Embedded SQL

John Ortiz

Why Isn't Interactive SQL Enough?

- How to do this using interactive SQL?
  - Print a well-formatted transcript of a student with either a name or an id
- What does it take for someone to use SQL?

  To know schemas, syntax, semantics, mathematics, logic, …
- Solution?

  Write application programs to help naïve users to manipulate the data.
- How to access database from within a program?

Idea of Embedded SQL

- Combine the power of both SQL & a general purpose programming language.
  - Use (embedded) SQL to perform data retrieval and updates.
  - Use the general purpose programming language (host PL) to perform more complex data processing and to provide a friendly user interface.

Oracle API

- Support embedded SQL through five host PLs (pro*languages)

  C/C++, Cobol, PL/I, Ada, Pascal
- Oracle8i supports Java/JDBC and SQLJ
- SQL stmts are placed in host PL programs
- Data flow from database to program variables and vice versa
- Two step compilation:
  - Precompilation: prog.pc → prog.cc
  - Compilation: prog.cc → prog.o

A Sample Pro*C/C++ Program

- The program is sample1.pc
- Common tasks:
  - Declare variables interfacing SQL & host PL
  - Prepare for any SQL error
    - Include sqlca (communication area)
    - Use whenever sqlerror, …
    - Provide for error processing
  - Connect to database
  - Issue SQL statements
  - Disconnect the database

Embedded SQL Statements

- Every SQL statement is preceded by *exec sql*
- Can use all SQL statements plus special ones.
  - Connect, disconnect
  - Whenever
  - Select … into … from …
  - Rollback
  - Commit
  - Statements declare and use cursors
  - Statements define and execute dynamic queries

Sample Program Using A Cursor

- How does a program handle query result containing more than one tuple?
- Use a cursor. See sample2.pc
- A cursor is a "window" through which one tuple can be accessed.
- A cursor must be declared with a query
- Open cursor executes the query
- Fetch cursor moves to the next tuple
- A cursor can be closed and re-opened

Dynamic SQL

- Create SQL statements at run time and then execute the newly created statements.
- General framework:
  - Declare a host string variable.
  - Place an SQL statement in the variable at run-time.
  - Let the DBMS parse & execute the SQL statement in the host variable.

Pro*C/C++ and PL/SQL

- On Oracle, Pro*C/C++ programs may contain any SQL statement and PL/SQL blocks.
- See this sample program
- Precompiled and stored PL/SQL procedures and functions can be used directly in embedded SQL statement

  exec sql execute p(…)

Oracle JDBC

- JDBC (Java Database Connectivity): API that allows Java applications and applets to connect to Oracle databases, send SQL statements, and receive data from databases.
- Need to set up CLASSPATH environment variable.
- See QueryUnivDB.java

JDBC: Program Tasks

- Import Java SQL API library
- Load JDBC driver: Class.forName(…)
- Create a DB connection
  DriveManager.getConnection(<connect string>)
- Create a query statement object
- Create a result object by executing the query
- Process the result
- Close the query
- Close the connection

Oracle SQLJ

- Java with embedded SQL (prec. w/ #sql)
- Part of the new SQL 1999 Standard.
- Much easier to use:
  - Connect to database
  - Create iterators for query results
- Oracle8i only supports JDK1.1
- Need two compilation steps:
  - sqlj file[.sqlj]
  - javac file.java
- See QueryUnivDB.sqlj

Embedded SQL Summary

- Many additional statements handling communication between database and PL system
- Special handling of multiple tuple result (cursor)
- Must handle errors and exceptions generated by DBMS
- Must pay special attention to program structure (goto, whenever not found,…)
- JDBC & SQLJ have much better DB-PL interface