Final Review #1

Wednesday, April 29, 2015
Agenda

• Next two classes: Final Review
• Today: Makeup Quiz

• Reminder: Course evaluations
The Final: Logistics

• Date: Wednesday, May 6, 2015
• Time: 5:00 - 6:30
• Where: In class
• Open book exam :)
• No computers :(
The Final: Content

5 Problems with Sub-Problems:
1. Queries
2. Data modeling
3. Transactions
4. Query processing
5. NoSQL systems
General Advice

• Some problems will require thinking
  – Use judgment
• Problem difficulty may be uneven:
  – do the easy ones first
Problem 1: Queries

- SQL
- MongoDB query language
SQL

- select-from-where
- order by and renamings
- joins
- group-by and having
- aggregations
- views
- insert, update, delete

Interesting question: *does one query return a subset of another?*
SQL Practice Problem #1

Find all students in the database:

```
SELECT *
FROM students
WHERE gpa >= 3.5 OR gpa < 3.5
```

Question: what’s wrong?
SQL Practice Problem #2

Products(product_id, product_name)
Sales(sale_id, product_id, quarter, year, quantity, price)

Find total sales by product for Q1 of this year:
SELECT p.product_name, SUM(s.quantity*s.price) AS total_sales
FROM products p, sales s
WHERE p.product_id = s.product_id
AND s.quarter = 1
AND s.year = 2015
GROUP BY product_name

Question: what’s wrong?
SQL Practice Problem #3

Old Schema:
ProductRequests(customer_id, customer_name, product_id, request_date)

New Schema:
Products(id, name, color, weight, number_available)
Customers(id, name, age, address, city)
Requests(customer_id, product_id, date)

Question: create ProductRequests view over new schema
MongoDB

- find and findOne
- comparison operators: $lt, $lte, $gt, and $gte
- logical operators: $in, $or, $and
- arrays: $all
- embedded documents: dot notation
- insert, update (with upsert), remove

An interesting question: write query to transform a JSON document
Problem 2: Data Modeling

- E/R diagrams
- Normal forms
- JSON and semi-structured data
E/R Diagrams

- Entities, attributes
- Relationships
- Inheritance
- Translation to relations
- SQL DDL:
  - Creating tables
  - Constraints

An interesting question: *translate an inheritance graph to relations*
Normal Forms

• Data anomalies
• Functional dependencies
• 1NF, 2NF and 3NF definitions
• Checking if a relation is in 3NF
• Decomposing into 3NF

An interesting question: does a FD hold on a table?
Revisiting Normal Forms
Recall: Functional Dependencies

Definition:

If two tuples agree on the attributes

\[ A_1, A_2, \ldots, A_n \]

then they must also agree on the attributes

\[ B_1, B_2, \ldots, B_n \]

Formally:

\[ A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_n \]
### Unnormalized to 1NF

Rule: A database schema is in 1NF *iff* all attributes have scalar values.

#### Students

<table>
<thead>
<tr>
<th>Student</th>
<th>Semester</th>
<th>GPA</th>
<th>Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>3.9</td>
<td>Math, DB, Alg</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>3.7</td>
<td>DB, Alg</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>3.5</td>
<td>Math, Alg</td>
</tr>
</tbody>
</table>

#### Students’ (1NF)

<table>
<thead>
<tr>
<th>Student</th>
<th>Semester</th>
<th>GPA</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>3.9</td>
<td>Math</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>3.9</td>
<td>DB</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>3.9</td>
<td>Alg</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>3.7</td>
<td>DB</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>3.7</td>
<td>Alg</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>3.5</td>
<td>Math</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>3.5</td>
<td>Alg</td>
</tr>
</tbody>
</table>

unnormalized vs 1NF
1NF to 2NF

Rule: A database schema is in 2NF iff it is in 1NF and there are no partial FDs on the primary key (i.e. all non-key attributes must be dependent on the entire PK)

Assumptions:
1. Student, Semester \rightarrow GPA
2. GPA is not functionally determined by course

<table>
<thead>
<tr>
<th>Students</th>
<th></th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Semester</td>
<td>Course</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>Math</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>DB</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>Alg</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>DB</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>Alg</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>Math</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>Alg</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enrolls</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Course</td>
<td>Semester</td>
</tr>
<tr>
<td>Alice</td>
<td>Math</td>
<td>Spring15</td>
</tr>
<tr>
<td>Alice</td>
<td>DB</td>
<td>Spring15</td>
</tr>
<tr>
<td>Alice</td>
<td>Alg</td>
<td>Spring15</td>
</tr>
<tr>
<td>Bob</td>
<td>DB</td>
<td>Spring15</td>
</tr>
<tr>
<td>Bob</td>
<td>Alg</td>
<td>Spring15</td>
</tr>
<tr>
<td>Carol</td>
<td>Math</td>
<td>Spring15</td>
</tr>
<tr>
<td>Carol</td>
<td>Alg</td>
<td>Spring15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GPA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Semester</td>
<td>GPA</td>
</tr>
<tr>
<td>Alice</td>
<td>Spring15</td>
<td>3.9</td>
</tr>
<tr>
<td>Bob</td>
<td>Spring15</td>
<td>3.7</td>
</tr>
<tr>
<td>Carol</td>
<td>Spring15</td>
<td>3.5</td>
</tr>
</tbody>
</table>
2NF to 3NF

Rule: A database schema is in 3NF iff it is in 2NF and there are no transitive dependencies.

**Assumptions:**

- EID → Name, Major
- Major → College
- By transitivity, EID → College

**Students**

<table>
<thead>
<tr>
<th>EID</th>
<th>Name</th>
<th>Major</th>
<th>College</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Math</td>
<td>Natural Sciences</td>
</tr>
<tr>
<td>200</td>
<td>Bob</td>
<td>CS</td>
<td>Natural Sciences</td>
</tr>
<tr>
<td>300</td>
<td>Carol</td>
<td>Finance</td>
<td>Business</td>
</tr>
</tbody>
</table>

**Students’**

<table>
<thead>
<tr>
<th>EID</th>
<th>Name</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Alice</td>
<td>Math</td>
</tr>
<tr>
<td>200</td>
<td>Bob</td>
<td>CS</td>
</tr>
<tr>
<td>300</td>
<td>Carol</td>
<td>Finance</td>
</tr>
</tbody>
</table>

**Majors**

<table>
<thead>
<tr>
<th>Major</th>
<th>College</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Natural Sciences</td>
</tr>
<tr>
<td>CS</td>
<td>Natural Sciences</td>
</tr>
<tr>
<td>Finance</td>
<td>Business</td>
</tr>
</tbody>
</table>
JSON

- JSON syntax
- From relations to JSON
- From JSON to relations

An interesting question: N/A
Problem 3: Transactions

• Data inconsistencies
• Concurrency control
• Distributed transaction processing
Data inconsistencies

- Dirty reads
- Non-repeatable reads
- Phantom reads

An interesting question: *find inconsistencies in a schedule*
Concurrency Control

• Serializability
• Repeatable Read
• Read Committed
• Read Uncommitted

An interesting question: explain what happens in a schedule
Distributed Transactions

• 2 phase commit
• “Eventual” consistency

An interesting question: *identify conflicting vector clocks*
Problem 4: Query Processing

- Query execution without indexes
- Query execution with indexes
- Types of indexes:
  - clustered index
  - unclustered index
- B⁺ trees

An interesting question: select index based on SQL queries
Problem 5: NoSQL Systems

• Data systems landscape
• MapReduce
• MongoDB
• Replication and “sharding”

An interesting question: convert a SQL query to MapReduce
COMMIT
(The End)
Make-up Quiz

Q1: What is the difference between horizontal and vertical partitioning?

Q2: When would you use a virtual view as opposed to a materialized view and why?

Q3: List out what ACID stands for and explain two of them

Q4: List one data model that is used by NoSQL systems