ICCS240 Database Management

SQL

Many slides in this lecture are either from or adapted from slides provided by

Theodoros Rekatsinas, UW-Madison

Abdu Alawini. UIUC

Why SQL?

The relational model of data is the most widely used model today

Main Concept: the *relation* - essentially, a table

Remember: The reason for using the relational model is data independence!

Logical data independence:
protection from changes in the *logical structure of the data*

SQL is a logical, declarative query language.
We use SQL because we happen to use the relational model.

What is Structured Query Language (SQL)?

SQL is a standard language for querying and manipulating data in RDBMS

It is an *attempt* to implement relational algebra as a query language

SQL is a **very high-level** programming language

Probably the world's most successful **parallel** programming language

Interlude: Database Languages

- Data Definition Language (DDL)
  - Syntax for defining the schema of a database

    `CREATE, ALTER, DROP`
- Data Manipulation Language (DML)
  - Syntax to manipulate data in database:

    `INSERT, DELETE, MODIFY`
  - Read-only selecting of data.

    `SELECT` … **`FROM`** … **`WHERE`** …
- Data Control Language (DCL)
  - Grant / revoke permissions on databases and their contents
- Transaction Control Language (TCL)
  - Manage transaction.

SQL is a …

DDL,

DML,

…

Basic SQL

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema

List: [1, 1, 2, 3]
Set: {1, 2, 3}
Multiset: {1, 1, 2, 3}

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

Attributes must have an **atomic** type in standard SQL, i.e. not a list, set, etc.

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **tuple** or **row** is a single entry in the table having the attributes specified by the schema

*Also referred to sometimes as a **record***

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

Data Types in SQL

- Atomic types:
  - Characters: CHAR(20), VARCHAR(50), …
  - Numbers: INT, BIGINT, FLOAT, …
  - Others: DATETIME, …

- Every attribute must have an atomic type
  - Therefore, tables are *flat*.

Table Schema

- The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Create Table Construct

An SQL relation is defined using the **create table** command:

```
CREATE TABLE R (A1 D1, A2 D2, …, An Dn,
                (integrity-constraint_1),
                ...,
                (integrity-constraint_k));
```

where

- `R` is the name of the relation
- each `Ai` is an attribute name in the schema of relation `R`
- `Di` is the data type of values in the domain of attribute `Ai`

Example:

```
create table instructor (
        ID              char(5),
        name            varchar(20),
        dept_name       varchar(20),
        salary          numeric(8,2));
```

Key Constraints

A key is a (**minimal**) **subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation
  - A **key** is an attribute whose values are unique.
  - i.e. if two tuples agree on the values of the key, then they must be the same tuple!

Students(sid:string, name:string, gpa: float)

1. Which would you select as a key?
2. Is a key always guaranteed to exist?
3. Can we have more than one key?

Integrity Constraints in CREATE Table

- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** R

Example:

```
create table instructor (
        ID              char(5),
        name            varchar(20),
        dept_name       varchar(20),
        salary          numeric(8,2),
        PRIMARY KEY (ID),
        FOREIGN KEY (dept_name) REFERENCES department(name)));
```

primary key declaration on an attribute automatically ensures **not null**

NULL

- To say “don’t know the value” we use NULL
  - NULL has (sometimes painful) semantics, more details later

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

Students(sid:string, name:string, gpa: float)

Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be **NOT NULL**, e.g., “name” in this table

And a few more relation definitions

```
create table student (
        ID              varchar(5),
        name            varchar(20) not null,
        dept_name       varchar(20),
        tot_cred        numeric(3,0),
        primary key (ID),
        foreign key (dept_name) references department(name));

create table takes (
        ID              varchar(5),
        course_id       varchar(8),
        sec_id          varchar(8),
        semester        varchar(6),
        year            numeric(4,0),
        grade           varchar(2),
        primary key (ID, course_id, sec_id, semester, year),
        foreign key (ID) references  student,
        foreign key (course_id, sec_id, semester, year) references section);
```

UPDATES to TABLES

- INSERT

  insert into *instructor* **values** ('10211','Smith','Biology',26000);

- DELETE – Remove all tuples from the *student* relation

  delete from *student* (where …)

- DROP TABLE

  drop table R

- ALTER – Change the table schema

  alter table *R* **add** *A D*

  - where *A* is the name of the attribute to be added to relation *R* and *D* is the domain of *A*.
  - All exiting tuples in the relation are assigned **NULL** as the value for the new attribute.

  alter table *R* **drop** *A*

  - where *A* is the name of an attribute of relation *R*
  - Dropping of attributes not supported by many databases.

Basic/typical form of SQL Query

```
SELECT <attributes: A1, …, Am>
FROM   <one or more relations: R1, …, Rk>
WHERE  <predicate/conditions: C>
```

Call this a select-from-where or **SFW** query.

The result of an SQL query is a relation.

Your playtime:

https://sqliteonline.com/

Try creating the tables that can hold the following dataset.

Define proper keys and constraint:
e.g.,

- Each name is unique in Person
- child in MotherChild or FatherChild must be some name in person
- A child can have a single mother and father
- Can age be null? Negative income?, …

Make sure that in the end you create this dataset!!!
(may try using your own delete, insert, alter, … statements)
See your table records: select * from <table>

Data from Werner Nutt

MotherChild

mother	child
Lisa	Mary
Lisa	Greg
Anne	Kim
Anne	Phil
Mary	Andy
Mary	Rob

FatherChild

father	child
Steve	Frank
Greg	Kim
Greg	Phil
Frank	Andy
Frank	Rob

Person

name	age	income
Andy	27	21
Rob	25	15
Mary	55	42
Anne	50	35
Phil	26	30
Greg	50	40
Frank	60	20
Kim	30	41
Mike	85	35
Lisa	75	87

Then populate the tables with the data provided in **sqlex_data1.sql**

Observe if there is any *errors* or constraint violations for a given statement, and think about why?

Your playtime:

Check more SQL script from sqlex_schema2.sql

See diagram of the schema!

Your playtime:

Try answering question in …

https://www.w3resource.com/mysql-exercises/create-table-exercises/

At least try these questions: 1, 5, 6, 7, 10, 16

Don’t look at the solution before trying to solve it!

General Constraints

- We can actually specify arbitrary assertions
  - E.g. "*There cannot be 25 people in the DB class*"

- In practice, we don't specify many such constraints. Why?
  - Performance!

Whenever we do something ugly
(or avoid doing something convenient)
it's for the sake of performance!