09. Programming Databases

Embedded SQL

- We allow SQL statements in a host program.
- Host program run on "client"?
- SQL run on DBMS server and response is brought to the client – same way your PgAdmin-III does?
- Below is a sample embedded SQL in ECPG (a C pre-processor for postgresql) almost compatible to oracle's Pro*C

```
#include <stdio.h>

EXEC SQL BEGIN DECLARE SECTION;
int c;
EXEC SQL END DECLARE SECTION;

int main() {

    EXEC SQL CONNECT TO test@10.100.71.21 USER test USING test;
    EXEC SQL set search_path to company;
    EXEC SQL SELECT count(*) INTO :c FROM employee;
    printf("emps: %d\n", c);
    return 0;
}
```

- In embedded SQL, there is notion of "host variables", these are variables declared within SQL DECLARE SECTION.
- Host variables can be mixed in SQL statements by pre-fixing colon. Often used to supply value into SQL statements or are used to collect values returned by executed SQL statement.
- For example above see use of host variable **c**. Example next shows usage of several host variables.

Compiling and Running Embedded SQL in C with PostgreSQL ECPG

(1) Pre-Process using following command. Produces C source file

$ ecpg prog1.pgc

(2) Compile and link C source file

$ gcc -I/opt/PostgreSQL/8.4/include -L/opt/PostgreSQL/8.4/lib ex1.c -lecpg

Note:

- You need postgresql having installed on your machine before you can compile
- May have to change the path (underlined) if postgresql is installed in some different directory.

- In the pre-processing step, all EXECSQL statements are converted to appropriate function C calls. For example SELECT statement in above program is converted to calling function ECPGdo, a function, part of ecpg library:
  ECPGdo(__LINE__, 0, 1, NULL, 0, ECPGst_normal, "select count (*) from employee", ECPGt_EOIT, ECPGt_int,&(c),(long)1,(long)1,sizeof(int), ECPGt_NO_INDICATOR, NULL , 0L, 0L, 0L, ECPGt_EORT);

Another example

```
#include <stdio.h>
#include <string.h>
EXEC SQL BEGIN DECLARE SECTION;
char query[100], name[30], buf[20];
int mdno, msal, sal;
EXEC SQL END DECLARE SECTION;

int main() {
        scanf("%d %d", &mdno, &msal);
        strcpy(query, "SELECT fname, salary FROM EMPLOYEE WHERE dno=");
        sprintf(buf, "%d", mdno);
        strcat(query, buf);
        strcat(query, " AND salary > ");
        sprintf(buf, "%d", msal);
        strcat(query, buf);
        printf("%s\n", query);
        EXEC SQL CONNECT TO test@10.100.71.21 USER test USING test;
        EXEC SQL set search_path to company;
        EXEC SQL PREPARE query_prep FROM :query;
        EXEC SQL DECLARE emp_cur CURSOR FOR query_prep;
        EXEC SQL OPEN emp_cur;
        while ( sqlca.sqlcode == 0 ) {
                EXEC SQL FETCH emp_cur INTO :name, :sal;
                printf("%-10s %10d\n", name, sal);
        }
        return 0;
}
```

Prepared Dynamic Query

- A dynamic Query is the one that uses some variables, and complete query expression is known only at run-time. For example
  query = "SELECT * FROM EMPLOYEE WHERE dno = " + dno
  Query is stored in a variable query, and becomes fully known at run-time?
- Sometime such a query is repeated run on DBMS with changed value of some parameter, for example dno in above case.
- When run repeatedly, we can ask dbms to create its “query execution plan” once and run it for following calls.
- Such queries are called prepared queries.

Benefits of Prepared Queries

- Faster Execution – since query plan can be created once and saved, this makes execution of dynamic query faster (gain is significant, when a dynamic query appears in a loop or so)
- Avoids Run time exceptions
- A popular technique to deal with SQL injections

May learn more at:

- http://intranet.daiict.ac.in/~pm_jat/postgres/html/ecpg.html
- Pro*C from Oracle has been around for much longer than ECPG; ECPG is supposed to be compatible with Pro*C, and documentation/discussions around on Pro*C should be equally good.

Data Type mis-match Issue in embedded SQL

For example

- Varchar to character array
- Date, Time etc to ??
- Row to struct
- Result-set to array of structs, and so forth.

Summary embedded SQL

SQL statements are placed in host programs.

There is notion of host variables; that are used for

(1) Making the SQL statements “parametric”

(2) Collecting data from results of executing SQL statements

It is important to note here is that SQL appearing in a host program may seem like executing on the client but actually not!

SQL statements are sent to remote DBMS.

SQL statements appearing in host program are converted to appropriate function calls. Refer pre-processed code provided here.