Introduction to Data Management
CSE 344

Lecture 2: Data Models
Announcements

• WQ1 and HW1 are out
  – Use your CSE ids to access the HW docs
• Use Piazza to post questions
• OHs are up on website
  – Check for HW specific TA OHs

• In-class note-taking policy
Class Overview

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

- **Semistructured Data Model**
  - JSON, CouchDB (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
- SQL
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
Data Models

• Relational
  – Data represented as relations
• Semi-structured (JSon)
  – Data represented as trees
• Key-value pairs
  – Used by NoSQL systems
• Graph
• Object-oriented
• We will study the first two in 344
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
Turing Awards in Data Management

Charles Bachman, 1973
*IDS and CODASYL*

Ted Codd, 1981
*Relational model*

Michael Stonebraker, 2014
*INGRES and Postgres*
The Relational Data Model

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

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

- "degree" or "arity" of a relation
  - Number of attributes

- Example types:
  - Strings: CHAR(20), VARCHAR(50), TEXT
  - Numbers: INT, SMALLINT, FLOAT
  - MONEY, DATETIME, ...
  - Usually vendor specific
  - Statically and strictly enforced
Keys

• An attribute that uniquely identifies a record
  – Example?

• A key can consist of multiple attributes
  – What does that mean?
Keys

• A relation can have many keys
  – But only one of them can be chosen to be the primary key

• Foreign key:
  – An attribute(s) that is a key for other relations
Relation Model: Example

- **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>

- **Schema**

  Company(cname, country, no_employees, for_profit)

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

Company\(\text{cname} , \text{country}, \text{no\_employees}, \text{for\_profit}\)

Country\(\text{name} , \text{population}\)

<table>
<thead>
<tr>
<th>\text{cname}</th>
<th>\text{country}</th>
<th>\text{no_employees}</th>
<th>\text{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>

<table>
<thead>
<tr>
<th>\text{name}</th>
<th>\text{population}</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>320M</td>
</tr>
<tr>
<td>Japan</td>
<td>127M</td>
</tr>
</tbody>
</table>
Semi-Structured Model: Example

Company(cname, country, no_employees, for_profit)

Country(name, population)
Query Language

• SQL
  – Structured Query Language
  – Developed by IBM in the 70s
  – Most widely used language to query relational data

• We will see other languages for the relational model later on
  – Relational algebra, relational calculus, etc.
Our First DBMS

- SQL Lite
- Will switch to SQL Server later in the quarter
Demo
Discussion

• Tables are NOT ordered
  – they are sets or multisets (bags)
• Tables are FLAT
  – No nested attributes
• Tables DO NOT prescribe how they are implemented / stored on disk
  – This is called **physical data independence**
Table Implementation

• How would you implement this?

<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>

• What happens when you alter a table?

**Physical data independence**

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

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

<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>
## 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>

Product(pname, price, category, manufacturer)

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