CS 5142
Scripting Languages
10/16/2015
Web Applications
Databases
Outline

• Databases
Why use a Database?

• Store data for months or years
  – Database may live longer than the web application(s) that you write for it

• Why not just use simple ad-hoc files?
  – Database remains consistent in the presence of multiple concurrent accesses
  – Database scales better when there is a lot of data, or a lot of accesses
  – Don’t reinvent the wheel
Relational Databases

- Database = collection of tables
- Table = relation = set of rows w/ same columns
- Row = tuple = one value per column
- Column = attribute = name+primitive type
- Only store primitive values, never nest tables

### Concepts

#### Recipe

<table>
<thead>
<tr>
<th>name</th>
<th>serves</th>
</tr>
</thead>
<tbody>
<tr>
<td>cake</td>
<td>4</td>
</tr>
<tr>
<td>gravy</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Ingredient

<table>
<thead>
<tr>
<th>name</th>
<th>taste</th>
</tr>
</thead>
<tbody>
<tr>
<td>flour</td>
<td>bland</td>
</tr>
<tr>
<td>salt</td>
<td>salty</td>
</tr>
</tbody>
</table>

#### Cuisine

<table>
<thead>
<tr>
<th>name</th>
<th>continent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italian</td>
<td>Europe</td>
</tr>
<tr>
<td>Indian</td>
<td>Asia</td>
</tr>
</tbody>
</table>

#### RecCui

<table>
<thead>
<tr>
<th>rec</th>
<th>cui</th>
</tr>
</thead>
<tbody>
<tr>
<td>daal</td>
<td>Indian</td>
</tr>
<tr>
<td>pizza</td>
<td>Italian</td>
</tr>
</tbody>
</table>

#### RecIng

<table>
<thead>
<tr>
<th>rec</th>
<th>ing</th>
<th>qty</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>cake</td>
<td>flour</td>
<td>2.5</td>
<td>cup</td>
</tr>
<tr>
<td>cake</td>
<td>milk</td>
<td>3</td>
<td>tbsp</td>
</tr>
<tr>
<td>gravy</td>
<td>salt</td>
<td>2</td>
<td>tsp</td>
</tr>
</tbody>
</table>

| gravy | sugar | 10 | g    |

CS 5142 Cornell University
10/16/13
Relational Algebra

- **Union**: Elements of R and elements of S
- **Difference**: Elements of R not in S
- **Cross Product**: Set of pairs \((r, s)\), such that \(r\) in R and \(s\) in S
- **Selection** \((\sigma)\): subset of the rows
- **Projection** \((\pi)\): subset of column attributes
- **Natural Join**: set of all combinations of tuples with common attributes
Non-Relational Data Stores

- Key-Value store:
  - *Cassandra*: distributed, tunable-consistency
  - *Memchached*: in-memory, used by Twitter, Facebook
- Document database:
  - *CouchDB*: JSON to store data, query with JavaScript
  - *MongoDB*
- Object-oriented databases:
  - *Hibernate*: ORM
  - *EJB*: Managed container (persistence + more)
From UML to Relations

- For each class, create a relation
  - Relation name = class name
  - Relation columns = class attributes
- For each association, create a relation
  - Relation name = combine class names
  - Relation columns = primary keys of both classes, plus attributes of association class, if any
UML with Associations

**Association** (binary relationship)

**Constraint on number of objects**

**Recipe**
- name PK
- serves

**Cuisine**
- name PK
- continent

**Ingredient**
- name PK
- taste

**Amount**
- qty
- unit

**PK = primary key**

**Class**

**Attribute**

**Association-class**
About SQL

• Structured Query Language
  – Query information from relational database
  – Declarative: describe what information to find, not how to find it

• SQL consists of two parts
  – DDL = Data Definition Language
  – DML = Data Manipulation Language

• Each database product (sqlite, MySQL, Oracle, DB2, …) has own SQL dialect
  – We use sqlite in this course
Most database products have additional primitive types.
How to Write + Run Code

• From PHP script
  → later in today’s lecture

• By hand, from command line

```
en-CS-CS5142:~$ sqlite3 test.db
SQLite version 3.6.20
Enter "help" for instructions
sqlite> CREATE TABLE RecIng(rec VARCHAR(50), ing VARCHAR(50), qty FLOAT, unit VARCHAR(10), PRIMARY KEY(rec, ing));
sqlite> INSERT INTO RecIng VALUES('cake','flour',2.5,'cup');
sqlite> INSERT INTO RecIng VALUES('cake','milk',3,'tbsp');
sqlite> INSERT INTO RecIng VALUES('gravy','salt',2,'tsp');
sqlite> INSERT INTO RecIng VALUES('gravy','sugar',10,'g');
sqlite> SELECT ing, qty FROM RecIng WHERE rec = 'cake';
flour|2.5
milk|3.0
sqlite>
```
CREATE TABLE Statement

CREATE TABLE RecIng(rec VARCHAR(50), ing VARCHAR(50), qty FLOAT, unit VARCHAR(10), PRIMARY KEY(rec, ing));
Insert Statement

```
INSERT INTO RecIng VALUES('cake','flour',2.5,'cup');
INSERT INTO RecIng VALUES('cake','milk',3,'tbsp');
INSERT INTO RecIng VALUES('gravy','salt',2,'tsp');
```

```
insert ::= 
  INSERT [OR conflictAlgorithm] INTO [id .] id [(id*)] VALUES(expr*)
insert ::= 
  INSERT [OR conflictAlgorithm] INTO [id .] id [(id*)] select
conflictAlgorithm ::= ROLLBACK | ABORT | FAIL | IGNORE | REPLACE
```
Select Statement

\[
\text{SELECT } [\text{ALL} \mid \text{DISTINCT}] \text{ result}^{+} \\
[\text{FROM } \text{table} (\text{joinOp} \text{ table} \text{ joinArgs})^{*}] \\
[\text{WHERE } \text{expr}] \\
[\text{GROUP BY } \text{expr}^{+}] \\
[\text{HAVING } \text{expr}] \\
(\text{compoundOp select})^{*} \\
[\text{ORDER BY } (\text{expr} [\text{sortOrder}])^{*} \\
[\text{LIMIT } \text{integer} [(\text{OFFSET} \mid ,) \text{ integer}] ]
\]

result ::= * | \text{tableName} .* | \text{expr} [[\text{AS}] \text{id}] 

\text{table} ::= \text{tableName} [\text{AS} \text{alias}] \mid (\text{select}) [\text{AS} \text{alias}] 

\text{joinOp} ::= , | [\text{NATURAL}] [\text{LEFT} \mid \text{RIGHT} \mid \text{FULL}] [\text{OUTER} \mid \text{INNER} \mid \text{CROSS}] \text{ JOIN} \text{table} \text{ joinArgs} ::= [\text{ON} \text{expr}] [\text{USING} (\text{id}^{+})] 

\text{sortOrder} ::= [\text{COLLATE} \text{collationName}] [\text{ASC} \mid \text{DESC}] 

\text{compoundOp} ::= \text{UNION} | \text{UNION ALL} | \text{INTERSECT} | \text{EXCEPT}

\text{Example:}
\begin{verbatim}
SELECT ing, qty 
FROM RecIng 
WHERE rec = 'cake';
\end{verbatim}
## List of sqlite Statements

### Data Definition Language (DDL)
- `(CREATE | ALTER | DROP) TABLE`
- `(CREATE | DROP) INDEX`
- `(CREATE | DROP) TRIGGER`
- `(CREATE | DROP) VIEW`
- `CREATE VIRTUAL TABLE`
- `ATTACH DATABASE`
- `DETACH DATABASE`
- `ANALYZE`
- `REINDEX`
- `VACUUM`

### Data Manipulation Language (DML)
- `INSERT`
- `SELECT`
- `UPDATE`
- `REPLACE`
- `DELETE`
- `BEGIN TRANSACTION`
- `COMMIT TRANSACTION`
- `ROLLBACK TRANSACTION`
- `END TRANSACTION`
- `EXPLAIN`
- `PRAGMA`
Lexical Peculiarities

- Case insensitive
- Commands end with semicolon (;)
- Single-line comments: --...
- Multi-line comment: /*...*/
- String literal: 's'
  - Escape single quote (') in string with another single quote ("'"), not with backslash (\')
- Identifier: simple id or quoted "id"
  - Quoted identifier can contain any character
  - Quoted identifier can be same as keyword
## Operators

<table>
<thead>
<tr>
<th>_OPERATOR</th>
<th>PRIORITY</th>
<th>CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>-,-, +, ~, NOT</td>
<td>1</td>
<td>Negation</td>
</tr>
<tr>
<td>CASE [expr] (WHEN expr THEN expr)+ [ELSE expr] END</td>
<td>1</td>
<td>Conditional</td>
</tr>
<tr>
<td>CAST (expr AS type)</td>
<td>1</td>
<td>Conversion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*, /, %</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>+, -</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>&lt;&lt;, &gt;&gt;, &amp;</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>&lt;, &lt;=, &gt;, &gt;=</td>
<td>2</td>
<td>Comparison</td>
</tr>
<tr>
<td>=, ==, ! =, &lt;&gt;, IN</td>
<td>2</td>
<td>Identity</td>
</tr>
<tr>
<td>expr [NOT](LIKE</td>
<td>GLOB</td>
<td>REGEXP</td>
</tr>
<tr>
<td>expr [NOT] BETWEEN expr AND expr</td>
<td>3</td>
<td>Comparison</td>
</tr>
<tr>
<td>AND, OR</td>
<td>2</td>
<td>Logic</td>
</tr>
<tr>
<td><a href="select">EXISTS</a></td>
<td>1</td>
<td>Query-in-expr</td>
</tr>
</tbody>
</table>
Library Functions

- **Aggregate**: avg, count, group_concat, max, min, sum, total
- **String**: glob, length, like, lower, ltrim, quote, replace, rtrim, soundex, substr, trim, upper
- **Number**: abs, max, min, random, round
- **Misc**: coalesce, ifnull, hex, last_insert_rowid, load_extension, nullif, randomblob, sqlite_version, typeof, zeroblob
- **Date+time**: date, time, datetime, julianday, strftime
SQL Documentation

- Take a class on databases
- Read a standard databases text book
- sqlite: http://www.sqlite.org
- MySQL: http://www.mysql.com
- Tutorial: http://www.w3schools.com/sql/default.asp
• Today’s lecture
  – Databases
  – SQL

• Next lecture
  – Session state
  – Form validation
  – AJAX