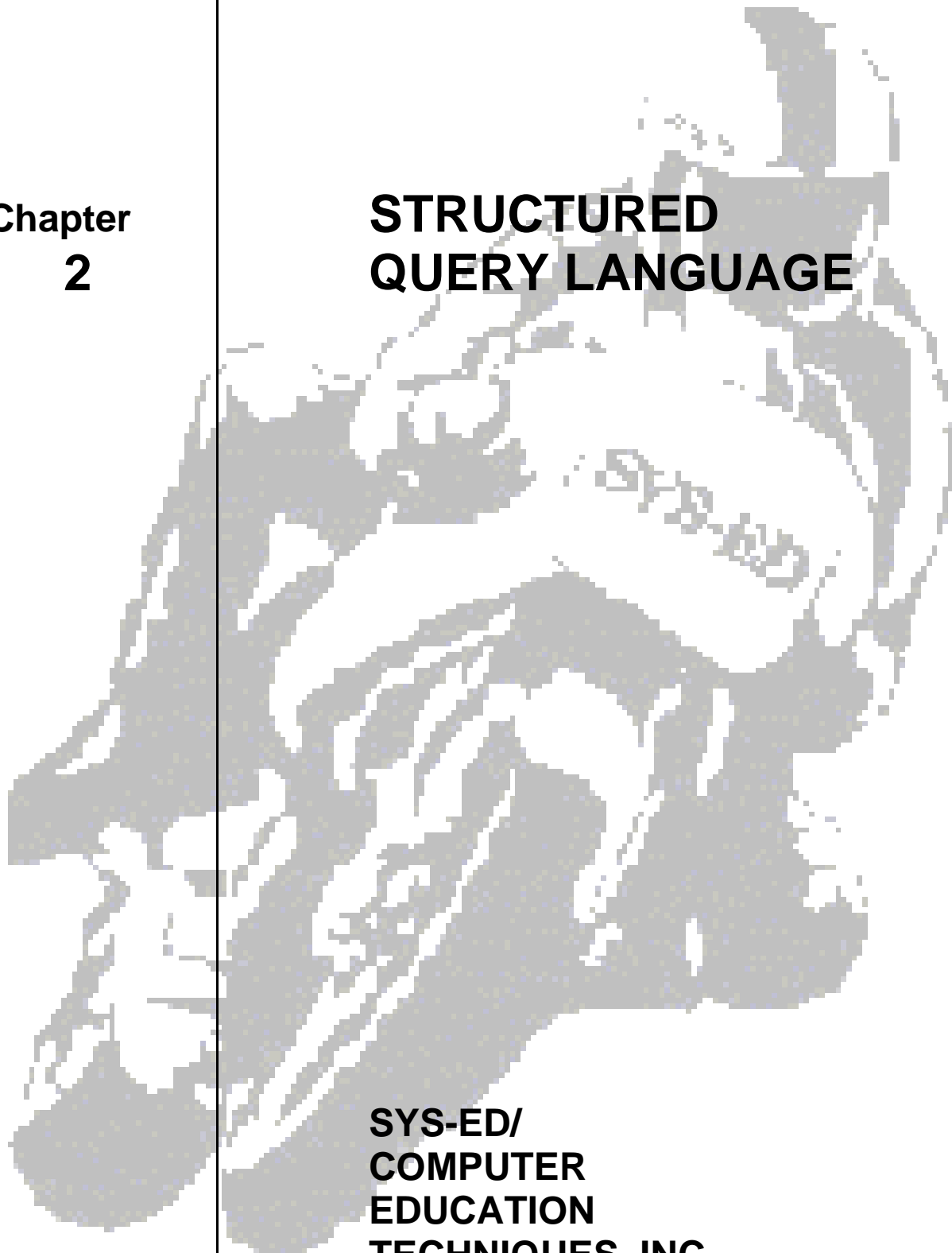


**Chapter
2**

**STRUCTURED
QUERY LANGUAGE**



**SYS-ED/
COMPUTER
EDUCATION
TECHNIQUES, INC.**

Objectives

You will learn:

- Code and test SQL with SPUFI.
- Prepare programs to be precompiled, compiled (assembled) and linked.
- Structure and features of DB2I.
- SQLCA record and its usage.

1 Introduction to SQL

SQL is a data independent non-procedural programming language.

It provides one language for manipulating DB2 objects and databases. SQL also provides a means for establishing security over the DB2 databases.

SQL statements can be stand-alone or embedded in an application program written in COBOL, VS COBOL II, COBOL for z/OS, Enterprise COBOL, JAVA, PL/I, and Assembler.

The functions and capabilities of SQL include the following:

- C Defining DB2 objects.
- C Altering DB2 objects.
- C Dropping/deleting DB2 objects.
- C Retrieving rows from DB2 database tables.
- C Adding rows to DB2 database tables.
- C Deleting rows from DB2 database tables.
- C Changing rows in DB2 database tables.
- C Using views for establishing security to DB2 databases.
- C Using GRANT to restrict access or REVOKE to take back granted privileges from the users.

2 DB2 through ISPF Menus

ISPF main menu might look as follows:

```
                ISPF MASTER MENU

1.   PDF   - ISPF/PROGRAM DEVELOPMENT FACILITY
2.   DB2   - DB2 PANELS
X.   EXIT  - TERMINATE ISPF

Enter End command(PF3) to terminate ISPF
```

2.1. DB2 Panels

DB2 main menu is shown below:

```
                DB2 MAIN MENU

OPTION )))<_

1. DB2I      - DB2 INTERACTIVE FUNCTION
2. DBEDIT    - DB2 EDIT
3. QMF       - QMF SQL AND QBE
4. QMFTOOL   - QMF MESSAGE TOOL

X. EXIT      - TERMINATE DB2

enter END command(PF3) to terminate DB2
```

2.2. DB2I Primary Selection Menu

The primary menu is the first in a series of panels.

It is used to select the procedure to be executed and to specify whether or not the panel associated with the procedure is to be displayed.

```
DB2I PRIMARY OPTION MENU

)))<

Select one of the following DB2 functions and press ENTER.

1  SPUFI                (Process SQL statements)
2  DCLGEN               (Generate SQL and source language declaration)
3  PROGRAM PREPARATION (Prepare a DB2 application to run)
4  PRECOMPILE           (Invoke DB2 compiler)
5  BIND/REBIND/FREE    (BIND, REBIND, or FREE application plans)
6  RUN                  (RUN an SQL program)
7  DB2 COMMANDS        (Issue DB2 commands)
8  UTILITIES           (Invoke DB2 utilities)
D  DB2 DEFAULTS        (Set global parameters)
X  EXIT                 (Leave DB2)

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

3 Program Development Aids

The interactive nature of DB2 (DB2I) provides the programmer/user with a wide range of tools for making the development of application programs easier.

The tools are an extension of TSO/ISPF.

- C SPUFI is used to execute and test SQL statements.
- C The program preparation facility is used to prepare an embedded SQL program for execution.
- C The program run facility is used to execute an application program.
- C The plan maintenance facility is used to prepare, alter, or delete an application plan.
- C The declarations generator facility is used to generate data declaration statements (copy book) for inclusion in COBOL or PL/I programs.
- C The on-line help facility, in the form of ISPF panels, provides tutorial information.

4 SPUFI: SQL Processing Using File Input

SPUFI is an interactive facility which allows a user to execute SQL commands on-line through TSO/ISPF. It stands for SQL Processor Using File Input. It is available under the DB2I option of ISPF.

SQL statements are stored in a sequential dataset or in a PDS member. The statements can then be edited using full ISPF edit capabilities. SQL commands can be immediately executed on-line.

The executed SQL statements can be viewed on-line and browsed using ISPF browse options.

4.1. How to Use SPUFI

1. Create a sequential dataset or member of a PDS containing SQL statements.
2. Specify an output file to contain the results of the SQL execution.
3. Automatically execute the SQL statements.
4. Automatic commit or rollback as necessary will be performed.
5. Browse the output file using the SPUFI menu.

4.2. SPUFI Menu

ENTER INPUT DATA SET NAME:

1. DATA SET NAME))< Sequential or PDS member.
2. VOLUME SERIAL))< Not Required.
3. DATA SET PSWD))< Not Required.

ENTER OUTPUT DATA SET NAME:

4. DATA SET NAME))< Must be sequential.
If not specified DB2 will create defaults from the default panel.

SPECIFY PROCESSING OPTIONS:

5. CHANGE DEFAULTS))< Y/N Display SPUFI default panel.
6. EDIT INPUT))< Y/N Lets you edit SQL statements.
7. EXECUTE))< Y/N Executes SQL statements in input data set.
8. AUTOCOMMIT))< Y/N Commits if run was successful.
9. BROWSE OUTPUT))< Y/N Lets you browse the output data set automatically after the command was processed.

4.3. SPUFI Default Menu

ENTER THE FOLLOWING TO CONTROL YOUR SPUFI SESSION:

1. ISOLATION LEVEL))< RR Repeatable read or CS for Cursor Stability.
2. MAXIMUM SELECT LINES))< 250 An integer representing the maximum number of lines to be returned.

OUTPUT DATA SET CHARACTERISTICS:

3. RECORD LENGTH))< 4092 At least 80 bytes.
4. BLOCK SIZE))< 4096 At least 80 bytes.
5. RECORD FORMAT))< VB F, FB, V, VB, FBA, VBA.
6. DEVICE TYPE))< SYSDA Required to be a DASD unit name.

OUTPUT FORMAT CHARACTERISTICS:

7. MAX. NUMERIC FIELD))< 20 Maximum width for numeric fields.
8. MAX. CHAR. FIELD))< 80 Maximum width for character fields.
9. COL. HEADING FIELD))< BOTH May be names, labels, any or both.

RECORD LENGTH))< The sum of all column widths plus two spaces between each column.

5 Embedding SQL Statements

SQL statements can be embedded in one of the traditional programming languages such as COBOL, FORTRAN, BAL, and PL/I or in CICS and DL/I application programs.

An application program can be developed and tested on-line using program preparation panels of the DB2I option of ISPF.

A program using SQL must go through the following stages before it can be executed:

1. The program must be precompiled.
2. A load module must be created.
3. An application plan must be created.

Option 3 of DB2I will present panels one by one to precompile the program, create the load module, create the application plan and to execute the program.

Output will appear in the following files:

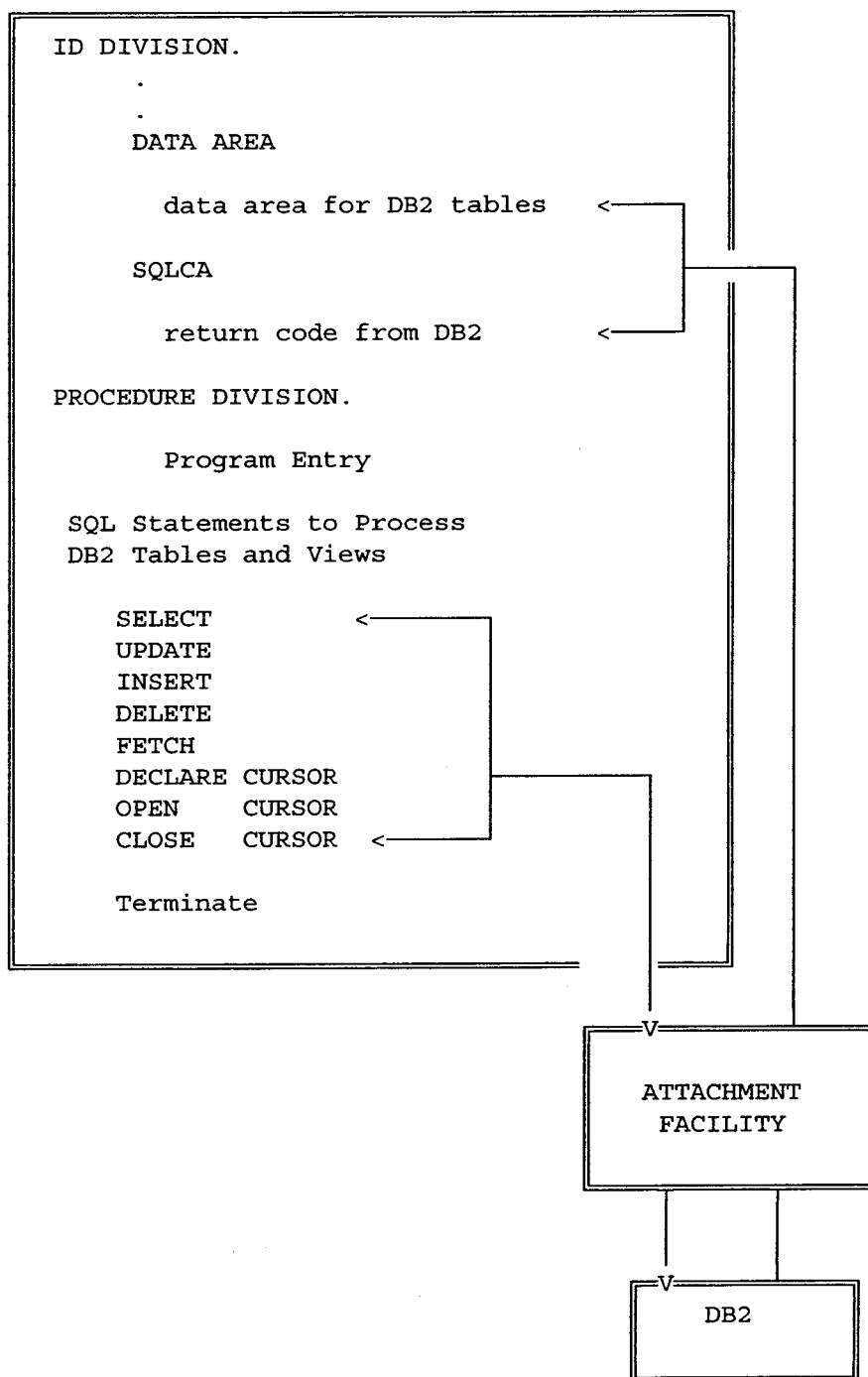
PRECOMPILE LISTING))< temp.PCLIST

COMPILER LISTING))< temp.LIST

PROGRAM LISTING))< temp.COBOL

Temp was given as a data set name qualifier in program preparation and/or precompile panels.

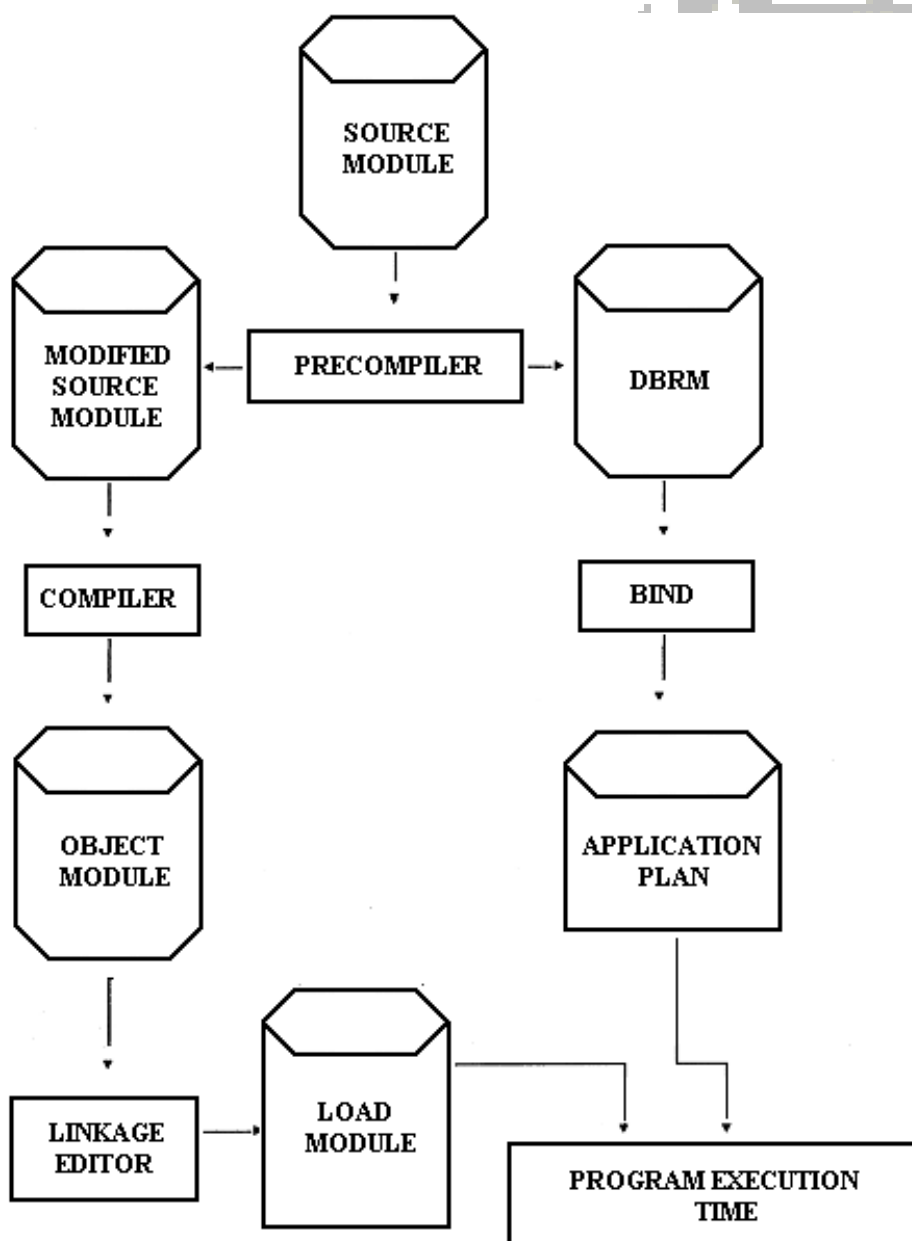
5.1. DB2 Application Program Structure



6 Program Preparation Process

Application programs which access DB2 databases must contain SQL statements. The SQL statements must be precompiled using the DB2 precompiler.

If a program also contains CICS/VS commands, it must be translated using the appropriate CICS/VS Command language translator.



7 Program Preparation Panels

Application programs developed using TSO and ISPF can be prepared for execution using the program preparation panels supplied by ISPF under DB2 (DB2I) option.

The panels lead the programmer through a step-by-step preparation process for executing a program.

There are other ways of preparing programs for execution.

Examples:

- C Invoking the DSNH CLIST.
- C Invoking the TSO prompter and DSN command processor.
- C Submitting the JCL.

The procedure for using the DB2 interactive facility (DB2I) is illustrated on the next several pages.

7.1. DB2 Program Preparation Menu

The program preparation panel is used for the preparation and execution of application programs.

```

))<

                                DB2 PROGRAM PREPARATION

Enter the following:
  1. INPUT DATA DET NAME .... ))<
  2. DATA SET NAME QUALIFIER ))< (For building data set name)
  3. PREPARATION ENVIRONMENT ))< (FOREGROUND, BACKGROUND, EDITJCL)
  4. RUN TIME ENVIRONMENT ... ))< (TSO, CICS, IMS)
  5. STOP IF RETURN CODE >= ))< (Lowest terminating)
  6. OTHER OPTIONS ))<

Select functions:                Display panel?  perform function?
  7. CHANGE DEFAULTS              ))<Y (Y/N)      .....
  8. PL/I MACRO PHASE.....       ))<N (Y/N)      ))<N (Y/N)
  9. PRECOMPILE.....             ))<Y (Y/N)      ))<Y (Y/N)
 10. CICS COMMAND TRANSLATION    ))<N (Y/N)
 11. BIND.....                   ))<Y (Y/N)      ))<Y (Y/N)
 12. COMPILE OR ASSEMBLE        ))<Y (Y/N)      ))<Y (Y/N)
 13. LINK.....                   ))<N (Y/N)      ))<Y (Y/N)
 14. RUN.....                     ))<N (Y/N)      ))<Y (Y/N)

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

```

7.2. DB2 Default Panel

This panel can be used to modify the DB2 system default parameters established at the time of the initial installation.

```
                                DB2I DEFAULTS
))<
Change defaults as desired:
1 DB2 NAME ..... ))< DSN      (installation defined)
2 DB2 CONNECTION RETRIES ))< 0      (How many retries for DB2 connection)
3 APPLICATION LANGUAGE . ))< COBOL  (COBOL, COB2, FORT ASM, ASMH, PL/I)
4 LINE/PAGE LISTING .... ))< 80    (A number from 5 to 999)
5 MESSAGE LEVEL ..... ))< I      (Information, Warning Error, Severe)
6 COBOL STRING DELIMITER ))< '    (DEFAULT, ' or ")
7 SQL STRING DELIMITER  ))< '    (DEFAULT, ' or ")
8 DECIMAL POINT ..... ))< .      (. or ,)
9 DB2 JOB STATEMENTS:  (option if your site has a SUBMIT exit)
   ))< //userid job (acctg. info), 'programmer', time=(,5),
   ))< // notify=userid,msgclass=x
   ))<
   ))<

PRESS: ENTER to save and exit  END to exit  HELP for more information
```

7.3. Precompile Panel

This panel is used to specify the INPUT AND INCLUDE LIBRARIES and any other options to be used in an application program.

```
                                PRECOMPILE
))<

Enter precompiler data sets:
1. INPUT DATA SET .... ))<  'userid.lib.type(member) '
2. INCLUDE LIBRARY ... ))<  lib containing dclgened member
3. DSNNAME QUALIFIER .. ))<  For building data set name)
4. DBRM DATA SET ..... ))<

Enter processing options as desired:

5. WHERE TO PRECOMPILE ))< FOREGROUND (FOREGROUND, BACKGROUND, or EDITJCL)
6. OTHER OPTIONS ..... ))<

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

7.4. Bind Panel

This panel is used to build an application plan.

All DB2 application programs require an application plan to allocate DB2 resources and support SQL requests during execution.

```
                                BIND

))<

Enter DBRM data set name(s):
 1. LIBRARY(s)   ))<
 2. MEMBER(s)   ))<
 3. PASSWORD(s) ))<
 4. MORE DBRMS? ))< NO

Enter options desired:
 5. PLAN NAME ..... ))<          (Required to create a plan)
 6. ACTION ON PLAN ..... ))< REPLACE (REPLACE or ADD)
 7. RETAIN EXECUTION AUTHORITY ))< YES  (YES to retain userlist)
 8. ISOLATION LEVEL ..... ))< RR      (RR or CS)
 9. PLAN VALIDATION TIME ..... ))< BIND (RUN or BIND)
10. RESOURCE ACQUISITION TIME ))< USE  (USE or ALLOCATE)
11. RESOURCE RELEASE TIME .... ))< COMMIT (COMMIT or DEALLOCATE)
12. EXPLAIN PATH SELECTION ... ))< NO  (NO or YES)

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

7.5. Compile, Link, and Run Program Preparation Panel

This panel is used to compile, link, and execute a DB2 application program.

```
PROGRAM PREPARATION:  COMPILER, LINK, AND RUN

))<

Enter compiler or assembler options:
 1. INCLUDE LIBRARY ))<
 2. INCLUDE LIBRARY ))<
 3. OPTIONS ..... ))<

Enter linkage editor options:
 4. INCLUDE LIBRARY ))<
 5. INCLUDE LIBRARY ))<
 6. INCLUDE LIBRARY ))<
 7. LOAD LIBRARY .. ))< runlib.load
 8. OPTIONS ..... ))<

Enter run options:
 9. PARAMETERS .... ))<
10. SYSIN DATA SET ))< TERM
11. SYSPRINT DS ... ))< TERM

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

8 SQLCA

Application programs which are to access DB2 databases must have a properly defined SQL communication area (SQLCA).

DB2 uses the SQLCA to indicate within a program whether or not an issued SQL statement was in its entirety successfully executed. To indicate the status of the last executed SQL statement, DB2 places a return in the SQLCA. An application program can test the SQL return code to determine the next processing step.

8.1. Defining the SQLCA in Working-Storage

The SQLCA must be included in the working-storage section of the application program.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
.
.
EXEC SQL
  INCLUDE SQLCA
END-EXEC
```

When the SQL include statement is used, the DB2 precompiler provides for the inclusion of the following SQLCA Cobol source statements in the working-storage section of the application program.

```
01  SQLCA.
    05  SQLCAID          PIC X(8) .
    05  SQLCABC          PIC S9(9) COMP .
    05  SQLCODE          PIC S9(9) COMP .
    05  SQLERRM.
        49  SQLERRML     PIC S9(4) COMP .
        49  SQLERRMC     PIC X(70) .
    05  SQLERRP          PIC X(8) .
    05  SQLERRD          OCCURS 6 TIMES
                        PIC S9(9) COMP .
    05  SQLWARN.
        10  SQLWARN0     PIC X(1) .
        10  SQLWARN1     PIC X(1) .
        10  SQLWARN2     PIC X(1) .
        10  SQLWARN3     PIC X(1) .
        10  SQLWARN4     PIC X(1) .
        10  SQLWARN5     PIC X(1) .
        10  SQLWARN6     PIC X(1) .
        10  SQLWARN7     PIC X(1) .
    05  SQLEXT          PIC X(8) .
```

8.2. SQLCA Fields

SQLCAID	An identifier of 'SQLCA', used to identify a DB2 storage dump.
SQLCABC	Indicates the length of the SQLCA (136 bytes).
SQLCODE	The SQL return code.
SQLERRM	A variable-length character string used to describe an error condition.
SQLERRML	Contains a value in the range of 0 to 70.
SQLERRMC	Will contain a character string if SQLERRML is greater than zero.
SQLERRD(3)	Used to indicate the extent to which a table was updated. After an update operation, the third occurrence of this field will be one of the following outcomes: C After an INSERT, the number of rows inserted into the table. C After an UPDATE, the number of rows updated in the table. C After a DELETE, the number of rows deleted from the table.

8.3. SQLCA: SQL Communication Area

SQLWARN0	If blank, all other SQLWARNn fields are also blank. Otherwise, one of the other SQLWARNn fields contains 'W'.
SQLWARN1	If 'W', at least one column's value was truncated when 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 not equal to the number of columns in the table or view being operated on by the statement. If 'W', a dynamic SQL UPDATE or DELETE statement does not include a WHERE clause.
SQLWARN5	If 'W', the program tried to execute a statement that applies only to SQL/DS.
SQLWARN6	Reserved.
SQLWARN7	Reserved.
SQLEXT	Reserved.

9 Whenever Statement

The SQL WHENEVER statement may also be used to check for exceptional conditions resulting from the execution of SQL statements.

The format of the WHENEVER statement and the three conditions which may occur are as follows:

```
EXEC SQL WHENEVER
```

NOT FOUND
SQLERROR
SQLWARNING

```
CONTINUE or GO TO host label
```

10 Result Table

Data can be retrieved using the SQL statement SELECT to specify a result table.

The data retrieved through SQL is always in the form of a table; this is known as a result table. As with the tables from which data is retrieved, a result table has rows and columns.

A program fetches this data one row at a time.

10.1. Data Types

When creating a DB2 table, each column has to be defined as a specific data type.

When performing operations on columns, the data must be compatible with the data type of the referenced column.

For example, character data, like a last name, can not be inserted into a column whose data type is numeric.

Similarly, columns containing incompatible data types can not be compared.

Examples:

Selecting all columns:

```
SELECT *
```

It is not necessary to know the column names for selecting DB2 data.

Use an asterisk (*) in the SELECT clause to indicate that you want to retrieve from each selected row of the named table.

This SQL statement selects all columns from the department table:

```
SELECT *  
FROM DSN8710.DEPT;
```

SELECT * is recommended mostly for use with dynamic SQL and view definitions.

SELECT * can be used in static SQL, but this is not recommended. If a column is added to the table to which SELECT * refers, the program might reference columns for which you have not defined receiving host variables.

10.2. Selecting Some Columns: SELECT column-name

Select the column or columns you want by naming each column. All columns appear in the order which has been specified, not in their order in the table.

```
SELECT column-name
```

Example:

This SQL statement selects only the MGRNO and DEPTNO columns from the department table:

```
SELECT MGRNO, DEPTNO  
FROM DSN8710.DEPT;
```

10.3. Naming Result Columns: AS

With AS, result columns can be named in a SELECT clause.

This is particularly useful for a column that is derived from an expression or a function.

Example:

The expression SALARY+BONUS+COMM has the name TOTAL_SAL.

```
SELECT SALARY+BONUS+COMM AS TOTAL_SAL  
FROM DSN8710.EMP  
ORDER BY TOTAL_SAL;
```

Result column names can be specified in the select-clause of a CREATE VIEW statement.

It is not necessary to supply the column list of CREATE VIEW, because the AS keyword names the derived column. The columns in the view EMP_SAL are EMPNO and TOTAL_SAL.

Example:

```
CREATE VIEW EMP_SAL AS  
SELECT EMPNO, SALARY+BONUS+COMM AS TOTAL_SAL  
FROM DSN8710.EMP;
```

10.4. SQL Rules for Processing a SELECT Statement

The rules of SQL dictate that a SELECT statement must generate the same rows as if the clauses in the statement had been evaluated in this order:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. ORDER

DB2 does not necessarily process the clauses in this order internally, but the results always look as if they had been processed in this order.

DB2 processes subselects from the innermost to the outermost subselect.

The ORDER BY clause can only be specified in the outermost SELECT statement.

If an AS clause is used for defining a name in the outermost SELECT clause, only the ORDER BY clause can refer to that name.

If an AS clause is used in a subselect, the name it defines can be defined outside of the subselect.

Example:

```
SELECT EMPNO, (SALARY + BONUS + COMM) AS TOTAL_SAL
FROM DSN8710.EMP
ORDER BY TOTAL_SAL;
```

10.5. Comparison Operators Used in Conditions

Type of Comparison	Specified with...	Example
Equal to null	IS NULL	PHONENO IS NULL
Equal to	=	DEPTNO = 'X01'
Not equal to	<>	DEPTNO <> 'X01'
Less than	<	AVG(SALARY) < 30000
Less than or equal to	<=	AGE <= 25
Not less than	>=	AGE >= 21
Greater than	>	SALARY > 2000
Greater than or equal to	>=	SALARY >= 5000
Not greater than	<=	SALARY <= 5000
Similar to another value	LIKE	NAME LIKE '%SMITH%' or STATUS LIKE 'N_'
At least one of two conditions	OR	HIREDATE < '1965-01-01' OR SALARY < 16000
Both of two conditions	AND	HIREDATE < '1965-01-01' AND SALARY < 16000
Between two values	BETWEEN	SALARY BETWEEN 20000 AND 40000
Equals a value in a set	IN (X, Y, Z)	DEPTNO IN ('B01', 'C01', 'D01')

10.6. Selecting Rows that Have Null Values

A null value indicates the absence of a column value in a row.

A null value is not the same as zero or all blanks.

A WHERE clause can be used for retrieving rows that contain a null value in some column.

Example:

Specify:

```
WHERE column-name IS NULL
WHERE column-name IS NOT NULL
```