Introduction to Database Systems: CS312
SQL Query Structures

Oliver Bonham-Carter

10 March 2022

Let's Spend a Moment to Recap

Automatic Cavalcade leaves you free to enjoy the show!

CHANGES SLIDES BY ITSELF!

New Kodak Cavalcade Projector, only $124.50

Now you can own a new, lower cost model of the famous $159.50 Kodak Cavalcade Projector. It has the same automatic action—you turn it on, it does the rest. But this one's only $124.50.

You put on big, bright shows—up to 40 slides in a row—with no work at all! Your Cavalcade changes slides quietly, dependably, *automatically*—at the time intervals you choose. You reverse, repeat, edit at will.

Your pictures stay sharp on the screen—no refocusing—because every slide is preconditioned by warm air. Your slides are safe . . . each in its own steel guard for smooth, jam-free showings.

And each slide is projected 500-watts-bright through the brilliant *f*/3.5 lens.

See the new Kodak Cavalcade Projector, Model 520, demonstrated at your Kodak dealer's. At only $124.50, it's a superb value. As little as $12.50 down at many dealers. Model 510 with remote-control cord, and *f*/2.8 lens for extra brilliance, $159.50.

Prices are list, include Federal Tax, and are subject to change without notice.

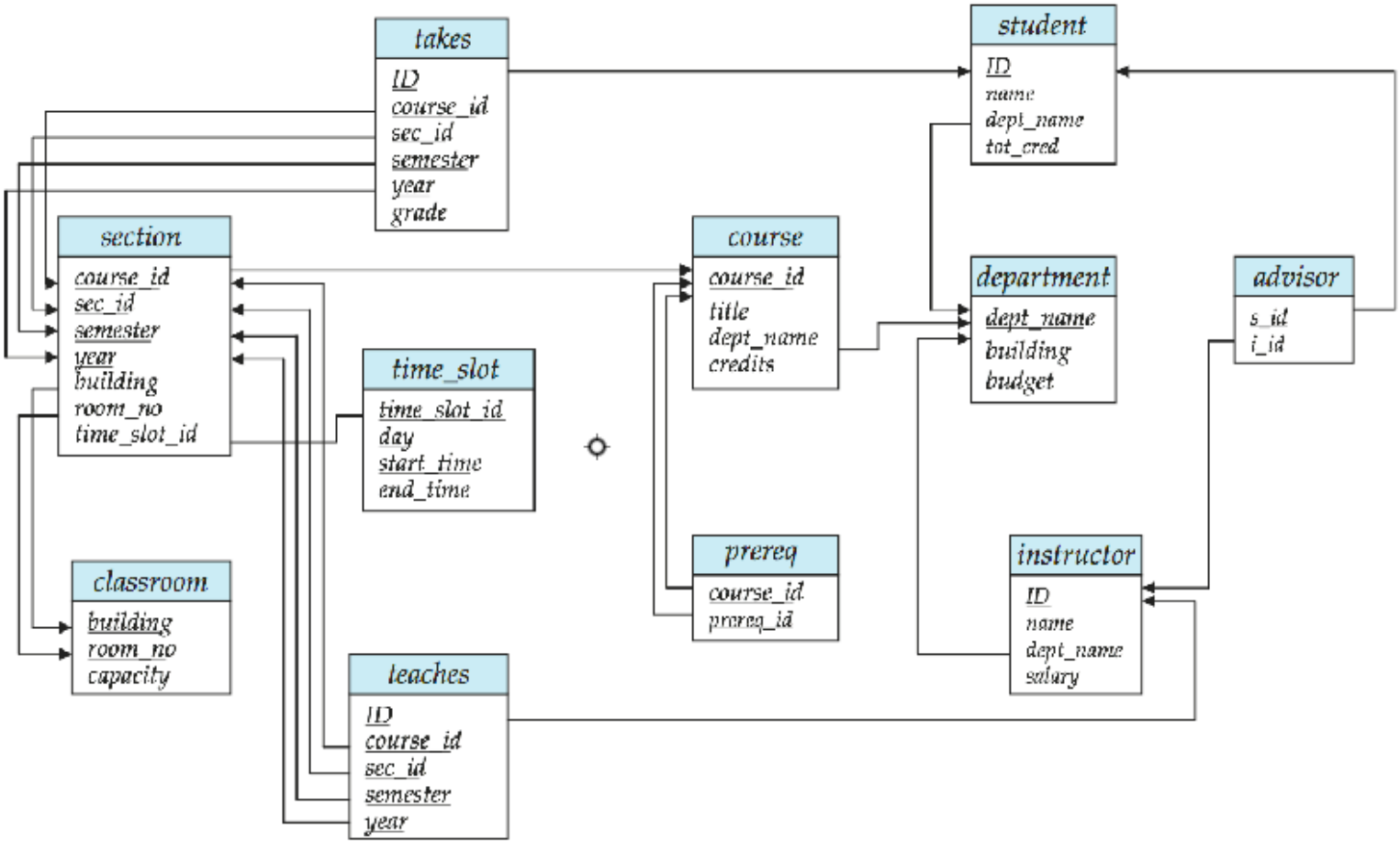
See Kodak's "The Ed Sullivan Show" and "The Adventures of Ozzie and Harriet"

EASTMAN KODAK COMPANY, Rochester 4, N.Y.

Kodak TRADEMARK

How to Connect The Information?

No free-standing tables allowed!



The Basic Query Structure

The SQL data-manipulation language (DML) provides the ability to query information, and insert, delete and update tuples

A typical SQL pseudo code query has the form:

```
SELECT A1, A2, ..., An
FROM r1, r2, ..., rm
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 filters out particular data from a table.

- SQL allows duplicates in relations as well as in query results.
- The SELECT statement has many optional clauses:
  - WHERE specifies which rows to retrieve.
  - GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group.
  - HAVING selects among the groups defined by the GROUP BY clause.
  - ORDER BY specifies an order in which to return the rows.
  - AS provides an alias which can be used to temporarily rename tables or columns..

Given table 'T'

Table "T"	Query	Result
C1 C2 1 a 2 b	`SELECT * FROM T;`	C1 C2 1 a 2 b
C1 C2 1 a 2 b	`SELECT C1 FROM T;`	C1 1 2
C1 C2 1 a 2 b	`SELECT * FROM T WHERE C1 = 1;`	C1 C2 1 a
C1 C2 1 a 2 b	`SELECT * FROM T ORDER BY C1 DESC;`	C1 C2 2 b 1 a

New Table Columns

Adding *cost* information to TeaDB

Build file: `sandbox/teaDB/teaDB_Build.txt`

```
cat builder_teaDB.txt | sqlite3 teaDB.sqlite3
```

The **SELECT** Clause

TeaDB

- Find everything in the `Department` table.
- SELECT * FROM Department;
- Find all entries for *dept*'s of the `Department` table
- SELECT dept from Department;
- Count entries of *dept*'s in `Department` table,
- SELECT COUNT(dept) FROM department;

The **SELECT** Clause

TeaDB

- Find all unique entries for *depts* in `Department` table,
- SELECT DISTINCT(dept) FROM department;
- Count unique entries of *depts* in `Department` table,
- SELECT COUNT(DISTINCT(dept)) FROM Department;
  /*count unique occurrences*/
- Return an exhaustive set of sandwiches that are being ordered.
- SELECT DISTINCT(sandwich) FROM Tea;
- What query to use to count these types of sandwiches?

The **WHERE** clause

TeaDB

The **WHERE** clause: **conditions** that the result must satisfy

- Corresponds to the selection predicate of the relational algebra
- Comparison results can be combined using the logical connectives and, or, and not
- Comparisons can be applied to results of arithmetic expressions

The **WHERE** clause

TeaDB

- Find out who is ordering a sandwich less than $15 (from the new cost column)

- SELECT * FROM tea WHERE cost $<$ 15;

- Find department, Session material, sandwich type for orders of sandwiches less than $15.

- SELECT Department.id, Session.session, tea.sandwich, tea.cost FROM Tea, Department, Session WHERE cost ¡ 15 AND Department.id == Session.id AND Department.id == Tea.id;

- Find out what kinds of *sandwich*es are going to each *dept*

- SELECT department.dept, tea.sandwich FROM department, tea WHERE department.id == tea.id;

The **WHERE** clause

TeaDB

- Find out which professors are presenting posters

- SELECT * FROM session WHERE material == “poster”; /* show all*/
- SELECT ID, material FROM session WHERE material == “poster”; /*which professor is doing what?*/

- Find how who is presenting a poster, having what kind of sandwich which costs over $10

- SELECT session.ID, session.material, tea.sandwich, tea.cost FROM session, tea WHERE session.material == “poster” AND tea.cost > 10 AND session.id == tea.id;

Moving On

And now this!

New Database Tables!

CampusDB

Build file: `sandbox/campusDB_Build.txt`

```
cat campusDB_build.txt | sqlite3 CampusDB.sqlite3
```

Abbreviations in queries

CampusDB

Find which students are working with what instructors.

- SELECT Instructor.ID, Instructor.name, Instructor.studentId, Student.name, Student.Id FROM Instructor, Student WHERE Instructor.studentId == Student.ID;

Shorter way to write query by using abbreviations

- SELECT **i**.ID, **i**.name, **i**.studentId, s.name, s.Id FROM Instructor **i**, Student s WHERE i.studentId == s.ID;

- The "**Instructor**" table name can be replaced with an **i**.
- The "Student" table name can be replaced by an "s".

Aggregate Functions
CampusDB

These functions operate on the multiset of values of a column of a relation, and return a value

- **avg**: average value
- **min**: minimum value
- **max**: maximum value
- **sum**: sum of values
- **count**: number of values

Mathematical Functions

CampusDB

To find all instructors in Comp. Sci. dept with salary > 80000

- SELECT name FROM instructor WHERE deptName = "CompSci" AND salary > 80000;

Using functions

- SELECT AVG (salary) FROM instructor WHERE deptName = "CompSci";
- SELECT MIN (salary) FROM instructor WHERE deptName = "CompSci";
- SELECT MAX (salary) FROM instructor WHERE deptName = "CompSci";
- SELECT SUM (salary) FROM instructor WHERE deptName = "CompSci";
- SELECT COUNT (salary) FROM instructor WHERE deptName = CompSci";

Faults of The WHERE Clause

CampusDB

- Find the ID, name and total credit students who are taking a course where the total credit is 3 or 4 hours.

Why will "AND" **NOT** work

- SELECT ID, name, totCred FROM student WHERE totCred == "3" OR totCred == "4";

Watch out for cross products that give no usable information!!

- SELECT s.name, i.name from student s, instructor i WHERE s.deptName == i.deptName and s.deptName == "CompSci";
- Common Solution – Use two queries instead
  - SELECT s.name from student s WHERE s.deptName == "CompSci";
  - SELECT i.name from instructor i WHERE i.deptName == "CompSci";

Using Count and Count(Distinct())

CampusDB

Find the number of tuples in the course relation

- SELECT COUNT(credits) FROM course;
- SELECT COUNT(distinct(credits)) FROM course;
- SELECT COUNT (*) FROM course;
- SELECT COUNT (distinct(*)) FROM course;
- Question: Why will the above *distinct* line **not** work?

Removing Tables or Data

CampusDB: Adding data to Student table

- DROP TABLE student
  - Deletes the table and its contents
- DELETE FROM student
  - Deletes all contents of table, but retains table

Remember, to use a builder file to re-create a database if it becomes corrupt or unstable.

Changing Table Contents

- ALTER TABLE
  - Alter TABLE r ADD AD
  - 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.
  - EX: `ALTER TABLE Department ADD Email varchar;`

- Change name of table:
  - `ALTER TABLE Department RENAME TO newDept;`

- Add a column to a table
  - `ALTER table course ADD COLUMN courseTag char(1);`
  - `Check your additional column:`
    - `.schema course`

Complex Queries

Instructor names, IDs and their Students?

```
SELECT
   instructor.name,
   instructor.id,
   instructor.studentID,
   student.ID,
   student.name
FROM
   instructor, student
WHERE
   student.id == instructor.studentID;
```

Output

```
Miller|10101|S1|S1|Michaels
Johnson|10102|S1|S1|Michaels
Charleson|10103|S2|S2|Peterson
Thompson|10104|S2|S2|Peterson
Mauler|10105|S3|S3|Mullen
```

Try these with `campusDB.sqlite3`!

Work by yourself or together with others

- What is the average salary of computer science teachers?
- What is the average salary of computer science teachers who make less than $98000?

Try these with `campusDB.sqlite3`!

Work by yourself or together with others

- What are the salaries of instructors who worked during the Spring?

- What are the salaries of instructors who worked during the Fall?

Try these with `campusDB.sqlite3`!

Work by yourself or together with others

- What are the Instructor names and their IDs who taught which Students (show names and IDs)?

- What are the Instructor names and their IDs who taught which Students (show names and IDs)?

Try these with `campusDB.sqlite3`!

Work by yourself or together with others

- What are the Instructor names and their IDs who taught which Students (show names and IDs) for classes taught in the year 2010?

- What are the Instructor names and their IDs who taught which Students (show names and IDs) for classes taught in the year 2010. In which semester were they teaching?

- Come-up with your own complex question and query solution.