSEM,R12
         SPACE
         LA    R15,RC00             INDICATE VALID FUNCTION
         LM    R2,R3,0(R1)          PARAMETER ADDRESSES
         LM    R5,R8,8(R1)          PARAMETER ADDRESSES
         LR    R4,R3                2ND BASE REG FOR TABLE IOAREA
         A     R4,=A(4096)          STARTING ADDRESS FOR 2ND BASE REG
         L     R15,=V(ADSNHLI)       GET ADDRESS OF ADSNHLI
         ST    R8,0(,R15)            SAVE DSNHLI ADDRESS
***********************************************************************
*                                                                     *
*        LOCATE REQUESTED FUNCTION                                    *
*                                                                     *
***********************************************************************
         LA    R9,BRANCHT            LOOP THROUGH BRANCH TABLE
LOOP1    L     R10,8(R9)             ADDR OF SQL ROUTINE
         CLC   0(8,R9),0(R2)         FUNCTION TO BE PERFORMED
         BER   R10                   EXECUTE SQL CALL
         LA    R9,12(R9)             GET NEXT ENTRY
         CLC   0(8,R9),=C'FFFFFFFF'  LAST ENTRY
         BNE   LOOP1                 NOT THE LAST - CONTINUE SEARCH
         LA    R15,RC08              INDICATE FUNCTION NOT VALID
RETURN   L     R13,4(,R13)           RESTORE CALLERS R13
         LM    R0,R12,20(R13)        RESTORE CALLERS R0-R12
         L     R14,12(R13)           GET RETURN POINT
         BR    R14                     & RETURN
         USING SAMPEMO,R3            SQL TABLE IOAREA
         USING SAMPEMO+4096,R4       2ND BASE REG FOR TABLE IOAREA
         USING SAMPEMK,R5            SQL SEARCH VALUES - ASSUME STANDARD
         USING SQLCAD,R6             SQLCA
         USING SQLDSECT,R7           SQLCA EXTENSION
         SPACE
         DC    C'SEMSAVE '           ID TO TABLE HANDLER RULE
SQLSAVE  DC    18A(0)                SAVE AREA
         EJECT
***********************************************************************
*                                                                     *
*        BRANCH TABLE                                                 *
*                                                                     *
***********************************************************************
BRANCHT  DS    0D                    BRANCH TABLE
         DC    C'CSELECTO',A(CSELECTO)  SEQUENTIAL SELECT WITH CURSOR
         DC    C'CSELECTC',A(CSELECTC)  SEQUENTIAL SELECT CLOSE CURSOR
         DC    C'INSERT  ',A(INSERT)    INSERT SINGLE ROW
         DC    C'CUPDATEO',A(CUPDATEO)  OPEN CURSOR FOR UPDATE (1 ROW)
         DC    C'CUPDATEU',A(CUPDATEU)  UPDATE FOR UPDATE CURSOR
         DC    C'CUPDATEC',A(CUPDATEC)  CLOSE UPDATE CURSOR
         DC    C'CDELETEO',A(CDELETEO)  OPEN CURSOR FOR DELETE (1 ROW)
         DC    C'CDELETED',A(CDELETED)  DELETE FOR DELETE CURSOR
         DC    C'CDELETEC',A(CDELETEC)  CLOSE DELETE CURSOR
         DC    C'KSELEC1O',A(KSELEC1O)  SEQUENTIAL SELECT WITH CURSOR
         DC    C'KSELEC1F',A(KSELEC1F)  FETCH WITHIN KSELECT CURSOR
         DC    C'KSELEC1C',A(KSELEC1C)  CLOSE KSELECT CURSOR
         DC    C'KSELEC2O',A(KSELEC2O)  SEQUENTIAL SELECT WITH CURSOR
         DC    C'KSELEC2F',A(KSELEC2F)  FETCH WITHIN KSELECT CURSOR
         DC    C'KSELEC2C',A(KSELEC2C)  CLOSE KSELECT CURSOR
         DC    C'KSELEC3O',A(KSELEC3O)  LABEL FOR USER KSELECT OPEN
         DC    C'KSELEC3F',A(KSELEC3F)  LABEL FOR USER KSELECT FETCH
         DC    C'KSELEC3C',A(KSELEC3C)  LABEL FOR USER KSELECT CLOSE
         DC    C'FFFFFFFF',A(0)      END OF BRANCH TABLE FLAG
         SPACE
SC100    DC    F'100'
         EJECT
***********************************************************************
*                                                                     *
*        CSELECTO FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CSELECTO EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               DECLARE CSELECT  CURSOR FOR                             X
               SELECT                                                  X
               EMPNO                                                  ,X
               FIRSTNME                                                ,X
               MIDINIT                                                 ,X
               LASTNAME                                                ,X
               WORKDEPT                                                ,X
               PHONENO                                                 ,X
               HIREDATE                                                ,X
               JOBCODE                                                 ,X
               EDUCLVL                                                 ,X
               SEX                                                     ,X
               BRTHDATE                                                ,X
               SALARY                                                   X
               FROM                                                     X
               DSN8.TEMPL                                               X
               WHERE                                                    X
               EMPNO                                                    X
               =:AEM00011
         EXEC  SQL                                                      X
               OPEN CSELECT
         ICM   R14,15,SQLCODE
         BM    RETURN                  SQLCODE < 0
         CLC   SQLCODE,SC100
         BE    RETURN                  SQLCODE = 100
         EXEC  SQL                                                      X
               FETCH CSELECT                                            X
               INTO                                                     X
               :AEM0001$                                                X
               :AEM0002$                                               ,X
               :AEM0003$                                               ,X
               :AEM0004$                                               ,X
               :AEM0005$                                               ,X
               :AEM0006$:AEM0006@                                      ,X
               :AEM0007$:AEM0007@                                      ,X
               :AEM0008$:AEM0008@                                      ,X
               :AEM0009$:AEM0009@                                      ,X
               :AEM0010$:AEM0010@                                      ,X
               :AEM0011$:AEM0011@                                      ,X
               :AEM0012$:AEM0012@
         LA    R15,RC21                SELECT/FETCH FUNCTION
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CSELECTC FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CSELECTC EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                      X
               CLOSE CSELECT
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        INSERT   FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
         INSERT   EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               INSERT INTO                                             X
               DSN8.TEMPL                                              X
               (                                                       X
               EMPNO                                               ,   X
               FIRSTNME                                            ,   X
               MIDINIT                                             ,   X
               LASTNAME                                            ,   X
               WORKDEPT                                            ,   X
               PHONENO                                             ,   X
               HIREDATE                                            ,   X
               JOBCODE                                             ,   X
               EDUCLVL                                             ,   X
               SEX                                                 ,   X
               BRTHDATE                                            ,   X
               SALARY                                                  X
               )                                                       X
               VALUES (                                                X
               :AEM0001$                                              ,X
               :AEM0002$                                              ,X
               :AEM0003$                                              ,X
               :AEM0004$                                              ,X
               :AEM0005$                                              ,X
               :AEM0006$:AEM0006@                                     ,X
               :AEM0007$:AEM0007@                                     ,X
               :AEM0008$:AEM0008@                                     ,X
               :AEM0009$:AEM0009@                                     ,X
               :AEM0010$:AEM0010@                                     ,X
               :AEM0011$:AEM0011@                                     ,X
               :AEM0012$:AEM0012@                                      X
               )
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CUPDATEO FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CUPDATEO EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               DECLARE CUPDATE CURSOR FOR                              X
               SELECT                                                  X
               EMPNO                                               ,   X
               FIRSTNME                                            ,   X
               MIDINIT                                             ,   X
               LASTNAME                                            ,   X
               WORKDEPT                                            ,   X
               PHONENO                                             ,   X
               HIREDATE                                            ,   X
               JOBCODE                                             ,   X
               EDUCLVL                                             ,   X
               SEX                                                 ,   X
               BRTHDATE                                            ,   X
               SALARY                                                  X
               FROM                                                    X
               DSN8.TEMPL                                              X
               WHERE                                                   X
               EMPNO                                                   X
               =:AEM00011                                              X
               FOR UPDATE OF                                           X
               FIRSTNME                                   ,            X
               MIDINIT                                    ,            X
               LASTNAME                                   ,            X
               WORKDEPT                                   ,            X
               PHONENO                                    ,            X
               HIREDATE                                   ,            X
               JOBCODE                                    ,            X
               EDUCLVL                                    ,            X
               SEX                                        ,            X
               BRTHDATE                                   ,            X
               SALARY
         EXEC  SQL                                                     X
               OPEN CUPDATE
         ICM   R14,15,SQLCODE
         BM    RETURN               SQLCODE < 0
         CLC   SQLCODE,SC100
         BE    RETURN               SQLCODE = 100
         EXEC  SQL                                                     X
               FETCH CUPDATE                                           X
               INTO                                                    X
               :AEM0001$                                              ,X
               :AEM0002$                                              ,X
               :AEM0003$                                              ,X
               :AEM0004$                                              ,X
               :AEM0005$                                              ,X
               :AEM0006$:AEM0006@                                     ,X
               :AEM0007$:AEM0007@                                     ,X
               :AEM0008$:AEM0008@                                     ,X
               :AEM0009$:AEM0009@                                     ,X
               :AEM0010$:AEM0010@                                     ,X
               :AEM0011$:AEM0011@                                     ,X
               :AEM0012$:AEM0012@
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CUPDATEU FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CUPDATEU EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               UPDATE                                                  X
               DSN8.TEMPL                                              X
               SET                                                     X
               FIRSTNME                                                X
               =:AEM0002$                                             ,X
               MIDINIT                                                 X
               =:AEM0003$                                             ,X
               LASTNAME                                                X
               =:AEM0004$                                             ,X
               WORKDEPT                                                X
               =:AEM0005$                                             ,X
               PHONENO                                                 X
               =:AEM0006$:AEM0006@                                    ,X
               HIREDATE                                                  X
               =:AEM0007$:AEM0007a                                      ,X
               JOBCODE                                                   X
               =:AEM0008$:AEM0008a                                      ,X
               EDUCLVL                                                   X
               =:AEM0009$:AEM0009a                                      ,X
               SEX                                                       X
               =:AEM0010$:AEM0010a                                      ,X
               BRTHDATE                                                  X
               =:AEM0011$:AEM0011a                                      ,X
               SALARY                                                    X
               =:AEM0012$:AEM0012a                                       X
               WHERE CURRENT OF CUPDATE
         LA    R15,RC22             UPDATE FUNCTION RETURN CODE
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CUPDATEC FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CUPDATEC EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                       X
               CLOSE CUPDATE
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CDELETEO FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CDELETEO EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                       X
               DECLARE CDELETE CURSOR FOR                                X
               SELECT                                                    X
               EMPNO                                            ,        X
               FIRSTNME                                         ,        X
               MIDINIT                                          ,        X
               LASTNAME                                         ,        X
               WORKDEPT                                         ,        X
               PHONENO                                          ,        X
               HIREDATE                                         ,        X
               JOBCODE                                          ,        X
               EDUCLVL                                          ,        X
               SEX                                              ,        X
               BRTHDATE                                         ,        X
               SALARY                                                    X
               FROM                                                      X
               DSN8.TEMPL                                                X
               WHERE                                                     X
               EMPNO                                                     X
               =:AEM00011
         EXEC  SQL                                                       X
               OPEN CDELETE
         ICM   R14,15,SQLCODE
         BM    RETURN                 SQLCODE < 0
         CLC   SQLCODE,SC100
         BE    RETURN                 SQLCODE = 100
         EXEC  SQL                                                       X
               FETCH CDELETE                                             X
               INTO                                                      X
               :AEM0001$                                                , X
               :AEM0002$                                                , X
               :AEM0003$                                                , X
               :AEM0004$                                                , X
               :AEM0005$                                                , X
               :AEM0006$:AEM0006@                                       , X
               :AEM0007$:AEM0007@                                       , X
               :AEM0008$:AEM0008@                                       , X
               :AEM0009$:AEM0009@                                       , X
               :AEM0010$:AEM0010@                                       , X
               :AEM0011$:AEM0011@                                       , X
               :AEM0012$:AEM0012@
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CDELETED FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CDELETED EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                       X
               DELETE                                                    X
               FROM                                                      X
               DSN8.TEMPL                                                X
               WHERE CURRENT OF CDELETE
         LA    R15,RC24               DELETE FUNCTION RETURN CODE
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        CDELETEC FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
CDELETEC EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                       X
```

```
               CLOSE CDELETE
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC10 FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC10 EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               DECLARE KSELECT1 CURSOR FOR                             X
               SELECT                                                  X
               EMPNO                                              ,    X
               FIRSTNME                                           ,    X
               MIDINIT                                            ,    X
               LASTNAME                                           ,    X
               WORKDEPT                                           ,    X
               PHONENO                                            ,    X
               HIREDATE                                           ,    X
               JOBCODE                                            ,    X
               EDUCLVL                                            ,    X
               SEX                                                ,    X
               BRTHDATE                                           ,    X
               SALARY                                                  X
               FROM                                                    X
               DSN8.TEMPL                                              X
               WHERE                                                   X
               (                                                       X
               EMPNO                                                   X
               >= :AEM00011                                            X
               )                                                       X
               ORDER BY                                                X
               EMPNO                                             ASC
         EXEC  SQL                                                     X
               OPEN KSELECT1
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC1F FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC1F EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               FETCH KSELECT1                                          X
               INTO                                                    X
               :AEM0001$                                             , X
               :AEM0002$                                             , X
               :AEM0003$                                              , X
               :AEM0004$                                              , X
               :AEM0005$                                              , X
               :AEM0006$:AEM0006@                                     , X
               :AEM0007$:AEM0007@                                     , X
               :AEM0008$:AEM0008@                                     , X
               :AEM0009$:AEM0009@                                     , X
               :AEM0010$:AEM0010@                                     , X
               :AEM0011$:AEM0011@                                     , X
               :AEM0012$:AEM0012@
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC1C FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC1C EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               CLOSE KSELECT1
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC2O FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC2O EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         EXEC  SQL                                                     X
               DECLARE KSELECT2 CURSOR FOR                             X
               SELECT                                                  X
               EMPNO                                              ,    X
               FIRSTNME                                           ,    X
               MIDINIT                                            ,    X
               LASTNAME                                           ,    X
               WORKDEPT                                           ,    X
               PHONENO                                            ,    X
               HIREDATE                                           ,    X
               JOBCODE                                            ,    X
               EDUCLVL                                            ,    X
               SEX                                                ,    X
               BRTHDATE                                           ,    X
               SALARY                                                  X
               FROM                                                    X
               DSN8.TEMPL                                              X
               WHERE                                                   X
               (                                                       X
               EMPNO                                                   X
```

```
            LIKE :AEM00012 AND                                        X
            EMPNO                                                     X
            >= :AEM00011                                              X
            )                                                         X
            ORDER BY                                                  X
            EMPNO                                                 ASC
      EXEC  SQL                                                       X
            OPEN KSELECT2
      B     RETURN
      DROP  R8
      DROP  R9
      DROP  R10
      DROP  R11
      LTORG
      EJECT
***********************************************************************
*                                                                     *
*       KSELEC2F FOR TABLE EM                                         *
*                                                                     *
***********************************************************************
KSELEC2F EQU   *
      BALR  R11,0
      USING *,R11
      USING *+4095,R10
      USING *+8190,R9
      USING *+12285,R8
      LA    R10,4095(R11)
      LA    R9,4095(R10)
      LA    R8,4095(R9)
      EXEC  SQL                                                       X
            FETCH KSELECT2                                            X
            INTO                                                      X
            :AEM0001$                                                ,X
            :AEM0002$                                                ,X
            :AEM0003$                                                ,X
            :AEM0004$                                                ,X
            :AEM0005$                                                ,X
            :AEM0006$:AEM0006@                                       ,X
            :AEM0007$:AEM0007@                                       ,X
            :AEM0008$:AEM0008@                                       ,X
            :AEM0009$:AEM0009@                                       ,X
            :AEM0010$:AEM0010@                                       ,X
            :AEM0011$:AEM0011@                                       ,X
            :AEM0012$:AEM0012@
      B     RETURN
      DROP  R8
      DROP  R9
      DROP  R10
      DROP  R11
      LTORG
      EJECT
***********************************************************************
*                                                                     *
*       KSELEC2C FOR TABLE EM                                         *
*                                                                     *
***********************************************************************
KSELEC2C EQU   *
      BALR  R11,0
      USING *,R11
      USING *+4095,R10
      USING *+8190,R9
      USING *+12285,R8
      LA    R10,4095(R11)
      LA    R9,4095(R10)
      LA    R8,4095(R9)
      EXEC  SQL                                                       X
            CLOSE KSELECT2
      B     RETURN
      DROP  R8
      DROP  R9
      DROP  R10
      DROP  R11
      LTORG
```

```
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC30 FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC30 EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         USING SAMPEMK3,R5
         EXEC  SQL                                                     X
               DECLARE KSELECT3 CURSOR FOR                             X
               SELECT                                                  X
               EMPNO                                           ,       X
               FIRSTNME                                        ,       X
               MIDINIT                                         ,       X
               LASTNAME                                        ,       X
               WORKDEPT                                        ,       X
               PHONENO                                         ,       X
               HIREDATE                                        ,       X
               JOBCODE                                         ,       X
               EDUCLVL                                         ,       X
               SEX                                             ,       X
               BRTHDATE                                        ,       X
               SALARY                                                  X
               FROM                                                    X
               DSN8.TEMPL                                              X
               WHERE EMPNO >= :AP1EMP#3 AND WORKDEPT = :AP1DEPT3       X
               ORDER BY EMPNO ASC, WORKDEPT ASC
         EXEC  SQL                                                     X
               OPEN KSELECT3
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC3F FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC3F EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         USING SAMPEMK3,R5
         EXEC  SQL                                                     X
               FETCH KSELECT3                                          X
               INTO                                                    X
               :AEM0001$                                             , X
               :AEM0002$                                             , X
               :AEM0003$                                             , X
               :AEM0004$                                             , X
               :AEM0005$                                             , X
               :AEM0006$:AEM0006@                                    , X
               :AEM0007$:AEM0007@                                    , X
               :AEM0008$:AEM0008@                                    , X
               :AEM0009$:AEM0009@                                    , X
               :AEM0010$:AEM0010@                                    , X
               :AEM0011$:AEM0011@                                    , X
               :AEM0012$:AEM0012@
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
***********************************************************************
*                                                                     *
*        KSELEC3C FOR TABLE EM                                        *
*                                                                     *
***********************************************************************
KSELEC3C EQU   *
         BALR  R11,0
         USING *,R11
         USING *+4095,R10
         USING *+8190,R9
         USING *+12285,R8
         LA    R10,4095(R11)
         LA    R9,4095(R10)
         LA    R8,4095(R9)
         USING SAMPEMK3,R5
         EXEC  SQL                                                     X
               CLOSE KSELECT3
         B     RETURN
         DROP  R8
         DROP  R9
         DROP  R10
         DROP  R11
         LTORG
         EJECT
SQLCAD   DSECT ,
         EXEC  SQL INCLUDE SQLCA
*                                    DSNHLI INTERFACE - ADSNHLI CONTAINS
*                                    THE ADDRESS OF THE SINGLE COPY OF
*                                    DFSLI000 (ENTRY POINT - DSNHLI)
*                                    IN LOAD MODULE ????V50.
MFC1EDSN CSECT                       CSECT FOR DSNHLI (NOTHING ELSE)
         DS    0H
         ENTRY DSNHLI                TELL LKED ABOUT IT
DSNHLI   EQU   *                     GET IN HERE
         USING *,15                  BASE IS R15 TO FIND THE VCON
         L     15,ADSNHLI            LOCATE THE DSNHLI ADDRESS
         DROP  15                    FORGET R15, IT'S NOT USABLE NOW
         BR    15                    GO TO DSNHLI AND RETURN TO
*                                    MAIN TABLE HANDLER RULE CSECT
         ENTRY ADSNHLI
ADSNHLI  DC    F'0'
         END
```

APPENDIX B. SPECIFICATION EXAMPLE

To demonstrate the potential use of IMSADF II accessing DB2 data bases, reference the following examples. Three transactions are defined using the DB2 distributed Employee Table (DSN8.TEMPL).

The first transaction 'EM' provides FETCH, UPDATE, INSERT, and DELETE functions for a row of the Employee Table.

Additionally, the 'EM' transaction provides several types of secondary key selection browsing against the Employee Table.

- all employees in the Table
- all employees in a requested work department in the Table
- all employees in the table that satisfy the LIKE PREDICATE entered, (that is, %000%% or 000___)

Note: All secondary key selection browse displays are ordered by the EMPNO Column in ascending sequence.

The second transaction 'ES' provides 3 simple queries on the Employee Table using the DB2 built in functions, COUNT, SUM, and AVG. Values are displayed for 3 derived Columns, COUNT of employees, SUM(SALARY), and AVG(SALARY) based on WORKDEPT, SEX or EDUCLVL.

The third transaction 'EX' demonstrates that both DL/I and DB2 data can be displayed and updated from the same transaction display screen. This transaction processes the Employee Table using the same Table Layout and Table Handler Rules defined in the 'EM' transaction.

Appendix C, "Sample Problem" on page C-1 contains the screen flow associated with the first two sample transactions, EM and ES.

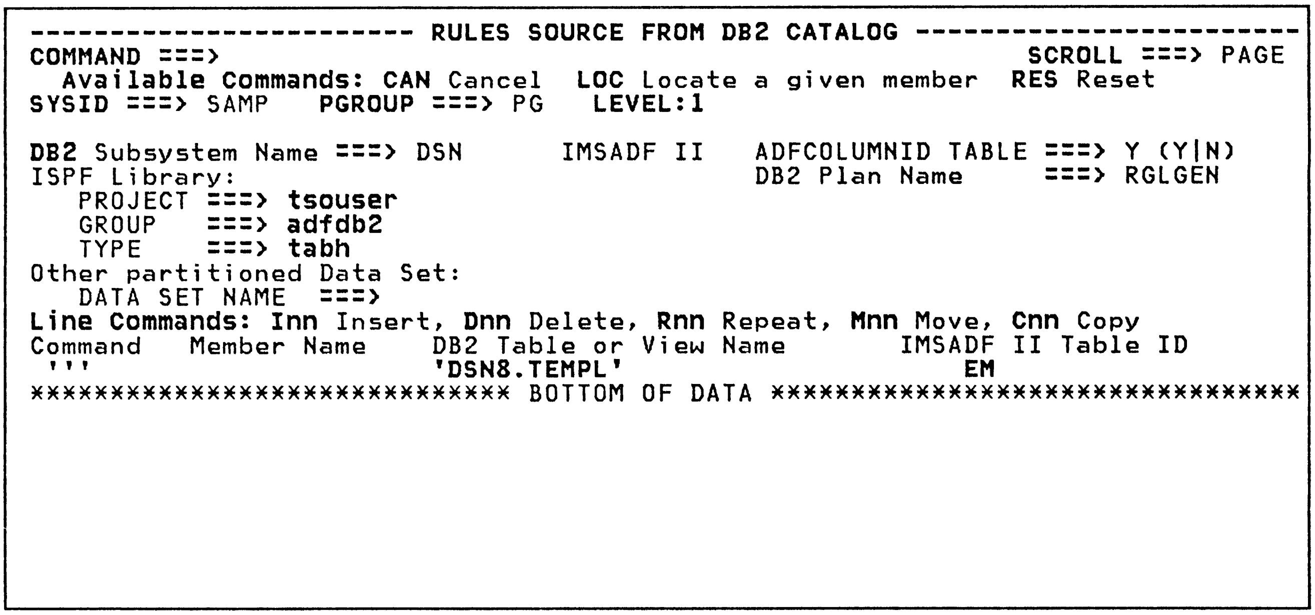
DB2 EMPLOYEE TABLE DEFINITION

Following is the DB2 Sample Problem Employee Table definition:

```
     CREATE TABLE DSN8.TEMPL
          (EMPNO     CHAR(6),       NOT NULL,
           FIRSTNME  VARCHAR(12)    NOT NULL,
           MIDINIT   CHAR(1),       NOT NULL,
           LASTNAME  VARCHAR(15)    NOT NULL,
           WORKDEPT  CHAR(3),       NOT NULL,
           PHONENO   CHAR(4),
           HIREDATE  DECIMAL(6),
           JOBCODE   DECIMAL(3),
           EDUCLVL   SMALLINT,
           SEX       CHAR(1),
           BRTHDATE  DECIMAL(6),
           SALARY    DECIMAL(8,2)
       VALIDPROC DSN8EAV1
       EDITPROC  DSN8EAE1
       IN DSN8DAPP.DSN8SEMP ;

     CREATE UNIQUE INDEX DSN8.XEMPL1
            ON DSN8.TEMPL
            (EMPNO ASC) CLUSTER;

     CREATE INDEX DSN8.XEMPL2
            ON DSN8.TEMPL
            (WORKDEPT ASC) ;
```

To extract Rules Generator source statements from the DB2 catalog for the DSN8.TEMPL Table use the IMSADF II RGLGEN utility. Refer to the RGLGEN Utility chapter for complete details.

Enter the following information on the IADF RGLGEN GENERATION panel to invoke the RGLGEN Utility.



```
----------------------- RULES SOURCE FROM DB2 CATALOG -----------------------
COMMAND ===>                                                    SCROLL ===> PAGE
  Available Commands: CAN Cancel  LOC Locate a given member  RES Reset
SYSID ===> SAMP   PGROUP ===> PG   LEVEL:1

DB2 Subsystem Name ===> DSN      IMSADF II   ADFCOLUMNID TABLE ===> Y (Y|N)
ISPF Library:                                DB2 Plan Name     ===> RGLGEN
   PROJECT ===> tsouser
   GROUP   ===> adfdb2
   TYPE    ===> tabh
Other partitioned Data Set:
   DATA SET NAME  ===>
Line Commands: Inn Insert, Dnn Delete, Rnn Repeat, Mnn Move, Cnn Copy
Command   Member Name    DB2 Table or View Name       IMSADF II Table ID
 '''                     'DSN8.TEMPL'                      EM
****************************** BOTTOM OF DATA ********************************
```

Figure B-1. Rules Source from DB2 Catalog Panel

In this example the output from the RGLGEN Utility is stored as member SAMPTBEM in the specified partitioned data set 'TSOUSER.ADFDB2.TABH'. The optional SYSADF.ADFCOLUMNID Table is used to create the IMSADF II Column ID's.

RULES GENERATOR INPUT

The following represents the Rules Generator input stream used to define the three sample DB2 transactions.

NOTE: The Rules Generator source statements required to create the following sample DB2 transactions are supplied when IMSADF II is installed.

- The IMSADF.RULES.SOURCE library, member RGLDB2S contains the Rules Generator source.
- The IMSADF.ADFIMG library, members EM, ES, and EX contain the associated screen image source.
- The Rules Generator source statements for the Employee Table are included in the RGLDB2S source member. However, one of the purposes of this appendix is to demonstrate using the RGLGEN Utility. Therefore this appendix shows the source being included using the Rules Generator INCLUDE function.

  The source extracted from the DB2 catalog is not complete. The FIELD MERGE and SEGMENT OVERRIDE functions of the Rules Generator are used to complete the definition of the DSN8.TEMPL Table.

```
***********************************************************************
*                                                                     *
*   APPLICATION DEFINITION - DLI PARTS DATA BASE - PA SEGMENT         *
*                            DB2 DSN8  DATA BASE - EM TABLE           *
*                                                                     *
***********************************************************************
*
SYSTEM   SYSID=SAMP,DBID=PA,                             RULE ID CHARS
         USRLANG=E,                                FORCE TO ENGLISH
         SOMTX=OR,                                 DEFAULT SECONDARY OPTION CODE
         PCBNO=1,                                  PCB NUMBER FOR DATA BASE
         SHEADING='S A M P L E   P R O B L E M',      GENERAL HEADING
         SFORMAT=DASH,                                   SCREEN FORMAT
         PGROUP=ZZ                                       PROJECT GROUP
*
***********************************************************************
*                                                                     *
*   APPLICATION DEFINITION INPUT FOR PA ROOT SEGMENT                  *
*                                                                     *
***********************************************************************
*
SEGMENT ID=PA,PARENT=0,NAME=PARTROOT,LENGTH=50,SKSEG=18
  FIELD ID=KEY,LENGTH=17,KEY=YES,NAME=PARTKEY,SNAME='PART NUMBER'
  FIELD ID=DESC,LENGTH=20,POS=27,SNAME='DESCRIPTION',REL=YES
*
INCLUDE MEMBERS=(SAMPTBEM)
*** expansion follows
***********************************************************************
*                                                                     *
*    TABLE:     ID=EM      DATE:   12/25/84    TIME:   00:04:29       *
*            NAME=DSN8.TEMPL                    MEMBER=SAMPTBEM       *
*                                                                     *
***********************************************************************
TABLE  ID=EM,TYPE=TBL,SQLNAME='DSN8.TEMPL',SQLIND=YES
COLUMN ID=0001,SQLNAME='EMPNO',SNAME='EMPNO',
       TYPE=C,LENGTH=006,SQLUPD=NO
COLUMN ID=0002,SQLNAME='FIRSTNME',SNAME='FIRSTNME',
       TYPE=V,LENGTH=012
COLUMN ID=0003,SQLNAME='MIDINIT',SNAME='MIDINIT',
       TYPE=C,LENGTH=001
COLUMN ID=0004,SQLNAME='LASTNAME',SNAME='LASTNAME',
       TYPE=V,LENGTH=015
COLUMN ID=0005,SQLNAME='WORKDEPT',SNAME='WORKDEPT',
       TYPE=C,LENGTH=003
COLUMN ID=0006,SQLNAME='PHONENO',SNAME='PHONENO',
       TYPE=C,LENGTH=004,SQLNULL=YES
COLUMN ID=0007,SQLNAME='HIREDATE',SNAME='HIREDATE',
       TYPE=P,LENGTH=004,SQLNULL=YES
COLUMN ID=0008,SQLNAME='JOBCODE',SNAME='JOBCODE',
       TYPE=P,LENGTH=002,SQLNULL=YES
COLUMN ID=0009,SQLNAME='EDUCLVL',SNAME='EDUCLVL',
       TYPE=I,LENGTH=002,SQLNULL=YES
COLUMN ID=0010,SQLNAME='SEX',SNAME='SEX',
       TYPE=C,LENGTH=001,SQLNULL=YES
COLUMN ID=0011,SQLNAME='BRTHDATE',SNAME='BRTHDATE',
       TYPE=P,LENGTH=004,SQLNULL=YES
COLUMN ID=0012,SQLNAME='SALARY',SNAME='SALARY',
       TYPE=P,LENGTH=005,DEC=02,SQLNULL=YES
*** expansion ends

*
* FIELD MERGE INFORMATION FOR 'EM' KEY SELECTION
*
 COLUMN ID=0001,KEY=YES,SNAME='EMPLOYEE SEARCH DATA',COL=01
 COLUMN ID=0004,RELATED=YES,COL=16    **related field on sks
 COLUMN ID=0005,RELATED=YES,COL=33    **related field on sks
 COLUMN ID=0012,RELATED=YES,COL=43    **related field on sks
*
* SEGMENT OVERRIDE INFORMATION FOR 'EM' KEY SELECTION
*
TABLE    OVERRIDE=ID,ID=EM,SKSEGS=15,
         SKLEFT='EMPLOYEE           LAST           WORK',
         SKLEFT='NUMBER             NAME           DEPT',
        SKRIGHT='       SALARY'
*
* PSEUDO SEGMENT DEFINITION - P1
*
* IF the user defined secondary key selection KSELECT3 SQL function
* is specified, the fields in this pseudo segment represent the
* Host Variables defined in the KSELECT3 WHERE clause.
*
* A key audit is used to move values to these fields.
*
SEGMENT ID=P1,TYPE=PS
 FIELD  ID=KS3,TYPE=C,LENGTH=09
 FIELD  ID=EMP#,TYPE=C,LENGTH=06,REDEF=KS3
 FIELD  ID=DEPT,TYPE=C,LENGTH=03,REDEF=KS3,OFFSET=6
*
* TABLE DEFINITION - ES
*
* This Table defines a VIEW of the DB2 Sample Employee
* Table (DSN8.TEMPL).  The defined Columns are derived
* using DB2 functions.
*
* This Table is processed by the 'ES' transaction.  It
* demonstrates the use of USER defined SQL functions.
*
* NOTE: This VIEW definition does not exist in the DB2
* catalog.  The RGLGEN Utility cannot be used.
*
* Even though no key is required for processing this Table,
* IMSADF II still requires that a key field be defined.
*
* SQLNULL=YES is specified for the derived Columns based
* on the SALARY Column because the SALARY Column in the
* DSN8.TEMPL Table is defined eligible for the NULL value.
*
TABLE   ID=ES,TYPE=TBL,SQLNAME='DSN8.TEMPL',SQLIND=YES
 COLUMN ID=ECNT,TYPE=I,LENGTH=04,SQLNAME='COUNT(*)'
 COLUMN ID=SSAL,TYPE=P,LENGTH=07,DEC=2,SQLNAME='SUM(SALARY)',
        SQLNULL=YES
 COLUMN ID=ASAL,TYPE=P,LENGTH=05,DEC=2,SQLNAME='AVG(SALARY)',
        SQLNULL=YES
 FIELD  ID=DKEY,TYPE=P,LENGTH=05,REDEF=ASAL,KEY=YES
*
* PSEUDO SEGMENT DEFINITION - P2
*
* The fields in this pseudo segment represent the Host Variables
* required to process the 3 USER defined SQL WHERE clauses in
* the ES Table Handler Rule.
*
* A process audit is used to trigger the USER defined SQL
* functions in the ES Table Handler Rule.
*
SEGMENT ID=P2,TYPE=PS
 FIELD  ID=DEPT,TYPE=C,LENGTH=03,AUDIT=YES
 FIELD  ID=JOBC,TYPE=P,LENGTH=02,AUDIT=YES
 FIELD  ID=EDUL,TYPE=I,LENGTH=02,AUDIT=YES
*
*********************************************************************
*                                                                   *
*     GENERATE SEGMENT LAYOUT AND HANDLER RULES FOR SEGMENT - PA    *
*     GENERATE SEGMENT LAYOUT RULE FOR PSEUDO SEGMENT - P1, P2      *
*     GENERATE TABLE LAYOUT RULE FOR DB2 TABLE - EM, ES             *
*                                                                   *
*********************************************************************
*
GENERATE OPTIONS=(SEGL,SEGH),SEGMENT=PA
GENERATE OPTIONS=SEGL,SEGMENTS=(P1,P2)
GENERATE OPTIONS=TABL,TABLES=(EM,ES)
*
* The following GENERATE OPTION=TABH Builds a Table Handler Rule
* for Table ID 'EM'.  One user defined SQL function is specified.
* The LABEL indicates it is to be used during Secondary Key
* Selection browse.
*
* THE SQLCALL operand implies that the following standard
* IMSADF II SQL functions are included in the Table Handler Rule -
* (CSELECT, INSERT, CUPDATE, CDELETE, KSELECT1, and KSELECT2).
*
GENERATE OPTIONS=TABH,TABLES=EM,SQLCALL=(DSQLCALL,KSELECT2),
         SQLUSER=YES,ASMREQ=YES
KSELECT3 SELECT   WHERE EMPNO >= :EMP#.P1 AND WORKDEPT = :DEPT.P1
               ORDER BY EMPNO ASC, WORKDEPT ASC
&SQLENDS
*
* The following GENERATE OPTION=TABH builds a Table Handler Rule
* for Table ID 'ES'.  3 USER defined SQL functions are specified.
*
* No standard IMSADF II SQL functions are defined for this Table.
*
GENERATE OPTIONS=TABH,TABLES=ES,SQLCALL=NONE,SQLUSER=YES,ASMREQ=YES
DEPTSELC SELECT   WHERE WORKDEPT = :DEPT.P2
JOBCSELC SELECT   WHERE JOBCODE = :JOBC.P2
EDULSELC SELECT   WHERE EDUCLVL >= :EDUL.P2
&SQLENDS
*
* The following GENERATE OPTION=CVALL builds a conversational
* Input Transaction Rule and MFS source for the Primary Key
* Selection and Transaction display screens for the EM
* Transaction.
*
GENERATE TRXID=EM,OPTIONS=CVALL,DBPATH=(EM),TSEGS=(P1),
         TRXNAME='DB2 EMPLOYEE TABLE',
         DEVNAME=(2,A3,A4),
         DEVCHRS=(0,0,0),
         DEVTYPE=(2,7,4),SPOS=SIMAGE
               'D B 2  'S A M P L E    'P R O B L E M
&=1
 &MODE                      TRANSACTION: 'DB2 'EMPLOYEE 'TABLE
 OPTION:&OPT TRX:&TRAN KEY:&KEY
    &SYSMSG
&=1
        EMPLOYEE NUMBER-------&50001.EM
&=1
        FIRST NAME------------&50002.EM
        MIDDLE INITIAL--------&50003.EM
        LAST NAME-------------&50004.EM
        DEPARTMENT NUMBER-----&50005.EM
        PHONE EXTENSION-------&50006.EM
        DATE HIRED------------&50007.EM
        JOB CODE--------------&50008.EM
        EDUCATION LEVEL-------&50009.EM
        SEX-------------------&50010.EM
        BIRTH DATE------------&50011.EM
        SALARY----------------&50012.EM
&*         1     1     2      3      3       4    5   5
&*---------2-----8-----4------1------8-------6----1---5
&*   ID   *SLEN *VROW *VCOL  *VMODE *ASTATUS*KSEL*CLR*XHL
&:0001.EM                                     KA  P
&:0002.EM                                         Y
&:0003.EM                                         Y
&:0004.EM                                         Y
&:0005.EM                                         Y
&:0006.EM                                         Y
&:0007.EM   6                                     Y
&:0008.EM   3                                     Y
&:0009.EM   5                                     Y
&:0010.EM                                         Y
&:0011.EM   6                                     Y
&:0012.EM   10                                    Y
&ENDS
*
* The following GENERATE OPTION=CVALL builds a conversational
* Input Transaction Rule and MFS source for the Transaction
* display screen for the ES Transaction.
*
* The ES Transaction is defined without Key Select Functions.
*
GENERATE TRXID=ES,OPTIONS=CVALL,KEYSL=NO,TSEGS=(ES,P2),DTRAN=NO,
         DKEY=NO,TRXNAME='DB2 EMPLOYEE STATISTICS',
         DEVNAME=(2,A3,A4),
         DEVCHRS=(0,0,0),
         DEVTYPE=(2,7,4),SPOS=SIMAGE
               'D B 2   'S A M P L E   'P R O B L E M
                      'EMPLOYEE 'STATISTICS
&=1
 OPTION:&OPT
    &SYSMSG
&=2
  COUNT OF EMPLOYEES        TOTAL SALARY        AVERAGE SALARY
       &6ECNT.ES        &6SSAL.ES               &6ASAL.ES
&=2
  ENTER ONE OF THE FOLLOWING:
&=1
     BY  WORK DEPARTMENT----: &4DEPT.P2
             OR
         JOB CODE (1 TO 60)-: &4JOBC.P2
             OR
         EDUCATION LEVEL----: &4EDUL.P2
         11 = LESS THAN HIGH SCHOOL
         12 = HIGH SCHOOL
         16 = COLLEGE DEGREE
         18 = MASTERS DEGREE
         20 = P.H.D.
&*          1     1     2      3      3       4    5   5
&*---------2-----8-----4------1------8-------6----1---5
&*   ID   *SLEN *VROW *VCOL  *VMODE *ASTATUS*KSEL*CLR*XHL
&:ECNT.ES    5                                   Y   U
&:SSAL.ES                                        Y   U
&:ASAL.ES                                        Y   U
&:DEPT.P2                                        R   R
&:JOBC.P2                                        R   R
&:EDUL.P2                                        R   R
&ENDS
*
* The following GENERATE OPTION=CVALL builds a conversational
* Input Transaction Rule and MFS source for the Primary Key
* Selection and Transaction display screens for the EX
* Transaction.
*
* This transaction allows both DL/I and DB2 data to be displayed
* and updated from the same transaction display screen.
*
GENERATE TRXID=EX,OPTIONS=CVALL,DBPATH=(PA,EM),TSEGS=(P1),
         TRXNAME='DLI-PART / DB2-EMPLOYEE',DATACOMP=(EM),
         DEVNAME=(2,A3,A4),
         DEVCHRS=(0,0,0),
         DEVTYPE=(2,7,4),SPOS=SIMAGE
               'D B 2   'S A M P L E   'P R O B L E M
&=1
 &MODE                      TRANSACTION: 'DLI PART / 'DB2 EMPLOYEE
 OPTION:&OPT TRX:&TRAN KEY:&KEY
    &SYSMSG
&=1
   DLI SEGMENT      PART NUMBER-----------&5KEY.PA
                    DESCRIPTION-----------&5DESC.PA
&=2
   DB2 TABLE        EMPLOYEE NUMBER-------&50001.EM
                    FIRST NAME------------&50002.EM
                    MIDDLE INITIAL--------&50003.EM
                    LAST NAME-------------&50004.EM
                    DEPARTMENT NUMBER-----&50005.EM
                    PHONE EXTENSION-------&50006.EM
                    DATE HIRED------------&50007.EM
                    JOB CODE--------------&50008.EM
                    EDUCATION LEVEL-------&50009.EM
                    SEX-------------------&50010.EM
                    BIRTH DATE------------&50011.EM
                    SALARY----------------&50012.EM
&*          1     1     2      3      3       4    5   5
&*---------2-----8-----4------1------8-------6----1---5
&*   ID   *SLEN *VROW *VCOL  *VMODE *ASTATUS*KSEL*CLR*XHL
&:KEY.PA                                         P
&:DESC.PA                                        G
```

```
&:0001.EM                                        KA  P
&:0002.EM                                            Y
&:0003.EM                                            Y
&:0004.EM                                            Y
&:0005.EM                                            Y
&:0006.EM                                            Y
&:0007.EM   6                                        Y
&:0008.EM   3                                        Y
&:0009.EM   5                                        Y
&:0010.EM                                            Y
&:0011.EM   6                                        Y
&:0012.EM   10                                       Y
&ENDS
*
* GENERATE the Secondary Option Menu Rule
*
GENERATE OPTIONS=SOM
*
* GENERATE the SIGNON screen and Primary Option Menu Rule
*
GENERATE OPTIONS=CVSYS
*
* GENERATE the Conversational Mini-Driver
*
GENERATE OPTIONS=STLE,PGMID=OR
```

TRANSACTION HELP FACILITY

The following is an example of the IMSADF II HELP facility for the EM transaction. The HELP facility is available in conversational processing, to describe the purpose and input requirements for the currently displayed screen. HELP is invoked through the entry of '?' in the OPTION field. The screen on which the '?' is entered will determine which help text will be retrieved and displayed.

This example will be displayed if a '?' were entered in the OPTION field on either the Primary or Secondary key selection screen for the EM transaction. The second page of this example will be displayed if the user presses the PF1 key.

```
*FOLLOWING IS THE HELP DATA TO BE ENTERED IN THE MESSAGE DATA BASE
*REPRESENTED BY BATCH DRIVER INPUT
MFC1B4HESAMPPEM@
MFC1B4HTSAMPPEM@0001
         H E L P   F O R   T R X I D   E M
          D B 2   S A M P L E  P R O B L E M
           K E Y   S E L E C T I O N   S C R E E N S
*
PRIMARY KEY SELECTION
*
EMPLOYEE SEARCH DATA ______
      ENTRY              ACTION
      NNNNNN             WHERE N IS 0 to 9 - EMPLOYEE DATA IS
                         DISPLAYED ON TRANSACTION DISPLAY
      NNNNN> OR <        WHERE N IS 1 TO 5 NUMBERS - SECONDARY KEY
                         SELECTION WITH EMPLOYEES STARTING WITH
                         THE PARTIAL KEY AND GREATER
      N_____ OR %        WHERE N IS 1 to 5 NUMBERS - SECONDARY KEY
                         SELECTION WITH EMPLOYEES WITH THESE CORRECT
                         PARTIAL VALUES
      DNNN               WHERE D INVOKES USER DEFINED KSELECT3 -
                         D INDICATES DISPLAY ALL EMPLOYEES IN THE
                         NNN DEPARTMENT ON THE SECONDARY KEY
                         SELECTION BROWSE SCREEN

PRESS PF1 TO DISPLAY NEXT PAGE $$
MFC1B4HTSAMPPEM@0002
         H E L P   F O R   T R X I D   E M
          D B 2   S A M P L E  P R O B L E M
*
SECONDARY KEY SELECTION
*
THE SECONDARY KEY SELECTION SCREEN DISPLAYS EMPLOYEE INFORMATION.
IN A TABULAR FORM. A SELECTION MAY BE MADE FROM THE CURRENT PAGE,
THE NEXT PAGE MAY BE DISPLAYED, OR THE TRX AND KEY CAN BE CHANGED
TO SWITCH TO A NEW TRANSACTION.
*
          EMPLOYEE          LAST           WORK        SALARY
          NUMBER            NAME           DEPT
 NNN      ______     _______________       __      ________.__
```

HIGH LEVEL AUDIT LANGUAGE

The following are samples of High Level Audit Language required to execute the 'EM', 'ES' and 'EX' transactions.

EM and EX Transactions

Test to see if the User enters a 'D' in the first position of the Employee Number field, (SAEM0001). If he has then set up to invoke the USER defined KSELECT3 function.

```
*  HIGH LEVEL AUDIT LANGUAGE FOR THE DB2 EM AND EX TRANSACTIONS

SYSID = SAMP
XPANDLBLS = YES
AGROUP = YYYY
SEGID = EM
FIELD = 0001
* FIELD 0001 = DB2 COLUMN EMPNO
KEY
*
*       PRIMARY KEY AUDIT
*
 P0
  IF SAEM0001 = NON$ 'D$$$$$'
*                             TEST FOR D IN 1ST POSITION
      SAP1DEPT = SUBSTR SAEM0001 2 : 3
*                             MOVE DEPT NUMBER TO PSEUDO SEGMENT
      SAP1EMP# = '000000'
      SPASQL = KSELECT3
*                             USE KSELECT3 FOR SEC KEY SEL
      SPAWHERE = SAP1KS3
*                             SET KSELECT3 HOST VARIABLES
  ENDIF
*
*       SECONDARY KEY AUDIT
*
 P1
  SAP1EMP# = SAEM0001
*                             SAVE FETCHED EMPNO FROM SEC KEY
*                             FOR REPOSITIONING OF NEXT SCREEN
```

ES Transaction

Invoke one of the three USER defined SQL functions defined in the ES Table Handler Rule based on the terminal input.

If the Work Department field is not blank then invoke the DEPTSELC function. If the Job Code field is not zero then invoke the JOBCSELC function. If the Education Level field is not zero then invoke the EDULSELC function.

```
  *  HIGH LEVEL AUDIT LANGUAGE FOR THE DB2 ES TRANSACTION

  SYSID = SAMP
  XPANDLBLS = YES
  AGROUP = YYYY
  SEGID = P2
  FIELD = DEPT
  PROCESS
   P1
    IF DEPT NE '   '
      IF SQL DEPTSELC SAP2DEPT ES NOT OK
        IF SQLCODE = 100
          SPAERMSG = 'EMPLOYEE STATS REQUEST COMPLETED - NO HITS'
          EXIT
        ELSE
          ERRORMSG = 1000
        ENDIF
      ELSE
        SAP2JOBC = 0
        SAP2EDUL = 0
      ENDIF
    ENDIF
  *
  FIELD = JOBC
  PROCESS
   P1
    IF JOBC NE 0
      IF SQL JOBCSELC SAP2JOBC ES NOT OK
        IF SQLCODE = 100
          SPAERMSG = 'EMPLOYEE STATS REQUEST COMPLETED - NO HITS'
          EXIT
        ELSE
          ERRORMSG = 1000
        ENDIF
      ELSE
        SAP2DEPT = '   '
        SAP2EDUL = 0
      ENDIF
    ENDIF
  *
  FIELD = EDUL
  PROCESS
   P1
    IF EDUL NE 0
      IF SQL EDULSELC SAP2EDUL ES NOT OK
        IF SQLCODE = 100
          SPAERMSG = 'EMPLOYEE STATS REQUEST COMPLETED - NO HITS'
          EXIT
        ELSE
          ERRORMSG = 1000
        ENDIF
      ELSE
        SAP2DEPT = '   '
        SAP2JOBC = 0
      ENDIF
    ENDIF
```

NOTE: Error message 1000 must be defined to the IMSADF II system.

THE BIND PROCESS

Prior to executing a DB2 transaction the associated Table Handler Rules must be defined to DB2 by creating an Application Plan.

The two Table Handler Rules defined in this example are shown here being BINDed into one DB2 Application Plan, (SAMPTOR). When IMSADF II transactions EM, EX, or ES are executed this DB2 Application Plan is used.

The following information to create the SAMPTOR Application Plan is supplied to the DB2I BIND Panel.

```
DSNEBP02                          BIND
===>

ENTER THE DBRM LIBRARY NAME(S):
1 DBRMLIB1 ===> 'db2.dbrmlib'                              2 PASSWORD1 ===>
3 DBRMLIB2 ===> 'imsadf.adfdbrm'                           4 PASSWORD2 ===>
5 DBRMLIB3 ===>                                            6 PASSWORD3 ===>
7 DBRMLIB4 ===>                                            8 PASSWORD4 ===>

ENTER THE MEMBER NAME(S) TO BE BOUND IN THIS PLAN:
         9 ===> sampsem   12 ===>          15 ===>          18 ===>
        10 ===> sampses   13 ===>          16 ===>          19 ===>
        11 ===>           14 ===>          17 ===>          20 ===>

SPECIFY OPTIONS AS DESIRED:
21   PLAN NAME ............. ===> samptor    Enter desired plan name.
22   ACTION ON PLAN ......... ===> add       Enter ADD or REPLACE.
23   RETAIN EXECUTION AUTH. . ===> yes       Enter YES to retain user list.
24   PLAN VALIDATION TIME ... ===> bind      Enter RUN or BIND.
25   ISOLATION LEVEL ........ ===> cs        Enter RR or CS.
26   MESSAGE LEVEL .......... ===> i         Enter I, W, E, or C.
27   DB2 NAME ............... ===> dsn       Enter DB2 subsystem name.
PRESS:  ENTER to process     END to exit     HELP for more information
```

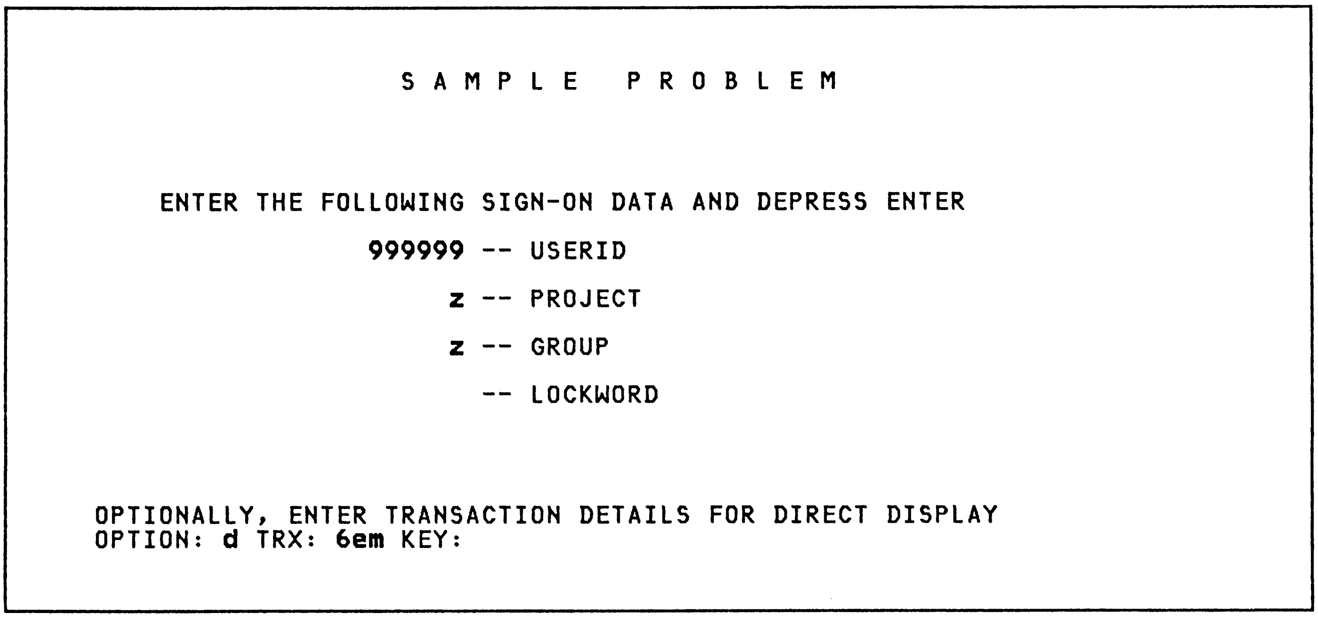
Figure B-2. DB2I Bind Panel Input

APPENDIX C. SAMPLE PROBLEM

The sample problem shown here is derived from the Specification Example defined in the previous appendix.

This sample problem screen flow demonstrates conversational transactions accessing and updating a DB2 Table.

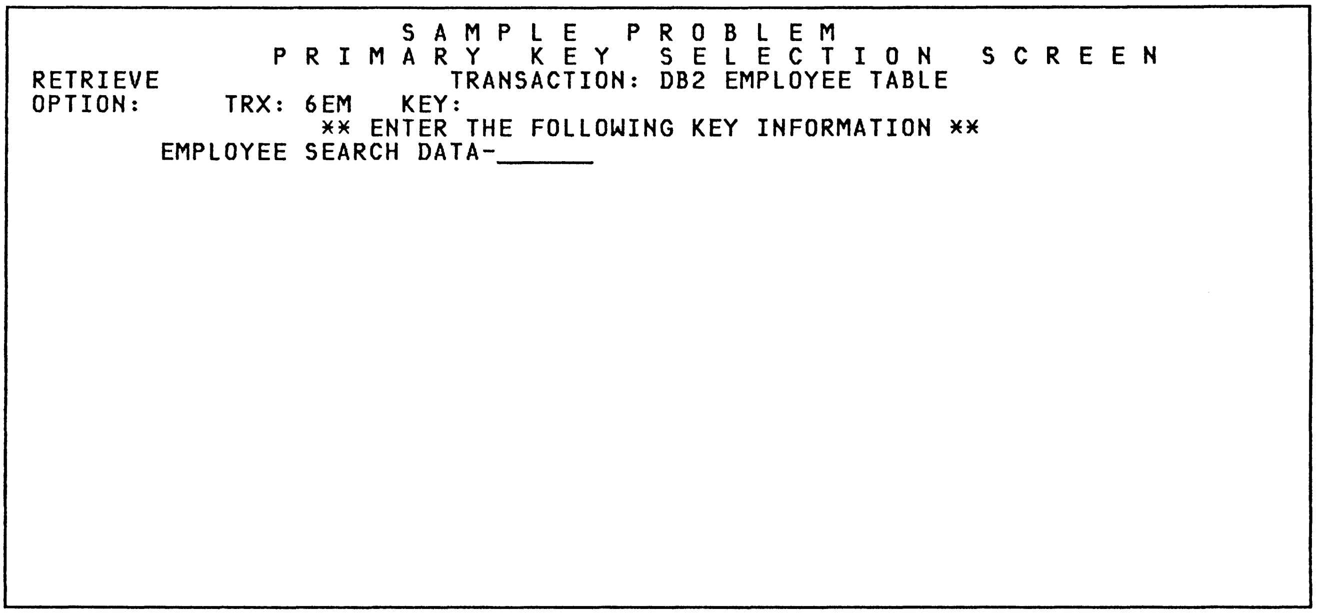
Following is the scenario of execution time screens:



```
                         S A M P L E   P R O B L E M


        ENTER THE FOLLOWING SIGN-ON DATA AND DEPRESS ENTER

                    999999 -- USERID

                         z -- PROJECT

                         z -- GROUP

                           -- LOCKWORD



     OPTIONALLY, ENTER TRANSACTION DETAILS FOR DIRECT DISPLAY
     OPTION: d TRX: 6em KEY:
```

Figure C-1. Sign-on Screen

Upon entry of USERID, PROJECT/GROUP, OPTION and TRX enough information has been provided to IMSADF II to bypass the Primary and Secondary Option menu screens. The next screen displayed is the Primary Key Selection screen for TRXID 'EM', Employee Data.

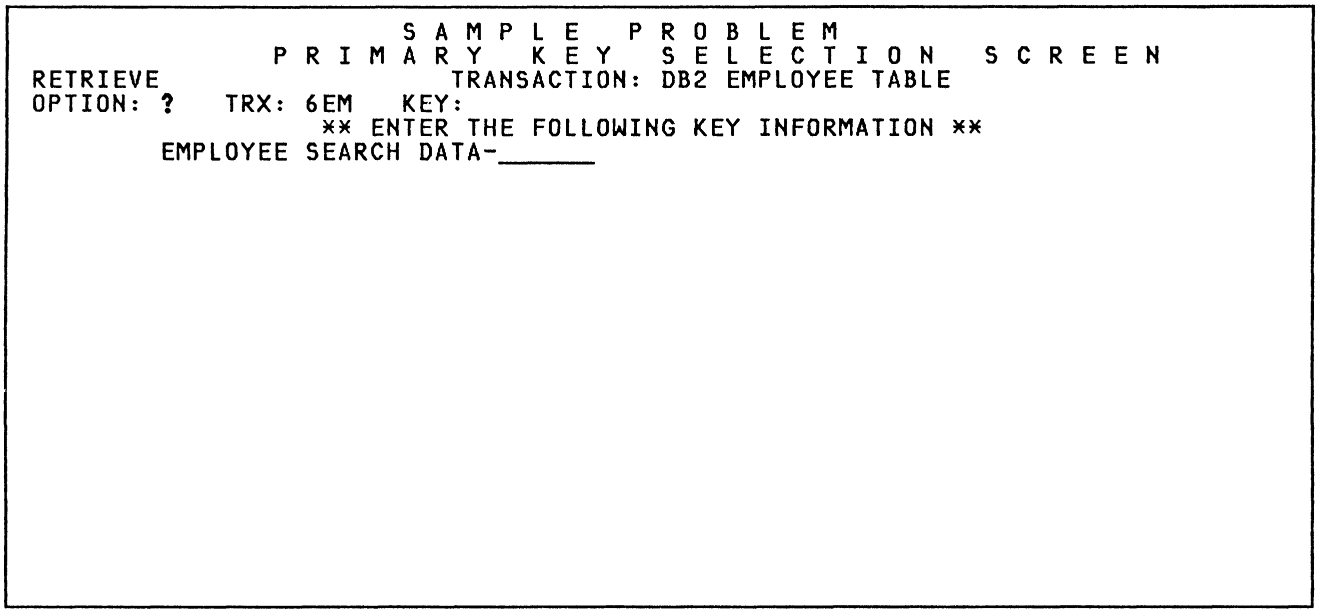


```
                    S A M P L E   P R O B L E M
              P R I M A R Y   K E Y   S E L E C T I O N   S C R E E N
RETRIEVE                   TRANSACTION: DB2 EMPLOYEE TABLE
OPTION:     TRX: 6EM   KEY:
                  ** ENTER THE FOLLOWING KEY INFORMATION **
      EMPLOYEE SEARCH DATA-______
```

Figure C-2. Primary Key Selection, Transaction EM

Enter the required Employee Search Data Key.

In this example the terminal operator does not know what is required and instead enters a '?' in the OPTION field.

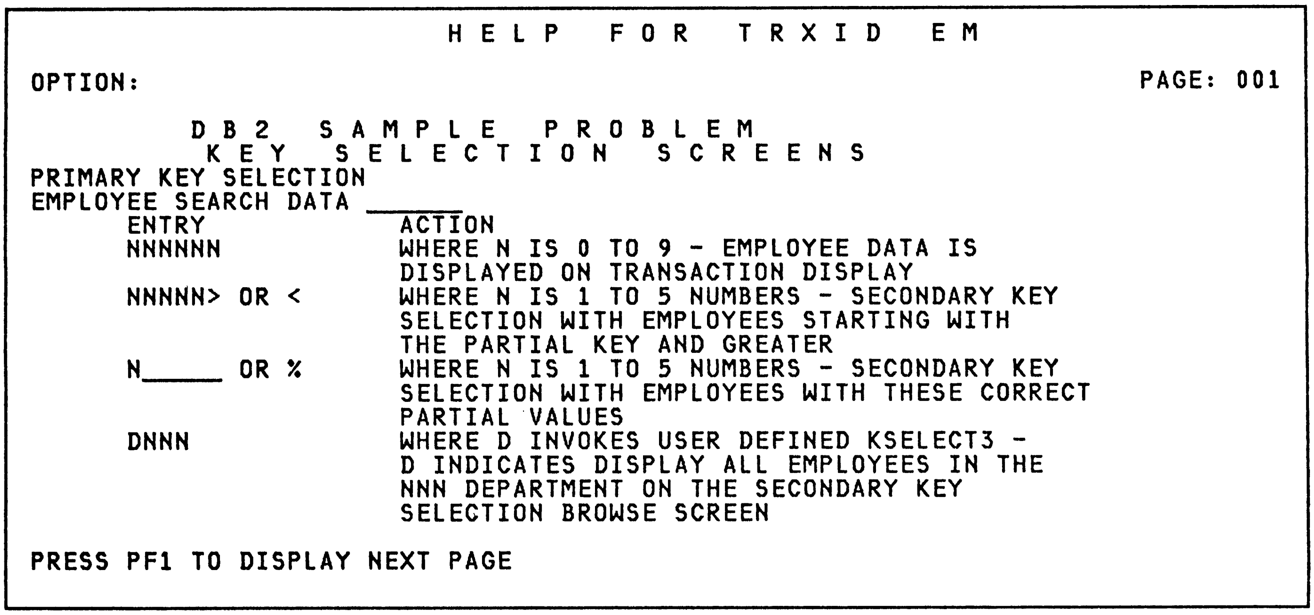


```
                    S A M P L E   P R O B L E M
              P R I M A R Y   K E Y   S E L E C T I O N   S C R E E N
RETRIEVE                   TRANSACTION: DB2 EMPLOYEE TABLE
OPTION: ?   TRX: 6EM   KEY:
                  ** ENTER THE FOLLOWING KEY INFORMATION **
      EMPLOYEE SEARCH DATA-______
```

Figure C-3. Primary Key Selection, Transaction EM, HELP REQUEST

By entering a '?' in the OPTION field the terminal operator has requested the IMSADF II HELP facility for the EM transaction.

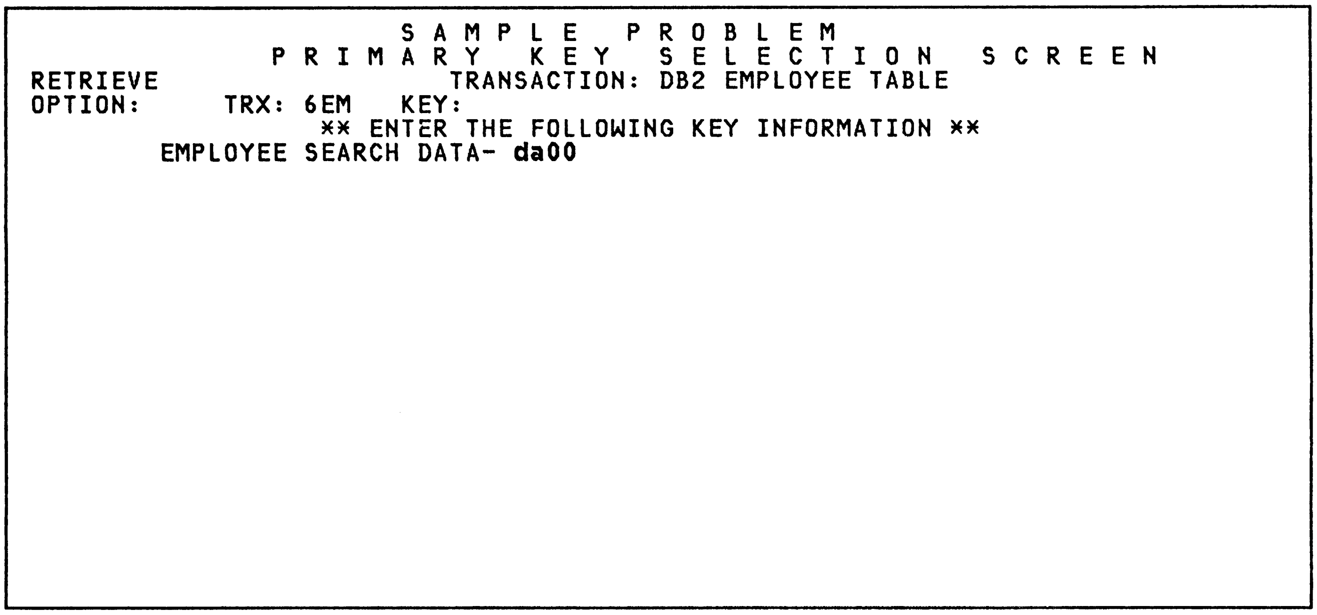
Upon entry of the HELP request, the HELP panel is displayed.



```
                          H E L P   F O R   T R X I D   E M
OPTION:                                                               PAGE: 001

          D B 2   S A M P L E   P R O B L E M
           K E Y   S E L E C T I O N   S C R E E N S
PRIMARY KEY SELECTION
EMPLOYEE SEARCH DATA ______
      ENTRY                ACTION
      NNNNNN               WHERE N IS 0 TO 9 - EMPLOYEE DATA IS
                           DISPLAYED ON TRANSACTION DISPLAY
      NNNNN> OR <          WHERE N IS 1 TO 5 NUMBERS - SECONDARY KEY
                           SELECTION WITH EMPLOYEES STARTING WITH
                           THE PARTIAL KEY AND GREATER
      N_____ OR %          WHERE N IS 1 TO 5 NUMBERS - SECONDARY KEY
                           SELECTION WITH EMPLOYEES WITH THESE CORRECT
                           PARTIAL VALUES
      DNNN                 WHERE D INVOKES USER DEFINED KSELECT3 -
                           D INDICATES DISPLAY ALL EMPLOYEES IN THE
                           NNN DEPARTMENT ON THE SECONDARY KEY
                           SELECTION BROWSE SCREEN

PRESS PF1 TO DISPLAY NEXT PAGE
```

Figure C-4. HELP Screen, Transaction EM

Upon entry, the Primary Key Selection screen is displayed again for input.

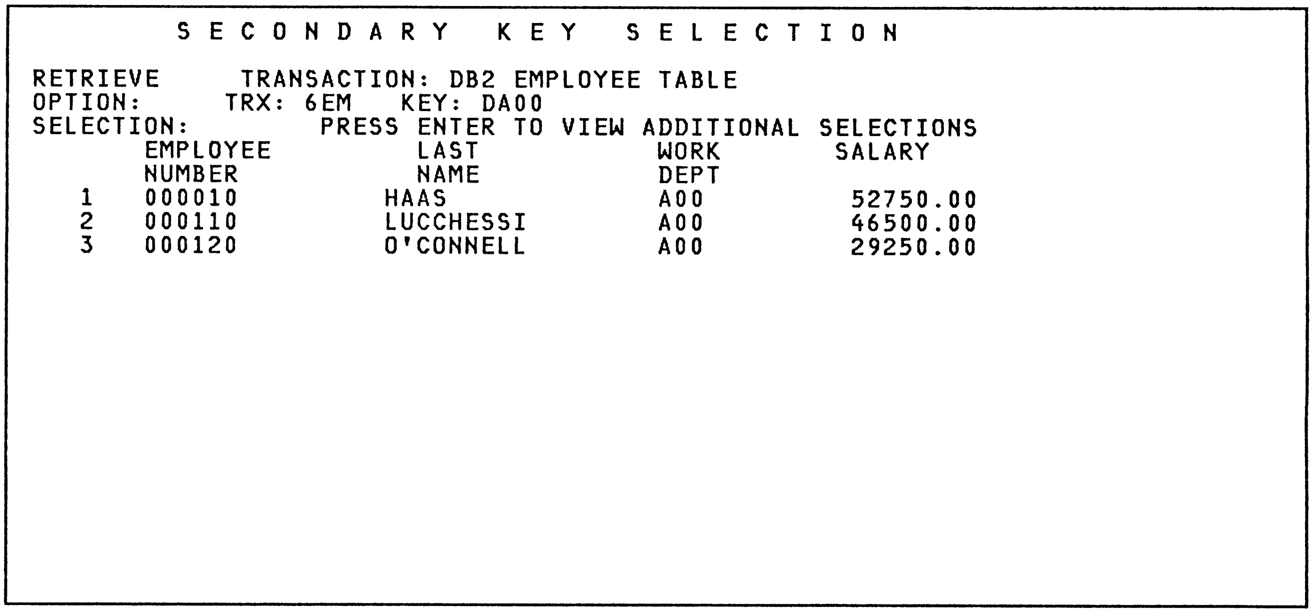


```
                       S A M P L E   P R O B L E M
               P R I M A R Y   K E Y   S E L E C T I O N   S C R E E N
RETRIEVE                   TRANSACTION: DB2 EMPLOYEE TABLE
OPTION:     TRX: 6EM   KEY:
                 ** ENTER THE FOLLOWING KEY INFORMATION **
        EMPLOYEE SEARCH DATA- da00
```

Figure C-5. Primary Key Selection, Transaction EM

By entering da00 the terminal operator has requested that Secondary Key Selection Browse be invoked to display all employees from department A00.

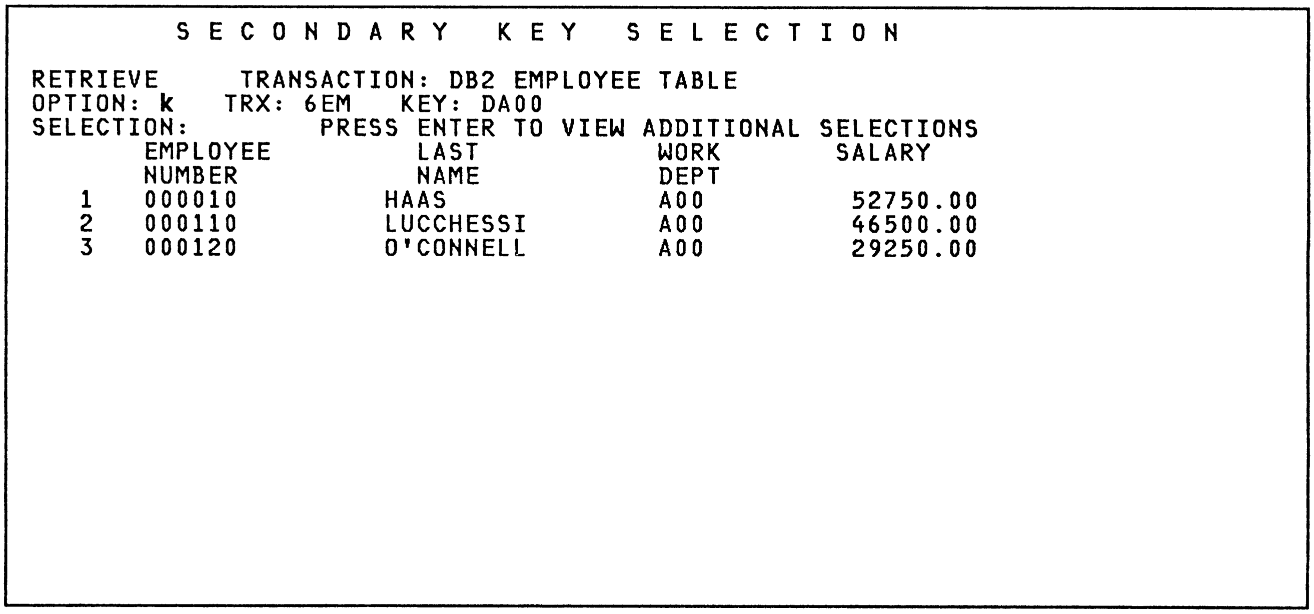
Upon entry, the Secondary Key Selection Browse screen is displayed.



```
          S E C O N D A R Y   K E Y   S E L E C T I O N

 RETRIEVE     TRANSACTION: DB2 EMPLOYEE TABLE
 OPTION:    TRX: 6EM   KEY: DA00
 SELECTION:        PRESS ENTER TO VIEW ADDITIONAL SELECTIONS
        EMPLOYEE          LAST           WORK        SALARY
        NUMBER            NAME           DEPT
    1   000010          HAAS             A00          52750.00
    2   000110          LUCCHESSI        A00          46500.00
    3   000120          O'CONNELL        A00          29250.00
```

Figure C-6. Secondary Key Selection, Transaction EM BY WORKDEPT

The information required is often contained in the related fields displayed on this screen, and there is no need to go to the transaction display screen.



```
          S E C O N D A R Y   K E Y   S E L E C T I O N

 RETRIEVE     TRANSACTION: DB2 EMPLOYEE TABLE
 OPTION: k  TRX: 6EM   KEY: DA00
 SELECTION:        PRESS ENTER TO VIEW ADDITIONAL SELECTIONS
        EMPLOYEE          LAST           WORK        SALARY
        NUMBER            NAME           DEPT
    1   000010          HAAS             A00          52750.00
    2   000110          LUCCHESSI        A00          46500.00
    3   000120          O'CONNELL        A00          29250.00
```

Figure C-7. Secondary Key Selection, Transaction EM BY WORKDEPT

The terminal operator enters 'k' in the OPTION field to return to the Primary Key Selection screen.

```
                  S A M P L E   P R O B L E M
          P R I M A R Y   K E Y   S E L E C T I O N   S C R E E N
RETRIEVE             TRANSACTION: DB2 EMPLOYEE TABLE
OPTION:     TRX: 6EM   KEY:
                  ** ENTER THE FOLLOWING KEY INFORMATION **
        EMPLOYEE SEARCH DATA- 0003>
```

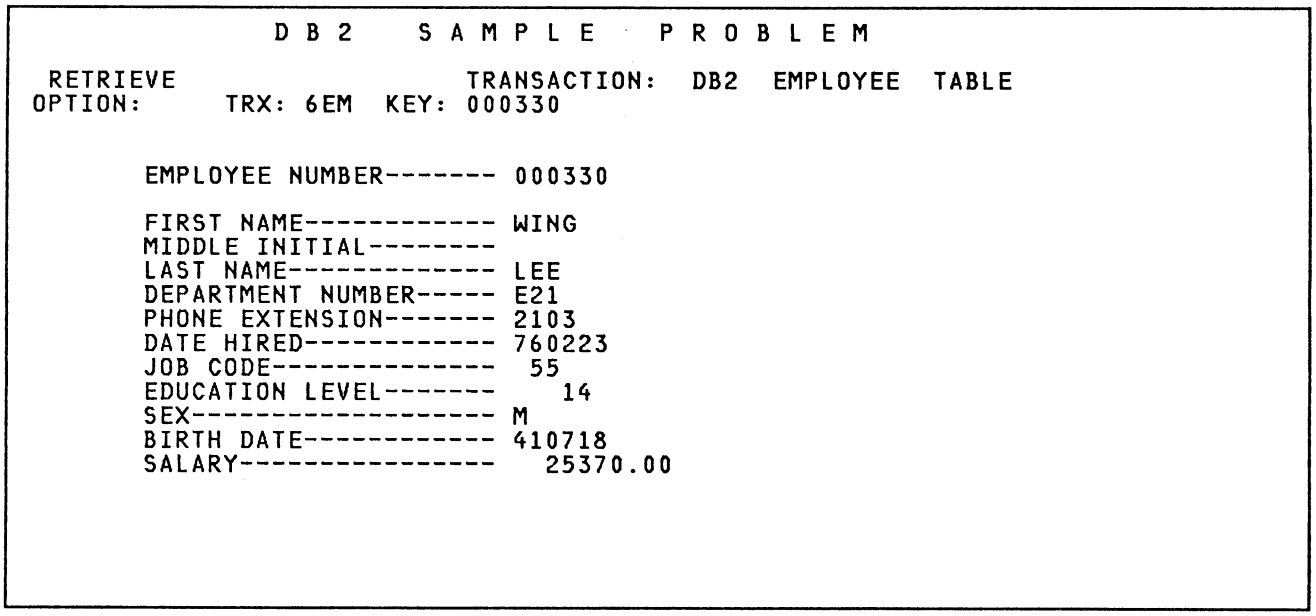
Figure C-8. Primary Key Selection, Transaction EM

Upon redisplay of the Primary Key Selection screen, a request is made for a generic search on the employee numbers beginning with 0003. The resulting Secondary Key Selection browse screen follows:

```
          S E C O N D A R Y   K E Y   S E L E C T I O N

RETRIEVE     TRANSACTION: DB2 EMPLOYEE TABLE
OPTION: F   TRX: 6EM   KEY: 0003>
SELECTION: 4      PRESS ENTER TO VIEW ADDITIONAL SELECTIONS
       EMPLOYEE         LAST            WORK       SALARY
       NUMBER           NAME            DEPT
   1   000300         SMITH             E11          17750.00
   2   000310         SETRIGHT          E11          15900.00
   3   000320         MEHTA             E21          19950.00
   4   000330         LEE               E21          25370.00
   5   000340         GOUNOT            E21          23840.00
```

Figure C-9. Secondary Key Selection, Transaction EM BY EMPNO

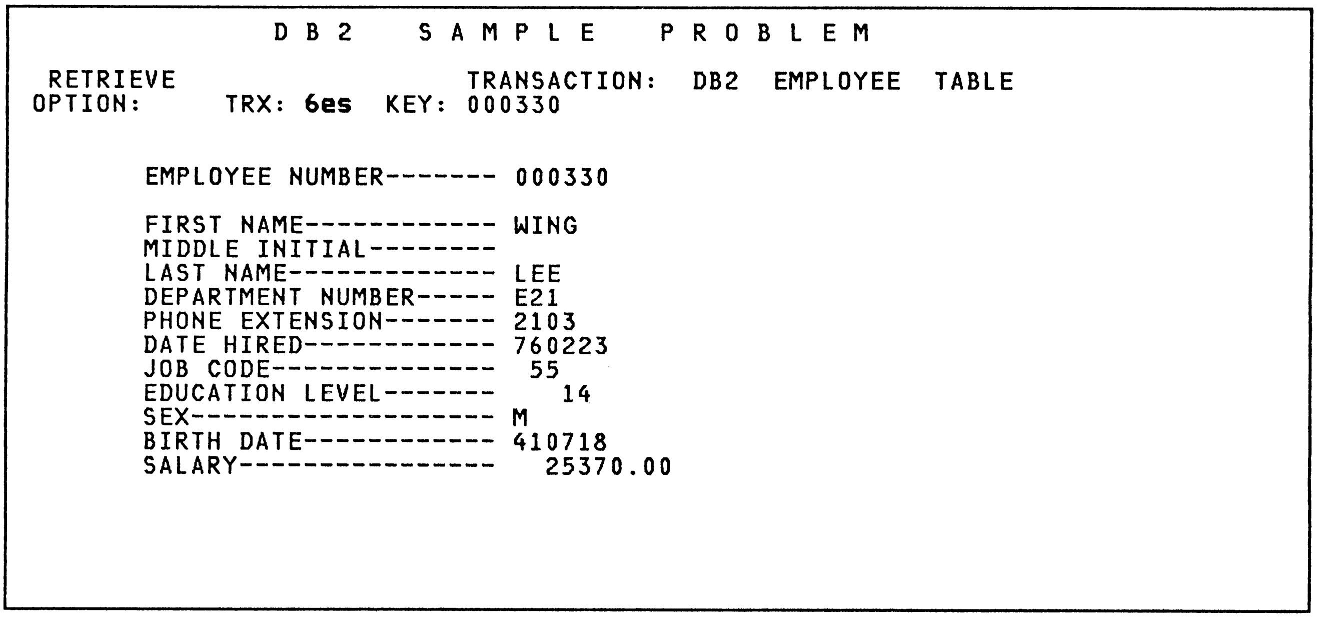
From the Secondary Key Selection screen, selection 4 is made to proceed to the Transaction Display screen.



```
                D B 2    S A M P L E    P R O B L E M

 RETRIEVE                       TRANSACTION:  DB2  EMPLOYEE  TABLE
OPTION:     TRX: 6EM  KEY: 000330


       EMPLOYEE NUMBER------- 000330

       FIRST NAME------------ WING
       MIDDLE INITIAL--------
       LAST NAME------------- LEE
       DEPARTMENT NUMBER----- E21
       PHONE EXTENSION------- 2103
       DATE HIRED------------ 760223
       JOB CODE--------------  55
       EDUCATION LEVEL-------    14
       SEX------------------- M
       BIRTH DATE------------ 410718
       SALARY----------------   25370.00
```

Figure C-10. Transaction Display Screen, Transaction EM

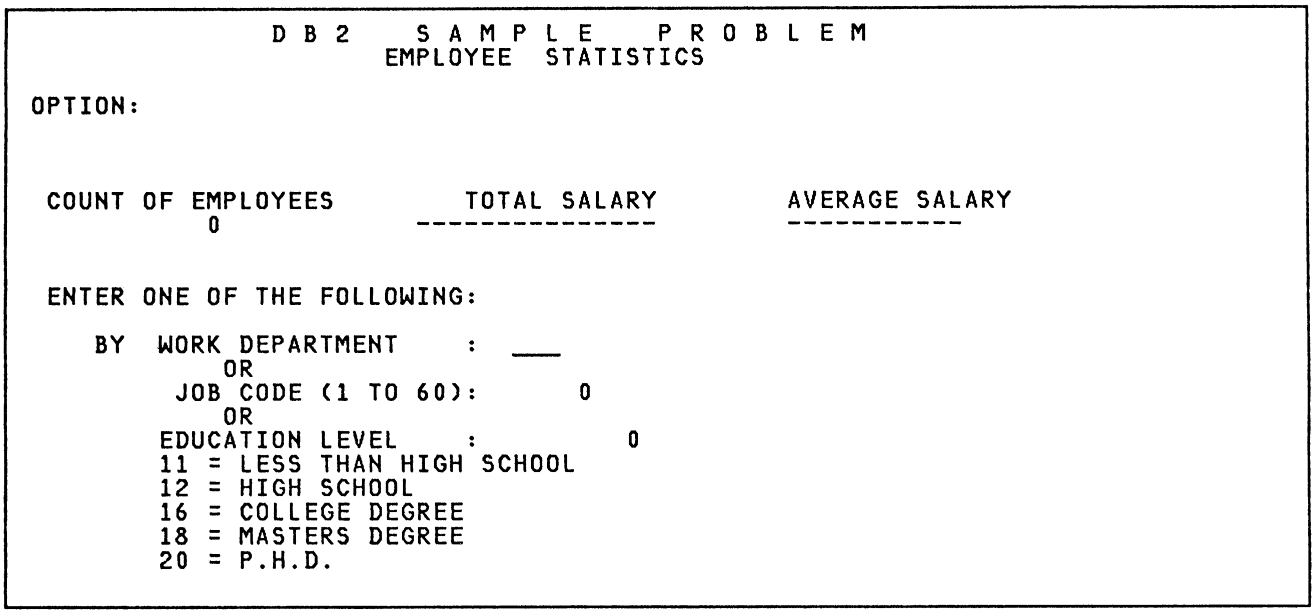
Because the transaction mode selected is 6 'RETRIEVE', updating is not allowed. If the transaction mode is changed to 5 'UPDATE', this same screen is used for updates.



```
                D B 2    S A M P L E    P R O B L E M

 RETRIEVE                       TRANSACTION:  DB2  EMPLOYEE  TABLE
OPTION:     TRX: 6es  KEY: 000330


       EMPLOYEE NUMBER------- 000330

       FIRST NAME------------ WING
       MIDDLE INITIAL--------
       LAST NAME------------- LEE
       DEPARTMENT NUMBER----- E21
       PHONE EXTENSION------- 2103
       DATE HIRED------------ 760223
       JOB CODE--------------  55
       EDUCATION LEVEL-------    14
       SEX------------------- M
       BIRTH DATE------------ 410718
       SALARY----------------   25370.00
```

Figure C-11. Transaction Display Screen, Transaction EM

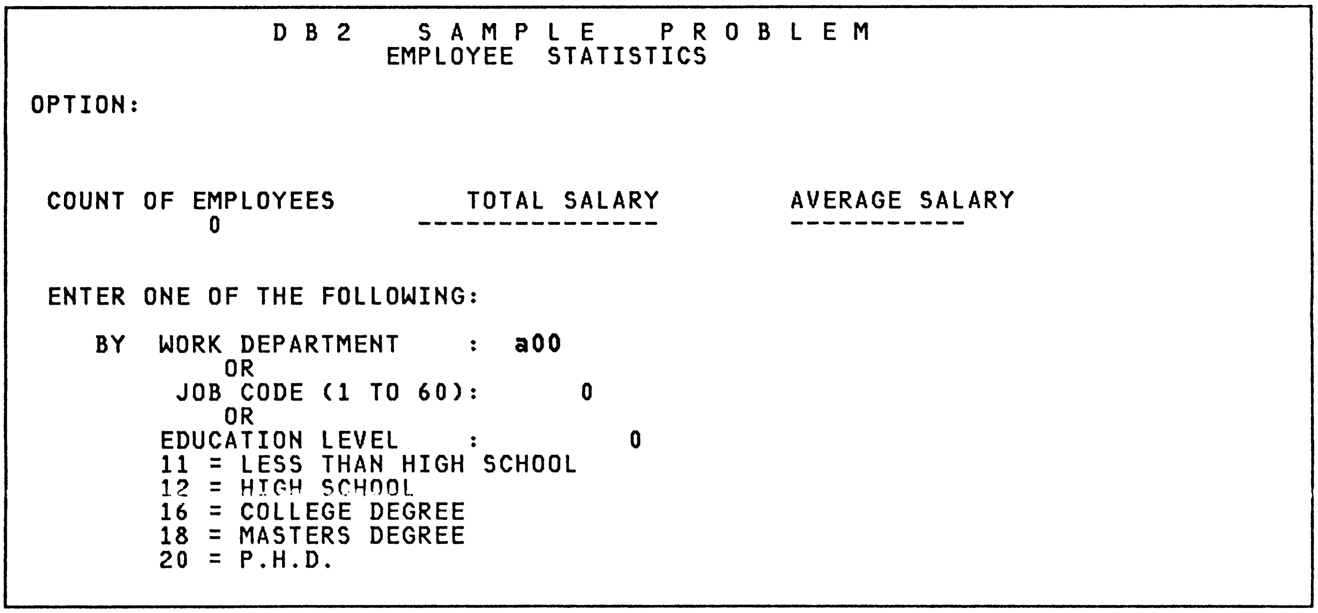
From the 'EM' Transaction Display Screen the terminal operator changes the TRX field to 6ES. Internally IMSADF II switches to the 'ES' transaction. The 'ES' transaction does not have a key selection phase. The 'ES' transaction display screen is shown.



```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS

OPTION:


 COUNT OF EMPLOYEES        TOTAL SALARY        AVERAGE SALARY
           0            ---------------        -----------


 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT    :  ___
            OR
         JOB CODE (1 TO 60):      0
            OR
        EDUCATION LEVEL    :         0
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

Figure C-12. Transaction Display Screen, Transaction ES

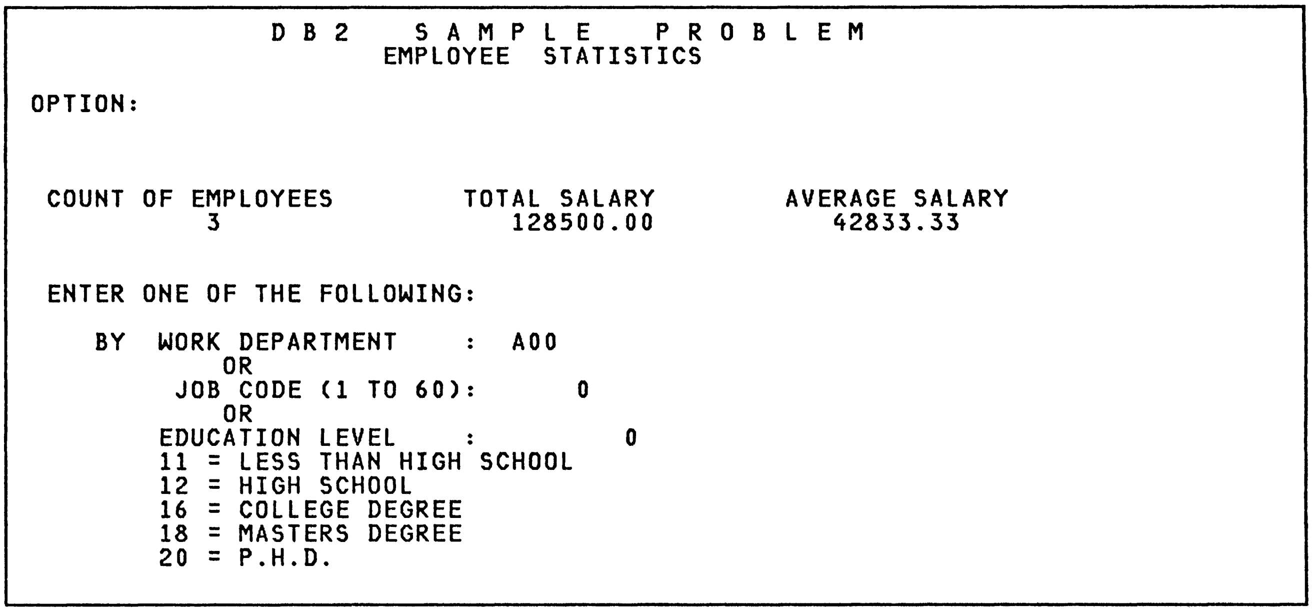
To request Employee Statistics by WORK DEPARTMENT enter a department number and PRESS enter.



```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS

OPTION:


 COUNT OF EMPLOYEES        TOTAL SALARY        AVERAGE SALARY
           0            ---------------        -----------


 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT    :  a00
            OR
         JOB CODE (1 TO 60):      0
            OR
        EDUCATION LEVEL    :         0
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

Figure C-13. Transaction Display Screen, Transaction ES

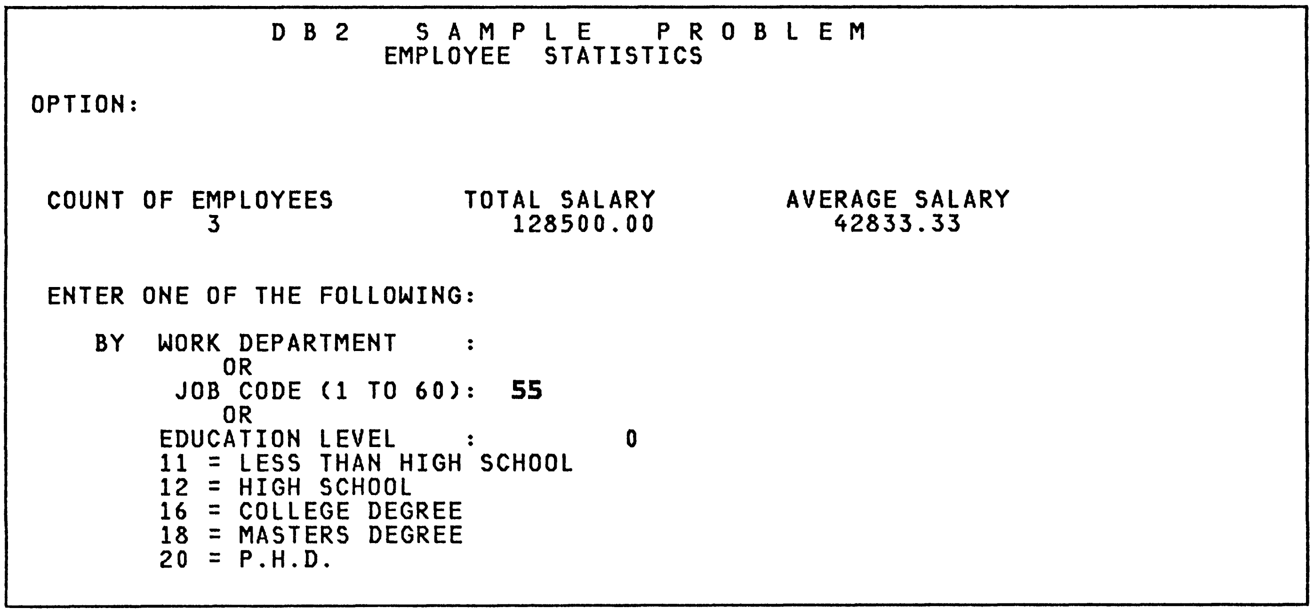
The terminal operator enters department number A00.



```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS

OPTION:



 COUNT OF EMPLOYEES          TOTAL SALARY          AVERAGE SALARY
          3                     128500.00             42833.33


 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT    :  A00
            OR
         JOB CODE (1 TO 60):      0
            OR
        EDUCATION LEVEL    :         0
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

Figure C-14. Transaction Display Screen, Transaction ES by DEPT A00

The ES Transaction Display screen is redisplayed with Employee Statistics for department A00.



```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS

OPTION:



 COUNT OF EMPLOYEES          TOTAL SALARY          AVERAGE SALARY
          3                     128500.00             42833.33


 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT    :
            OR
         JOB CODE (1 TO 60):  55
            OR
        EDUCATION LEVEL    :         0
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

Figure C-15. Transaction Display Screen, Transaction ES by DEPT A00

The terminal operator blanks out WORK DEPARTMENT and enters 55 in the JOB CODE to request statistics for all employees with a job code of 55.

```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS
OPTION:


 COUNT OF EMPLOYEES          TOTAL SALARY          AVERAGE SALARY
          9                     250170.00             27796.66


 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT     :  ___
            OR
         JOB CODE (1 TO 60):      55
            OR
        EDUCATION LEVEL     :         0
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

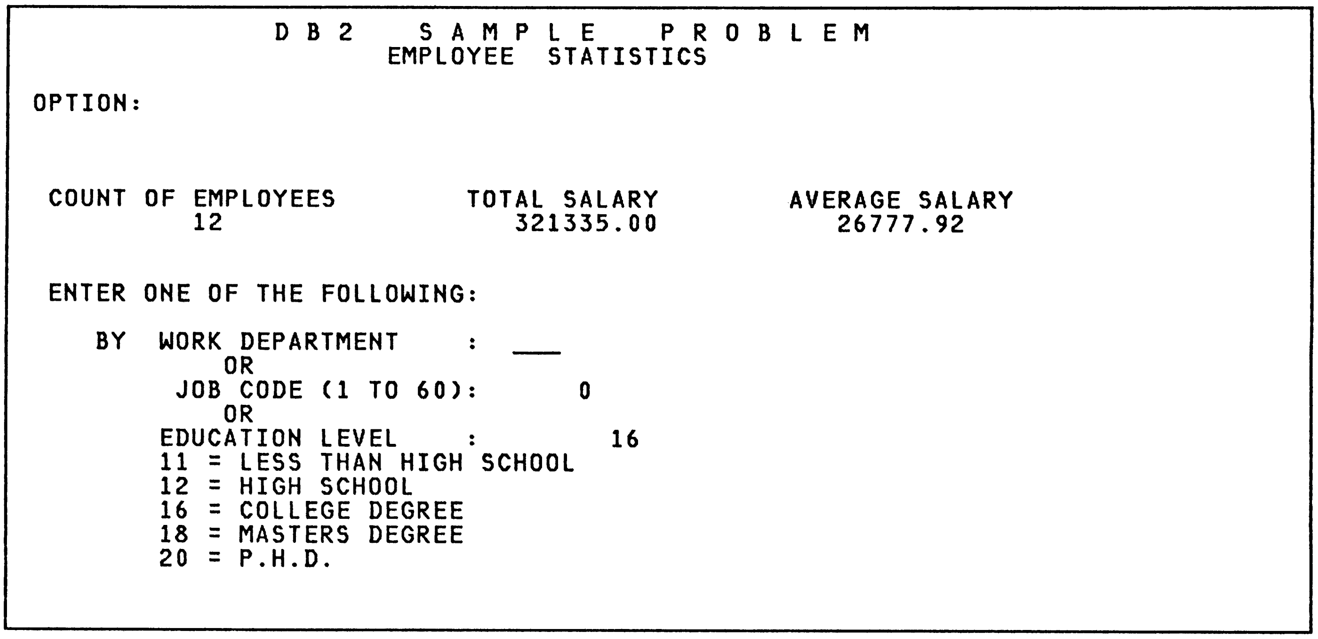
Figure C-16. Transaction Display Screen, Transaction ES by JOBCODE 55

The resulting Data Display shows the count of employees, total salary and average salary for job code 55.

```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS
OPTION:


 COUNT OF EMPLOYEES          TOTAL SALARY          AVERAGE SALARY
          9                     250170.00             27796.66

 ENTER ONE OF THE FOLLOWING:

    BY  WORK DEPARTMENT     :  ___
            OR
         JOB CODE (1 TO 60):       0
            OR
        EDUCATION LEVEL     :        16
        11 = LESS THAN HIGH SCHOOL
        12 = HIGH SCHOOL
        16 = COLLEGE DEGREE
        18 = MASTERS DEGREE
        20 = P.H.D.
```

Figure C-17. Transaction Display Screen, Transaction ES by JOBCODE 55

The terminal operator enters a zero in the JOB CODE field and enters a 16 in the EDUCATION LEVEL field to request statistics for all employees with an education level of 16.



```
                D B 2    S A M P L E    P R O B L E M
                       EMPLOYEE  STATISTICS
 OPTION:



  COUNT OF EMPLOYEES          TOTAL SALARY          AVERAGE SALARY
           12                    321335.00             26777.92


  ENTER ONE OF THE FOLLOWING:

     BY  WORK DEPARTMENT     :  ___
             OR
          JOB CODE (1 TO 60):      0
             OR
         EDUCATION LEVEL     :        16
         11 = LESS THAN HIGH SCHOOL
         12 = HIGH SCHOOL
         16 = COLLEGE DEGREE
         18 = MASTERS DEGREE
         20 = P.H.D.
```

Figure C-18. Transaction Display Screen, Transaction ES by EDUCLVL 16

The resulting Data Display shows the count of employees, total salary and average salary for education level 16.

APPENDIX D. BTS IN AN IMS/VS - DB2 ENVIRONMENT

The IMS/VS Batch Terminal Simulator (BTS), program product number 5668-948, Release 2, supports the tracing of SQL calls in a format similar to that of DL/I calls. Refer to the IMS/VS BTS Program Reference and Operations Manual, SH20-5523, for complete details.

To run BTS as a batch job, a DFSESL DD statement must be added to the BTSBMP procedure.

To run BTS in TSO foreground, add an ALLOC command for the DFSESL data set to the BTS CLIST. Execute the BTS CLIST and specify the KW(BMP) option.

BTS INPUT COMMANDS

The BTSIN data set is used to input BTS command statements.

```
./D DDOF=327029
./O DB=YES Q=YES ATR=NO
./T TC=MFC1T01 MBR=MFC1TOM SPA=6000 LANG=ASM TYPE=MSG PLC=20
./T TC=MFC1T02 MBR=MFC1TOM SPA=6000 LANG=ASM TYPE=MSG PLC=20
./T TC=MFC1T03 MBR=MFC1TOM SPA=6000 LANG=ASM TYPE=MSG PLC=20
./T TC=MFC1T99 MBR=MFC1T99 SPA=6000 LANG=ASM TYPE=MSG PLC=20
./T TC=SAMPTOR MBR=SAMPTOR SPA=6000 LANG=ASM TYPE=MSG PLC=20
./T TC=LTERM3 MDL=2
./T TC=IOPCB  MDL=2
```

Figure D-1. Sample BTSIN Data for an IMSADF II - DB2 Transaction

BTS OUTPUT

```
****  SQL CALL- FUNC=CALL,                   RTRNCD= 0000
             ----.----1----.----2----.----3----.----4----.----5----
      RDIIN =  H   SAMPSES   #    I        8   8  V
             028001ECDDECE4107F1B3800002000FB00FC0A0E
             08800E2147252033BD6C090106420A8C0A8C050F
 INPUT VARS =     0           V           V     *
             00020F00003500000E00003500360E0000350036
             0008100004C1000015D204C504C3159204CC04C5
OUTPUT VARS =     D  ,L
             00010C00006D0000
             0000140307B30000
BTS0096I DB2 SQL CALLS = 00001
```

Figure D-2. BTS SQL Trace

This figure displays the BTS SQL trace output produced, when the IMSADF II sample transaction 'ES', defined in appendices B and C, executes the DEPTSELC SQL function.

RTRNCD For each SQL statement traced the RTRNCD presents the SQL return code from the SQLCA.

RDIIN The first forty bytes of the RDIIN are displayed (fixed portion). The RDIIN precedes each run time SQL statement in a compilation. Bytes 37 and 38 contain the SQL statement number, and can be used to identify this statement in a compilation listing.

VARS INPUT and/or OUTPUT VARS contain the host variables referenced in the SQL statement.

- The first four bytes contain the length of the displayed area.
- Twelve bytes are displayed for each variable:

— Host Variable type	- 2 bytes
— Host Variable length	- 2 bytes
— Host Variable address	- 4 bytes
— Host Variable Indicator address	- 4 bytes

  Refer to the variable part of the RDIIN in a compilation listing.

APPENDIX E. IMSADF II TRACE FACILITY

A trace capability is provided as part of IMSADF II which can be used for detailed tracing of internal control and flow module by module.

Reference the _IMS Application Development Facility II Version 2 Release 2 Diagnosis Guide_ for complete details on using the IMSADF II trace facility.

If tracing is required for an IMSADF II transaction that accesses DB2 Tables/Views through the Table Handler Rule interface set the IMSADF II trace options as follows:

```
  FLOW=Y,EXTEND=Y,MODULES=(MFC1V09S)
```

The traces in module MFC1V09S display:

- Parameters passed to MFC1V09S
- Parameters MFC1V09S passes to the Table Handler Rule
- SQL Communication Area
- Data Compare information

If tracing is required while executing the RGLGEN Utility, set the IMSADF II trace options as follows:

```
  FLOW=Y,EXTEND=Y,MODULES=(MFC1Y25)
```

The traces in module MFC1Y25 display:

- Parameters passed to MFC1Y25
- SQL Communication Area
- MFC1Y25 return codes and data areas

APPENDIX F. RGLGEN UTILITY LINK-EDIT PLAN

????Y25	RGLGEN Utility link-edit load module name. ???? is the installed ADFID in effect at link-edit time. This is the PROGRAM-NAME on the TSO RUN command.
????SYSP	IMSADF II installation options, control block
MFC1Y25	RGLGEN Utility Main Routine
MFC1Y35	RGLGEN Utility Message Building
MFC1Y36	RGLGEN Utility Message Text
MFC1Y37	RGLGEN Utility Message Tailoring
MFC1V38	IMSADF II DDNAME Checker
MFC1V39	IMSADF II Trace Interface
MFC1V48	IMSADF II PDS Access Function
MFC1A16	IMSADF II Date Time Routine
DSNELI	DB2 TSO Language Interface

Additional modules: Optionally loaded if required

MFC1V40	IMSADF II Trace Output Function
????FLLM	IMSADF II Trace Control Function

INDEX

IMS Application Development Facility II Version 2 Release 2
DATABASE 2 Application Specification Guide
SH20-6603-01

READER'S COMMENT FORM

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation? ________________________

Number of latest Newsletter associated with this publication: ________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SH20-6603-01

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 8D8
220 Las Colinas Boulevard
Irving, Texas 75039-5513

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Cut or Fold Along Line

IMS Application Development Facility II Version 2 Release 2 DATABASE 2 Application Specification Guide SH20-6603-01

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

IMS Application Development Facility II Version 2 Release 2
DATABASE 2 Application Specification Guide
SH20-6603-01

READER'S COMMENT FORM

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

____

____

____

____

What is your occupation? ______________________________

Number of latest Newsletter associated with this publication: ______________________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SH20-6603-01

Reader's Comment Form

Fold and tape　　Please Do Not Staple　　Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS　　PERMIT NO. 40　　ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 8D8
220 Las Colinas Boulevard
Irving, Texas 75039-5513

Fold and tape　　Please Do Not Staple　　Fold and tape

IBM®

Cut or Fold Along Line

IMS Application Development Facility II Version 2 Release 2 DATABASE 2 Application Specification Guide SH20-6603-01

Publication Number
SH20-6603-01

File Number
S/370/4300-32

Program Number
5665-348

SH20-6603-01