Practice and Applications of Data Management
CMPSCI 345

Lecture 02: Data Models
Interesting stuff about databases

- It used to be about boring stuff: employee records, bank records, etc.

- Today, the field covers all the largest sources of data, with many new ideas.
  - Web search.
  - Data mining.
  - Scientific and medical databases.
  - Integrating information.
Interesting stuff about databases

- You may not notice it, but databases are behind almost everything you do on the Web.
  - Google searches.
  - Queries at Amazon, eBay, etc.

- Databases make sure that
  - Multiple users can access the same data without creating confusion or inconsistencies
  - Important data isn’t lost after a crash
What is a data model?

- Mathematical representation of data.
  - relational model = tables
  - semi-structured model = trees/graphs.
  - key-value pairs: noSQL systems

- Operations on data.

- Constraints.
Relational databases

- Relational Database = a set of relations

- Relation:
  - Schema: name of the relation, name and type of each column
  - Instance: a table, with rows and columns

- Restriction:
  - All attributes are atomic, no nested tables

- Database Schema:
  - Collection of relation schemas
Relational instance: tables

**Arity** (number of attributes) is 3

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234545</td>
<td>John</td>
<td>20000</td>
</tr>
<tr>
<td>5423341</td>
<td>Smith</td>
<td>60000</td>
</tr>
<tr>
<td>4352342</td>
<td>Fred</td>
<td>50000</td>
</tr>
</tbody>
</table>

A relation is a set of tuples: no tuple can occur more than once - Not true in practice, real systems allow duplicates
Why relations?

- Very simple model.
- *Often* matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.
Database schemas in SQL

- SQL is primarily a query language, for getting information from a database.

- But SQL also includes a *data-definition* component for describing database schemas.
Creating (declaring) a relation

- Simplest form is:
  
  CREATE TABLE <name> (  
  <list of elements>
  );

- To delete a relation:
  
  DROP TABLE <name>;}
Elements of table declarations

- Most basic element: an attribute and its type.

- Types are vendor-specific
  - Character strings: CHAR(20), VARCHAR(50), TEXT
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATE, TIME

- Types are static and strictly enforced
  - Exception: SQLite has dynamic types
Practice

- Start a new database on SQLite:
  >sqlite3 simpleDB

- Practice the CREATE TABLE command
  E.g.: CREATE TABLE Company(id, name);

- Insert a few tuples:
  E.g.: INSERT INTO Company VALUES (1, 'SONY');

- You can also drop your table:
  E.g.: DROP TABLE Company;
How to check your work

- Execute:
  - SELECT * FROM Company;

- What do you see?

- Improve the format (SQLite-specific):
  - .header on
  - .mode column
  - .nullvalue NULL

or whatever table name you used
Comment on syntax

- COMPANY = Company = company

- Company(name, country)
- Person(name, country)

 repeated attributes are ok
NULL values

- Whenever we don’t know the value, we can set it to NULL

- E.g.: INSERT INTO Company VALUES (NULL, 'IBM');
Declaring keys

- An attribute or list of attributes may be declared PRIMARY KEY or UNIQUE.

- Either says that no two tuples of the relation may agree in all the attribute(s) on the list.

- There are a few distinctions to be mentioned later.
Single-attribute keys

- Place PRIMARY KEY or UNIQUE after the type in the declaration of the attribute.

Example:

```sql
CREATE TABLE Company
    (cname VARCHAR(20) PRIMARY KEY,
     country VARCHAR(20),
     no_employees int,
     for_profit CHAR(1));
```
Practice primary keys

- Populate the table:
  - `insert into Company  values ('GizmoWorks', 'USA', 20000,'y');`
  - `insert into Company  values ('Canon', 'Japan', 50000,'y');`
  - `insert into Company  values ('Hitachi', 'Japan', 30000,'y');`
  - `insert into Company  values('Charity', 'Canada', 500,'n');`

- How can you test the primary key constraint?
  - E.g.: `insert into Company  values ('Canon', 'USA', 2000,'n');`
PRIMARY KEY vs. UNIQUE

- There can be only one PRIMARY KEY for a relation, but several UNIQUE attributes.

- No attribute of a PRIMARY KEY can ever be NULL in any tuple.

- Attributes declared UNIQUE may have NULLs, and there may be several tuples with NULL.
Practice primary keys

Try:

- insert into Company values (NULL, 'Somewhere', 0, 'n');

What happened?

oops... SQLite allows keys to be NULL!!!
Deleting tuples

- Delete the bad tuple:
  
  ```sql
  DELETE FROM Company WHERE country='Somewhere';
  ```

- How can you delete all the companies with less than 25000 employees?
  
  ```sql
  DELETE FROM Company WHERE no_employees < 25000;
  ```

- What does this command do?
  
  ```sql
  DELETE FROM Company;
  ```
Multi-attribute keys

Company(cname, country, no_employees, for_profit)

- What if we want a separate record for each country-branch of each company?
  - CREATE TABLE Company
    (cname VARCHAR(20),
     country VARCHAR(20),
     no_employees int,
     for_profit CHAR(1),
     PRIMARY KEY(cname, country));
Discussion

- Tables are not ordered. They represent sets or bags.
- Tables do not describe how they are implemented.
  - Physical data independence
- Tables are flat.

- How would you implement a table?
Discussion

- Why is physical data independence important?
Altering a table

ALTER TABLE Company ADD ceo VARCHAR(20);

UPDATE Company
SET ceo='Mitarai'
WHERE cname = 'Canon';

Can you ALTER the Company table to add the products they manufacture?
Multiple tables and keys

- Create a separate table:
  - CREATE TABLE Product
    (pname VARCHAR(20) PRIMARY KEY,
     price FLOAT,
     category VARCHAR(20),
     manufacturer varchar(20) references Company);

- Enable foreign keys in SQLite:
  PRAGMA foreign_keys=ON;