UFCEKG-20-2 : Data, Schemas and Applications
Multitier Architecture, MySQL & PHP
Multitier architecture:

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client–server architecture in which presentation, application processing, and data management functions are logically separated. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture. The most widespread use of multi-tier architecture is the three-tier architecture.

N-tier application architecture provides a model by which developers can create flexible and reusable applications. By segregating an application into tiers, developers acquire the option of modifying or adding a specific layer, instead of reworking the entire application. Three-tier architectures typically comprise a presentation tier, a business or data access [logic] tier, and a data tier.

(Wikipedia: Multitier Architecture)
3-tier architecture (application view)

**Presentation tier**
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

**Logic tier**
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

**Data tier**
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Advantages of the 3-tier architecture approach:

- The ability to **separate logical components** of an application ensures that applications are easy to manage and understand.
  i.e. experts can be employed that specialise in one of the layers e.g. user interface design

- Because **communication can be controlled** between each logical tier of an application, changes in one tier, for example, the database access tier, do not have to affect the client component.
  i.e. a change from one DBMS to another would only require a change to the component in the data access layer with little or no effect on the business/logic (middle) or UI layer.

- **Specific tools and technologies suited to each layer** can be deployed (and may evolve at a different pace).
Typical web-oriented 3-tier architecture
Web-oriented 3-tier architecture: tools & technologies

- **Presentation tier** – Browser / custom client, Client Side Scripting (JavaScript, ActionScript, VBScript etc.), Applets.

- **Logical Tier** – Web Server (Apache, IIS, Websphere etc.); Scripting Languages (PHP, Perl etc.), Programming Languages (Java, C, C# etc), Application Frameworks (Ruby on Rails etc.)

- **Data Tier** – Database Management System (DBMS) (Oracle, MySQL, SQL Server, DB2 etc.), XMLDB
MySQL in the persistence / data tier:

MySQL

- **Open Source** (relational) database server
  - Runs on many platforms (Unix & Windows)

- **Networked server** – no fancy GUI like MS Access.
  - You can find *clients (such as phpMyAdmin)* that provide a GUI.

- Great for small, medium to large-sized applications (ebay, amazon, facebook etc. all make use of it)
MySQL in the persistence /data tier:

MySQL Strengths

- **High performance**
  benchmarks very well against commercial dbs

- **Low-cost**
  no cost under open source licence

- **Easy to configure and learn**
  easy to set up, SQL compliant

- **Portable**
  Linux, Unix and Windows versions

- **Open Source**
  source code is available for modification
phpMyAdmin

- A MySQL client written in PHP

- Via a browser you can:
  - Manage Databases
  - Manage MySQL users
  - Create tables, add/edit/delete data, browse data
  - Submit queries (SQL)
  - Import and Export tables

- A great way to learn SQL!

- phpMyAdmin @ localhost: [http://localhost/phpmyadmin](http://localhost/phpmyadmin)
phpMyAdmin interface:

- Home
- Pop-up help
- Sql window
- Phpmyadmin help
- Mysql help
- Db status
- Manage users
- Import data
- Export db
- Databases
There are three main API options when considering connecting to a MySQL database server:

- **PHP's MySQL Extension** - original extension which provides a procedural interface and is intended for use only with MySQL versions older than 4.1.3. Can be used with versions of MySQL 4.1.3 or newer, but not all of the latest MySQL server features will be available.

- **PHP's mysqli Extension** - MySQL improved extension takes advantage of new features found in MySQL versions 4.1.3 and newer. The mysqli extension is included with PHP versions 5 and later.

- **PHP Data Objects (PDO)** - PHP Data Objects, or PDO, is a database abstraction layer that provides a consistent API regardless of the type of database server. In theory, it allows for the switch of the database server, from say Firebird to MySQL, with only minor changes to the PHP code.
Advantages of the mysqli API

- Object-oriented interface
- Support for Prepared Statements
- Support for Multiple Statements
- Support for Transactions
  - Enhanced debugging capabilities
  - Embedded server support

**Note:**
If using MySQL versions 4.1.3 or later it is *strongly* recommended that the **mysqli** extension is used.
### Example records (3)

<table>
<thead>
<tr>
<th>qid</th>
<th>quote</th>
<th>author</th>
<th>dob</th>
<th>dod</th>
<th>url</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>There is no remedy but to love more.</td>
<td>Henry David Thoreau</td>
<td>1817</td>
<td>1862</td>
<td><a href="http://en.wikipedia.org/wiki/Henry_David_Thoreau">http://en.wikipedia.org/wiki/Henry_David_Thoreau</a></td>
<td>love</td>
</tr>
<tr>
<td>6</td>
<td>Work is the curse of the drinking classes.</td>
<td>Oscar Wilde</td>
<td>1854</td>
<td>1900</td>
<td><a href="http://en.wikipedia.org/wiki/Oscar_wilde">http://en.wikipedia.org/wiki/Oscar_wilde</a></td>
<td>humour</td>
</tr>
<tr>
<td>11</td>
<td>Religion is what keeps the poor from murdering the rich.</td>
<td>Napoleon Bonaparte</td>
<td>1769</td>
<td>1821</td>
<td><a href="http://en.wikipedia.org/wiki/Napoleon">http://en.wikipedia.org/wiki/Napoleon</a></td>
<td>politics</td>
</tr>
</tbody>
</table>

**Entity Model**

- **PK**: qid
- **quote**
- **author**
- **dob**
- **dod**
- **url**
- **category**

The diagram shows a table with columns for qid (primary key), quote, author, dob, dod, url, and category.
php script using mysqli (select example)

// Connect to the db
$mysqli = new mysqli('hostname','username','password','database');

// Send the query to the database and pull the records in a
// certain category using the SELECT statement
// If the result returns true
if ($result = $mysqli->query("SELECT quote, url FROM quote
WHERE category='love'")) {

    // print out the number of records retrieved
    echo 'For the category "love", there are ' . $result->num_rows.' records.<br/>';

    // The "fetch_object()" method allows access to the returned
    // rows within the resource object ($result in this case).
    while ($row = $result->fetch_object()) {
        echo 'Quote: '.$row->quote.' '<br/>
        echo 'URL: <a href='.$row->url.'>'.$row->url.'<a><br/>';}
}

else { // it’s an error & the query failed
    echo $mysqli->error;
} // end else

$mysqli->close();