CS422 Principles of Database Systems
Embedded SQL and JDBC

Chengyu Sun
California State University, Los Angeles

Applications and Databases

Call-level Interface (CLI)

A library of functions that allow applications to
- connect to the database
- send SQL statements to the database
- get the results back from the database

```c
String url = "jdbc:mysql://localhost/user=sql\&password=sql";
Connection c = DriverManager.getConnection(url);
Statement stmt = c.createStatement();
ResultSet rs = stmt.executeQuery("select * from Products");
while (rs.next) {
    // some processing here
}
```

Embedded SQL

- Embed SQL statements directly in host language (specifically, C)
  - Check SQL syntax at compilation time
  - Variables shared by SQL and the host language
  - Part of the SQL standard

ECPG Example

- `artists.pgc`
- `EXEC SQL`
- `Compilation`
  - Preprocessing: `.pgc` to `.c`
  - Compile: `.c` to `.o`
  - Link: `.o` to executable
    - `-lecpg`

Error Handling

```c
EXEC SQL WHENEVER condition action;
```

- `condition`
  - SQLERROR
  - SQLWARNING
  - NOT FOUND

- `action`
  - CONTINUE
  - STOP
  - SQLPRINT
  - GOTO label
  - CALL name (args)
Connections

EXEC SQL CONNECT TO target USER user;
EXEC SQL DISCONNECT;

- target
  - dbname@hostname
- user
  - username USING password

Shared Variables

EXEC SQL BEGIN DECLARE SECTION;
EXEC SQL END DECLARE SECTION;

- Shared (host) variables
  - Referenced as :variable_name in SQL statements
  - Referenced as variable_name in host language

SELECT INTO

EXEC SQL SELECT col(s) INTO variable(s) WHERE cond(s);

- Needs indicator to handle null values
  - 0: result not null
  - negative: null
  - positive: result truncated
- Only for single result

Cursor and FETCH INTO

EXEC SQL DECLARE name CURSOR FOR query;
EXEC SQL OPEN name;
do {
  ...
  EXEC SQL FETCH NEXT FROM name INTO variables;
  ...
} while ( strcmp(sqlca.sqlstate,"02000") != 0 );

- SQLSTATE and sqlca
  - http://www.postgresql.org/docs/7.4/static/errcodes-appendix.html

Updates

EXEC SQL update;
EXEC SQL COMMIT;

- update could be any update statement
  - create table, insert, delete, update ...
- Examples:
  - EXEC SQL CREATE TABLE foo (number integer, ascii char[16]);
  - EXEC SQL CREATE UNIQUE INDEX num ON foo(number);
  - EXEC SQL COMMIT;
  - EXEC SQL INSERT INTO foo (number, ascii) VALUES (9999, 'ทดสอบ');
  - EXEC SQL COMMIT;

Dynamic SQL

EXEC SQL BEGIN DECLARE SECTION;
char *stmt;
EXEC SQL END DECLARE SECTION;
EXEC SQL PREPARE name FROM :stmt;
EXEC SQL EXECUTE name [INTO variables] [USING values];

- When SQL statements themselves are generated by the host language
JDBC Basics ...

- import java.sql.*;
- Load driver
  - Class.forName("org.postgresql.Driver")
- Create connection
  - Connection c = DriverManager.getConnection( url, username, password );
  - URL
    - jdbc:postgresql://hostname/dname

... JDBC Basics

- Create statement
  - Statement stmt = c.createStatement();
    - stmt.executeQuery()
    - stmt.executeUpdate()
- Get result back
  - ResultSet rs

  http://java.sun.com/j2se/1.3/docs/guide/jdbc/

DB Query Results

- In a program, we want to
  - select * from UserInfo;
  - Access each row
  - Access column in a row
  - Access column names

<table>
<thead>
<tr>
<th>user</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>csun</td>
<td>cs122</td>
</tr>
<tr>
<td>csun</td>
<td>cs201</td>
</tr>
<tr>
<td>csun</td>
<td>cs320</td>
</tr>
</tbody>
</table>

JDBC ResultSet – Row Access

- next() – move cursor down one row
  - true if the current row is valid
  - false if no more rows
  - Cursor starts from before the 1st row

JDBC ResultSet – Column Access

- Access the columns of current row
  - getXX( String columnName )
    - E.g. getString("user");
  - getXX( int columnIndex )
    - columnIndex starts from 1
    - E.g. getString(1);
JDBC ResultSet – Access Column Names

```java
ResultSetMetaData meta = rs.getMetaData();
```

**ResultSetMetaData**
- `getColumnName( columnIndex )`
  - Column name
- `getColumnLabel( columnIndex )`
  - Column title for display or printout