IBM DB2 Universal Database Road Map to DB2 Programming Version 5

Document Number S10J-8155-00

IBM DB2 Universal Database

Road Map to DB2 Programming

Version 5

S10J-8155-00

IBM DB2 Universal Database

Road Map to DB2 Programming

Version 5

S10J-8155-00

Before using this information and the product it supports, be sure to read the general information under Appendix C, “Notices” on page 61.

Contents

Welcome

Welcome to the world of DB2 programming! This is a client/server environment that provides a rich set of features to develop applications that access DB2 databases and that perform administrative functions.

DB2 provides several different ways to access DB2 databases. You can use embedded SQL, the DB2 Call Level Interface (DB2 CLI), Java, or Open Database Connectivity (ODBC) end-user tools. To perform administrative functions, you can use the DB2 application programming interfaces (APIs).

DB2 also provides many features to build advanced applications: triggers, constraints, stored procedures, user-defined functions, user-defined data types, and support for large objects (such as audio, video, and image) to build multimedia applications.

This book is your road map through these topics. It introduces you to the fundamental programming methods you should understand, describes DB2 features, and points to detailed sources of information.

This book is divided into the following chapters. We suggest you read each one before going to the detailed programming information:

Chapter 1, "About the DB2 Programming Environment" on page 1	Describes the DB2 client/server environment, various development kits, and the sample programs and database.
Chapter 2, "Accessing DB2 Databases" on page 7	Describes different ways your applications can access DB2 databases, and their respective merits and limitations.
Chapter 3, "Using DB2 Features" on page 29	Lists DB2 features you can use to extend or supplement your applications.
Chapter 4, "Using Application Development Tools" on page 43	Lists various tools you can use to develop your applications.

This book also contains a couple of appendixes that provide general information about DB2:

Appendix A, "About DB2 Universal Database" on page 47	Lists the products that make up DB2 Universal Database.
Appendix B, "How the DB2 Library Is Structured" on page 51	Describes the online help, SmartGuides, and books that make up the DB2 library.

Conventions

This book uses the following graphics to help you quickly identify detailed sources of information about the topics being discussed:

Identifies related books in the DB2 library.

Appendix B, “How the DB2 Library Is Structured” on page 51 provides a complete listing of the online help, SmartGuides, and books in the DB2 library.

Identifies related World Wide Web sites.

Identifies some related sample programs.

“About the Sample Programs” on page 4 describes DB2 sample programs in more detail.

Chapter 1. About the DB2 Programming Environment

Quick Beginnings	Provides details about DB2 client/server configurations, and contains installation instructions.

DB2 runs in a client/server environment. In this environment, computers are linked together in a network, and can act as a server, a client, or both. A server provides services to other computers on the network, while a client requests services. In the DB2 client/server environment, servers manage databases, and clients run applications that access databases.

DB2 supports many different client/server configurations. You can have several clients on different platforms connected to one server (as shown in Figure 1), or you can have multiple clients connected to several servers, all on different platforms. You can even have Web-based applications that access DB2 databases across the Internet.

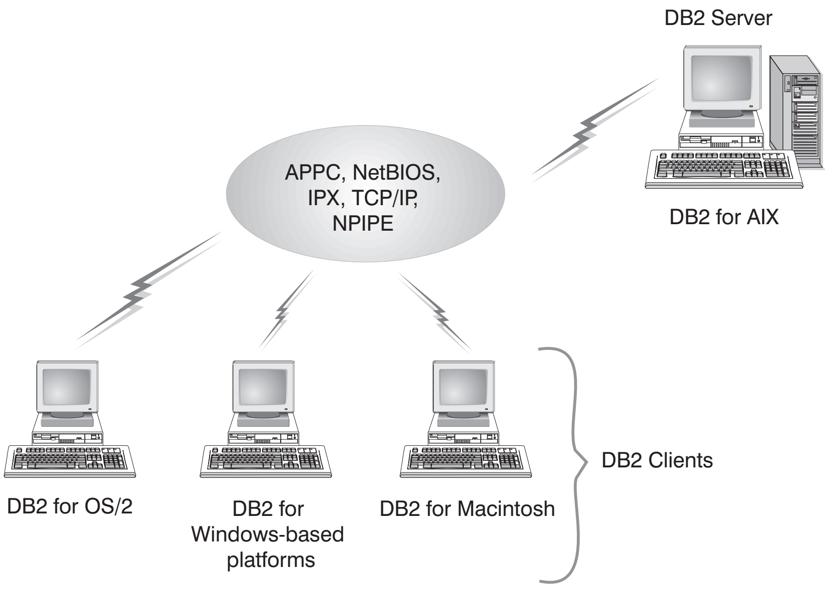

DB2 Server
APPC, NetBIOS, IPX, TCP/IP, NPIPE
DB2 for AIX
DB2 Clients
DB2 for OS/2
DB2 for Windows-based platforms
DB2 for Macintosh

Figure 1. One possible DB2 client/server configuration

A client application that needs to access a DB2 database sends a request to the server. The request is coded in Structured Query Language (SQL), the language used to access DB2 databases. The server processes the request, and returns the data to the client application. In this context, DB2 programming is the development of client applications that access DB2 databases to retrieve, store, or manipulate data.

DB2 programming also includes developing user-defined functions and stored procedures which run on the server. These topics are discussed in “User-defined Functions” on page 35 and “Stored Procedures” on page 39.

You can develop client applications using the following:

- Application Developer's Kit
- Java Development Kit
- An Open Database Connectivity (ODBC) end-user tool such as Lotus Approach. “Using ODBC End-User Tools” on page 21 discusses these tools.

About the Application Developer's Kit

Building Applications for Windows and OS/2 Environments *Building Applications for UNIX Environments*	Each book provides more information about the Software Developer's Kit. Read the book that is appropriate for your platform.

http://www.software.ibm.com/data/	Provides information about DB2 Universal Database, and contains links to DB2 Extenders, Net.Data, and Lotus Approach.
http://www.software.ibm.com/ad/vabasic/vabasic.htm	Provides information about VisualAge for Basic.

The Application Developer's Kit gives you all the tools you need to create database applicatons that run on a variety of platforms, and that connect to any DB2 database. The Application Developer's Kit includes the following products. Each product comes with online documentation which provides additional information, including installation instructions.

Note: Other promotional products might be added from time-to-time.

DB2 Universal Database
A relational database management system that contains features and tools that enable users to create, update, control, and manage relational databases using SQL.

Software Developer's Kits
Provide the tools and environment you need to develop applications that access DB2 databases using embedded SQL or the DB2 Call Level Interface. Software Developer's Kits are available for OS/2, several Windows platforms including the Windows NT operating system, and a variety of UNIX platforms.

DB2 Extenders
Enable your applications to include other types of data besides character and numeric data, such as images, video, audio, and complex documents.

VisualAge for Basic
A suite of application development tools built around an advanced implementation of the Basic programming language that you can use to create GUI client applications, DB2 stored procedures, and DB2 user-defined functions.

Net.Data
A comprehensive World Wide Web development environment to create simple dynamic Web pages or complex Web-based applications that can access DB2 databases.

Lotus Approach
Provides a graphical interface to perform queries, develop reports, and analyze data. You can also develop applications using LotusScript, a full-featured, object-oriented programming language.

About the Java Development Kit

http://www.software.ibm.com/data/db2/java/	Provides information about the Java Development Kit, and the Java Database Connectivity API.

The Java Development Kit provides the tools and environment you need to develop Java applications and applets. To access DB2 databases, you also need to use the DB2 JDBC driver that comes with DB2. The driver supports Sun Microsystem's Java Database Connectivity (JDBC) application programming interface (API), which provides a standard way to access databases from Java code.

Java applications require the DB2 Client Application Enabler to access DB2 databases. Java applets run using a Java-enabled Web browser, and do not require the DB2 Client Application Enabler to be installed on the client.

You can also use the Java programming language to develop user-defined functions and stored procedures which run on the server. These topics are discussed in "User-defined Functions" on page 35 and "Stored Procedures" on page 39.

For more information about developing Java applications and applets, refer to "Using Java" on page 26.

About the Sample Programs

Embedded SQL Programming Guide	Each book provides a complete list of the sample programs, and describes their locations.
Building Applications for Windows and OS/2 Environments	
Building Applications for UNIX Environments	

DB2 supplies many sample programs that show you the different ways you can access DB2 databases: embedded SQL, DB2 Call Level Interface, and DB2 application programming interfaces. You can use the sample programs to learn how to code your applications.

The sample programs are written in a variety of programming languages:

C	FORTRAN
C++	REXX
COBOL	Java

And they are available on a variety of platforms:

AIX	OS/2
HP-UX	Windows 3.1
SCO OpenServer	Windows 95
Silicon Graphics IRIX	The Windows NT operating environment
SINIX	Macintosh
The Solaris Operating Environment	

In subsequent chapters, this book lists some sample programs that are related to the topics under discussion.

About the Sample Database

Administration Guide	Each book contains a detailed listing of the sample database.
SQL Reference	

DB2 supplies a sample database. The sample database contains tables with a variety of data types. You can run the sample programs using the database to see how they work.

Chapter 2. Accessing DB2 Databases

You can choose several different ways to access DB2 databases. You can:

- Embed static and dynamic SQL statements in your applications.
- Code function calls in your applications to the DB2 Call Level Interface (DB2 CLI) to invoke dynamic SQL statements.
- Develop Java applications and applets that call the Java Database Connectivity application programming interface (JDBC API).
- Develop applications using Open Database Connectivity (ODBC) end-user tools such as Lotus Approach and its programming language LotusScript.

To perform administrative functions such as backing up and restoring databases, your applications can call DB2 APIs.

The way your application accesses DB2 databases will depend on the type of application you want to develop. For example, if you want a data entry application, you might choose to embed static SQL statements in your application. If you want an application that performs queries over the World Wide Web, you would probably choose to develop Java applications. Or, if you just want a simple application to query a database, you might code it using LotusScript.

The following sections provide an overview of the ways you can access databases, including their respective merits and limitations. They also point to detailed sources of information.

Using Embedded SQL Statements

SQL Getting Started	Introduces SQL concepts and provides examples for many constructs and tasks.
SQL Reference	Describes SQL syntax, semantics, and the rules of the language.
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL.
Building Applications for UNIX Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a UNIX system.
Building Applications for Windows and OS/2 Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Windows or OS/2 system.

`cursor`	Demonstrates the use of a cursor using static SQL
`delet`	Deletes items from a database using static SQL.
`dynamic`	Demonstrates the use of a cursor using dynamic SQL.
`openftch`	Fetches rows from the database and then updates or deletes them using static SQL.
`static`	Retrieves information using static SQL
`updat`	Updates a database using static SQL.

Structured Query Language (SQL) is the database interface language used to access and manipulate data in DB2 databases. You can embed SQL statements in your applications, enabling them to perform any task supported by SQL, such as retrieving or storing data. Using DB2, you can code your embedded SQL applications in the COBOL, FORTRAN, C, C++, and REXX programming languages. Other products enable you to code your applications in other languages, such as Basic and PL/I. For details about those products, refer to “About the Application Developer's Kit” on page 2 and “Companion Products” on page 44.

An application in which you embed SQL statements is called a *host program*. A programming language that you compile, and in which you embed SQL statements, is called a *host language*. The program and language are defined this way because they host or accommodate SQL statements.

The following table shows parts of host programs written in C and COBOL. The SQL statements begin with EXEC SQL, distinguishing them from the host language statements written in C and COBOL. SQL statement **1** declares an SQLCA data structure, which your application can examine to determine the results of an SQL statement. SQL statement 2 declares a host variable, which you can use to pass data between your application and the database.

C Host Program

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlenv.h>

EXEC SQL INCLUDE SQLCA; 1

int main(void)
{
   EXEC SQL BEGIN DECLARE SECTION; 2
      char statement [81] = " ";
   EXEC SQL END DECLARE SECTION;
```

COBOL Host Program

```
DATA DIVISION.
WORKING-STORAGE SECTION.

COPY "sql.cbl".
COPY "sqlenv.cbl".

EXEC SQL INCLUDE SQLCA END-EXEC. 1

EXEC SQL BEGIN DECLARE SECTION END-EXEC. 2
  01 STATEMENT    PIC X(81).
EXEC SQL END DECLARE SECTION END-EXEC.
```

A Typical Embedded SQL Application

Your application will usually contain the following parts. (Figure 2 on page 10 also shows the parts in pseudocode format. Refer to the programming guides for the exact syntax of the SQL statements.)

Initialization

You declare all the variables and data structures that the database manager uses to interact with the host program. The pseudocode example declares the variables 1 `USERID` and 2 `PW`, and the data structure 3 `SQLCA`. The variables are then referenced in the SQL statement:

5 `EXEC SQL CONNECT TO` *databaseA* `USER :USERID USING :PW`

You also include one or more SQL statements to handle warnings and errors. In the example, an SQL statement causes control to pass to the statements identified by the host label ERRCHK:

4 `EXEC SQL WHENEVER SQLERROR GOTO ERRCHK`

Transaction Processing

Transaction processing consists of one or more *units of work* or *transactions*. A unit of work or transaction is a sequence of SQL statements (possibly with intervening host language code) that the database manager treats as a whole. For example, a transaction can be used to deduct money from one account and to add it to another.

A transaction begins implicitly with the first executable SQL statement. For example, this can be the CONNECT statement 5 to connect to the database you want to access. Or it can be some other SQL statement, such as SELECT to select a row from the database, or INSERT to insert a row into the database.

```
Start Program                                                          ┐
EXEC SQL BEGIN DECLARE SECTION                                         │
     DECLARE USERID FIXED CHARACTER (8) 1                              │
     DECLARE PW FIXED CHARACTER (8) 2                                  │
          ●                                                            │
          ●                                                            │ Initialization
     (other host variable declarations)                                │
          ●                                                            │
          ●                                                            │
EXEC SQL END DECLARE SECTION                                           │
EXEC SQL INCLUDE SQLCA 3                                               │
EXEC SQL WHENEVER SQLERROR GOTO ERRCHK 4                               ┘
          ●
          ●
          ●
     EXEC SQL CONNECT TO databaseA USER  :USERID USING  :PW  5         ┐ Transaction
     EXEC SQL SELECT  . . .                                            │ Processing
     EXEC SQL INSERT  . . .                                            │
          ●                                                            │ First Unit
     EXEC SQL COMMIT                                                   ┘ of Work
          ●
          ●
          ●
     EXEC SQL CONNECT TO databaseB USER  :USERID USING  :PW            ┐
     EXEC SQL SELECT  . . .                                            │ Second Unit
     EXEC SQL DELETE  . . .                                            │ of Work
          ●                                                            │
     EXEC SQL COMMIT                                                   ┘
          ●
          ●
          ●
EXEC SQL CONNECT RESET 6                                               ┐
ERRCHK                                                                 │
          ●                                                            │ Termination
          ●                                                            │
          ●                                                            │
End Program                                                            ┘
```

Figure 2. A typical host program in pseudocode format

A transaction ends with either a COMMIT or a ROLLBACK statement, or when the program ends. To commit means to complete the transaction. To rollback means to undo the transaction.

Termination

You release your connection to the database, and clean up resources used by the program, such as temporary storage or data structures. The pseudocode example releases the connection to the database with the statement:

```
6 EXEC SQL CONNECT RESET
```

Building Your Embedded SQL Application

When you embed SQL statements in your application, you must precompile and bind your application to a database, following the steps listed below and in Figure 3 on page 12.

1 Create source files that contain programs with embedded SQL statements.

2 Connect to a database, then precompile each source file.

The precompiler converts the SQL statements in each source file into DB2 run-time API calls to the database manager. The precompiler also produces an *access package* in the database and, optionally, a bind file, if you specify that you want one created.

The access package (or *package*) contains access plans selected by the DB2 optimizer for the *static* SQL statements in your application. The access plans contain the information required by the database manager to execute the static SQL statements in the most efficient manner as determined by the optimizer. For *dynamic* SQL statements, the optimizer creates access plans when you run your application. Static and dynamic SQL statements are explained in "Comparing Static SQL versus Dynamic SQL" on page 12.

The *bind file* contains the SQL statements and other data required to create an access package. You can use the bind file to rebind your application later without having to precompile it first. Rebinding creates access plans that are optimized for current database conditions. You need to rebind your applications if your application will access a different database from the one against which it was precompiled.

3 Compile the modified source files (and other files without SQL statements) using the host language compiler.

4 Link the object files with the DB2 and host language libraries to produce an executable program.

5 Bind the bind file to create the access package if this was not already done at precompile time, or if a different database is going to be accessed.

6 Run the application. The application accesses the database using the access plan in the package.

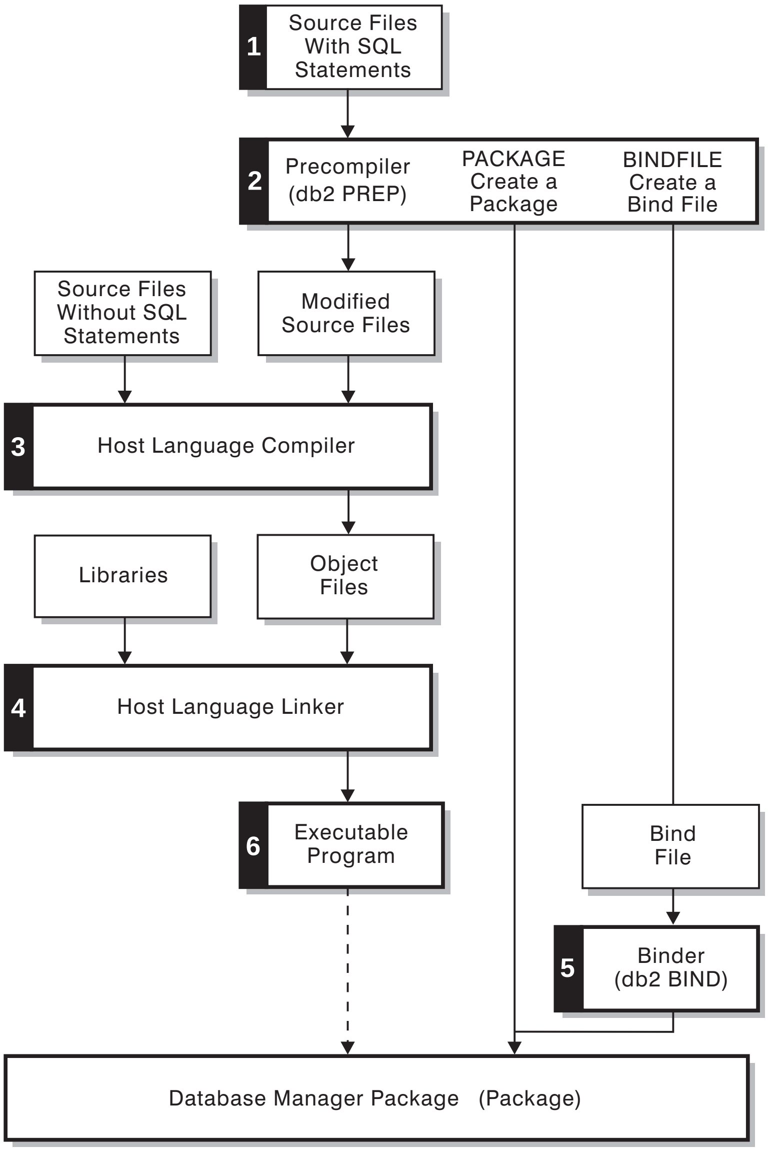


Figure 3. Building your embedded SQL application

Comparing Static SQL versus Dynamic SQL

Static SQL statements are ones where you know, before compile time, the SQL statement type and the table and column names. The only unknowns are the specific data values the statement is searching for or updating. You can represent those values in host language variables. You prepare or build static SQL statements before you run your application.

In contrast, dynamic SQL statements are those statements that your application builds and executes at run time. An interactive application that prompts the end user for key parts of an SQL statement, such as the names of the tables and columns to be searched, is a good example of dynamic SQL. The application builds the SQL statement while it's running, and then submits the statement for processing.

You can write applications that have only static SQL statements or only dynamic SQL statements or a mix of both.

Static SQL

Merits	Limitations
Ease of programming: Static SQL applications are generally straight forward to program. You simply embed SQL statements in your source file. For dynamic SQL applications, you need to use host variables or a data structure to handle the parts of the SQL statement that aren't known during development, as well as additional SQL statements such as PREPARE, EXECUTE, and DESCRIBE. **Persistence:** The access plan for static SQL statements is retained in the database in a ready-to-execute package, and can be reused. For dynamic SQL statements, the access plan must be created each time you run your application. **Encapsulation and security:** With static SQL applications, you can limit the access end users have to data. For example, you can grant a user permission to run an application that updates a table without giving the user general access to the table. You do this by granting authorizations to run an access package when you bind your application. This encapsulates access authority in the package, giving selected users permission only to run a package, not to access the entire table. In dynamic SQL applications, authorizations are validated at run time on a per statement basis. Therefore, in order to run a dynamic SQL statement, a user must have explicit access privileges for each database object.	**Fixed access paths:** The performance of static SQL statements depends on the state of the database the last time you bound your application. If you add an index to the database at a later time, your static application cannot take advantage of the index unless you rebind it. Similarly, adding a large number of rows to a table will change the statistics that were used when the static SQL statement was originally bound to the database. Since dynamic SQL statements are parsed and optimized at run time, your dynamic application can take advantage of or react to any changes to the database every time you run it. **Inflexibility:** Applications that use static SQL are less flexible because you must code the transactions that the end user is going to issue against the database, and you cannot change them without recoding, recompiling, and rebinding the application to the database. Interactive applications with dynamic SQL are more flexible because the end user can specify the SQL statements at run time, either manually or using software.

Generally, static SQL statements are well-suited for high-performance applications with predefined transactions. A reservation system is a good example of such an application.

Dynamic SQL

Merits	Limitations
Flexibility: You can develop flexible applications in which end users specify the SQL statements at run time. This is ideal for applications where you don't know the complete SQL statements during development. For static SQL applications, you must know all the parts except the specific data values the statement is searching for or updating which you can represent with a host language variable. **Late access path binding:** Dynamic SQL statements always use an optimal access plan because the statements are parsed and optimized at run time. This is ideal if the state of the database changes often, and can improve the performance of your application. Static SQL statements use the access plan created at precompile time. **Object independence:** Database objects that dynamic SQL statements access do not have to exist at precompile or bind time, only at run time. For static SQL statements, all the objects they reference must exist at bind time, but not necessarily at precompile time.	**Start-up costs:** In general, an application that uses dynamic SQL has a higher start-up cost per SQL statement than static SQL because dynamic SQL statements need to be compiled before your application can use them. However, each time your application runs a dynamic SQL statement, the initial cost to compile becomes less of a factor. If multiple users run the same dynamic application with the same statements, only the first application to issue the statement incurs the compilation cost. **Unpredictability:** The performance behavior of dynamic SQL applications might differ during each execution because dynamic statements are prepared at run time, and the status of the database may change from one execution to the next. Static SQL applications run using an access package created at precompile time. The performance behavior of the application remains constant until you rebind it.

Generally, dynamic SQL statements are well-suited for applications that run against a rapidly changing database where transactions need to be specified at run time. An interactive query interface is a good example of such an application.

Using the DB2 Call Level Interface

SQL Getting Started	Introduces SQL concepts and provides examples for many constructs and tasks.
SQL Reference	Describes SQL syntax, semantics, and the rules of the language.
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface.
Building Applications for UNIX Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a UNIX system.
Building Applications for Windows and OS/2 Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Windows or OS/2 system.

`basiccon.c`	Shows a basic connection.
`colpriv.c`	Lists column privileges.
`columns.c`	Lists all columns for table search string.
`compnd.c`	Demonstrates compound SQL.
`fetch.c`	Demonstrates a simple fetch sequence.
`getattrs.c`	Lists some common environment, connection, and statement options/attributes.
`samputil.c`	Demonstrates utility functions used by most DB2 CLI samples.
`samputil.h`	Is the header file for samputil.c, included by most samples.
`setcolat.c`	Sets column attributes using SQLSetColAttributes().
`tables.c`	Lists all tables.
`xfetch.c`	Shows extended fetch, multiple rows per fetch.

DB2 CLI is a programming interface that your C or C++ applications can use to access DB2 databases. DB2 CLI is based on the Microsoft Open Database Connectivity (ODBC) specification, and the ISO CLI standard. Basing DB2 CLI on industry standards might result in a shorter learning curve for application programmers who are already familiar with these database interfaces.

When you use DB2 CLI, your application passes dynamic SQL statements as function arguments to the database manager for processing. As such, DB2 CLI is an alternative to embedded dynamic SQL, providing another way to access DB2 databases using dynamic SQL.

A Typical DB2 CLI Application

Your application will usually contain the following parts:

Initialization

Initialization allocates environment and connection handles, and connects your application to the database. A handle is a variable that refers to a data object controlled by the DB2 CLI. There are environment, connection, statement, and descriptor handles.

Using handles frees your application from having to allocate and manage global variables or data structures such as the SQLCA used in embedded SQL applications.

Figure 4 on page 17 shows the function call sequences for initialization.

Transaction Processing

Transaction processing consists of the five steps shown below and in Figure 5 on page 18.

1 Allocate statement handles before any SQL statements can be executed. A statement handle refers to the data object that contains information about an SQL statement managed by DB2 CLI. This includes information such as dynamic parameters, cursor information, result values, and status information.

2 Prepare and execute SQL statements using one of two methods. The transaction starts implicitly with the first access to the database.

- Prepare then execute, which splits the preparation of the statement from the execution. You would use this method if the statement will be executed repeatedly, usually with different parameter values.
- Execute directly, which combines the prepare and execute steps into one. You would use this method if the statement will be executed only once. This avoids having to call two functions to execute the same statement.

3 Process the results of the SQL statement. The method you use depends on the type of SQL statement:

- Receive Query Results: If the statement is a query, you usually need to perform the following steps to retrieve each row of the result set:
  1. Establish (describe) the structure of the result set, number of columns, column types and lengths, using `SQLNumResultCols()` and `SQLDescribeCol()` or `SQLColAttribute()`.
  2. Optionally bind application variables to columns in order to receive the data, using `SQLBindCol()`.
  3. Repeatedly fetch the next row of data, and receive it into the bound application variables, using `SQLFetch()`.

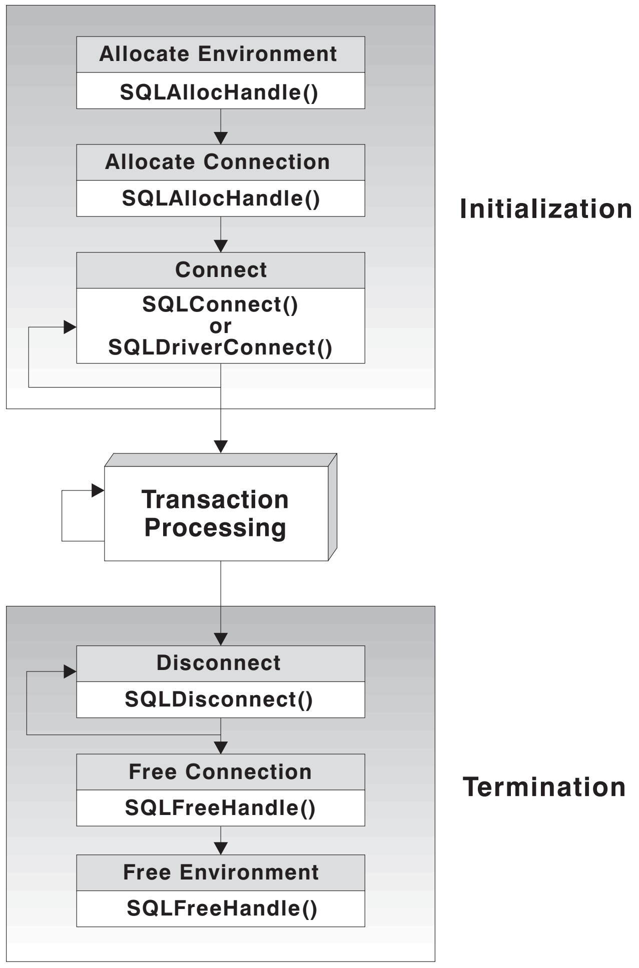


Figure 4. DB2 CLI function calls for initialization and termination

4. Optionally retrieve columns that were not previously bound, using `SQLGetData()` after each successful fetch.

- Update Data: If the statement is modifying data, your application needs only to check for diagnostic messages. Your application can use `SQLRowCount()` to get the number of rows affected by the SQL statement.
- Other: If the statement neither queries nor modifies the data, your application needs only to check for diagnostic messages.

4 Free the statement handle ending processing for that statement.

5 Commit (complete) or roll back (undo) the transaction. The transaction ends at this point.

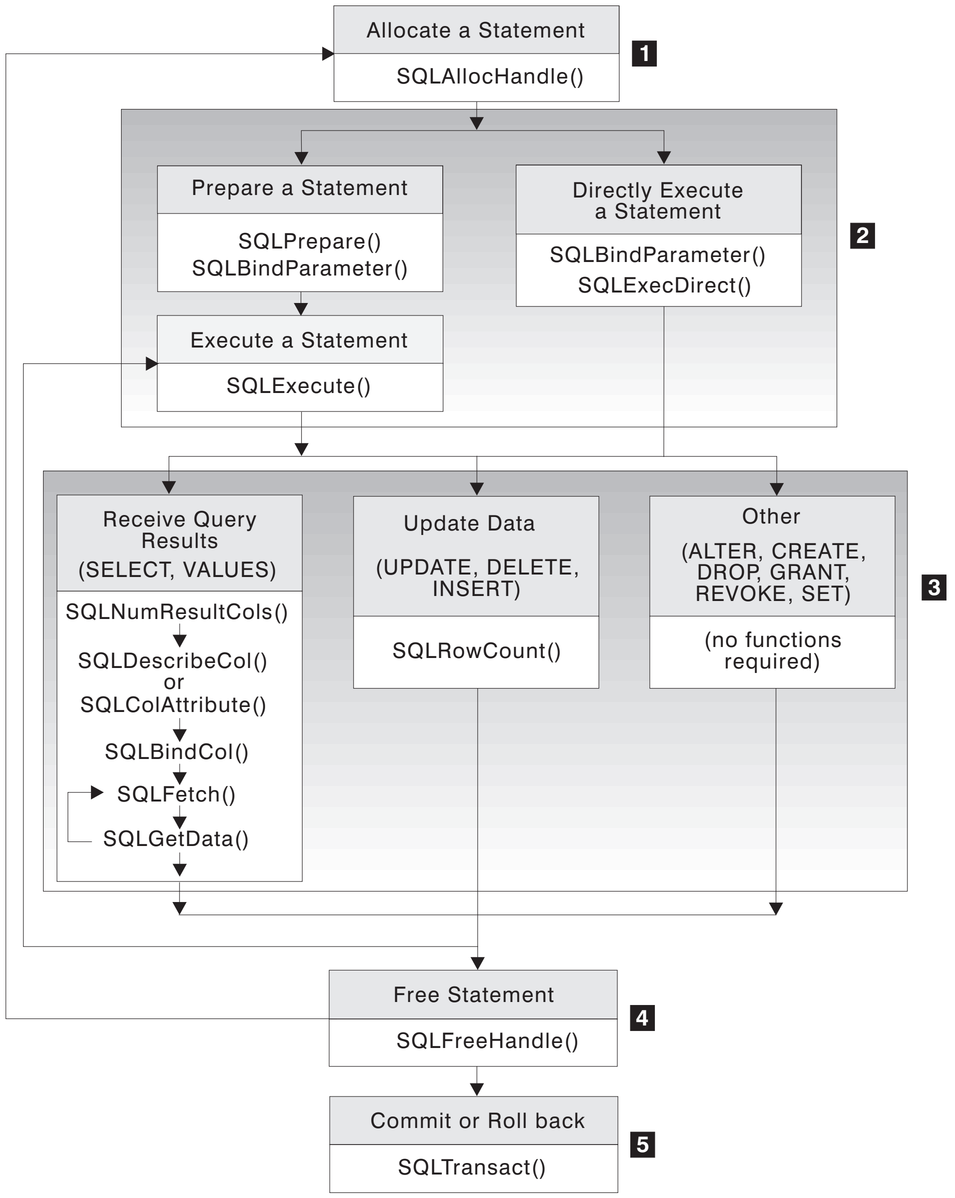


Figure 5. DB2 CLI function calls for transaction processing

Termination

Termination disconnects your application from the database, and frees the handles.

Figure 4 on page 17 shows the function call sequences for termination.

Building Your DB2 CLI Application

You do not need to precompile or bind DB2 CLI applications because they use common access packages provided with DB2. You simply compile and link your application.

However, before your DB2 CLI or ODBC applications can access DB2 databases, the DB2 CLI bind files must be bound on each DB2 database that will be accessed. This occurs automatically on the first connection to the database, but we recommend that the database administrator bind the bind files from one client on each platform that will access a DB2 database.

For example, suppose you have OS/2, AIX, and Windows 95 clients that each access two DB2 databases. The administrator must bind the bind files from one OS/2 client on each database that will be accessed. Next, the administrator must bind the bind files from one AIX client on each database that will be accessed. Finally, the administrator must do the same on one Windows 95 client.

Comparing DB2 CLI Versus Embedded Dynamic SQL

You can develop dynamic applications using either embedded dynamic SQL statements or DB2 CLI. In both cases, SQL statements are prepared and processed at run time. Each method has unique advantages listed below.

DB2 CLI Advantages

Portability
DB2 CLI applications use a standard set of functions to pass SQL statements to the database. All you need to do is compile and link DB2 CLI applications before you can run them. In contrast, you must precompile embedded SQL applications, compile them, and then bind them to the database before you can run them. This process effectively ties your application to a particular database.

No binding
You do not need to bind individual DB2 CLI applications to each database they access. You only need to bind the bind files that are shipped with DB2 CLI once for all your DB2 CLI applications. This can significantly reduce the amount of time you spend managing your applications.

Extended fetching and input
DB2 CLI functions enable you to retrieve multiple rows in the database into an array with a single call. They also let you execute an SQL statement many times using an array of input variables.

Consistent interface to catalog

Database systems often contain catalog tables that have information about the database and its users. The form of these catalogs can vary among systems. DB2 CLI provides a consistent interface to query catalog information about tables, columns, foreign and primary keys, and user privileges. This shields your application from catalog changes across releases of database servers, and from differences among database servers. You don't have to write catalog queries that are specific to a particular server or product version.

Extended data conversion

DB2 CLI automatically converts data between SQL and C data types. For example, fetching any SQL data type into a C `char` data type converts it into a character-string representation. This makes DB2 CLI well-suited for interactive query applications

No global data areas

DB2 CLI eliminates the need for application controlled, often complex global data areas, such as SQLDA and SQLCA, typically associated with embedded SQL applications. Instead, DB2 CLI automatically allocates and controls the necessary data structures, and provides a handle for your application to reference them.

Retrieve result sets from stored procedures

DB2 CLI applications can retrieve multiple rows and result sets generated from a stored procedure residing on the server.

Scrollable cursors

DB2 CLI supports server-side scrollable cursors that can be used in conjunction with array output. This is useful in GUI applications that display database information in scroll boxes that make use of the Page Up, Page Down, Home and End keys. You can declare a read-only cursor as scrollable and then move forwards or backwards through the result set by one or more rows. You can also fetch rows by specifying an offset from the current row, the beginning or end of a result set, or a specific row you bookmarked previously.

Embedded Dynamic SQL Advantages

Supports many programming languages

Embedded SQL supports more that just C and C++. This might be an advantage if you prefer to code your applications in another language.

Consistent with static SQL

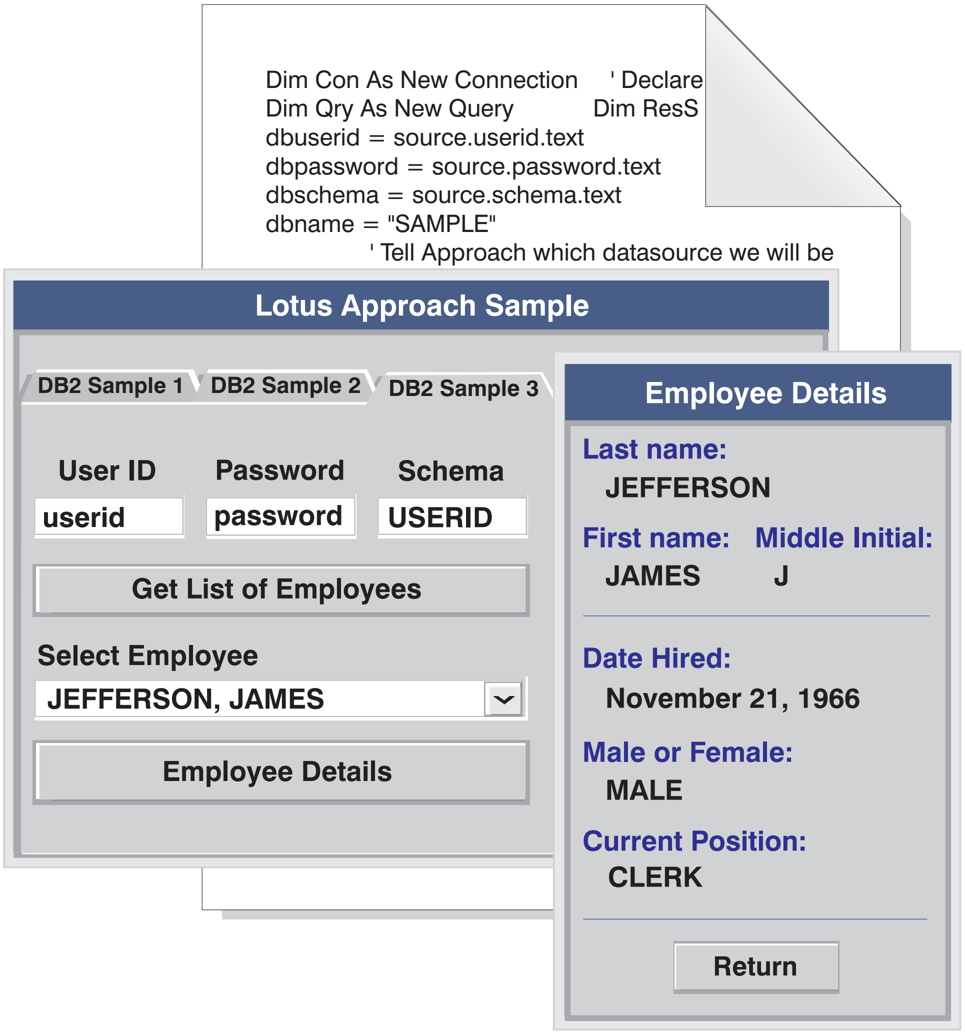
Dynamic SQL is generally more consistent with static SQL. If you already know how to program static SQL, moving to dynamic SQL might not be as difficult as moving to DB2 CLI. You might prefer dynamic SQL if you also need to use static SQL statements in your dynamic SQL applications, instead of mixing static SQL and DB2 CLI.

Using ODBC End-User Tools

http://www.software.ibm.com/data/ The url takes you to the Data Management page. From there, click on DB2 Family, and then DB2 and Lotus Approach.	Provides more details about DB2 and Lotus Approach.

There might be cases where you need an application to perform a basic task, such as querying the database. You can use ODBC end-user tools such as Lotus Approach, Microsoft Access, and Microsoft VisualBasic to create these applications. ODBC tools provide a simpler alternative to developing applications than using a high-level programming language.

Lotus Approach provides two ways to access DB2 data. You can use the graphical interface to perform queries, develop reports, and analyze data. Or you can develop applications using LotusScript, a full-featured, object-oriented programming language that comes with a wide array of objects, events, methods, and properties, along with a built-in program editor. Figure 6 on page 22 shows a query application developed using LotusScript.



```
Dim Con As New Connection     ' Declare
Dim Qry As New Query          Dim ResS
dbuserid = source.userid.text
dbpassword = source.password.text
dbschema = source.schema.text
dbname = "SAMPLE"
                ' Tell Approach which datasource we will be
```

User ID	Password	Schema
userid	password	USERID

Select Employee: JEFFERSON, JAMES

Employee Details

Last name: JEFFERSON

First name: JAMES Middle Initial: J

Date Hired: November 21, 1966

Male or Female: MALE

Current Position: CLERK

Figure 6. A simple query application developed using LotusScript

Merits and Limitations

Merits	Limitations
Quick access to data: Tools such as Lotus Approach provide a quick way to access data using their graphical interface. **Quick application development:** Tools such as Lotus Approach provide a quick way to develop relatively simple applications using its scripting language. **Rapid skills transfer:** If you have previous ODBC knowledge, you can use that knowledge to develop ODBC applications that access DB2 databases.	**Not suitable for complex applications:** The scripting language for some ODBC tools might not be suitable for developing complex applications. A high-level language such as C might be a more appropriate choice. **May not be able to access DB2 features and functions:** You might not be able to access some DB2 features (such as large object support), or perform some DB2 functions (such as creating databases or indexes).

Using DB2 APIs

API Reference	Describes the DB2 application programming interfaces (APIs) and data structures you can use to manage your databases. Explains how to call APIs from your applications.
Administration Getting Started	Introduces basic DB2 database administration concepts and tasks, and describes server administration tools.

	APIs Demonstrated:
backrest	Backup/Recovery: BACKUP DATABASE RESTORE DATABASE ROLL FORWARD DATABASE
d_dbconf	Database Configuration: GET DATABASE CONFIGURATION DEFAULTS
dbcat	Node Directory Management: CATALOG DATABASE CLOSE DATABASE DIRECTORY SCAN GET NEXT DATABASE DIRECTORY ENTRY OPEN DATABASE DIRECTORY SCAN UNCATALOG DATABASE
dbconf	Database Control/Configuration: CREATE DATABASE DROP DATABASE GET DATABASE CONFIGURATION RESET DATABASE CONFIGURATION UPDATE DATABASE CONFIGURATION
dbstart	Database Manager Control: START DATABASE MANAGER
dbstop	Database Manager Control: FORCE USERS STOP DATABASE MANAGER
impexp	Data Utilities: EXPORT IMPORT
makeapi	Application Preparation: BIND PRECOMPILE PROGRAM START DATABASE MANAGER STOP DATABASE MANAGER
tabspace	Table Space Management: TABLESPACE QUERY SINGLE TABLESPACE QUERY OPEN TABLESPACE QUERY FETCH TABLESPACE QUERY GET TABLESPACE STATISTICS CLOSE TABLESPACE QUERY

When writing your applications, you might need to perform some database administration tasks, such as creating, activating, backing up, or restoring a database. DB2 provides numerous APIs so you can perform these tasks from your applications, including embedded SQL and DB2 CLI applications. This enables you to program the same administrative functions into your applications that you can perform using the DB2 server administration tools.

Additionally, you might need to perform specific tasks that can only be performed using the DB2 APIs. For example, you might want to retrieve the text of an error message so your application can display it to the end user. To retrieve the message, you must use the Get Error Message API.

Tasks

The following table lists some of the tasks you can perform using DB2 APIs:

APIs	Sample Tasks
Backup/Recovery	Back up, restore, roll forward database; open history file scan, prune and update history file
Database Control	Activate, create, deactivate, drop, migrate, restart database
Database Manager Control	Start and stop database manager
Database Directory Management	Catalog and uncatalog database; open and close database directory scan
Node Directory Management	Catalog and uncatalog node; open and close node directory scan
Database Configuration	Get, reset, and update database configuration
Database Monitoring	Get monitor switches; get snapshot
Operational Utilities	Force application; quiesce tablespaces for table; reorganize table; run statistics
Data Utilities	Export; import; load; load query
General Application Programming	Get instance; get error message; dereference address; copy memory; free memory; get address
Application Preparation	Precompile program; bind; rebind
Remote Server Utilities	Attach; detach
Table Space Management	Tablespace query; open, fetch, and close tablespace query; get tablespace statistics
Miscellaneous	Get authorizations; query client; initialize and link to device; delete committed session

Merits and Limitations

Merits	Limitations
Flexibility: You can use DB2 APIs in both embedded SQL and DB2 CLI applications. **Many programming languages:** You can code applications in C, COBOL, FORTRAN, and REXX to call DB2 APIs.	**Not easily ported:** Applications that use DB2 APIs cannot be ported easily to other database products.

Using Java

SQL Getting Started	Introduces SQL concepts, and provides examples for many constructs and tasks.
SQL Reference	Describes SQL syntax, semantics, and the rules of the language.
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL.
Building Applications for Windows and OS/2 Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Windows or OS/2 system.
Building Applications for UNIX Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a UNIX system.

http://www.software.ibm.com/data/db2/java/	Provides information about the Java Database Connectivity (JDBC) API.

There might be cases where you need an application that can access DB2 databases across the Internet. Using the Java programming language, you can develop applications and applets that access and manipulate data in DB2 databases.

DB2 makes this possible by providing support for the Sun Microsystem's Java Database Connectivity (JDBC) API. DB2 provides this support through a DB2 JDBC driver that comes with DB2. The JDBC API, which is similar to ODBC APIs, provides a standard way to access databases from Java code. Your Java code passes SQL statements as function arguments to the DB2 JDBC driver. The driver handles the JDBC API calls from your client Java code.

Java's portability enables you to deliver DB2 access to clients on multiple platforms, requiring only a Java-enabled Web browser.

You can also use the Java programming language to develop user-defined functions and stored procedures which run on the server. These topics are discussed in “User-defined Functions” on page 35 and “Stored Procedures” on page 39.

Java Applications

Java applications rely on the DB2 Client Application Enabler to connect to DB2. You start your application from the desktop or command line, like any other application. The DB2 JDBC driver handles the JDBC API calls from your application, and uses the Client Application Enabler to communicate the requests to the server and to receive the results.

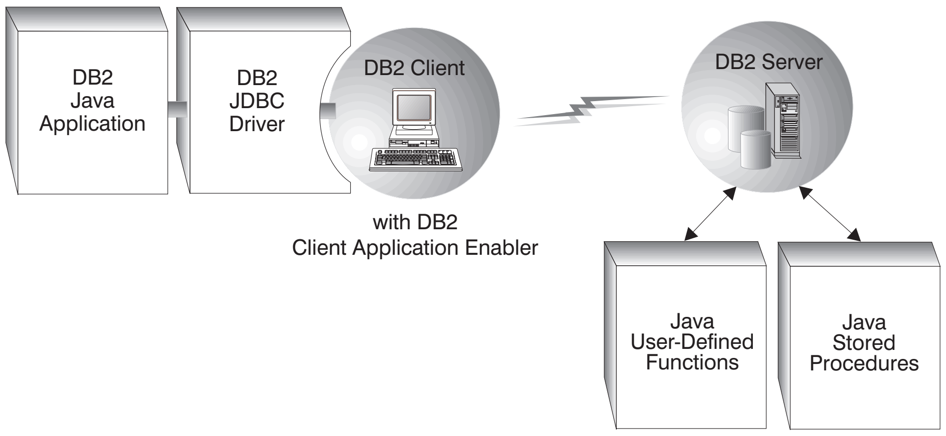


Figure 7. Java applications

Java Applets

Java applets do not require the DB2 Client Application Enabler on the client. Typically, you would imbed the applet in a HyperText Markup Language (HTML) page.

To run your applet, you need only a Java-enabled Web browser on the client machine. When you load your HTML page, the browser downloads the Java applet to your machine, which then downloads the Java class files and DB2's JDBC driver. When your applet calls the JDBC API to connect to DB2, the JDBC driver establishes a separate network connection with the DB2 database through the JDBC applet server residing on the Web server.

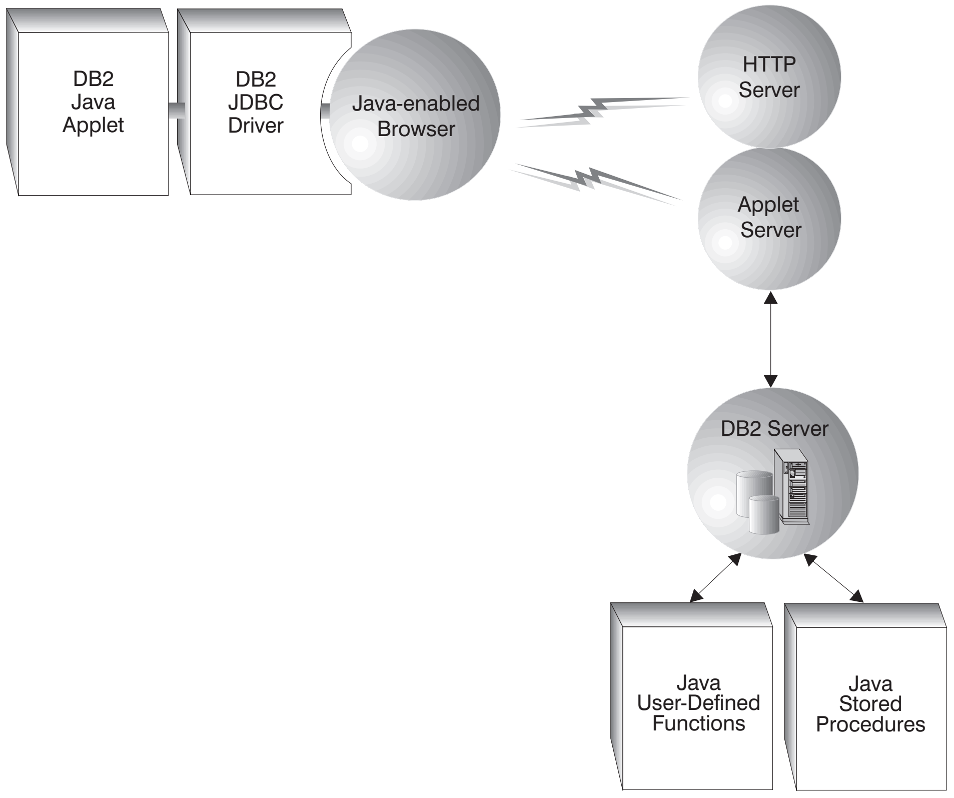


Figure 8. Java applets

Java UDFs and Stored Procedures

You can also develop user-defined functions (UDFs) and stored procedures in the Java language. This process is similar to creating UDFs and stored procedures in any other programming language. Once you create and register your Java UDFs and stored procedures, you can call them from applications coded in any supported programming language. These topics are discussed in “User-defined Functions” on page 35 and “Stored Procedures” on page 39.

Chapter 3. Using DB2 Features

DB2 comes with a variety of features that run on the server which you can use to supplement or extend your applications. When you use DB2 features, you do not have to write your own code to perform the same tasks.

DB2 also lets you store some parts of your code at the server instead of keeping all of it in your client application. This can have performance and maintenance benefits.

The following table lists some key DB2 features, and what they provide. There are features to protect data and to define relationships between data. And there are object-relational features to create flexible, advanced applications. You can use some features in more than one way, such as constraints, which enable you to protect data and to define relationships between data values.

	Protect Data	**Define Relation-ships**	**Object Relational**
"Constraints" on page 30	X	X	
"User-defined Types and Large Objects" on page 33	X		X
"User-defined Functions" on page 35			X
"Triggers" on page 37	X	X	
"Stored Procedures" on page 39	X		

To decide whether or not to use DB2 features, consider the following points:

Application independence
: You can make your application independent of the data it processes. Using DB2 features that run at the database enables you to maintain and change the logic surrounding the data without affecting your application. If you need to make a change to that logic, you only need to change it in one place; at the server, and not in each application that accesses the data.

Performance
: You can make your application perform more quickly by storing and running parts of your application on the server. This shifts some processing to generally more powerful server machines, and can reduce network traffic between your client application and the server.

Application requirements
: Your application might have unique logic that other applications do not. For example, if your application processes data entry errors in a particular order that would be inappropriate for other applications, you might want to write your own code to handle this situation.

In some cases, you might decide to use DB2 features that run on the server because they can be used by several applications. In other cases, you might decide to keep logic in your application because it is used by your application only.

The following sections provide an overview of DB2 features, and point to detailed sources of information.

Constraints

SQL Reference	Describes SQL syntax, semantics, and the rules of the language. Chapter 2 contains details about constraints.
Administration Guide	Contains information required to design, implement, and maintain a database. Chapter 4 contains details about constraints.

	Embedded SQL	**DB2 CLI**	
`trigsql`	X		Demonstrates triggers and constraints.

To protect data and to define relationships between your data, you usually define *business rules* (or *rules*). Rules define what data values are valid for a column in a table, or how columns in one or more tables are related to each other.

DB2 provides *constraints* as a way to enforce those rules using the database system. By using the database system to enforce business rules, you don't have to write code in your application to enforce them. However, if a business rule applies to one application only, you should code it in the application instead of using a global database constraint.

DB2 provides different kinds of constraints. You define constraints using the SQL statements CREATE TABLE and ALTER TABLE.

NOT NULL constraints
NOT NULL constraints prevent null values from being entered into a column.

UNIQUE constraints
UNIQUE constraints ensure that the values in a set of columns are unique and not null for all rows in the table. For example, a typical UNIQUE constraint in a DEPARTMENT table might be that the department number is unique and not null.

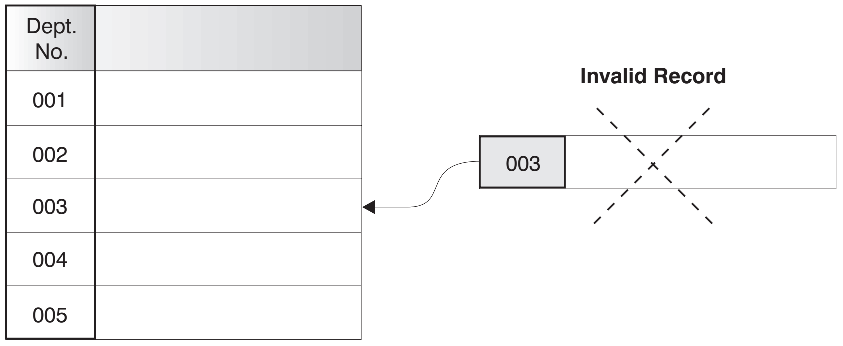


Figure 9. UNIQUE constraints prevent duplicate data

The database manager enforces the constraint during insert and update operations, ensuring data integrity.

PRIMARY KEY constraint

Each table can have one PRIMARY KEY. A PRIMARY KEY is a column or combination of columns that has the same properties as a UNIQUE constraint.

FOREIGN KEY constraints

FOREIGN KEY constraints (also known as referential integrity constraints) enable you to define required relationships between and within tables.

A FOREIGN KEY references a set of columns in either the same table or another table that comprise a PRIMARY KEY or a UNIQUE constraint. This creates a constraint so that any value in the FOREIGN KEY must match an existing value in the referenced key (either the PRIMARY KEY or a UNIQUE constraint).

For example, you might want to ensure that every employee in the EMPLOYEE table is a member of an existing department as defined in the DEPARTMENT table. To establish this relationship, you define the department number in the EMPLOYEE table as the FOREIGN KEY, and the department number in the DEPARTMENT table as the PRIMARY KEY.

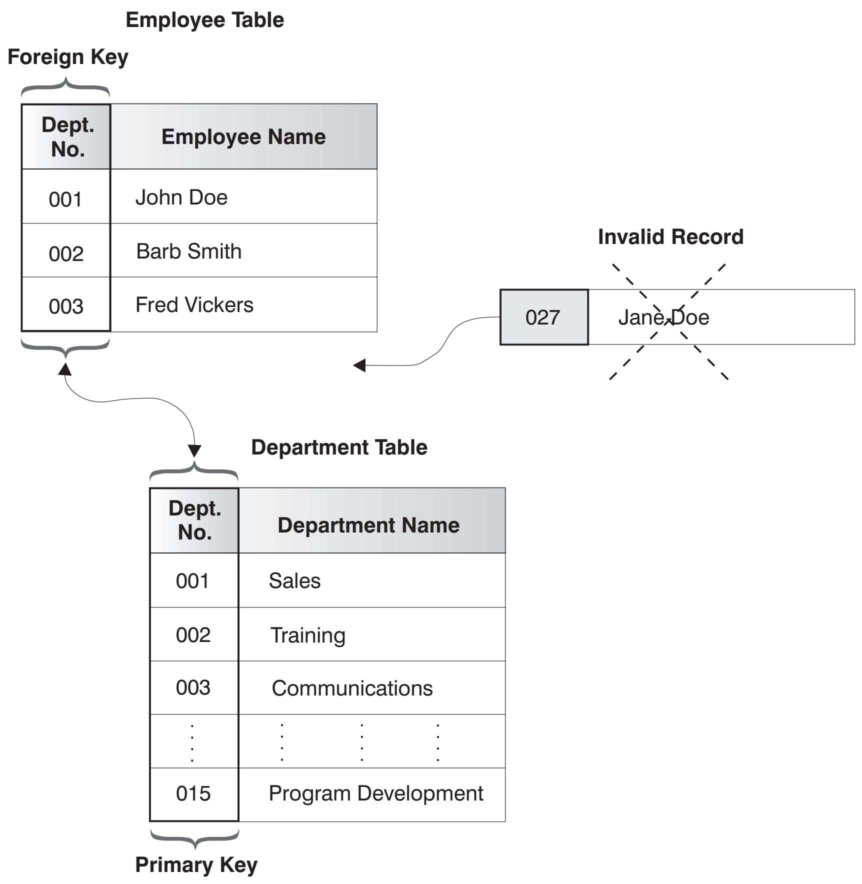


Figure 10. FOREIGN and PRIMARY KEY constraints define relationships and protect data

With this relationship, you can control updates to the EMPLOYEE table by ensuring that the department number is valid, as defined in the DEPARTMENT table. The database can reject any EMPLOYEE record with a department number that is not valid. This ensures data integrity.

CHECK constraints

A CHECK constraint is a rule in the database that specifies the values allowed in one or more columns of every row of a table.

For example, in an Employee table, you can define the Type of Job column to be 'Sales', 'Manager', or 'Clerk'. With this constraint, any record with a different value in the Type of Job column is invalid and would be rejected, enforcing rules about the type of data allowed in the table

User-defined Types and Large Objects

SQL Reference	Describes SQL syntax, semantics, and the rules of the language. Chapters 2, 3 and 6 contain details about user-defined types and large objects.
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL. Chapter 6 contains details about user-defined types and large objects.
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface. Chapter 3 contains details about user-defined types and large objects.
Administration Guide	Contains information required to design, implement, and maintain a database. Chapter 4 contains details about user-defined types and large objects.

	Embedded SQL	**DB2 CLI**	
`lobeval`	X		Demonstrates the use of LOB locators, and deferring the evaluation of the actual LOB data.
`lobfile`	X		Demonstrates the use of LOB handles.
`lobloc`	X		Demonstrates the use of LOB locators.
`lobval`	X		Demonstrates the use of LOBs.
`showpic.c`		X	Extracts a binary large object (BLOB) picture to file using SQLBindColToFile() then displays it using xwud on XWindows, iconedit on OS/2.
`showpic2.c`		X	Extracts a BLOB picture to a file using piecewise output then displays it using xwud on XWindows, iconedit on OS/2.
`picin.c`		X	Loads graphic BLOBs into the emp_photo sample table directly from a file using SQLBindParamToFile().
`picin2.c`		X	Loads graphic BLOBs into the emp_photo sample table using SQLPutData.

Every data element in the database is stored in a column of a table, and each column is defined to have a data type. The data type places limits on the types of values you can put into the column and the operations you can perform on them. For example, a column of integers can only contain numbers within a fixed range. DB2 includes a set of built-in data types with defined characteristics and behaviors: character strings, numerics, datetime values, and large objects.

Sometimes, however, the built-in data types might not serve the needs of your applications. DB2 provides *user-defined types* (UDTs) which enable you to define the distinct data types you need for your applications.

UDTs are based on the built-in data types. When you define a UDT, you also define the operations that are valid for the UDT. For example, you might define a MONEY data type that is based on the DECIMAL data type. However, for the MONEY data type, you might allow only addition and subtraction operations, but not multiplication and division operations.

Large objects (LOBs) enable you to store and manipulate large, complex data objects in the database; objects such as audio, video, images, and large documents. UDTs are often based on LOBs.

The combination of UDTs and LOBs gives you considerable power. You are no longer restricted to using the built-in data types provided by DB2 to model your business data, and to capture the semantics of that data. You can use UDTs to define large, complex data structures for advanced applications.

In addition to extending built-in data types, UDTs provide several other benefits:

Support for object-oriented programming in your applications
You can group similar objects into related data types. These types have a name, an internal representation, and a specific behavior. By using UDTs, you can tell DB2 the name of your new type and how it is represented internally. A LOB is one of the possible internal representations for your new type, and is the most suitable representation for large, complex data structures.

Data integrity through strong typing and encapsulation
Strong typing guarantees that only functions and operations defined on the distinct type can be applied to the type. Encapsulation ensures that the behavior of UDTs is restricted by the functions and operators that can be applied to them.

Performance through integration into the database manager
Because UDTs are represented internally, the same way as built-in data types, they share the same efficient code as built-in data types to implement built-in functions, comparison operators, indexes, and other functions.The exception to this is UDTs that utilize LOBs, which cannot be used with comparison operators and indexes.

User-defined Functions

SQL Reference	Describes SQL syntax, semantics, and the rules of the language. Chapters 3 and 6 contain details about user-defined functions.
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL. Chapters 6 and 7 contain details about user-defined functions.
Administration Guide	Contains information required to design, implement, and maintain a database. Chapter 4 contains details about user-defined functions.
Building Applications for UNIX Environments	Provides instructions to compile, link, and run user-defined functions on a UNIX system.
Building Applications for Windows and OS/2 Environments	Provides instructions to compile, link, and run user-defined functions on a Windows or OS/2 system.

	Embedded SQL	**DB2 CLI**	
`calludf`	X		Uses the library of UDFs created by the sample program `udf`.
`sampudf`	X		Demonstrates the use of UDTs and UDFs.
`udf`	X		Creates a library of UDFs made specifically for the SAMPLE tables, but can be used with tables of compatible column types.
`order.c`		X	UDF library code.

The built-in capabilities supplied through SQL may not satisfy all of your application needs. To allow you to extend those capabilties, DB2 supports *user-defined functions* (UDFs). You can write your own code in C, C++, Java, and Basic to perform operations within any SQL statement that returns a single scalar value or a table.

This gives you significant flexibility. Your applications can return single scalar values such as select lists from databases, or they can return whole tables from non-database sources such as spreadsheets.

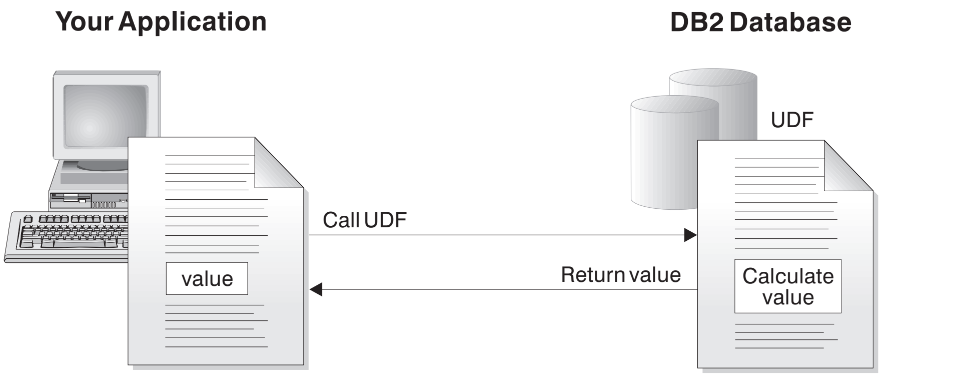


Figure 11. UDFs extend DB2 SQL

UDFs provide a way to standardize your applications. By implementing a common set of functions in UDFs, many applications can use the UDFs to process data in the same way ensuring consistent results.

User-defined functions also support object-oriented programming in your applications. UDFs provide for abstraction, allowing you to define the methods that can be used to perform operations on data objects. And UDFs provide for encapsulation, allowing you to control access to the underlying data of an object, protecting it from direct manipulation and possible corruption.

OLE Automation UDFs

OLE (Object Linking and Embedding) automation is part of the OLE 2.0 architecture from Microsoft Corporation. With OLE automation, applications can expose their properties and methods in OLE automation objects. Other applications can then instantiate those objects, use their properties, and invoke their methods.

DB2 for Windows NT provides access to OLE automation objects using UDFs. To access OLE automation objects and invoke their methods, you must register the methods of the objects as UDFs. DB2 applications can then invoke the methods by calling the UDFs. The UDFs can be scalar or table functions.

For example, you can develop an application that queries data in a spreadsheet created using a product such as Microsoft Excel. To do this, you would develop an OLE automation table function that retrieves data from the worksheet, and returns it to DB2. DB2 can then process the data, perform online analytical processing (OLAP), and return the query result to your application.

Triggers

SQL Reference	Describes SQL syntax, semantics, and the rules of the language. Chapters 2 and 6 contains details about triggers.
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL. Chapter 8 contains details about triggers.
Administration Guide	Contains information required to design, implement, and maintain a database. Chapter 4 contains details about triggers.

	Embedded SQL	**DB2 CLI**	
`trigsql`	X		Demonstrates triggers and constraints.

A *trigger* defines a set of actions that are executed at, or triggered by, a delete, insert or update operation on a specified table. When such an SQL operation is executed, the trigger is said to be activated. The trigger can be activated before the SQL operation or after it. You define a trigger using the SQL statement CREATE TRIGGER.

You can use triggers that run before an update or insert in several ways:

- To check or modify values before they are actually updated or inserted in the database. This is useful if you need to transform data from the way the user sees it to some internal database format.
- To run other non-database operations coded in user-defined functions.

Similarly, you can use triggers that run after an update or insert in several ways:

- To update data in other tables. This is useful for maintaining relationships between data or in keeping audit trail information.
- To check against other data in the table or in other tables. This is useful to ensure data integrity when referential integrity constraints aren't appropriate, or when table check constraints limit checking to the current table only.
- To run non-database operations coded in user-defined functions. This is useful when issuing alerts or to update information outside the database.

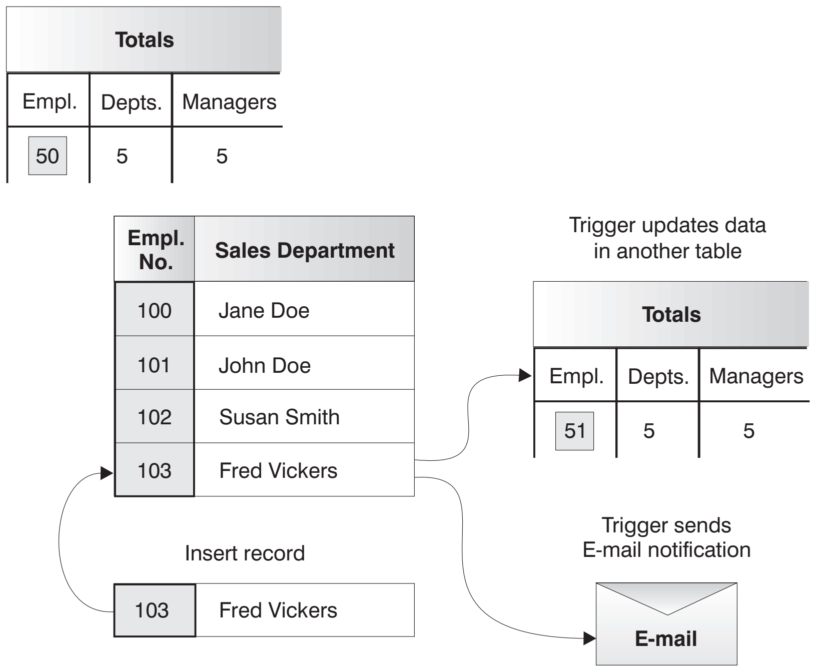


Figure 12. Inserts can activate a trigger

You gain several benefits using triggers:

Faster application development
Triggers are stored in the database, and are available to all applications. This relieves you of the need to code equivalent functions for each application.

Global enforcement of business rules
Triggers are defined once, and are used by all applications that use the data governed by the triggers.

Easier maintenance
Any changes need to be made only once in the database instead of in every application that uses a trigger.

Stored Procedures

Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL. Chapter 5 contains details about stored procedures.
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface. Chapter 3 contains details about stored procedures.
Building Applications for UNIX Environments	Provides instructions to compile, link, and run stored procedures on a UNIX system.
Building Applications for Windows and OS/2 Environments	Provides instructions to compile, link, and run stored procedures on a Windows or OS/2 system.

	Embedded SQL	**DB2 CLI**	
`inpcli`	X		Demonstrates stored procedures using either the SQLDA structure or host variables. This is the client program of a client/server example. The server program is `inpsrv`. `inpcli` fills the SQLDA with information and passes it to the server program for further processing.
`inpsrv`	X		Creates a table in the SAMPLE database with the information received in the SQLDA. Returns the SQLCA status to the client program `inpcli`.
`outcli`	X		Demonstrates stored procedures using the SQLDA. This is the client program of a client/server example. The server program is `outsrv`. `outcli` allocates and initialized a one variable SQLDA, and passes it to the server program for further processing.
`outsrv`	X		Finds the median salary, fills the SQLDA with the median, and returns the SQLDA and SQLCA status to the client program `outcli`.

	Embedded SQL	DB2 CLI	
`inpcli.c`		X	Calls embedded SQL stored procedure `inpsrv`.
`inpcli2.c`		X	Calls DB2 CLI stored procedure `inpsrv2.c`.
`inpsrv2.c`		X	Stored procedure (rewrite of embedded SQL sample `inpsrv`).

Typically, applications access the database across the network. This can result in a lot of data being transmitted across the network, and poor performance.

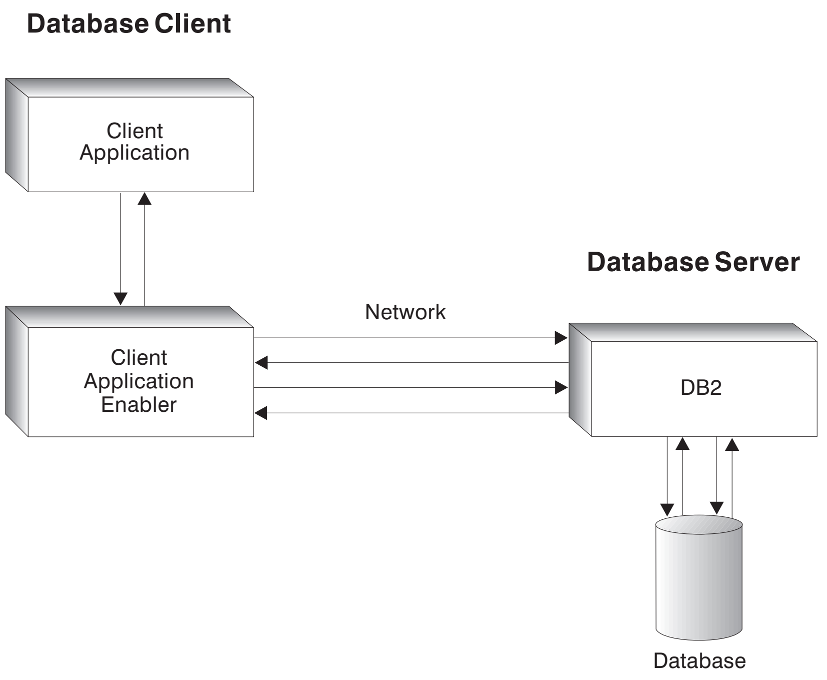


Figure 13. Application accessing a database across the network

A *stored procedure* is a part of your application that runs on the database server. Your client application passes control to the stored procedure allowing it to perform intermediate processing on the server without transmitting unnecessary data across the network. Only the records your application needs are transmitted by the stored procedure.

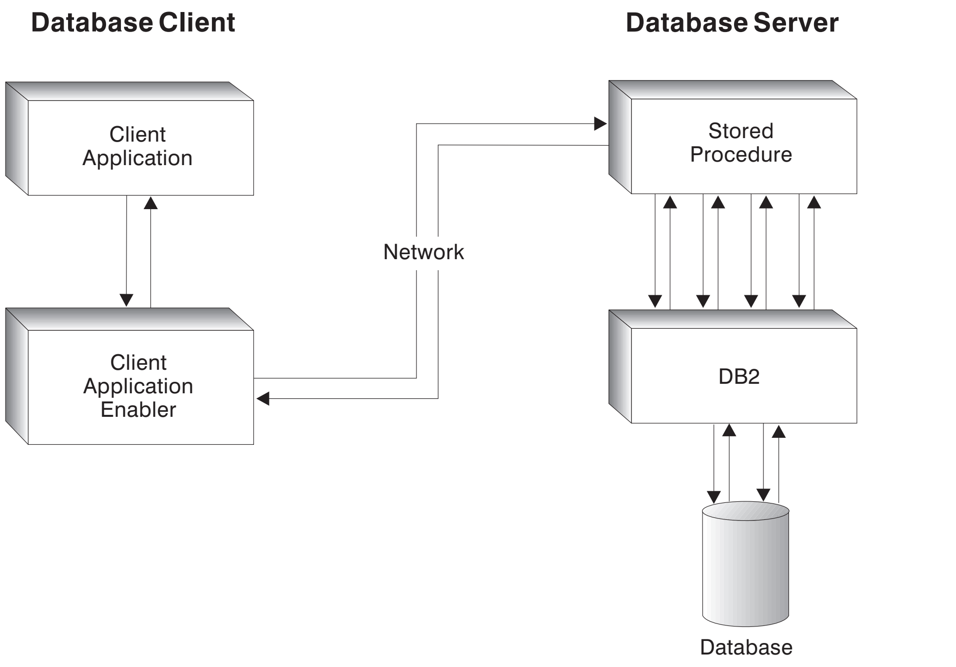


Figure 14. Application using a stored procedure

You gain several benefits using stored procedures:

Reduced network traffic
: Grouping SQL statements together can save on network traffic. A typical application requires two trips across the network for each SQL statement. Grouping SQL statements results in two trips across the network for each group of statements, resulting in better performance for applicatons.

Access to features that exist only on the server
: Stored procedures can have access to commands that run only on the server, such as LIST DATABASE DIRECTORY and LIST NODE DIRECTORY; they might have the advantages of increased memory and disk space on server machines; and they can access any additional software installed on the server.

Enforcement of business rules
: You can use stored procedures to define business rules that are common to several applications. This is another way to define business rules, in addition to using constraints and triggers.

When an application calls the stored procedure, it will process data in a consistent way according to the rules defined in the stored procedure. If you need to change the rules, you only need to make the change once in the stored procedure, not in every application that calls the stored procedure.

Chapter 4. Using Application Development Tools

You can use a variety of different tools when developing your applications. DB2 Universal Database supplies tools to help you write and test the SQL statements in your applications, and to help you monitor their performance. There are also companion products that you can use to extend your applications beyond traditional data types, and to code your applications in a variety of programming languages.

DB2 Universal Database Tools

DB2 supplies several tools that you can use when developing your applications. The tools are summarized below. “DB2 Universal Database Tools” on page 49 provides more details, and lists additional tools.

Command Center
: Provides an interactive window that you can use to execute SQL statements and DB2 commands.

Command Line Processor
: Use from the system command prompt to submit SQL statements and DB2 commands for execution.

Control Center
: A graphical interface that you can use to list applications connected to a database, and to force one or all of the applications off the database.

Performance Monitor
: A graphical interface you can use for performance tuning and to analyze query access plans.

Visual Explain
: A graphical tool for analyzing and tuning SQL statements. Visual Explain helps you understand the structure and potential execution performance of SQL statements.

SmartGuides

DB2 SmartGuides help you complete administration tasks by taking you through each task one step at a time. SmartGuides that might be of particular interest to application developers are summarized below. “SmartGuides” on page 51 provides a complete list.

Create Database
Helps you create a database, and perform some basic configuration tasks.

Create Table Space
Helps you create a new table space.

Create Table
Helps you select basic data types, and create a primary key for the table.

Companion Products

http://www.software.ibm.com/ad/pli/	Provides information about the PL/I family.
http://www.software.ibm.com/ software/ad/visgen/	Provides information about VisualAge Generator.

In addition to the products described in “About the Application Developer's Kit” on page 2, there are other products you can use when developing DB2 applications:

PL/I Family
Products that provide tools to enable a visual programming environment to build new applications and to support existing applications. PL/I products contain a built-in precompiler that works with the DB2 Software Developer's Kit.

VisualAge Generator
Formerly named VisualGen, this product is a powerful high-end application development solution for building mission critical multi-tier client/server systems.

From a single desktop environment you can rapidly develop and test both client and server logic. Once your system is completely tested, you invoke a sophisticated generation engine which partitions your logic, transforms it into an optimized 3GL (COBOL or C++), and deploys it to your target platforms. As your platforms, databases or transactional systems change, you can retarget and generate without modifying your source.

VisualAge Generator includes support for DB2 distributed unit of work, which enables you to connect multiple DB2 databases in a single transaction.

Appendix A. About DB2 Universal Database

DB2 is a relational database management system that is web-enabled with Java support; scalable from single processors to symmetric multi-processors; and multimedia capable with image, audio, video, and text support. The DB2 product consists of three products and components: *DB2 Workgroup Edition*, *DB2 Enterprise Edition*, and *DB2 Client Application Enabler*. These products have the following features.

DB2 Workgroup Edition

DB2 Workgroup Edition includes:

- *DB2 Universal Database server*, which enables local and remote clients and applications to create, update, control, and manage relational databases using Structured Query Language (SQL), ODBC, or CLI.
- *DB2 Client Pack* CD-ROM, which contains all the latest DB2 Client Application Enablers. With DB2 Client Application Enabler, clients from a variety of platforms can connect to any DB2 server, including DataJoiner.
- *DB2 Net.Data* CD-ROM, which contains all supported DB2 Net.Data (formerly known as DB2 World Wide Web Connection) products. DB2 Net.Data enables application developers to create Internet applications that access data from DB2 databases.

DB2 Workgroup Edition is licensed on a per user basis.

DB2 Enterprise Edition

DB2 Enterprise Edition includes all the functions provided in DB2 Workgroup Edition, plus support for host connectivity providing users with access to DB2 databases residing on host systems such as MVS/ESA, OS/390, OS/400, VM, and VSE.

DB2 Enterprise Edition supports unlimited user licensing.

DB2 Client Application Enabler

DB2 Client Application Enabler enables a client workstation to access the DB2 server.

Other DB2 Products

There are a variety of other DB2 products that you can order separately:

DB2 Application Developer's Kit
The DB2 Application Developer's Kit contains a collection of DB2 Universal Database products, clients, DB2 Connect products, Software Developer's Kits, and application development tools for all supported platforms. The AD Kit gives you all the tools you need to create multimedia database applications that can run on a variety of platforms and can connect to any DB2 Server, including DataJoiner.

DB2 Universal Database Personal Edition
DB2 Universal Database Personal Edition enables you to create and use local databases and to access remote databases if they are available. This product is available only for the OS/2, Windows 95, and the Windows NT operating system.

DB2 Connect Enterprise Edition
DB2 Connect Enterprise Edition (formerly known as DDCS Multi-User Gateway) provides access from clients on the network to DB2 databases residing on host systems such as MVS/ESA, OS/390, OS/400, VM, and VSE.

DB2 Connect Personal Edition
DB2 Connect Personal Edition (formerly known as DDCS Single-User) provides access from a single workstation to DB2 databases residing on host systems such as MVS/ESA, OS/390, OS/400, VM, and VSE. This product is available only for the OS/2, Windows 95, and the Windows NT operating system.

DB2 Universal Database Extended Edition
DB2 Universal Database Extended Edition (formerly known as DB2 Parallel Edition) provides the ability for a database to be partitioned across multiple independent computers of a common platform, each with its own processor(s), memory, and DASD. To the end-user and application developer, the database still appears as a single database on a single computer. This enables an application to use a database that is simply too large for a single computer to handle efficiently. SQL operations can operate in parallel on the individual database partitions, thereby speeding up the execution time of a single query.

DB2 Universal Database Tools

DB2 Universal Database includes the following tools to perform server administration tasks.

Note: Not all tools are available on every platform.

Control Center

A graphical interface that displays database objects (such as databases, tables, and packages) and their relationship to each other. Use the Control Center to perform administrative tasks such as configuring the system, managing directories, backing up and recovering the system, scheduling jobs, and managing media.

The Control Center includes the following facilities:

- *Command Center:* to enter DB2 commands and SQL statements in an interactive window, and to see the execution result in a result window. You can scroll through the results and save the output to a file.
- *Script Center:* to create mini applications called scripts, which you can store and invoke at a later time. These scripts can contain DB2 commands, SQL statements, or operating system commands. You can schedule scripts to run unattended. You can run these jobs once or you can set them up to run on a repeating schedule. A repeating schedule is particularly useful for tasks like backups.
- *Journal:* to view the following types of information: all available information about jobs that are pending execution, executing, or that have completed execution; the recovery history log; the alerts log; and the messages log. You can also use the Journal to review the results of jobs that run unattended.
- *Alert Center:* to monitor your system for early warnings of potential problems, or to automate actions to correct problems.
- *Tools Setting:* to change the settings for the Control Center, Alert Center, and Replication.

Performance Monitor

An installable option for the Control Center, the Performance Monitor is a graphical interface that provides comprehensive performance data collection, viewing, reporting, analysis, and alerting capabilities for your DB2 system. Use the Performance Monitor for performance tuning.

You can choose to monitor snapshots or events. The Snapshot Monitor enables you to capture point-in-time information at specified intervals. The Event Monitor allows you to record performance information over the duration of an event, such as a connection.

Visual Explain

An installable option for the Control Center, Visual Explain is a graphical interface that enables you to analyze and tune SQL statements, including viewing access plans chosen by the optimizer for SQL statements.

Appendix B. How the DB2 Library Is Structured

The DB2 Universal Database library consists of SmartGuides, online help, and books. This section describes the information that is provided, and how to access it.

To help you access product information online, DB2 provides the Information Center on OS/2, Windows 95, and the Windows NT operating systems. You can view task information, DB2 books, troubleshooting information, sample programs, and DB2 information on the Web. “About the Information Center” on page 58 has more details.

SmartGuides

SmartGuides help you complete some administration tasks by taking you through each task one step at a time. SmartGuides are available on OS/2, Windows 95, and the Windows NT operating systems. The following table lists the SmartGuides.

SmartGuide	Helps you to...	How to Access...
Add Database	Catalog a database on a client workstation.	From the Client Configuration Assistant, click on **Add**.
Create Database	Create a database, and to perform some basic configuration tasks.	From the Control Center, click with the right mouse button on the **Databases** icon and select **Create**->**New**.
Performance Configuration	Tune the performance of a database by updating configuration parameters to match your business requirements.	From the Control Center, click with the right mouse button on the database you want to tune and select **Configure performance**.
Backup Database	Determine, create, and schedule a backup plan.	From the Control Center, click with the right mouse button on the database you want to backup and select **Backup**->**Database using SmartGuide**.
Restore Database	Recover a database after a failure. It helps you understand which backup to use, and which logs to replay.	From the Control Center, click with the right mouse button on the database you want to restore and select **Restore**->**Database using SmartGuide**.

SmartGuide	Helps you to...	How to Access...
Create Table	Select basic data types, and create a primary key for the table.	From the Control Center, click with the right mouse button on the **Tables** icon and select **Create**->**Table using SmartGuide**.
Create Table Space	Create a new table space.	From the Control Center, click with the right mouse button on the **Table spaces** icon and select **Create**->**Table space using SmartGuide**.

Online Help

Online help is available with all DB2 components. The following table describes the various types of help.

Type of Help	Contents	How to Access...
Command Help	Explains the syntax of commands in the command line processor.	From the command line processor in interactive mode, enter: **?** *command* where *command* is a keyword or the entire command. For example, **?** *catalog* displays help for all the CATALOG commands, whereas **?** *catalog database* displays help for the CATALOG DATABASE command.
Control Center Help	Explains the tasks you can perform in a window or notebook. The help includes prerequisite information you need to know, and describes how to use the window or notebook controls.	From a window or notebook, click on the **Help** push button or press the F1 key.

Type of Help	Contents	How to Access...
Message Help	Describes the cause of a message number, and any action you should take.	From the command line processor in interactive mode, enter: **?** *message number* where *message number* is a valid message number. For example, **?** *SQL30081* displays help about the SQL30081 message. To view message help one screen at a time, enter: **?** *XXXnnnnn* **| more** where *XXX* is the message prefix, such as SQL, and *nnnnn* is the message number, such as 30081. To save message help in a file, enter: **?** *XXXnnnnn* > *filename.ext* where *filename.ext* is the file where you want to save the message help. **Note:** On UNIX-based systems, enter: **\?** *XXXnnnnn* **| more** or **\?** *XXXnnnnn* > *filename.ext*
SQL Help	Explains the syntax of SQL statements.	From the command line processor in interactive mode, enter: **help** *statement* where *statement* is an SQL statement. For example, **help** *SELECT* displays help about the SELECT statement.
SQLSTATE Help	Explains SQL states and class codes.	From the command line processor in interactive mode, enter: **?** *sqlstate* or **?** *class-code* where *sqlstate* is a valid five digit SQL state and *class-code* is a valid two digit class code. For example, **?** *08003* displays help for the 08003 SQL state, whereas **?** *08* displays help for the 08 class code.

DB2 Books

The table in this section lists the DB2 books. They are divided into two groups:

- Cross-platform books: These books are for DB2 on any of the supported platforms.
- Platform-specific books: These books are for DB2 on a specific platform. For example, there is a separate *Quick Beginnings* book for DB2 on OS/2, Windows NT, and UNIX-based operating systems.

Most books are available in HTML and PostScript format, and in hardcopy that you can order from IBM. The exceptions are noted in the table.

You can obtain DB2 books and access information in a variety of different ways:

View To view an HTML book, you can do the following:

- If you are running DB2 administration tools on OS/2, Windows 95, or the Windows NT operating systems, you can use the Information Center. "About the Information Center" on page 58 has more details.
- Use the open file function of the Web browser supplied by DB2 (or one of your own) to open the following page:

  ```
  sqllib/doc/html/index.htm
  ```

  The page contains descriptions of and links to the DB2 books. The path is located on the drive where DB2 is installed.

  You can also open the page by double-clicking on the **DB2 Online Books** icon. Depending on the system you are using, the icon is in the main product folder or the Windows Start menu.

Search To search for information in the HTML books, you can do the following:

- Click on **Search the DB2 Books** at the bottom of any page in the HTML books. Use the search form to find a specific topic.
- Click on **Index** at the bottom of any page in an HTML book. Use the Index to find a specific topic in the book.
- Display the Table of Contents or Index of the HTML book, and then use the find function of the Web browser to find a specific topic in the book.
- Use the bookmark function of the Web browser to quickly return to a specific topic.
- Use the search function of the Information Center to find specific topics. "About the Information Center" on page 58 has more details.

Print To print a book on a PostScript printer, look for the file name shown in the table.

Order To order a hardcopy book from IBM, use the form number.

Book Name	Book Description	Form Number File Name
	Cross-Platform Books	
Administration Getting Started	Introduces basic DB2 database administration concepts and tasks, and walks you through the primary administrative tasks.	S10J-8154 db2k0x50
Administration Guide	Contains information required to design, implement, and maintain a database to be accessed either locally or in a client/server environment.	S10J-8157 db2d0x50
API Reference	Describes the DB2 application programming interfaces (APIs) and data structures you can use to manage your databases. Explains how to call APIs from your applications.	S10J-8167 db2b0x50
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification.	S10J-8159 db2l0x50
Command Reference	Explains how to use the command line processor, and describes the DB2 commands you can use to manage your database.	S10J-8166 db2n0x50
DB2 Connect Enterprise Edition Quick Beginnings	Provides planning, installing, configuring, and using information for DB2 Connect Enterprise Edition. Also contains installation and setup information for all supported clients.	S10J-7888 db2cyx50
DB2 Connect Personal Edition Quick Beginnings	Provides planning, installing, configuring, and using information for DB2 Connect Personal Edition.	S10J-8162 db2c1x50
DB2 Connect User's Guide	Provides concepts, programming and general using information about the DB2 Connect products.	S10J-8163 db2c0x50
DB2 Connectivity Supplement	Provides setup and reference information for customers who want to use DB2 for AS/400, DB2 for OS/390, DB2 for MVS, or DB2 for VM as DRDA Application Requesters with DB2 Universal Database servers, and customers who want to use DRDA Application Servers with DB2 Connect (formerly DDCS) application requesters. **Note:** Available in HTML and PostScript formats only.	No form number db2h1x50
Embedded SQL Programming Guide	Explains how to develop applications that access DB2 databases using embedded SQL, and includes discussions about programming techniques and performance considerations.	S10J-8158 db2a0x50
Glossary	Provides a comprehensive list of all DB2 terms and definitions. **Note:** Available in HTML format only.	No form number db2t0x50

Book Name	Book Description	Form Number File Name
Installing and Configuring DB2 Clients	Provides installation and setup information for all DB2 Client Application Enablers and DB2 Software Developer's Kits. **Note:** Available in HTML and PostScript formats only.	No form number db2iyx50
Master Index	Contains a cross reference to the major topics covered in the DB2 library. **Note:** Available in PostScript format and hardcopy only.	S10J-8170 db2w0x50
Message Reference	Lists messages and codes issued by DB2, and describes the actions you should take.	S10J-8168 db2m0x50
Replication Guide and Reference	Provides planning, configuring, administering, and using information for the IBM Replication tools supplied with DB2.	S95H-0999 db2e0x50
Road Map to DB2 Programming	Introduces the different ways your applications can access DB2, describes key DB2 features you can use in your applications, and points to detailed sources of information for DB2 programming.	S10J-8155 db2u0x50
SQL Getting Started	Introduces SQL concepts, and provides examples for many constructs and tasks.	S10J-8156 db2y0x50
SQL Reference	Describes SQL syntax, semantics, and the rules of the language. Also includes information about release-to-release incompatibilities, product limits, and catalog views.	S10J-8165 db2s0x50
System Monitor Guide and Reference	Describes how to collect different kinds of information about your database and the database manager. Explains how you can use the information to understand database activity, improve performance, and determine the cause of problems.	S10J-8164 db2f0x50
Troubleshooting Guide	Helps you determine the source of errors, recover from problems, and use diagnostic tools in consultation with DB2 Customer Service.	S10J-8169 db2p0x50
What's New	Describes the new features, functions, and enhancements in DB2 Universal Database. **Note:** Available in HTML and PostScript formats only.	No form number db2q0x50
	Platform-Specific Books	
Building Applications for UNIX Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a UNIX system.	S10J-8161 db2axx50
Building Applications for Windows and OS/2 Environments	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Windows or OS/2 system.	S10J-8160 db2a1x50

Book Name	Book Description	Form Number File Name
DB2 Extended Enterprise Edition Quick Beginnings	Provides planning, installing, configuring, and using information for DB2 Universal Database Extended Enterprise Edition for AIX.	S72H-9620 db2v3x50
DB2 Personal Edition Quick Beginnings	Provides planning, installing, configuring, and using information for DB2 Universal Database Personal Edition on OS/2, Windows 95, and the Windows NT operating systems.	S10J-8150 db2i1x50
DB2 SDK for Macintosh Building Your Applications	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Macintosh system. **Note:** Available in PostScript format and hardcopy for DB2 Version 2.1.2 only.	S50H-0528 sqla7x02
DB2 SDK for SCO OpenServer Building Your Applications	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a SCO OpenServer system. **Note:** Available for DB2 Version 2.1.2 only.	S89H-3242 sqla9x02
DB2 SDK for Silicon Graphics IRIX Building Your Applications	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a Silicon Graphics system. **Note:** Available in PostScript format and hardcopy for DB2 Version 2.1.2 only.	S89H-4032 sqlaax02
DB2 SDK for SINIX Building Your Applications	Provides environment setup information and step-by-step instructions to compile, link, and run DB2 applications on a SINIX system. **Note:** Available in PostScript format and hardcopy for DB2 Version 2.1.2 only.	S50H-0530 sqla8x00
Quick Beginnings for OS/2	Provides planning, installing, configuring, and using information for DB2 Universal Database on OS/2. Also contains installing and setup information for all supported clients.	S10J-8147 db2i2x50
Quick Beginnings for UNIX	Provides planning, installing, configuring, and using information for DB2 Universal Database on UNIX-based platforms. Also contains installing and setup information for all supported clients.	S10J-8148 db2ixx50
Quick Beginnings for Windows NT	Provides planning, installing, configuring, and using information for DB2 Universal Database on the Windows NT operating system. Also contains installing and setup information for all supported clients.	S10J-8149 db2i6x50

Notes:

1. The character in the sixth position of the file name indicates the language of a book. For example, the file name `db2d0e50` indicates that the *Administration Guide* is in English. The following letters are used in the file names to indicate the language of a book:

Language	Identifier	Language	Identifier
Brazilian Portuguese	B	Hungarian	H
Bulgarian	U	Italian	I
Czech	X	Norwegian	N
Danish	D	Polish	P
English	E	Russian	R
Finnish	Y	Slovenian	L
French	F	Spanish	Z
German	G	Swedish	S

2. For late breaking information that could not be included in the DB2 books, see the README file. Each DB2 product includes a README file which you can find in the directory where the product is installed.

About the Information Center

The Information Center provides quick access to DB2 product information. The Information Center is available on OS/2, Windows 95, and the Windows NT operating systems. You must install the DB2 administration tools to see the Information Center.

Depending on your system, you can access the Information Center from the:

- Main product folder
- Toolbar in the Control Center
- Windows Start menu.

The Information Center provides the following kinds of information. Click on the appropriate tab to look at the information:

Tasks	Lists tasks you can perform using DB2.
Reference	Lists DB2 reference information, such as keywords, commands, and APIs.
Books	Lists DB2 books.
Troubleshooting	Lists categories of error messages and their recovery actions.
Sample Programs	Lists sample programs that come with the DB2 Software Developer's Kit. If the Software Developer's Kit is not installed, this tab is not displayed.

Web	Lists DB2 information on the World Wide Web. To access this information, you must have a connection to the Web from your system.

When you select an item in one of the lists, the Information Center launches a viewer to display the information. The viewer might be the system help viewer, an editor, or a Web browser, depending on the kind of information you select.

The Information Center provides search capabilities so you can look for specific topics, and filter capabilities to limit the scope of your searches.

Appendix C. Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the

IBM Director of Licensing,
IBM Corporation,
500 Columbus Avenue,
Thornwood, NY, 10594
USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Department 071
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

This publication may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries:

ACF/VTAM
ADSTAR
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
CICS
C Set++
C/370
DATABASE 2
DatagLANce
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
Distributed Relational Database Architecture
DRDA
Extended Services
FFST
First Failure Support Technology
IBM
IMS
Lan Distance
MVS/ESA
MVS/XA
NetView
OS/400
OS/390
OS/2
PowerPC
QMF
RACF
RISC System/6000
SAA
SP
SQL/DS
SQL/400
S/370
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WIN-OS/2

Trademarks of Other Companies

The following terms are trademarks or registered trademarks of the companies listed:

C-bus is a trademark of Corollary, Inc.

HP-UX is a trademark of Hewlett-Packard.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Solaris is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, or service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

Index

Contacting IBM

This section lists ways you can get more information from IBM.

If you have a technical problem, please take the time to review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. Depending on the nature of your problem or concern, this guide will suggest information you can gather to help us to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

Telephone

If you live in the U.S.A., call one of the following numbers:

- 1-800-237-5511 to learn about available service options.
- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, see Appendix A of the IBM Software Support Handbook. You can access this document by selecting the "Roadmap to IBM Support" item at: http://www.ibm.com/support/.

Note that in some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

World Wide Web
http://www.software.ibm.com/data/
http://www.software.ibm.com/data/db2/library/

The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more. The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information. (Note that this information may be in English only.)

Anonymous FTP Sites
ftp.software.ibm.com

Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools concerning DB2 and many related products.

Internet Newsgroups
comp.databases.ibm-db2, bit.listserv.db2-l

These newsgroups are available for users to discuss their experiences with DB2 products.

CompuServe
GO IBMDB2 to access the IBM DB2 Family forums

All DB2 products are supported through these forums.

To find out about the IBM Professional Certification Program for DB2 Universal Database, go to http://www.software.ibm.com/data/db2/db2tech/db2cert.html

Part Number: 10J8155

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

10J8155

S10J-8155-00