CSE 530A
Database Management Systems

Introduction

Washington University
Fall 2013
Overview

• Time: Mon/Wed 7:00-8:30 PM
• Location: Crow 206
• Instructor: Michael Plezbert
• TA: Gene Lee
• Websites:
  – [http://classes.engineering.wustl.edu/cse530/](http://classes.engineering.wustl.edu/cse530/)
  – [http://bb.wustl.edu](http://bb.wustl.edu)
# Grading

<table>
<thead>
<tr>
<th>Part</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm</td>
<td>15%</td>
</tr>
<tr>
<td>Final</td>
<td>15%</td>
</tr>
<tr>
<td>Homework (~5)</td>
<td>30%</td>
</tr>
<tr>
<td>Lab (~5)</td>
<td>35%</td>
</tr>
<tr>
<td>Class Participation</td>
<td>5%</td>
</tr>
</tbody>
</table>
What this class will cover

• Focus will be on the use of databases (though we’ll touch on theory and implementation)

• Specifically, relational databases (though we’ll touch on other options)
What this class will cover

• Database design
• SQL
• Programmatic access (ODBC, JDBC)
• Object-relational mapping (Hibernate)
• Database implementation
• Non-relational databases
Tools we will use

- PostgreSQL
- Java
- Eclipse
- SSH
- SVN
- JDBC
- Hibernate
- HTML/Javascript
Why study databases?

- Arguably most common way of storing data
- Usage spans from large scale, highly concurrent transactional systems (back-end for most business software) to single-user embedded systems (Android, iOS, Blackberry, Windows Phone, HTML5)
Ways of storing data

• Unstructured data
  – e.g. text files

• Semi-structured data
  – e.g. XML, JSON

• Structured data
  – Hierarchical, network, relational, …
What is a database?

• A *database* is a structured collection of data
• A *database management system* (DBMS) is a system for managing databases
• A *relational* database organizes data using a relational model based on first-order predicate logic
Relational databases

• A database consists of zero or more tables
  – A table is defined as having one or more fields (a.k.a. columns or attributes)
    • The number of columns is the degree (or arity) of the table
      – Also sometimes called the width, though that can also refer to the size of the columns rather than the number
  – The data is organized as rows (or records) in the table
    • The current number of rows is the cardinality
Example

<table>
<thead>
<tr>
<th>year</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1789</td>
<td>George</td>
<td>Washington</td>
</tr>
<tr>
<td>1797</td>
<td>John</td>
<td>Adams</td>
</tr>
<tr>
<td>1801</td>
<td>Thomas</td>
<td>Jefferson</td>
</tr>
<tr>
<td>1809</td>
<td>James</td>
<td>Madison</td>
</tr>
<tr>
<td>1817</td>
<td>James</td>
<td>Monroe</td>
</tr>
</tbody>
</table>
Relational databases

• Table definition is generally static
• Records are unordered
• Databases often consist of multiple tables with keys linking records between tables
• In relational theory, duplicate tuples (records) are typically not allowed
  – But most RDBMSs allow duplicates unless explicitly forbidden
• In RDBMSs, columns have specified types
SQL

- Structured Query Language
  - Developed in the early 1970s
    - Based on relational model described in 1970 by Edgar Codd but does not adhere strictly to it
  - Standardized in 1986, with several updates since
  - Declarative language with procedural elements
SQL

• Can generally be divided into two categories
• Data definition (DDL)
  – Creating and altering tables, constraints, etc.
• Data manipulation (DML)
  – Select, Insert, Update, Delete
Creating a table

```sql
CREATE TABLE presidents (  
    year integer,  
    first_name text,  
    last_name text
);
```

- Line breaks, indentation, and capitalization of key words does not matter
  - Some RDBMSs distinguish between case in table or column names, but not all

- Note: The text type is not SQL standard but is supported by many RDBMSs (such as PostgreSQL and MS SQL Server)
Destroying a table

DROP TABLE presidents;
Inserting data

```
INSERT INTO presidents (year, first_name, last_name)
VALUES (1789, 'George', 'Washington');

INSERT INTO presidents
VALUES (1797, 'John', 'Jefferson');
```

- First option allows specification of order of values or subset of fields.
- Note use of single quotes around literal string values
Updating data

UPDATE presidents SET last_name = 'Adams'
WHERE last_name = 'Jefferson';

• Updates *all* rows that match the predicate
Null and three-valued logic

• NULL indicates missing information
  – Value is *unknown*
    • NULL ≠ TRUE
    • NULL ≠ FALSE
    • NULL ≠ " (empty string)
    • NULL ≠ NULL
# Three-valued logic

<table>
<thead>
<tr>
<th>AND</th>
<th>true</th>
<th>false</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>null</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>false</td>
<td>null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>true</th>
<th>false</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>null</td>
</tr>
<tr>
<td>null</td>
<td>true</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>=</th>
<th>true</th>
<th>false</th>
<th>null</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>null</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>null</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>NOT p</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>null</td>
<td>null</td>
</tr>
</tbody>
</table>
IS NULL

INSERT INTO presidents (year, first_name)
VALUES (1801, 'Thomas');

UPDATE presidents SET last_name = 'Jefferson'
WHERE last_name IS NULL;
NOT NULL Constraint

CREATE TABLE presidents (  
    year integer NOT NULL,  
    first_name text NOT NULL,  
    last_name text NOT NULL  
);

INSERT INTO presidents (year, first_name)  
VALUES (1801, 'Thomas');

- Insert will *fail* due to violation of NOT NULL constraint
Constraints

CREATE TABLE presidents (  
    year integer CHECK (year >= 1789),  
    first_name text NOT NULL,  
    last_name text NOT NULL  
);  

• Note that check passes if true or null, so probably actually want…

CREATE TABLE presidents (  
    year integer NOT NULL CHECK (year >= 1789),  
    first_name text NOT NULL,  
    last_name text NOT NULL  
);
Queries

SELECT year, first_name, last_name FROM presidents;

SELECT * FROM presidents;

• Second option is shorthand for all columns in table-defined order

<table>
<thead>
<tr>
<th>year</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1789</td>
<td>George</td>
<td>Washington</td>
</tr>
<tr>
<td>1801</td>
<td>Thomas</td>
<td>Jefferson</td>
</tr>
<tr>
<td>1809</td>
<td>James</td>
<td>Madison</td>
</tr>
<tr>
<td>1817</td>
<td>James</td>
<td>Monroe</td>
</tr>
<tr>
<td>1797</td>
<td>John</td>
<td>Adams</td>
</tr>
</tbody>
</table>

(5 rows)

• Note arbitrary order of rows
Predicates

SELECT * FROM presidents
    WHERE first_name = 'James';

<table>
<thead>
<tr>
<th>year</th>
<th>first_name</th>
<th>last_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1809</td>
<td>James</td>
<td>Madison</td>
</tr>
<tr>
<td>1817</td>
<td>James</td>
<td>Monroe</td>
</tr>
</tbody>
</table>

(2 rows)

- Only returns rows for which the predicate is true