Database Management Systems
CS 457/657

Lecture 2: Data Model, SQL
Data Models

• What are data models?
  – A data model is a general, conceptual way of structuring data
  – Think about it…

• Data models studied in this course:
  – Relational data model (data->relation)
  – Semi-structured data model (XML)
Data Models

• Other popular models:
  – Key-value stores (e.g., NoSQL)
  – Graph data model
  – Object-oriented

• A data model describes both
  – The data
  – And the query language
Data Models

• Schema vs. Instance
  – Schema: the structure of a particular database under a certain data model
  – Instance: the actual data
The Relational Data Model

• Database instance:
  – "table" or "relation"
  – "column" or "attribute" or "field"
  – "row" or "tuple" or "record"

• Cardinality of relation instance: number of tuples
The Relational Data Model

• Database schema
  – "table name" or "relation name"
  – "column name" or "attribute name"
  – each attribute has a "type"

• Degree (or arity) of relation: number of attributes
The Relational Data Model

• Types or datatypes
  – Character strings: CHAR(20), VARCHAR(50), TEXT
  – Numbers: INT, BIGINT, SMALLINT, FLOAT
  – Others: MONEY, DATETIME

• Types are vendor specific
Database Keys

• An attribute is called a "key" if it uniquely identifies a record
• We can have a key with multiple attributes: each unique combination of values for those attributes uniquely determines the record
• Candidate key: Minimal set of fields that uniquely identify each tuple in a relation
Database Keys

• Primary key
  – one candidate key can be selected as primary key

• Foreign keys
  – other tuples may use key values as "logical pointers"
Examples

• Without types
  – Company(cname, country, no_employees, for_profit)

• With types
  – Company(cname: varchar(30), country: char(20), no_employees:int, for_profit:char(1))

• Other example
  – Movie(id int, name varchar(30), producer varchar(30))
SQL

- We will use SQLite in class
- Making sure SQL Lite shows us the data in a nicer format (These commands are specific to SQLite!)
  - .header on
  - .mode column
  - .nullvalue NULL
**SQL: creating tables**

```sql
create table Company
    (cname varchar(20) primary key,
     country varchar(20),
     no_employees int,
     for_profit char(1));

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');

select * from Company;
```
upper/lower case; name conflicts

• Company, company, COMPANY
  – all the same

• Company(cname, country), Person(pname, country)
  – repeated 'country’ OK

• Company(cname, country), Person(pname, company)
  – the attribute 'company' not ok
Null values

• whenever we don't know the value, we can set it to NULL

```sql
insert into Company values('MobileWorks', 'China', null, null);
select * from Company;
```
Deleting tuples

delete from Company where cname = 'Hitachi';
select * from Company;
More about tables

• NOT ordered
  – They represent sets or bags

• NOT prescribe how they should be implemented
  – PHYSICAL DATA INDEPENDENCE!

• FLAT
  – all attributes are base types
Discussion

• how would you implement a table?
  – row oriented
  – column oriented
  – vertically partitioned
  – horizontally partitioned
Discussion

• Why is physical data independence important?
  – we can optimize the data layout on disk for performance without breaking the applications written on top of the database
Altering a table in SQL

```
ALTER TABLE Company ADD ceo varchar(20);
select * from Company;

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

SELECT * FROM Company;
```
Multiple Tables, Foreign Keys

```
create table Product
    (pname varchar(20) primary key,
    price float,
    category varchar(20),
    manufacturer varchar(20) references Company);
```
constraint Violation

PRAGMA foreign_keys=ON;

insert into Product values('Gizmo', 19.99, 'gadget', 'GizmoWorks');
insert into Product values('PowerGizmo', 29.99, 'gadget', 'GizmoWorks');
insert into Product values('SingleTouch', 149.99, 'photography', 'Canon');
insert into Product values('MultiTouch', 199.99, 'photography', 'MobileWorks');
insert into Product values('SuperGizmo', 49.99, 'gadget', 'MobileWorks');

select * from Product;

-- If we try:
insert into Product values('MultiTouch2', 199.99, 'photography', 'H2');
-- We should get an error if foreign keys got enforced
-- Error: foreign key constraint failed
Basic SQL

- Range query
  
  ```sql
  select *
  from Product
  where price > 100.0;
  ```

- Projection
  
  ```sql
  select price, category
  from Product;
  ```