



Dynamic SQL with dynamic parameters

- Only used in the execution of non query SQL statements. Use *place holder* to realize dynamic parameter in SQL statement. Some like the macro processing method in C.

:

```
EXEC SQL BEGIN DECLARE SECTION;
```

```
char sqlstring[200];
```

```
int birth_year;
```

```
EXEC SQL END DECLARE SECTION;
```

```
strcpy( sqlstring, "DELETE FROM STUDENT WHERE  
YEAR(BDATE) <= :y; ");
```

```
printf(" Enter birth year for delete :");
```

```
scanf("%d", &birth_year);
```

```
EXEC SQL PREPARE purge FROM :sqlstring;
```

```
EXEC SQL EXECUTE purge USING :birth_year;
```

:



Dynamic SQL for query

- Used to form query statement dynamically

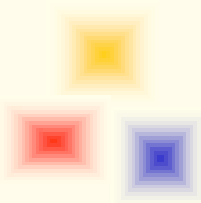
:

```
EXEC SQL BEGIN DECLARE SECTION;
char sqlstring[200];
char SNO[7];
float GRADE;
short GRADEI;
char GIVENCNO[6];
EXEC SQL END DECLARE SECTION;
char orderby[150];
strcpy( sqlstring, "SELECT SNO,GRADE FROM SC WHERE CNO= :c ");
printf(" Enter the ORDER BY clause :");
scanf("%s", orderby);
strcat( sqlstring, orderby);
printf(" Enter the course number :");
scanf("%s", GIVENCNO);
EXEC SQL PREPARE query FROM :sqlstring;
EXEC SQL DECLARE grade_cursor CURSOR FOR query;
EXEC SQL OPEN grade_cursor USING :GIVENCNO;
```



Dynamic SQL for query (Cont.)

```
if (SQLCA.SQLCODE<0) exit(1);           /* There is error in query*/
while (1) {
    EXEC SQL FETCH grade_cursor INTO :SNO, :GRADE :GRADEI
    if (SQLCA.SQLCODE==100) break;
    /* treat data fetched from cursor, omitted*/
    :
}
EXEC SQL CLOSE grade_cursor;
:
```



Stored Procedure

- Used to improve performance and facilitate users. With it, user can take frequently used database access program as a procedure, and store it in the database after compiling, then call it directly while need.
 - Make user convenient. User can call them directly and don't need code again. They are reusable.
 - Improve performance. The stored procedures have been compiled, so they don't need parsing and query optimization again while being used.
 - Expand function of DBMS. (can write script)



Example of a Stored Procedure

EXEC SQL

```
CREATE PROCEDURE drop_student  
    (IN student_no CHAR(7),  
     OUT message CHAR(30))
```

```
BEGIN ATOMIC
```

```
    DELETE FROM STUDENT
```

```
        WHERE SNO=student_no;
```

```
    DELETE FROM SC
```

```
        WHERE SNO=student_no;
```

```
    SET message=student_no || 'dropped';
```

```
END;
```

EXEC SQL

:

```
CALL drop_student(...);
```

:

/* call this stored procedure later*/