Database Concepts and Practices

Dr. John Yoon

What is?

- Database
- Database Management Systems
- Database System

Database Models

- Flat Database Model (File-based)
- Network Database Model
- Hierarchical Database Model
- Relational Database Model
  - Weakness:
    - Multivalued
    - Composite values
- Object-Oriented Database Model
- Object-Relational Database Model
- Others

Concepts

- Schema
- Relation
- Relationship
- Relation Schema
- Database Schema

Start with This Database Construction

Database Schema: a set of relation schemas

Relation Schema: a set of attributes with their types for a table

Student (cwid int, age int, major text)

Department (name text, chair text)

Faculty (name text, dept text)

Class (fac text, stud int)

Database Management System (DBMS): a collection of database management packages
Database System (DBS): a DB running on a DBMS
Database (DB): a collection of data in a specific modeling, e.g., relational database, OODB, ORDB

Relation: table
Relational Database: a database in relational modelling, i.e., a set of relations

Primary Key (PK): a set of attributes that can identify one record (tuple) from any other
Foreign Key (FK): an attribute that is used as a PK at another table (relation)

"John"	"CSEC"
"Peter"	"Math"
"Sam"	"CS"
"James"	"CSEC"

← Any relationships? →

"1111"	"27"	"CSEC"
"1112"	"23"	"CSEC"
"1113"	"22"	"Math"
"1114"	"24"	"CSEC"
"1115"	"32"	"CS"
"1116"	"29"	"CSEC"

"John"	"CSEC"
"Peter"	"Math"
"Sam"	"CS"
"James"	"CSEC"

"James"	"1111"
"John"	"1114"

"1111"	"27"	"CSEC"
"1112"	"23"	"CSEC"
"1113"	"22"	"Math"
"1114"	"24"	"CSEC"
"1115"	"32"	"CS"
"1116"	"29"	"CSEC"

Relationship

Object-Relational Database

Database Schema: a set of relation schemas

Student (cwid int, age int, major Department)

Department (name text, chair text)

Faculty (name text, dept Department)

Class (fac Faculty, stud Student)

Start with This Database Construction

Student (cwid int, age int, major text)

FK FK

Enroll (student int, course text)

Course (title text, credit int)

Introduction to SQL

- Data Definition
- Basic Query Structure
- Additional Basic Operations
- Set Operations
- Null Values
- Aggregate Functions
- Nested Subqueries
- Modification of the Database

Domain Types in SQL

- **char(n).** Fixed length character string, with user-specified length *n.*
- **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
- **int.** Integer (a finite subset of the integers that is machine-dependent).
- **smallint.** Small integer (a machine-dependent subset of the integer domain type).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.
- More are covered in Chapter 4.

Create Table Construct

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

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

  - *r* is the name of the relation
  - each A_i is an attribute name in the schema of relation *r*
  - D_i is the data type of values in the domain of attribute A_i
- Example:

  create table *instructor* (
  ID **char**(5),
  name **varchar**(20) **not null,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))
- **insert into** *instructor* **values** (‘10211’, ’Smith’, ’Biology’, 66000);
- **insert into** *instructor* **values** (‘10211’, null, ’Biology’, 66000);

Integrity Constraints in Create Table

- **not null**
- **primary key** (A_1, ..., A_n)
- **foreign key** (A_m, ..., A_n) **references** *r*

Example: Declare *branch_name* as the primary key for *branch*

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department)*

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

And a Few More Relation Definitions

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

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

And more still

- **create table** *course* (
  course_id **varchar**(8) **primary key**,
  title **varchar(**50),
  dept_name **varchar**(20),
  credits **numeric**(2,0),
  foreign key *(dept_name*) **references** *department)*);

Drop and Alter Table Constructs

- **drop table**
- **alter table**
  - **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 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 Query Structure

- A typical SQL query has the form:

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.
- The result of an SQL query is a relation.

The select Clause

- The **select** clause list the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: find the names of all instructors:

  select *name*
  from *instructor*
- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g., *Name* ≡ *NAME* ≡ *name*
  - Some people use upper case wherever we use bold font.

The select Clause (Cont.)

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select.
- Find the names of all departments with instructor, and remove duplicates

  select distinct *dept_name*
  from *instructor*

- The keyword **all** specifies that duplicates not be removed.

  select all *dept_name*
  from *instructor*

The select Clause (Cont.)

- An asterisk in the select clause denotes “all attributes”

  select *
  from *instructor*

- The **select** clause can contain arithmetic expressions involving the operation, +, –, ∗, and /, and operating on constants or attributes of tuples.
- The query:

  select *ID, name, salary/12*
  from *instructor*

  would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.

The where Clause

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all instructors in Comp. Sci. dept with salary > 80000

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000
- Comparison results can be combined using the logical connectives **and, or,** and **not.**
- Comparisons can be applied to results of arithmetic expressions.

The from Clause

- The **from** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.
- Find the Cartesian product *instructor X teaches*

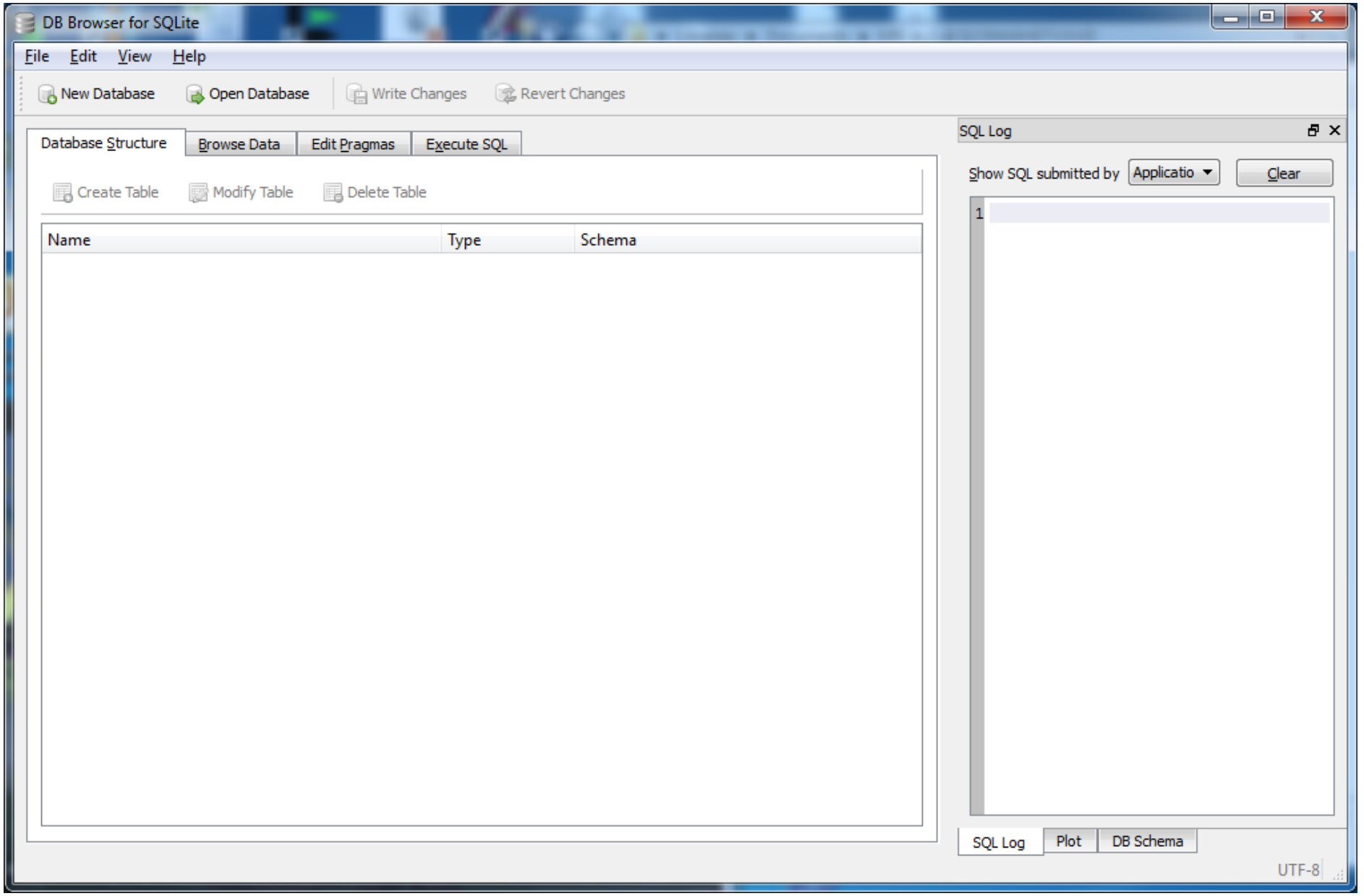
  select ∗
  from *instructor, teaches*

  - generates every possible instructor – teaches pair, with all attributes from both relations.
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra).

Exercise: SQLite

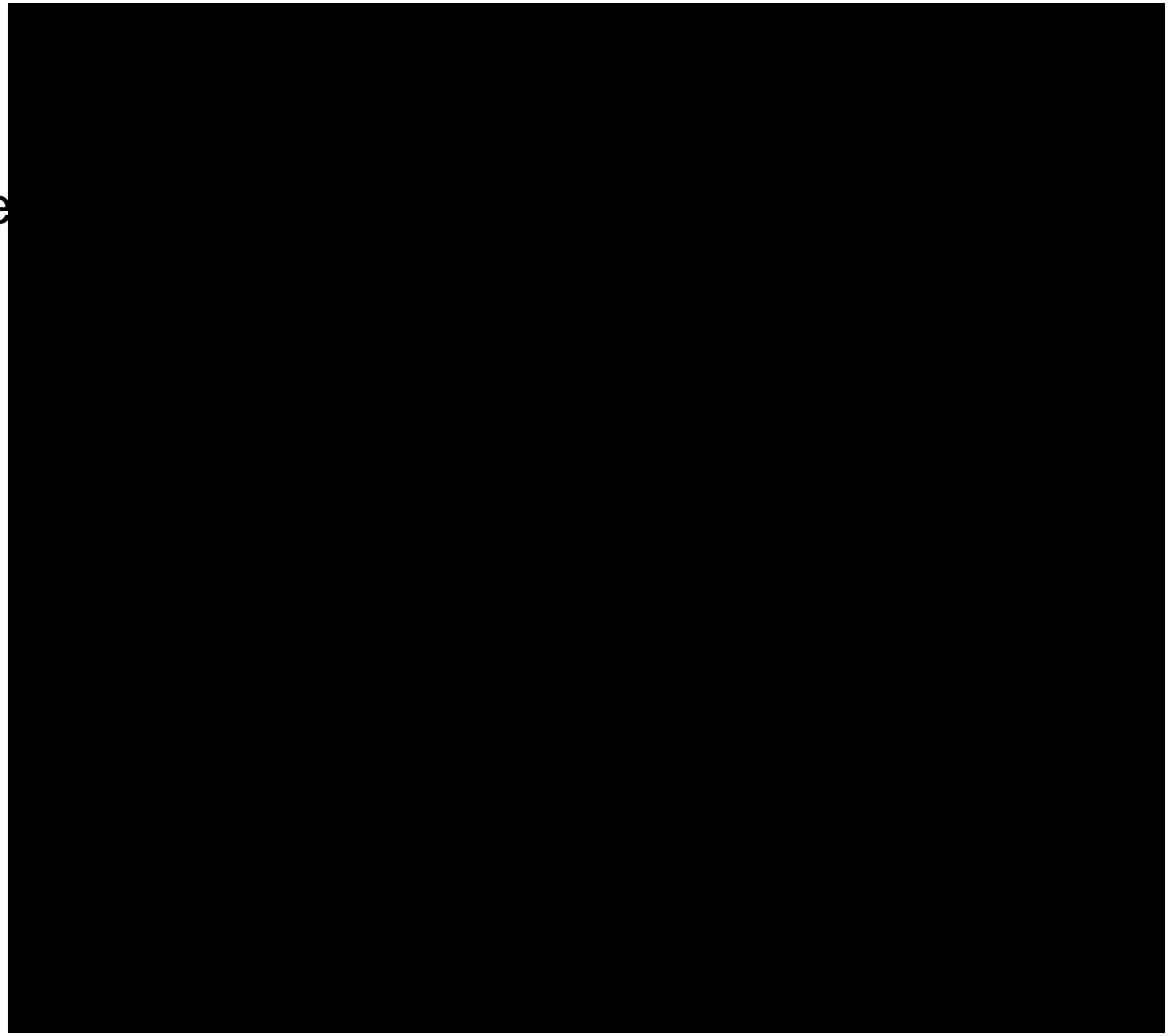
- Download
  - SQLite Browser: **http://sqlitebrowser.org/**
    - Install it
      - By clicking to let it go
    - Installed in the directory Program File
  - SQLite Command-line: **https://www.sqlite.org/download.html**
    - Simply double click the executable file sqlite3 in the unzipped folder

Exercise: SQLite Browser



Exercise: SQLite Command-line

- In sqlite shell
  - Find a database
  - Open the database
  - List tables
  - Select
  - Show schema
  - Exiting



Joins

- For all instructors who have taught courses, find their names and the course ID of the courses they taught.

  select *name, course_id*
  from *instructor, teaches*
  where *instructor.ID = teaches.ID*

- Find the course ID, semester, year and title of each course offered by the Comp. Sci. department

  select *section.course_id, semester, year, title*
  from *section, course*
  where *section.course_id = course.course_id* **and**
  dept_name = 'Comp. Sci.'