CS411
Database Systems

07: SQL System Aspects

Kazuhiro Minami

System Aspects of SQL

(Chapter 9: Four more ways to make SQL calls from outside the DBMS)

Call-Level Interface
- PHP
- Java Database Connectivity

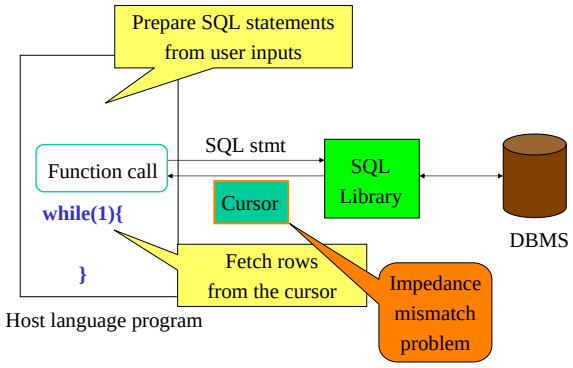
Stored procedures

Embedded SQL

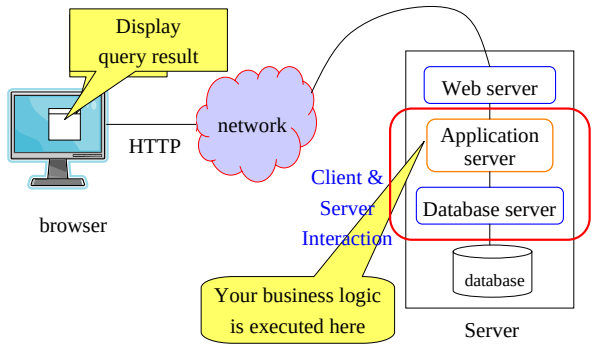
Connecting SQL to the Host Language

- Outside
  - API Approach:
    - Vendor specific libraries[80's-]
      - MySQL API for PHP
    - Open interfaces [90's -]
      - JDBC, ODBC
- Embedded SQL [70's-]
  - Embedded SQL for C/C++.
  - Not widely used.
- Inside
  - Stored procedures/functions: [80's-]

SQL/Host Language Interface (CLI)



The Three-Tier Architecture of Database Applications



MySQL + PHP

What is PHP?

- Stands for Hypertext Preprocessor
- A server-side scripting language
- PHP scripts are executed on the server
- Supports many databases (MysQL, Infomix, Oracle, etc.)

What is a PHP file?

- Contain text, HTML tags and scripts
- PHP files are returned to a browser as plain HTML
- Have a file extension of ".php"

Steps for writing a DB application

Set up a table

1. SSH to a csil Linux machine (e.g., csil-linux-ts1)
2. Login to MySQL server
   % mysql -h csil-projects.cs.uiuc.edu -u netid –p
3. Choose a database
   mysql > use <your database>;
4. Create a table “hello”
   mysql > CREATE TABLE hello (varchar(20));
5. Insert a tuple
   mysql > INSERT INTO hello VALUES (‘Hello World!’);
6. Quit MySQL
   mysql > quit

Steps for writing a DB application

Write a PHP program

1. Go to the directory ~/csil-projects/public_html
   % cd csil-projects/public_html
2. Write hello_world.php
3. Open http://csil-projects.cs.uiuc.edu/~username/hello_world.php with a web brower

hello_world.php

```
<html>
<body>
<?php
$host = 'csil-projects.cs.uiuc.edu';
$user = 'minami'; $password = 'password';
$link = mysql_connect($host, $user, $password) or die ('Could not
    connect: ' . mysql_error());
mysql_select_db('minami_db') or die ('Could not select database<br>');
$query = 'SELECT * FROM hello';
$result = mysql_query($query);
while ($row = mysql_fetch_array($result)) {
   echo ”$row[message]<br>”;
}
mysql_free_result($result);
mysql_close($link);
?>
</body>
</html>
```


PHP Basics

- All PHP code exist inside HTML text
  ```
  <?php
    PHP code goes here
  ?>
  ```
- Variables
  - Untyped and need not be declared
  - Begins with ‘$’
- Strings
  - Surrounded by either single or double quotes
    - $host = 'csil-projects.cs.uiuc.edu';
    - $x = ‘A host is $host.’
    - $x = “A message is $host.”
  - Concatination of strings
    - 'Could not connect: ' . mysql_error()

PHP Basics (Cont.)

- Arrays
  - Ordinary arrays
    - $a = array(30, 20, 10, 0) with $a[0] equal to 30, $a[1] equal to 20 and so on
  - Associative arrays
    - $seasons = array(‘spring’ => ‘warm’,
      ‘summer’ => ‘hot’,
      ‘fall’ => ‘warm’,
      ‘winter’ => ‘cold’);

      Then, seasons[‘summer’] has the value ‘hot’.

Creating a Database Connection

- Before you can access data in a database, you must create a connection to the database
- Syntax: mysql_connect(servername, username, password);
- Example:

```
<?php
$con = mysql_connect("localhost",”user",”pwd");
if (!$con)  {  die('Could not connect: ' . mysql_error());  }//
   some code
?>
```


Executing SQL Statements

- Choose a database

```
mysql_select_db('minami_db')
   or die ('Could not select database<br>');
```

- Execute a SQL statement

```
$query = 'SELECT * FROM hello';
$result = mysql_query($query);
```


Cursor Operations: Fetching results

- Use the mysql_fetch_array() function to return the first row from the recordset as an array.
- Each call to mysql_fetch_array() returns the next row in the recordset.
- The while loop loops through all the records in the recordset.
- To refer to the value of “message” attribute, we use the PHP $row variable ($row[message]).

```
while ($row = mysql_fetch_array($result)) {
   echo ”$row[message]<br>";
}
```

Insert Data From a Form Into a Database

- When a user clicks the submit button in the HTML form, the form data is sent to "insert.php".

```
<html>
<form action="insert.php" method="post">
ISBN: <input type="text" name="isbn" />
Title: <input type="text" name="bname" />
<input type="submit" / value="Add">
</form>
</html>
```


Insert Data From a Form Into a Database

- The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables.

```
$book = $_POST["bname"];
$isbn = $_POST["isbn"];
$sql = "INSERT INTO book(isbn, name) VALUES ($isbn, '$book')";
mysql_query($sql))
```


JDBC

All these methods follow the basic PHP paradigm

1. Connect to a DB server.
2. Say what database you want to use.
3. Assemble a string containing an SQL statement.
4. Get the DBMS to prepare a plan for executing the statement.
5. Execute the statement.
6. Extract the results into variables in the local programming language.

JDBC

- Java Database Connectivity (JDBC) is a library similar to SQL/CLI, but with Java as the host language.
- JDBC/CLI differences are often related to the object-oriented style of Java, but there are other differences.

The brainchild of a former UIUC undergrad

Connections

- A *connection object* is obtained from the environment in a somewhat implementation-dependent way.
- We'll start by assuming we have myCon, a connection object.

Statements

- JDBC provides two classes:
  1. Statement = an object that can accept a string that is an SQL statement and can execute such a string.
  2. PreparedStatement = an object that has an associated SQL statement ready to execute.

Creating Statements

- The Connection class has methods to create Statements and PreparedStatements.

```
Statement stat1 = myCon.createStatement();
PreparedStatement stat2 =
  myCon.createStatement(
      "SELECT beer, price FROM Sells" +
      "WHERE bar = 'Joe''s Bar'"
  );
```

Java trick: + concatenates strings.

createStatement with no argument returns a Statement; with one argument it returns a PreparedStatement.

Executing SQL Statements

- JDBC distinguishes queries from modifications, which it calls "updates."
- Statement and PreparedStatement each have methods executeQuery and executeUpdate.
  - For Statements, these methods have one argument: the query or modification to be executed.
  - For PreparedStatements: no argument.

Example: Update

- stat1 is a Statement.
- We can use it to insert a tuple as:

```
stat1.executeUpdate(
    "INSERT INTO Sells" +
    "VALUES('Brass Rail', 'Bud', 3.00)"
);
```

Example: Query

- stat2 is a PreparedStatement holding the query "SELECT beer, price FROM Sells WHERE bar = 'Joe''s Bar'".
- executeQuery returns an object of class ResultSet --- we'll examine it later.
- The query:

```
ResultSet Menu = stat2.executeQuery();
```

Accessing the ResultSet

- An object of type ResultSet is something like a cursor.
- Method Next() advances the "cursor" to the next tuple.
  - The first time Next() is applied, it gets the first tuple.
  - If there are no more tuples, Next() returns the value FALSE.

Accessing Components of Tuples

- When a ResultSet is referring to a tuple, we can get the components of that tuple by applying certain methods to the ResultSet.
- Method get*X* (*i*), where *X* is some type, and i is the component number, returns the value of that component.
  - The value must have type *X*.

Example: Accessing Components

- Menu is the ResultSet for the query "SELECT beer, price FROM Sells WHERE bar = 'Joe''s Bar'".
- Access the beer and price from each tuple by:

```
while ( Menu.Next() ) {
  theBeer = Menu.getString(1);
  thePrice = Menu.getFloat(2);
     /* do something with theBeer and
        thePrice */
}
```