

CS3743 Introduction to Database Systems

Embedded SQL

Weining Zhang

Department of Computer Science
University of Texas at San Antonio

August 15, 2009

Database Programming

- Objective: To access a database from an application program (as opposed to interactive interfaces)
 - Why? A majority of database operations are made thru application program
- Main techniques of DB programming
 - Embedded commands: Database commands are embedded in a general-purpose (host) programming language
 - API (Application Program Interface): Database functions are available to the host language as a library
 - A brand new, full-fledged language with native database functions

Outline

- 1 Embedded SQL
 - Overview
- 2 Oracle Pro*Languages
 - Program Structure
 - Stored Procedures
 - Dynamic SQL
- 3 Java DB Programming
 - JDBC and SQLJ

Embedded SQL

- Idea:
 - Use (embedded) SQL to access and update data
 - Use a host programming language for data processing and other tasks
- Typical program performs following tasks
 - Declare variables interfacing SQL & host PL
 - Prepare for handling any SQL error
 - Connect to database
 - Issue SQL statements and process results
 - Disconnect the database
- Two-step compilation:
 - Pre-compilation. Translate embedded SQL to host language
 - Compilation. Translate host language to machine language

Oracle Database Programming

- Oracle pro*languages provide embedded SQL through five host languages: C/C++, Cobol, PL/I, Ada, and Pascal
 - Host program can contain all interactive SQL statements, special embedded SQL statements, PL/SQL blocks, stored PL/SQL procedures and functions
 - Programs communicate with DBMS through specially data structures defined as SQLCA
 - SQL in program has a prefix
EXEC SQL ...
- Oracle also provides Java database programming via JDBC and SQLJ

Variables Interfacing DBMS

- Declared with SQL types in


```
EXEC SQL begin declare section;
...
EXEC SQL end declare section;
```

 - Types must be compatible with types of table columns
- Accessed in host blocks as varName and in SQL block as :varName
- Conversion between SQL and C types are performed by pre-compiler

Pro*C Programs

- Steps and tasks
 - Include header file sqlca
 - Declare variables that interface with DBMS (using SQL types)
 - Include error handling code (using whenever sqlerror statement)
 - Connect to database with a user name & password
 - Access database (using EXEC SQL statement)
 - Process query result (using loop or cursor)
 - Close database connection (using Commit/Rollback, Release)

Example

A **pro*C program** that retrieves employee name, salary, commission, by a given employee number

Using Cursors

Example

A **pro*C program** that uses a cursor to retrieve name, salary, and commission of salesmen among employees

- PL/SQL blocks can be embedded into C program
- Stored PL/SQL procedures and functions as well as packaged can be used in C program

Example

A sample **pro*C program** that uses a PL/SQL block to debit an account.

- Objective: Composing and executing new (not previously compiled) SQL statements at run-time
 - a program accepts SQL statements from the keyboard at run-time
 - a point-and-click operation translates to certain SQL query
- Dynamic update is relatively simple; dynamic query can be complex
 - because the type and number of retrieved attributes are unknown at compile time

Example

Assume a relation Projects(Pno, Name, Budget, Duration). We want to delete some projects based on some conditions that only become available at run time of a program.

```
delete Projects where <unknown>
```

Possible conditions may be something like

```
Budget > 2000000
Budget > 2000000 and Duration > 24
Name = 'Manned Spacecraft to Mars'
```

- General framework
 - Declare a host string variable
 - Construct an SQL statement at run-time and assign the SQL to the variable
 - Let the DBMS parses & executes the SQL statement in the host variable.
- Methods of dynamic SQL
 - 1 Update involving no host variable
 - 2 Update involving a fixed number of variables
 - 3 Query involving a fixed select-clause and fixed number of variables
 - 4 Query with unknown select list and unknown number of variables

Execute Immediate

An embedded SQL statement that causes a dynamic SQL statement to be compiled and executed immediately.

```
exec sql execute immediate :host_variable;
```

Example

A sample **pro*C program** that uses dynamic SQL Method 1 to create a table, insert a row, commit the insert, then drop the table.

Java Database Connectivity

- JDBC (Java DataBase Connectivity):
 - SQL connection function calls for Java programming
- A Java program with JDBC functions can access any relational DBMS that has a JDBC driver
- JDBC allows a program to connect to several databases (known as data sources)

Prepare-Execute

First construct and compile an SQL update statement using a fixed number of variables. The statement can then be executed multiple times

```
prepare stmt_name from :host_variable
```

- Implement Method 2: The host_variable can contain any SQL statement other than a query and can have a known number of placeholders.
- The stmt_name is an SQL identifier and needs not be explicitly declared.

Example

A sample **pro*C program** that uses dynamic SQL Method 2 to insert two rows into * the EMP table, then delete them.

Java JDBC Programming

- Import JDBC library (java.sql.*)
- Load JDBC driver.
(Class.forName("oracle.jdbc.driver.ThinDriver"))
- Connect to DB (getConnection(<connect string>))
- Create a query statement object from Statement class
- Create a result object (by executeQuery())
- Process the result (in a ResultSet object)
- Close the query
- Close the DB connection

Example

A sample **Java/JDBC program**.

SQLJ

- Java with embedded SQL (prec. w/ #sql)
- A part of SQL 1999 Standard.
- Easy to use:
 - Connect to database
 - Create iterators for query results
- Need two compilation steps:

```
sqlj file[.sqlj]  
javac file.java
```

Example

A sample **SQLJ** program