Embedded SQL

- In order to access database in programs, and take further process to the query results, need to combine SQL and programming language (such as C / C++, etc.)
- Problems should be solved:
  - How to accept SQL statements in programming language
  - How to exchange data and messages between programming language and DBMS
  - The query result of DBMS is a set, how to transfer it to the variables in programming language
  - The data type of DBMS and programming language may not the same exactly.

General Solutions

- Embedded SQL
  - The most basic method. Through pre-compiling, transfer the embedded SQL statements to inner library functions call to access database.
- Programming APIs
  - Offer a set of library functions or DLLs to programmer directly, linking with application program while compiling.
- Class Library
  - Supported after emerging of OOP. Envelope the library functions to access database as a set of class, offering easier way to treat database in programming language.

Usage of Embedded SQL (in C)

- SQL statements can be used in C program directly:
  - Begin with *EXEC SQL*, end with ';'
  - Through *host variables* to transfer information between C and SQL. Host variables should be defined begin with *EXEC SQL*.
  - In SQL statements, should add ':' before host variables to distinguish with SQL's own variable or attributes' name.
  - In host language (such as C), host variables are used as general variables.
  - Can't define host variables as Array or Structure.
  - A special host variable, SQLCA (SQL Communication Area)

    EXEC SQL INCLUDE SQLCA
  - Use SQLCA.SQLCODE to justify the state of result.
  - Use *indicator* (short int) to treat *NULL* in host language.

Example of *host variables* defining

```
EXEC SQL BEGIN DECLARE SECTION;
    char SNO[7];
    char GIVENSNO[7];
    char CNO[6];
    char GIVENCNO[6];
    float GRADE;
    short GRADEI;          /*indicator of GRADE*/
EXEC SQL END DECLARE SECTION;
```

Executable Statements

- CONNECT
  - EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
- Execute DDL or DML Statements

  -
    ```
    EXEC SQL INSERT INTO SC(SNO,CNO,GRADE)
                  VALUES(:SNO, :CNO, :GRADE);
    ```
- Execute Query Statements

  -
    ```
    EXEC SQL SELECT GRADE
             INTO  :GRADE :GRADEI
             FROM SC
             WHERE SNO=:GIVENSNO AND
                   CNO=:GIVENCNO;
    ```
- Because {SNO,CNO} is the key of SC, the result of this query has only one tuple. How to treat result if it has a set of tuples?

Cursor

1. Define a cursor
   - EXEC SQL DECLARE <*cursor name*> CURSOR FOR
     SELECT …
     FROM …
     WHERE …
2. EXEC SQL OPEN <*cursor name*>
   - Some like open a file
3. Fetch data from cursor
   - EXEC SQL FETCH <*cursor name*>
     INTO :hostvar1, :hostvar2, …;
4. SQLCA.SQLCODE will return 100 when arriving the end of cursor
5. CLOSE CURSOR <*cursor name*>