Lecture #27: DB Administration and Modern Architecture: The last real lecture.
Administrivia

- Homework
  - HW 9 Is Out

- Do Some Reading
  - Check Out Python and MongoDB Book
On The Menu

- Modern Architecture
  - PHP Stack vs. AJAX Stack
  - Simple Twitter via Apis
- Database Administration
  - Roles
  - Application Architecture
  - Environments
- Database Migrations
- Optimization
- Concurrency Issues
Our Architecture

[Diagram showing a database connected to PHP, which in turn connects to HTML via Chrome browser]
Alternative (Modern Architecture)
API Design in PHP

- Easy - instead of writing out HTML we'll write out some other type of data - in this example we'll use JSON

- Two New Commands to do this
  - header('Content-Type: application/json');
  - json_encode([php_array]);

- Twitter example
  - User profile
  - Tweets
Real Database Application Architecture

User

HTTP Port 80

Load Balancer

HTTP Port 80

Firewall

Application Server

DB Connection

Database Server
**Database Roles**

- Often a database application will have many classes of users
  - DBA, Developer, Analytics, End User
- MySQL Allows Administrators to grant certain classes of users certain privileges.
  - SELECT, UPDATE, INSERT, DELETE, ALTER
  - All customizable on a user level
  - All customizable on a table level
## Processing Rights: Art Gallery

<table>
<thead>
<tr>
<th>Data Staff</th>
<th>Customers</th>
<th>Sales</th>
<th>Paintings</th>
<th>Artists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Staff</td>
<td>Insert Change Query</td>
<td>Insert Query</td>
<td>Query</td>
<td>Query</td>
</tr>
<tr>
<td>Curator</td>
<td>Query</td>
<td>Query</td>
<td>Insert Change Query</td>
<td>Insert Change Query</td>
</tr>
<tr>
<td>Gallery Owner</td>
<td>Insert Change Query</td>
<td>Insert Change Query</td>
<td>Insert Change Query</td>
<td>Insert Change Query</td>
</tr>
<tr>
<td>DBMS Admin.</td>
<td>Modify structures; Grant rights</td>
<td>Modify structures; Grant rights</td>
<td>Modify structures; Grant rights</td>
<td>Modify structures; Grant rights</td>
</tr>
</tbody>
</table>
Environments

- Often applications are developed in a multi-tiered environment
  - Development
    - The place where the code gets written
    - Should be similar to staging, but often it's somebody's laptop
  - Staging
    - Computing environment that is technically identical to production except for scale and load
  - Production
    - Where the code is run for real
Database Roles & Environments

- **Development**
  - The developer is has God-like powers over all aspects of the system

- **Staging**
  - The developer has some access to staging servers and database

- **Production**
  - The developer likely does not have access to production servers.
Database Migration

- Often in production applications it becomes necessary to change the underlying schema or data without interrupting operations.
- A Database migration is an sequence of database operations designed to alter a database without disturbing the overall integrity of its data.
- Generally written by developers but applied by database administrators.
Database Security

- Run the DBMS behind a firewall!!
- Apply *all* patches to the OS and DBMS
- Use the least functionality possible
  - Disable all unnecessary access and network protocols; restrict user access; disable guest users, etc
- Protect the computer itself (lock it up!!!)
- Manage accounts and user groups carefully
  - E.g. forbid null passwords
- Plan for security emergencies
Database Optimization

- In production performance is often overriding priority, so optimization often becomes hugely important.

- Optimizations include
  - Index Creation
  - View Materialization
  - Query Optimization
  - De-normalization (gasp)
Terminology: Concurrency Control

- Definition: to insure that one user’s work does not inappropriately influence another user’s work.
- Example: A user should be able to set up an airline reservation for a customer and get the same result whether there are 5, 50, or 500 users on the system at the time.
Concurrency Control: Locks

- Concurrency control requires trade-offs
- For example, one user could obtain strict concurrency control by locking the entire database ... and then nobody else could work!
- All multi-user systems require a schema for managing concurrency control, not just database systems
Terminology: Transaction Logic

- Group together database actions based on the task
- These are commonly designated as
  - Start transaction
  - Commit transaction
  - Rollback transaction *if needed*
Terminology: Resource Locking

- Locks can be implicit (set by the system) or explicit (set by the programmer)
- Granularity of a lock: indicates the size. For example, the large granularity of locking an entire file insures the transaction but prevents other users from working. Locks with small granularity require much more code (e.g. locking one record) but conflicts among users are less common.
Terminology: Atomic Transactions

- LUWs: logical units of work
- For example, given a sale:
  - Change the customer’s record: increase the amount due
  - Change the salesperson’s record to increase the commission earned
  - Insert the new order record into the database
You might have heard of the “deadly embrace” ...

- When two users are vying for the same resource and each has locked the other’s.
- For example:
  - User A has locked the clients file to add a client
  - User B has locked the accounts file to add an account
  - User A needs to add an account for the new client
  - User B needs to assign a client to the new account
  - Both users cannot relinquish the lock on their respective file before completing the transaction(s).
- Every database system has algorithms to deal with this!
Technical Interviews - A Word

- Not your typical job interview
- Interviewer wants to establish both your technical proficiency and problem solving ability.
- Problem solving is easier to gauge in an interview
  - Sketch out a system
  - Solve a very specific technical problem
Finally - The Final

- Areas we will cover
  - Relational Design
  - MySQL
  - PHP
  - PHP + MySQL
  - SQLite
  - MongoDB