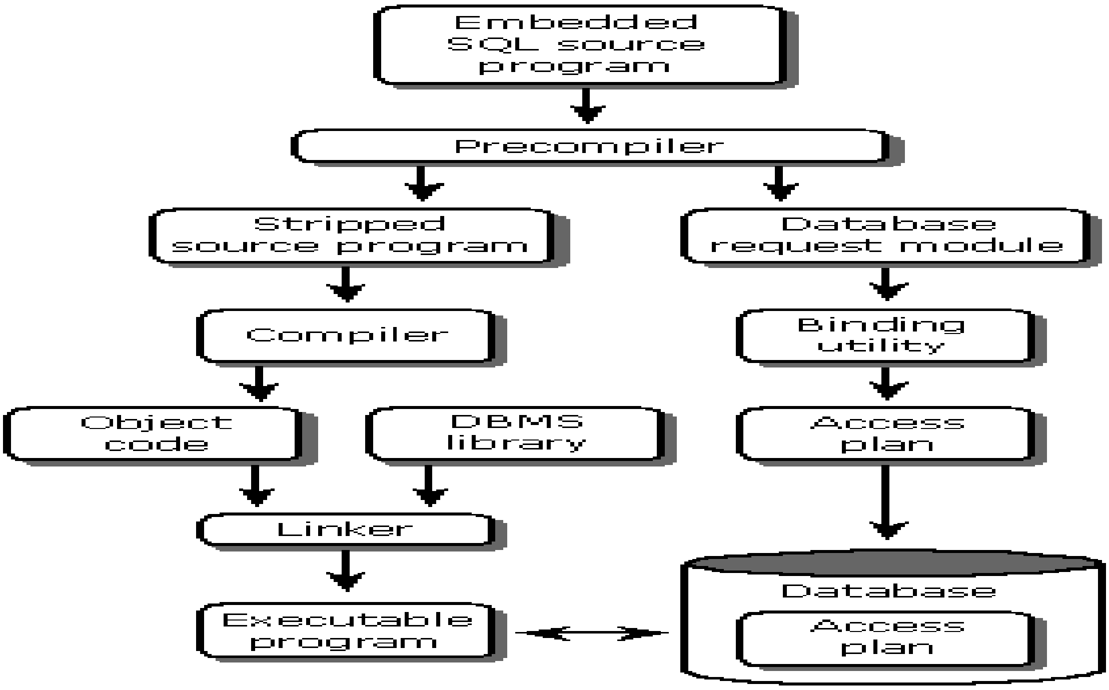
Embedded SQL

Embedded SQL is **a method of combining the computing power of a programming language and the database manipulation capabilities of SQL**. Embedded SQL statements are SQL statements written inline with the program source code, of the host language.



Embedded SQL applications connect to databases and execute embedded SQL statements. The embedded SQL statements are contained in a package that must be bound to the target database server.

You can develop embedded SQL applications for the Db2® database in the following host programming languages: C, C++, and COBOL.

Static and dynamic SQL

SQL statements can be executed in one of two ways: statically or dynamically.

Statically executed SQL statements

For statically executed SQL statements, the syntax is fully known at precompile time. The structure of an SQL statement must be completely specified for a statement to be considered static. For example, the names for the columns and tables referenced in a statement must be fully known at precompile time. The only information that can be specified at run time are values for any host variables referenced by the statement. However, host variable information, such as data types, must still be precompiled. You precompile, bind, and compile statically executed SQL statements before you run your application. Static SQL is best used on databases whose statistics do not change a great deal.

Dynamically executed SQL statements

Dynamically executed SQL statements are built and executed by an application 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 a situation suited for dynamic SQL.

- **Embedding SQL statements in a host language**

  Structured Query Language (SQL) is a standardized language that you can use to manipulate database objects and the data that they contain. Despite differences between host languages, embedded SQL applications are made up of three main elements that are required to setup and issue an SQL statement.
- **Supported development software for embedded SQL applications**

  Before you begin writing embedded SQL applications, you must determine if your development software is supported. The operating system that you are developing for determines which compilers, interpreters, and development software you must use.
- **Setting up the embedded SQL development environment**

  Before you can start building embedded SQL applications, install the supported compiler for the host language you will be using to develop your applications and set up the embedded SQL environment.
- **Designing embedded SQL applications**

  When designing embedded SQL applications you must use static or dynamic executed SQL statements.
- **Programming embedded SQL applications**

  Programming embedded SQL applications involves the same steps required to assemble an application in your host programming language.
- **Building embedded SQL applications**

  After you have created the source code for your embedded SQL application, you must follow additional steps to build the application. You should consider building 64-bit executable files when developing new embedded SQL database applications. Along with compiling and linking your program, you must precompile and bind it.

- **Deploying and running embedded SQL applications**

  Embedded SQL applications are portable and can be placed in remote database components. You can compile the application in one location and run the package on a different component.
- **Compatibility features for migration**

  The Db2 database manager provides features that facilitate the migration of embedded SQL C applications from other database systems.

How to Embed SQL in High-Level Languages?

For using embedded SQL, we need some tools in each high-level language. In some cases, we have inbuilt libraries which provide us with the basic building block.

While in some cases we need to import or use some packages to perform the desired tasks.

For example, in Java, we need a connection class. We first create a connection by using the connection class and further we open the connection bypassing the required parameters to connect with the database.

Example: How to connect to a database (using JAVA).
Code[with embedded SQL]:

```
Class.forName("com.mysql.jdbc.Driver");
Connection connection = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/DataFlair","user","root");
Statement statement = connection.createStatement();
```

Here, DataFlair is the name of the database and user is the username and root is the password.

Advantages of Embedded SQL

Some of the advantages of using SQL embedded in high-level languages are as follows:

- Helps to access databases from anywhere.
- Allows integrating authentication service for large scale applications.
- Provides extra security to database transactions.
- Avoids logical errors while performing transactions on our database.
- Makes it easy to integrate the frontend and the backend of our application.