Chapter 9, More SQL: Assertions, Views, and Programming Techniques

9.2 Embedded SQL

SQL statements can be embedded in a general purpose programming language, such as C, C++, COBOL,...

9.2.1 Retrieving Single Tuples with Embedded SQL

- exec sql include sqlca;
  exec sql begin declare section;
  char first_name[NAMESIZE];
  char last_name[NAMESIZE];
  char ssn[10];
  exec sql end declare section;

  strcpy(ssn, “987987987”);
  exec sql select fname, lname
  into :first_name, :last_name
  from employee
  where ssn = :ssn;
  if (sqlca.sqlcode == 0)
    printf(“%s, %s”, first_name, last_name);
  else printf(“No matching employee”);

- The embedded SQL statements is distinguished from the programming language statements by prefixing it with a command, EXEC SQL, so that a preprocessor can separate them from the host language code, and the SQL statements are terminated by a matching END-EXEC, or “;”.

- Host variable/shared variable: Within embedded SQL statements, we can refer to program variables (we call them shared variables), which are prefixed by a “:” sign.
This allows shared variables and database objects, such as attributes and relations, to have the same names.

- The shared variables used in embedded SQL statements should be declared somewhere else in the program. The declaration should preceded by `exec sql begin declare section;` and ended by `exec sql end declare section;`.

- **SQL communication area (sqlca):** After each SQL statement is executed, the DBMS provides feedback on whether the statement worked properly. This information is returned, via a collection of variables, to an area called sqlca (memory) that is shared by the host programming language and the SQL DBMS.

  - “`exec sql include sqlca;`” should be put somewhere in the program, the SQL compiler will insert the sqlca variables in place of the “`exec sql include sqlca;`” statement.

  - A variable in sqlca is called `sqlcode` which returns the status of each SQL statement execution. 0 means a successful execution; 100 means no more data/not found; < 0 means errors.

  - another variable in sqlca is `sqlstate`, which is a string of 5 characters. A value of “00000” means no error or exception; other values indicates various errors or exceptions.

### 9.2.2 Retrieving Multiple Tuples with Embedded SQL Using Cursors

- `exec sql begin declare section;`
  
  ```
  char first_name[NAMESIZE];
  char last_name[NAMESIZE];
  ```

- `exec sql end declare section;`

- `exec sql declare emp_dept cursor for`
  
  ```
  select  fname, lname
  ```
from employee
where dno= :dnumber;

exec sql open emp_dept;
while (sqlca.sqlcode == 0) {
    exec sql fetch emp_dept into :first_name, :last_name;
    if (sqlca.sqlcode == 0)
        printf("First Name: %s, Last Name: %s", first_name, last_name);
    else
        printf("ERROR MESSAGE");
}
exec sql close emp_dept;

- The Cursor structure represents an area in memory allocated for temporarily storing and processing the results of an SQL SELECT statement.
  - The cursor-name (emp_dept in above example) is the name assigned to the cursor structure.
  - The select statement defines the query.
  - The declare cursor statement is declarative; the query is not executed at this time.
  - The open statement open the cursor, and the select statement defined in declare cursor statement is executed and the set of tuples is stored in the cursor structure. This open statement will set a pointer (current pointer) pointing to the position before the first row of the query result.
  - The fetch statement fetches one row from the result into the host variables and moves the pointer to the next row in the result of the query.
  - The close statement closes the cursor.

- Update command in embedded SQL:

```sql
```
Update without cursor structure:

```sql
exec sql update employee
set salary = salary * 1.1
where dno=5;
```

Update with cursor structure:

```sql
exec sql declare emp_d5 cursor for
select ssn, salary
from employee
where dno=5
for update of salary;

exec sql update employee
set salary = salary * 1.1
where current of emp_d5;
```

* The cursor structure must be opened and positioned (using FETCH) on a row before the UPDATE command can be executed.

* Each execution of the UPDATE statement updates one row - the row at which the cursor is positioned.

* The only columns that can be updated are those listed in the FOR UPDATE OF clause of the DECLARE CURSOR statement.

* The cursor is not moved by the execution of the UPDATE statement. The FETCH statement moves the cursor.

### 9.2.3 Specifying Queries at Runtime Using Dynamic SQL

- Dynamic SQL allows a program to form an SQL statement during execution.

- Parparing a statement:

  ```sql
  exec sql prepare update_salary from
  “ update employee
  ```
set salary = salary * (1 + ?/100)
where dno = ?”;

- The question mark indicates that when the statement is executed, the value of the shared variable will be used.

- Executing prepared SQL:

  - Statements other than SELECT statements, and SELECT statements that return only a single row, are executed with the EXECUTE statement.
  
  ```sql
  exec sql execute update_salary using :rate, :dnumber;
  ```

- Using prepare statement and declare cursor statement:

  ```sql
  exec sql prepare update_salary from
  “ update employee
  set salary = salary * (1 + ?/100)
  where current of emp”;

  exec sql declare cursor emp for
  select ssn, salary
  where dno = :dnumber
  for update of salary;

  exec sql open emp;
  exec sql fetch emp into :ssn, :salary;
  exec sql execute update_salary using :rate;
  ```