CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

DIGH 402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2014

Week 2

Databases - Data Integrity

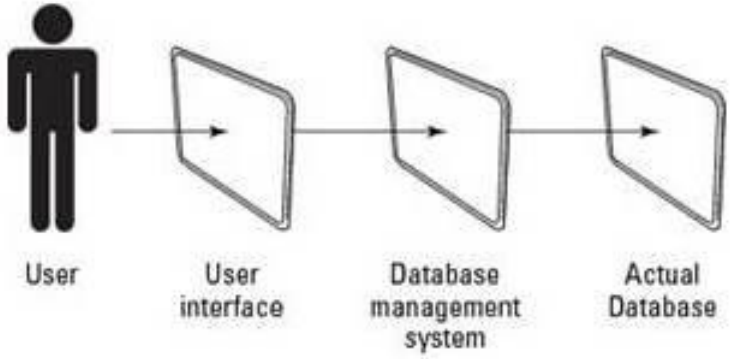
- multiple users accessing and editing a database
- ensuring data is both accurate and updated
- can cause an issue when multiple users are updating data
- data locking
- lock all data a user plans to modify
- rollback update choice to previous data state
- requires a lot of testing...

Databases - Data mining

- large chunks of data often referred to as 'data warehouses'
- data mining simply looks at separate databases to find information that's not obvious otherwise
- link databases and mine the data for combined patterns, inferences...
- data mining is useful for helping us to find hidden data in seemingly innocuous databases
- criminal profiling, political activists, genetic disposition...

Databases - Data programming

- if we want to store large amounts of data and manipulate it in different ways
- three parts of a database program to consider
    - user interface
    - database management system
    - actual information stored in a database



- UI should allow a user to use and manipulate the data without having to know how the data is stored or how to write commands to manipulate the data
- commands for manipulating data may include printing, searching, sorting...
- customised management layer with chosen programming language

Databases - SQL

- structured query language
- a way of accessing and manipulating a database

options and tools

- create a new database, rename...
- create tables in a database
- query a specified database and retrieve data
- insert, update, delete records within a database
- set privileges and permissions
- create groupings, views, orders...

there's a lot we can do with an SQL database...

Databases - SQL

generically we can consider SQL in two parts

1. Data Manipulation
    - select
    - insert into
    - update
    - delete

2. Data Definition
    - create database
    - drop database
    - create table
    - drop table
    - create index
    - drop index

and so on...

Databases - MySQL

creating DBs and tables:

- names (formally known as *identifiers*)
    - should be clear, meaningful, and ‘easy to type’ in queries etc
    - should only contain letters, numbers, and underscore (no spaces)
    - should not be the same as an existing keyword (eg: SQL term etc)
    - column names are case-insensitive (often best to use lowercase throughout)
    - no longer than 64 bytes
    - unique with DB scope (DB realm)
        eg: a table cannot have two columns with the same name
            a DB cannot have two tables with the same name

to name DB data:

- determine the DB name
- determine the table name
- determine the column name in the table

Databases - MySQL

column types:

- determine data type for a column
    - needs to be explicitly specified
    - specifies type of data that can stored in the column
- three primary types
    - Text (strings..)
    - Numbers
    - Dates and Times
- each primary type has a number of variants
    - Text (CHAR, VARCHAR, TEXT…)
    - Numbers (INT, FLOAT, DOUBLE...)
    - Dates and TImes (DATE, DATETIME, TIMESTAMP...)
- many types can take an optional ‘length’
    - eg: INT(10), VARCHAR(150)
    - excessive length will be truncated
    - lengths for numeric types determines output & not input length
- TIMESTAMP automatically set to current date and time
- other variants are available such as BLOB...

Databases - MYSQL

Choosing a column type

- decide whether column should store text, number, or date/time
    - often an obvious decision
    - date/time can be stored either specifically as a date or as text or as a number
        - further manipulation and use dependent on type

- select subtype for data per column
    - userid = int (10)
    - username = varchar(200)
    - usercreated = timestamp

- CHAR or VARCHAR?
    - CHAR always uses specified full string length
    - VARCHAR only uses space required for stored data
    - speed of fixed size CHAR vs flexibility of VARCHAR
        - char(bat)

Databases - MYSQL

Choosing any other column properties

- every column can be set to NOT NULL
    - NULL = no value

- default value for any column can also be set
    - if no default value, and no value set, NULL will be set for a new record
    - NOT NULL set, and no value set, error will occur for a new record

- UNSIGNED limits the stored data to positive numbers and zero
    - used with number types

- ZEROFILL pads extra space with zeros
    - used with number types
    - automatically UNSIGNED

Databases - MYSQL

Finishing your columns

- define and identify your PRIMARY KEY
    - nearly always a numerical value & must always have a value
    - unique way to refer to a given record
        eg: userid
    - can be defined arbitrarily
    - critical for referencing and finding a record

- identify those columns that cannot store a NULL value

- always specify a numeric type as UNSIGNED if a negative is either not necessary or illegal

- if applicable establish the default value for a column

- check and double check that your table can store the required data

Databases - MYSQL

Finishing your columns

- INDEX basically maintains a record of the values in a specified column of a given table
    - allows FULLTEXT search etc
    - improves performance when retrieving records
    - slightly reduces performance when inserting or updating records

- KEY (Primary and Foreign)
    - each table should have one primary key
    - primary key in one table often linked as a foreign key in another
        - acts as a way to reference one table to another

- AUTO_INCREMENT
    - added to a given column
    - automatically increments the column upon new insert by next highest number
    - eg: 1, 2, 3, 4 etc

etc, etc...

Databases - SQL

Example Queries

```
SELECT FirstName from Employees
SELECT FirstName from Employees WHERE FirstName = 'Emma'

INSERT INTO Employees VALUES ('Emma', 'Smith', '773-749-9246')

DELETE FROM Employees WHERE LastName = 'Smith'

UPDATE Employees SET PhoneNumber = '779-751-9248'  WHERE LastName = 'Smith'
UPDATE Employees SET PhoneNumber = '779-751-9248'  WHERE LastName = 'Smith'
AND FirstName = 'Emma'
UPDATE Employees SET PhoneNumber = '779-751-9248'  WHERE id = 223
```

- DISTINCT

```
SELECT DISTINCT FirstName from Employees
```

- SQL is not case sensitive
- NB: this is not an exhaustive list!

Databases - SQL & PHPMyAdmin

Development processes...

- PHP front-end to manage and control installed MYSQL
    - create new database
    - populate DB from scratch or import existing DB
    - create tables
    - add data per table and field
    - perform batch operations on a selected DB
    - manage users and privileges per DB
    - export DB or table for backup...

and on and on...

- effectively provides a front-end, graphical solution for MYSQL management
    - terminal/command line is standard interface for MYSQL management without a 3rd party tool such as PHPMyAdmin

Databases - PHPMYADMIN

Setup

- LAMP, XAMPP, WAMP

Raspberry Pi Setup

- install required software

    sudo apt-get update (and sudo apt-get upgrade if necessary)
    sudo apt-get install apache2 php5 mysql-client mysql-server tomcat6 vsftpd
    sudo apt-get install phpmyadmin (follow install instructions)
    cd /var/www

- add project directory, add HTML, CSS, JS, PHP etc files

- load phpmyadmin and setup mysql database

Databases - PHPMYADMIN

Create database and table using PHPMYADMIN - USERS

- open PHPMYADMIN
    - http://localhost/phpmyadmin/

- create new database with UTF-8 Unicode collation and name ‘402framework’ & storage engine will be MyISAM

- create a new table and name ‘users’
    - add 5 columns to your new table
    - add the following column names
        userid, username, firstname, lastname, usercreated

Databases - PHPMYADMIN

Create database and table using PHPMYADMIN - USERS

	Type	Collation	Attributes	Null	Default	Extra	Primary
userid	int		unsigned	No	None	auto_increment	Yes
username	varchar(30)	utf8_unicode_ci		No	None		
firstname	varchar(30)	“ ”		No	None		
lastname	varchar(50)	“ “		No	None		
usercreated	timestamp	“ “		No	CURRENT_TIMESTAMP		

Databases - PHPMYADMIN

Create database and table - USERS

- CREATE DATABASE 402framework;

- USE 402framework;

- CREATE TABLE users {
    userid INT UNSIGNED NOT NULL AUTO_INCREMENT,
    username VARCHAR(30) NOT NULL,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    usercreated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

    PRIMARY KEY (user_id)

}

- SHOW TABLES;

- SHOW COLUMNS FROM users;

Databases - PHPMYADMIN

INSERT INTO database and table - USERS

- INSERT INTO users (userid, username, firstname, lastname, usercreated) VALUES(NULL, ‘user1’, ‘yvaine’, ‘smith’, NOW());

OR

INSERT INTO users (username, firstname, lastname) VALUES(‘user1’, ‘yvaine’, ‘smith’);

Databases - PHPMYADMIN

UPDATE database and table - USERS

- UPDATE users SET username='tristanwall' WHERE userid=6;

Databases - PHPMYADMIN

SELECT FROM database and table - USERS

- SELECT * FROM users;

PHP Example