Introduction to Data Management
CSE 344

Lecture 2: Data Models
(Ch. 2.1-2.3)
Announcements

• WQ1 and HW1 are out
  – Make sure you have signed up for Gradiance. Check Spam Folder for Signup Email!

• Use Piazza to post questions

• If you have a laptop, bring to section Th
  – also look at HW1 for installing sqlite
  – can go through the examples yourself
Announcements (cont.)

• Let us know if you have suggestions for the class
  – Don’t wait till course evals!

• Feedback to speak slower and louder

• Slides will be available before lecture (12pm)
  – We can bring hardcopies if needed
Class Overview

• **Relational Data Model**
  – SQL, Relational Algebra, Relational Calculus, datalog
  – Query processing and optimization

• **Semistructured Data Model**
  – JSON (NoSQL)

• **Conceptual design**
  – E/R diagrams, Views, and Database normalization

• **Transactions and their implementations**

• **Parallel databases**
  – MapReduce, and Spark
Today

- Data models
- Relational data model
- Basic SQL statements
Review

• What is a database?
  – A collection of files storing related data

• What is a DBMS?
  – An application program that allows us to manage efficiently the collection of data files
Data Models

- Suppose we have book data: author, title, publisher, pub date, price, etc
  - How should we organize such data in files?

Data model: a general, conceptual way of structuring data
What Is A Data Models

• language / notation for talking about data

• models we will use:
  – relational: data is a collection of tables
  – semi-structured: data is a tree

• other models:
  – key-value pairs: used by NoSQL systems
  – graph data model: used by RDF (semi-structured can also do)
  – object oriented: often layered on relational, J2EE
3 Elements of Data Models

• Instance
  – The actual data

• Schema
  – Describe what data is being stored

• Query language
  – How data can be retrieved and manipulated

Each Data Model Does These Differently
The Relational Data Model

• Instance
  – Organized as “table” or “relation”
  – Consists of
    • “column” aka “attribute” aka “field”
    • “row” aka “tuple” aka “record”

• Schema
  – “table name” aka “relation name”
  – “column name” aka “attribute name”
  – Each attribute has a “type” aka “domain” aka “data type”
Relational Model

• Data is a collection of relations / tables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Employees</th>
<th>For_Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
<td>20000</td>
<td>True</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>True</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>True</td>
</tr>
<tr>
<td>HappyCam</td>
<td>Canada</td>
<td>500</td>
<td>False</td>
</tr>
</tbody>
</table>

• mathematically, relation is a set of tuples
  – each tuple appears 0 or 1 times in the table
  – order of the rows is unspecified
The Relational Data Model

• “degree” or “arity” of a relation
  – Number of attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Employees</th>
<th>For_Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>GizmoWorks</td>
<td>USA</td>
<td>20000</td>
<td>True</td>
</tr>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>True</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>True</td>
</tr>
<tr>
<td>HappyCam</td>
<td>Canada</td>
<td>500</td>
<td>False</td>
</tr>
</tbody>
</table>

What is the arity of our relation?
The Relational Data Model

- Attributes must have a data type:
  - Strings: CHAR(20), VARCHAR(50), TEXT
  - Numbers: INT, SMALLINT, FLOAT
  - MONEY, DATETIME, …
  - Usually vendor specific
  - Statically and strictly enforced

In SQLite these are column “affinities”
Keys

THE ANY KEY
Keys

- An attribute that uniquely identifies a record.

- Is City Name A Key?
  - Seattle, Bellevue, Kathmandu, Ouagadougou

There are 41 cities in the US names Springfield
Keys

There are 41 cities in the US names Springfield

• What to do about it?

• A key can consist of multiple attributes
  – What does that mean?
  – (Springfield, ME), (Springfield, MA),
    (Springfield, MI), (Springfield, MN),
    (Springfield, M) ….

But Wisconsin has 5 Springfields!
Keys (cont.)

- A relation can have many keys
  - But only one of them can be chosen to be the *primary key*
  - DBMS often makes searches by primary key fastest
  - other keys are called “secondary”
Relation Schema

• Names and types form part of the table “schema”:

  Company(cname, country, no_employees, for_profit)

  Company(cname: varchar(30), country: char(20),
          no_employees: int, for_profit: char(1))

• What is a good primary key for Company?
Relation Schema

- Names and types form part of the table “schema”:
  Company(cname, country, no_employees, for_profit)

  Company(cname: varchar(30), country: char(20),
          no_employees: int, for_profit: char(1))

- Instance

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>
Query Language

• SQL ("sequel")
  – Structured Query Language
  – Developed by IBM in the 70s

• Declarative language

Declarative Language: Programming paradigm that expresses the logic of computation without describing its control flow
SQL is like Wiki Syntax

There are multiple standards and in practice each DBMS implements its own.

<table>
<thead>
<tr>
<th>Language</th>
<th>HTML export tool</th>
<th>HTML import tool</th>
<th>Tables</th>
<th>Link titles</th>
<th>class attribute</th>
<th>id attribute</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsciiDoc</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>November 25, 2002¹¹</td>
</tr>
<tr>
<td>BBCode</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>1996²²[better source needed]</td>
</tr>
<tr>
<td>Creole</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>July 4, 2007³³</td>
</tr>
<tr>
<td>GitHub Flavored Markdown</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Markdown</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes/No</td>
<td>Yes</td>
<td>Yes/No</td>
<td>Yes/No</td>
<td>March 19, 2004⁴⁴⁵⁵</td>
</tr>
<tr>
<td>Markdown Extra</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes[⁶]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>MediaWiki</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>2002⁷⁷</td>
</tr>
<tr>
<td>MultiMarkdown</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>?</td>
</tr>
<tr>
<td>Org-mode</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>November 19, 2016⁸⁹</td>
</tr>
<tr>
<td>PmWiki</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>POD</td>
<td>Yes</td>
<td>?</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>reStructuredText</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>auto</td>
<td>April 2, 2002¹⁰</td>
</tr>
<tr>
<td>Textile</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>December 26, 2002¹¹</td>
</tr>
<tr>
<td>Texy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>2004¹²</td>
</tr>
<tr>
<td>txt2tags</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes/No</td>
<td>Yes/No</td>
<td>July 26, 2001¹⁵</td>
</tr>
</tbody>
</table>
SQL is like Wiki Syntax

https://xkcd.com/927/

How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc)

<table>
<thead>
<tr>
<th>Situation:</th>
<th>There are 14 competing standards.</th>
</tr>
</thead>
<tbody>
<tr>
<td>14?! RIDICULOUS! We need to develop one universal standard that covers everyone's use cases.</td>
<td>YEAH!</td>
</tr>
<tr>
<td>Soon:</td>
<td>Situation: There are 15 competing standards.</td>
</tr>
</tbody>
</table>

**SQL:1999** It introduced a large number of new features, many of which required clarifications in the subsequent SQL:2003.
Our First DBMS

• SQL Lite
• Will switch to SQL Server later in the quarter
SQL statements

• create table ...
• drop table ...
• alter table ... add/remove ...
• insert into ... values ...
• delete from ... where ...
• update ... set ... where ...

See: http://www.sqlite.org/lang.html for details
Demo
Discussion

• Tables are FLAT
  – No nested attributes

• Tables DO NOT prescribe how they are implemented / stored on disk
  – This is called **physical data independence**

**Physical data independence**
The logical definition of the data remains unchanged, even when we make changes to the actual implementation
Adding Attributes

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

- Let’s add a list of product that each company produces
  - How? Recall that tables are flat!
Foreign Keys

- A column (or columns) whose value is a key of another table
  - i.e., a reference to another row in another table

<table>
<thead>
<tr>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

Product(pname, price, category,
       manufacturer_name REFERENCES Company.cname,
       manufacture_country REFERENCES Company.country
)
## Best Practice: Use Integer Primary Key

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>country</th>
<th>no_employees</th>
<th>for_profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Canon</td>
<td>Japan</td>
<td>50000</td>
<td>Y</td>
</tr>
<tr>
<td>2</td>
<td>Hitachi</td>
<td>Japan</td>
<td>30000</td>
<td>Y</td>
</tr>
</tbody>
</table>

Product(pid, pname, price, category, manufacturer Foreign Key Company.cid)

<table>
<thead>
<tr>
<th>pid</th>
<th>pname</th>
<th>price</th>
<th>category</th>
<th>manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SingleTouch</td>
<td>149.99</td>
<td>photography</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>AC</td>
<td>300</td>
<td>Appliance</td>
<td>2</td>
</tr>
</tbody>
</table>
Demo