SQL Course

Writing Database Applications using Embedded SQL

- To define what **embedded SQL** means.
- To distinguish between **static** and **dynamic** SQL.
- To list the rules for embedding SQL statements.

There are three different ways of writing your application when you want to use SQL:

- **Direct Invocation:** Directly invoke SQL statements.
- **Embedded SQL:** Embed SQL statements directly in your programming language.
- **Module Language:** Write SQL statements separately in a module and call them from your host language program.

Hence, SQL statements and declarations can be embedded, or directly included, in another, more traditional programming language. Embedded SQL commands may be inserted into the programming language wherever a programming command could be specified, and if a programming command in that place could be labelled or numbered or otherwise given an identity so can the embedded SQL command.
SQL provides support for seven languages; Ada, C, COBOL, FORTRAN, MUMPS, Pascal (including extended Pascal) and PL/I.

Traditional programming languages can consist of embedded SQL statements and declarations. The actions of the programming language cause the SQL statements to be executed. Execution is either **static** or **dynamic**:

Static Execution
If you know the exact text of your SQL statements at the time you wrote your application, then you can specify the exact wording of your SQL statements long before you have to execute your application. This paradigm is often called **static SQL** because the source text of the statements does not change while your application is running.
Dynamic Execution
If you do not know the percise text of the statements when you write your application, then the application programmer cannot include these statements in advance. They must be processed during program execution. This paradigm is called **dynamic SQL**.

The rules for embedding the actual SQL statements are rather simple:

- Every embedded SQL statement has to be preceded by **EXEC SQL**, and this has to appear on a single line of the source program with nothing except white spaces between the keywords.
  Exceptions to this rule are **BEGIN DECLARE SECTION**, **END DECLARE SECTION** and the **EXCAPE** clause.
- Depending on the programming language (called host language) into which you're embedding, you may have to end the embedded SQL statement with some sort of terminator. The table below consists of some examples.
- The rules for the placement of host language comments in SQL constructs are those of the host language. SQL constructs embedded in host language comments are treated as comments.

- An SQL comment may appear anywhere in an SQL construct that a separator may appear (except between the keywords in **EXEC SQL**, **EXEC SQL BEGIN DECLARE SECTION**, and **EXEC SQL END DECLARE SECTION**). SQL comments are ignored during the compilation of the SQL construct.
- Lower-case letters are allowed for SQL keywords and user-defined names.

SQL prefixes and terminators by language

Language	*SQL prefix*	*SQL Terminator*
Ada	EXEC SQL	;
C	EXEC SQL	;
COBOL	EXEC SQL	end-exec
FORTRAN	EXEC SQL	(none)
MUMPS	&SQL(	)
Pascal	EXEC SQL	;
PL/I	EXEC SQL	;

Special Rules for C:

- The SQL terminator is a semicolon.
- SQL constructs must **not** be contained within an **#include** file and must **not** be modified by a **#define** directive.
- SQL statements may be specified anywhere a **C** statement may be specified within a function block. A label may precede **EXEC SQL** on the line, subject to the usual restrictions in **C**.