Chapter 9  SQL in a server environment

- SQL in a Programming Environment
  - embedded SQL
  - persistent stored modules
- Database-Connection Libraries
  - Call-level interface (CLI)
  - JDBC
  - PHP
The third approach to connecting databases to conventional languages is to use library calls.

1. C + CLI
2. Java + JDBC
3. PHP + PEAR/DB
Three-Tier Architecture:

1. **Web servers** --- talk to the user.
2. **Application servers** --- execute the business logic.
3. **Database servers** --- get what the app servers need from the database.
DBMS environment:

- **Schemas**: collections of tables, views, assertions, domains and so on.
- **Catalog**: collections of schemas, information about all the schemas in the catalog.
- **Clusters**: each user has an associated cluster, so in a sense, a cluster is “the database” as seen by a particular user.
The database is, in many DB-access languages, an *environment*.

Database servers maintain some number of *connections*, so app servers can ask queries or perform modifications.

The app server issues *statements*: queries and modifications, usually.
Diagram to Remember

Environment

Connection

Statement
Java Database Connectivity (JDBC) is a library with Java as the host language.
import java.sql.*;
Class.forName("com.mysql.jdbc.Driver");
Connection myCon = DriverManager.getConnection("...");

The JDBC classes

Loaded by forName

URL of the database your name, and password go here.

The driver for mySql; others exist
Statements

- JDBC provides two classes:
  1. *Statement* = an object that can accept a string that is a SQL statement and can execute such a string.
  2. *PreparedStatement* = an object that has an associated SQL statement ready to execute.
The `Connection` class has methods to create `Statements` and `PreparedStatements`.

Statement `stat1 = myCon.createStatement();`

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

`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: one argument: the query or modification to be executed.
  - For PreparedStatements: no argument.
Example: Update

- stat1 is a Statement.
- Use it to insert a tuple as:

```java
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:

  ```java
  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, get the components of that tuple by applying certain methods to the ResultSet.

- Method $\text{get}X(i)$, 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 = ResultSet for query “SELECT beer, price FROM Sells WHERE bar = ’Joe’s Bar’ ”.
- Access beer and price from each tuple by:

```java
while ( menu.next() ) {
    theBeer = Menu.getString(1);
    thePrice = Menu.getFloat(2);
    /*something with theBeer and thePrice*/
}
```
PHP (personal home page)

- A scripting language to be used for actions within HTML text.
- DB library exists within PEAR (PHP Extension and Application Repository).
  - Include with `include(DB.php)`.
Variables in PHP

- Must begin with `$`
- OK not to declare a type for a variable.
- But you give a variable a value that belongs to a “class”, in which case, methods of that class are available to it.
String Values

- PHP solves a very important problem for languages that commonly construct strings as values:
  - How do I tell whether a substring needs to be interpreted as a variable and replaced by its value?
- PHP solution: Double quotes means replace; single quotes means don’t.
Example: Replace or Not?

$100 = "one hundred dollars";
$sue = 'You owe me $100.';
$joe = "You owe me $100.";

- Value of $sue is ‘You owe me $100’, while the value of $joe is ‘You owe me one hundred dollars’. 
PHP Arrays

- Two kinds: *numeric* and *associative*.
- Numeric arrays are ordinary, indexed 0, 1, ...
  - Example: 
    ```php
    $a = array("Paul", "George", "John", "Ringo");
    ```
    - Then $a[0]$ is "Paul", $a[1]$ is "George", and so on.
Associative Arrays

- Elements of an associative array $a$ are pairs $x => y$, where $x$ is a key string and $y$ is any value.
- If $x => y$ is an element of $a$, then $a[x]$ is $y$. 
Example: Associative Arrays

- An environment can be expressed as an associative array, e.g.:

```php
$myEnv = array(
    "phptype" => "oracle",
    "hostspec" => "www.stanford.edu",
    "database" => "cs145db",
    "username" => "ullman",
    "password" => "notMyPW");
```
Making a Connection

- With the DB library imported and the array $myEnv available:

  ```php
  $myCon = DB::connect($myEnv);
  ```

  Function `connect` in the DB library

  Class is `Connection` because it is returned by `DB::connect()`.
Executing SQL Statements

- Method `query` applies to a Connection object.
- It takes a string argument and returns a result.
  - Could be **an error code** or the **relation** returned by a query.
Example: Executing a Query

- Find all the bars that sell a beer given by the variable $beer.

```
$beer = 'Bud';
$result = $myCon->query(
    "SELECT bar FROM Sells"
    "WHERE beer = $beer ;";
);
```

Remember this variable is replaced by its value.
Cursors in PHP

- The result of a query is the tuples returned.
- Method `fetchRow` applies to the result and returns the next tuple, or FALSE if there is none.
Example: Cursors

```php
while ($bar = $result->fetchRow()) {
    // do something with $bar
}
```
Summary

- Embedded SQL (shared variables, EXEC SQL, Cursor), Dynamic SQL
- SQL/PSM
- Call-level Interface (SQL/CLI)
- JDBC
- PHP more info: http://www.w3school.com.cn/php