Embedding SQL in an Application Program

Objectives

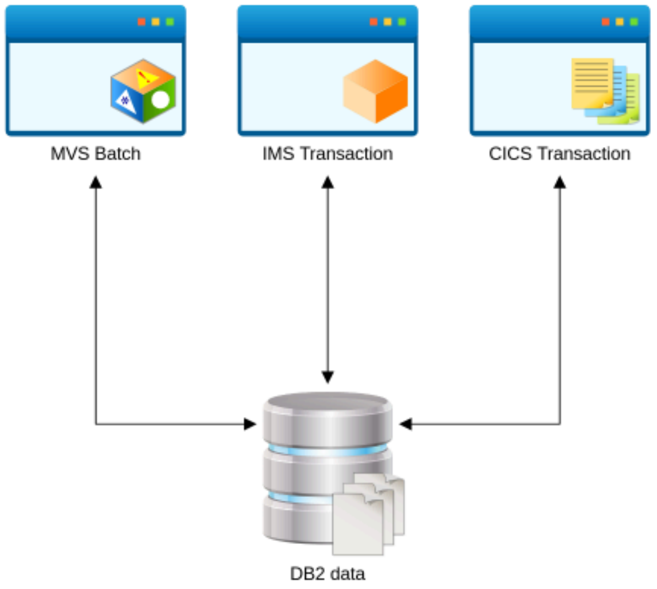
Embedding SQL in an Application Program

Db2 SQL statements can be included in various applications programs, allowing them to access and update Db2 data. Extra steps are required in the compilation process to enable this.

This module will examine those extra steps, namely the precompile and bind.

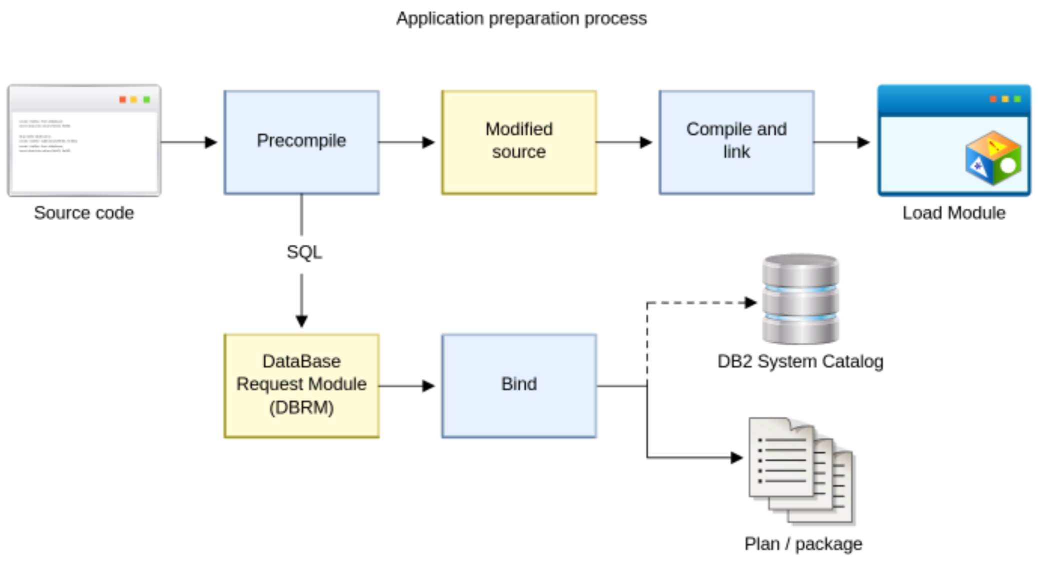
After completing this module you will be able to:

- Describe the Steps Required to Prepare an Application Program With Embedded SQL for Execution



In most installations it is usual to have application programs that access Db2 data. These programs could run in batch or be online transactions running under CICS or IMS.

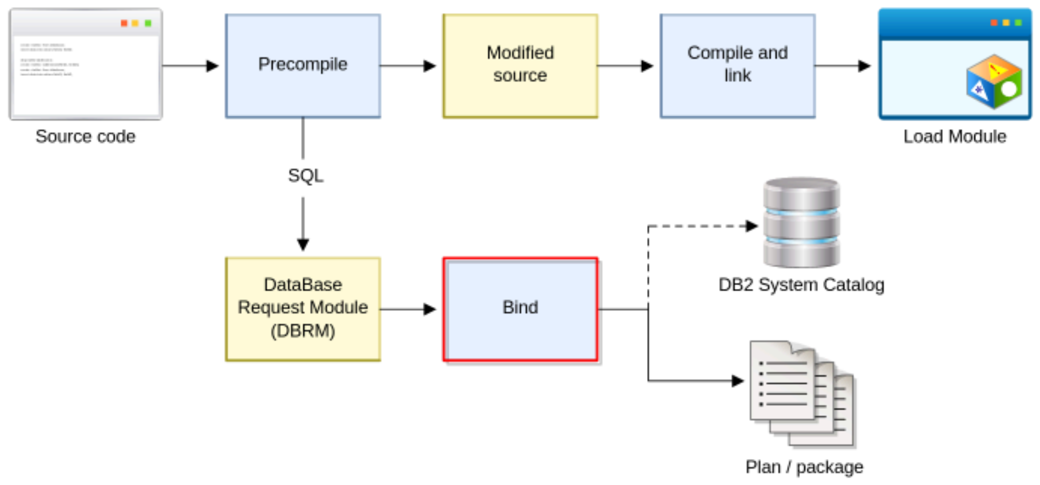
Provided a few simple rules are followed, the process of embedding SQL in an application program is straightforward.



The development cycle of an application program that contains SQL will have a few extra steps.

It is necessary to convert the SQL into something the compiler understands, and process it so Db2 understands how to retrieve the data from the tables.

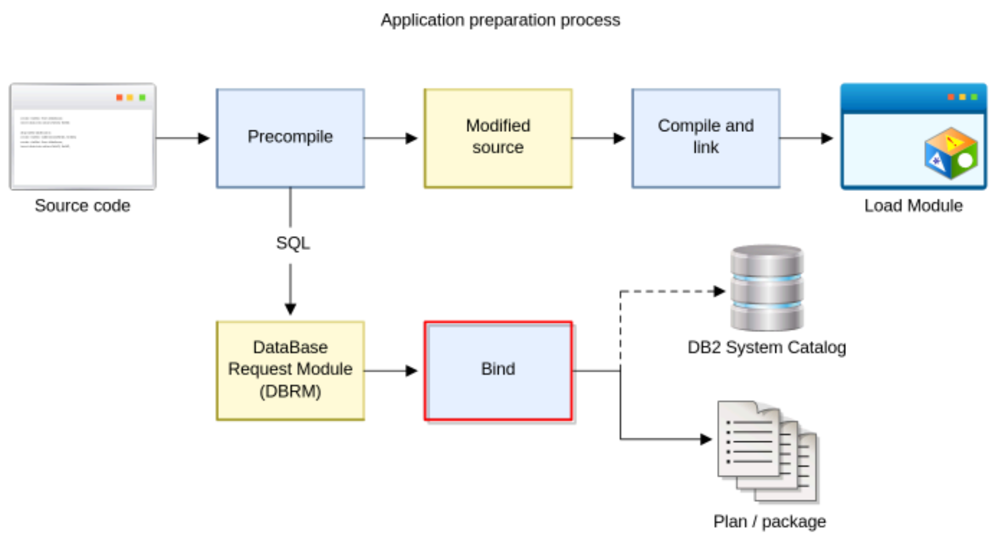
Application preparation process



For an application program, the bind is a separate step that must be performed before the program is ready for execution.

The result of a bind is a package. The package contains all the necessary information to satisfy the request for data by the application program.

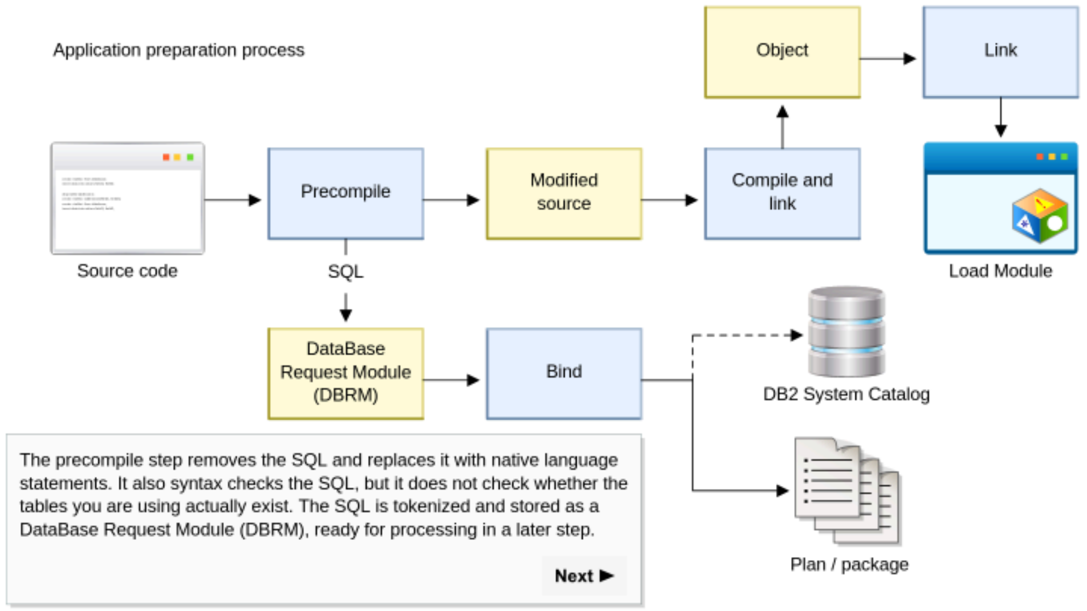
The choice between binding to a plan or a package will be explained later.



There are two significant advantages of this approach. The first is the SQL can be tested before it goes into an application program.

After the SQL is proven to be correct, it can be copied into the program and the necessary changes can be made.

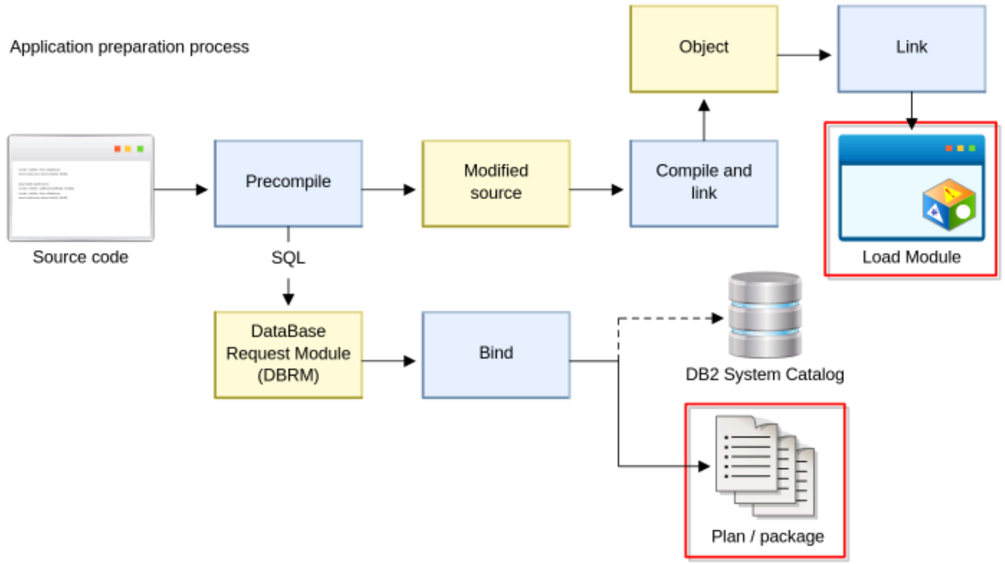
Another advantage is the SQL is compiled during the bind process. This is also referred to as static SQL and it gives Db2 a performance advantage over interpreted or dynamic SQL systems.



The precompile step removes the SQL and replaces it with native language statements. It also syntax checks the SQL, but it does not check whether the tables you are using actually exist. The SQL is tokenized and stored as a DataBase Request Module (DBRM), ready for processing in a later step.

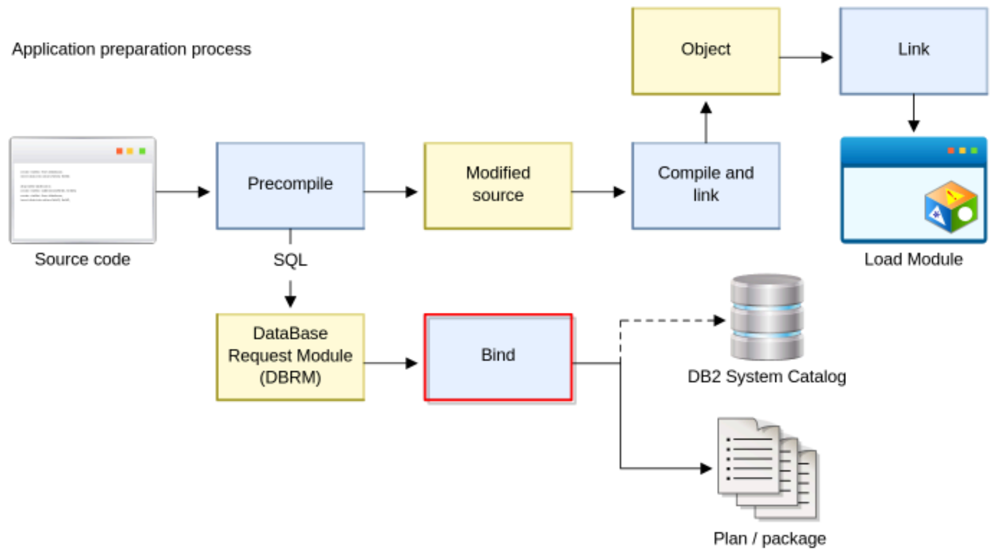
When the program is ready for testing, it is processed in the steps shown here.

Click **Next** in the graphic above to see these steps.



Execution of a Db2 program requires that both the load module and the plan are available and that the load module matches the package being executed. To do this the precompiler adds a token (CONTOKEN) in the load module and the DBRM, and ensures that these match at execution time. When the program requests data from a table, Db2 executes the plan and returns the data to the program.

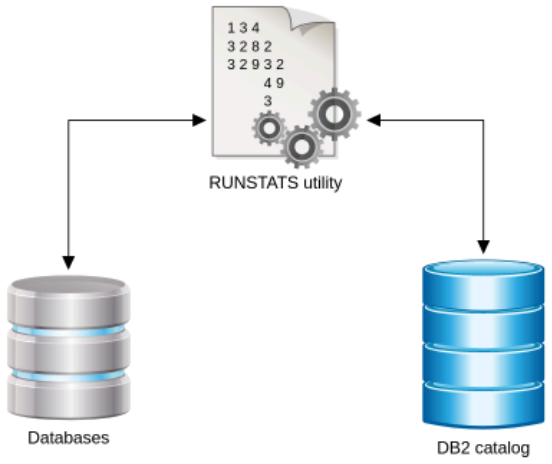
When the data has been passed to the program, it is processed like any other data.



Binding is a very important process because it determines how the data will be accessed. The bind process uses information in the Db2 Catalog at the time of the bind, to determine the optimum access path. Db2 then saves time by not referencing this data at run time and relying on the information in the plan.

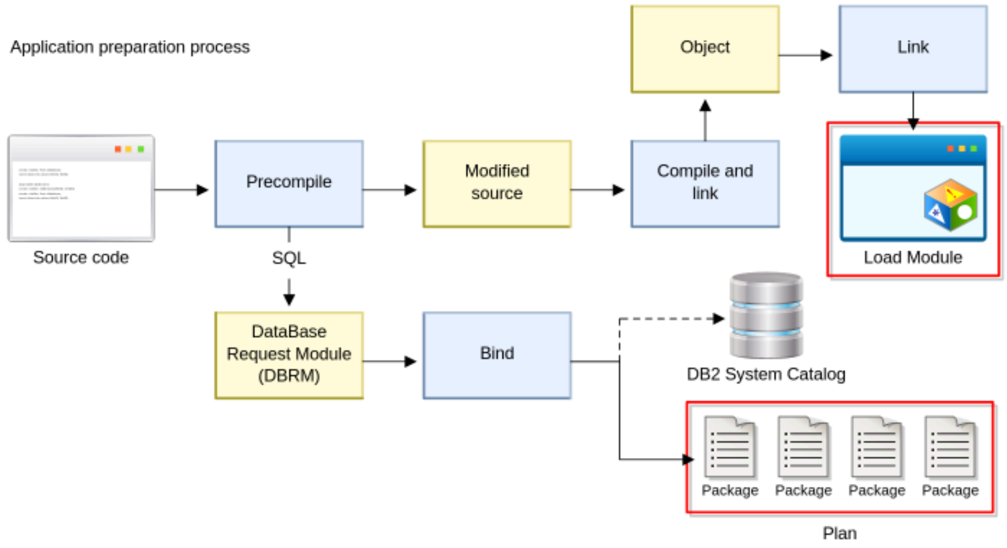
If the amount of data changes significantly or, for example, a new index is created, it is advisable to rebind, otherwise, Db2 could be using out of date and inefficient information.

With some changes, such as changes to the table structure referenced by the plan, Db2 will force you to rebind before allowing a program or plan to run.



The RUNSTATS utility obtains information about Db2 data in table spaces, indexes, and partitions and records this information in the Db2 catalog.

This utility should be run to update this information before a rebind of the object is performed because the new catalog statistics may allow Db2 to choose a better access path to the data.



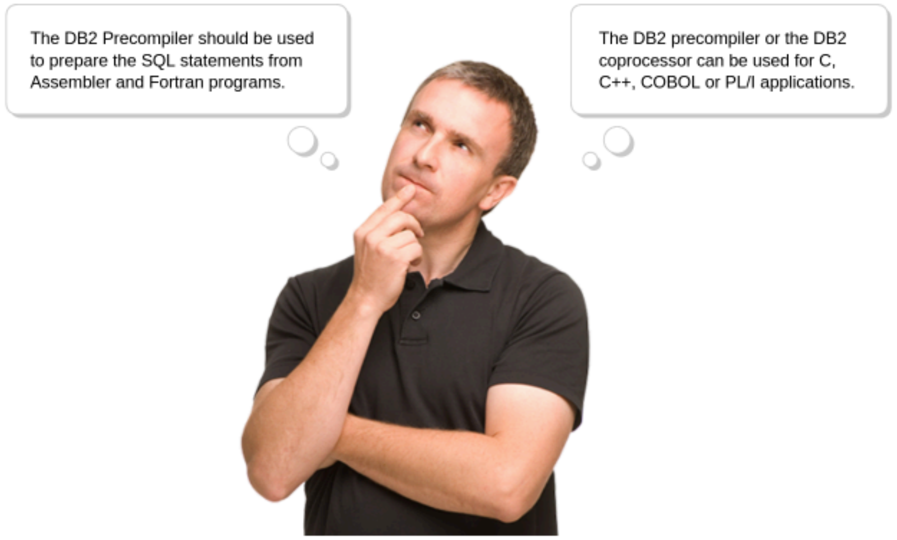
Every time a program is compiled, or recompiled, a DBRM is generated. The Bind process associated with the DBRM converts the SQL into executable code and creates a package. You can imagine that there are likely to be many packages created for your system, especially when programs call other programs and so on. In this scenario, packages need to be grouped together for a common purpose. Db2 achieves this by grouping the packages together in Collections.

The Load module created at the top right of the process above needs to know where the packages it needs can be located. It does this through a plan. A plan contains a list of Collections that the load module will use to search for the package.

Programming language	Precompiler procedure
High-level assembler	DSNHASM
C	DSNHC
C++	DSNHCPP
Enterprise COBOL	DSNHICOB
Fortran	DSNHFOR
PL/I	DSNHPLI
SQL	DSNHSQL

Precompilers are provided for many programming languages in the mainframe environment.

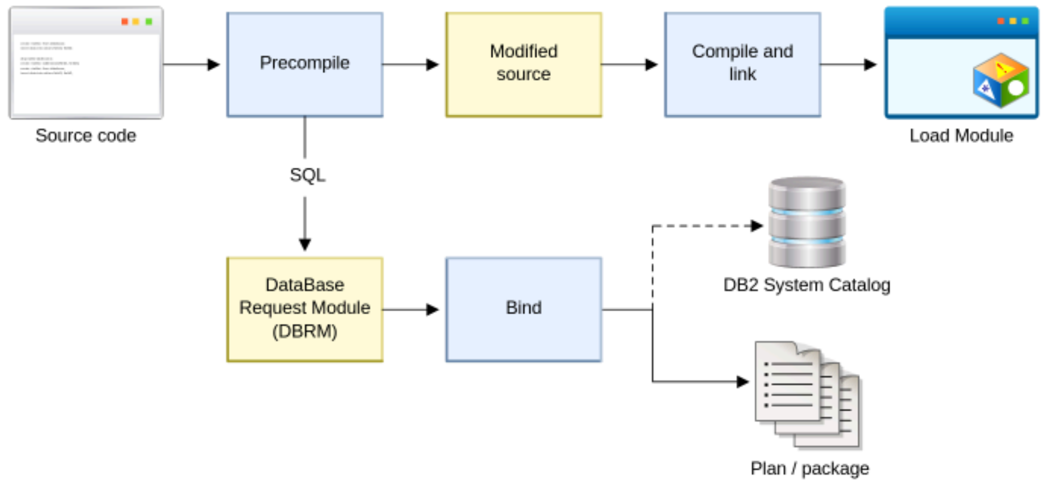
Programs in these languages can be precompiled using the Db2 interactive dialogs or, more commonly, by adding the precompiler procedures shown before the compile step in JCL.


The DB2 Precompiler should be used to prepare the SQL statements from Assembler and Fortran programs.
The DB2 precompiler or the DB2 coprocessor can be used for C, C++, COBOL or PL/I applications.

So far, you have seen how the precompiler performs the actions required when SQL statements are encountered in the source code. An alternative to the precompiler is the Db2 coprocessor. While the Db2 precompiler processes your SQL statements before you compile your program, the Db2 coprocessor performs this task while you are actually compiling the program.

The Db2 coprocessor provides fewer restrictions when it comes to SQL processing. For example, you can include SQL statements at any level of a nested program, and incorporate nested SQL INCLUDE statements.

Application preparation process



These are the steps required to prepare an application program with embedded SQL for execution:

- Precompile: Remove the SQL and put it in a DBRM.
- Compile: Standard compile.
- Bind: Parse the SQL in the DBRM. Check whether all the objects exist, check the authorities, and choose the access paths to the data.
- Link: Standard link.