SQLite Queries

Recap

relational database: collection of tables

- columns have names and types
- each row contains a record (a piece of data)

Example: Customer Table

From a database for a business that wants to keep track of customer info:

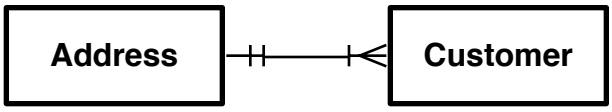
Customer ID (int)	First Name (text)	Last Name (text)	E-Mail (text)	...
1	Mary	Smith	mary.smith@sakilacustomer.org	
2	Patricia	Johnson	patricia.johnson@sakilacustomer.org	
3	Linda	Smith	linda.williams@sakilacustomer.org	
4	Barbara	Jones	barbara.jones@sakilacustomer.org	

Schema

Define -- collection of tables, how they are related

Often described by Entity-Relation Diagrams (ERD)

- one box per table
- links between boxes specify relationship (one-to-one, one-to-many, etc)
- all are “has-a” relations



today we’re just interested in table names —
later we’ll look at how tables are connected

SQLite3

Our projects this term will use SQLite

Download from sqlite.org

Command line application, start from terminal window:

```
% sqlite3 file
```

Here `file` is the name of the database file

- convention for names: end with .db
- if the DB does not exist it is created
- if DB exists already it is opened

Example (using sample DB from Perkovic):

```
% sqlite3 links.db
```

[demo]

NOTE: if you misspell the name of a DB you will get a new empty DB with the name you typed….

Dot Commands

A SQLIte command that starts with a period is a request to SQLite, not a query

Do not end these commands with a semicolon

Some useful commands:

```
.help
.quit
.databases
.tables
```

[demo]

Note commands can be abbreviated (`.h`, `.q`, etc)

The two commands in `.sqliterc` are also “dot commands”:

```
.header on
.mode column
```

Sakila

The database file we will use for projects is named `sakila211.db`

A collection of tables from a hypothetical video rental business

- customers
- stores
- films (descriptions of movies)
- inventory (how many copies of movies are at each store)
- rentals (which copies of movies were rented)

DB was created by MySQL to test their system

- http://dev.mysql.com/doc/sakila/en/
- has since been ported to several other RDBMs

To run the examples in this document:

```
% sqlite3 sakila211.db
```

SELECT

A SELECT statement asks the database system to fetch data from a table

```
SELECT col1, col2, ... FROM table
```

Example: the Category table has information about types of films. This query asks for the names of all the categories:

```
sqlite> SELECT name FROM category;

name
----------
Action
Animation
Children
...
```

Note semicolons at end of statement

Language is not case sensitive; convention puts keywords in caps so they stand out

Example: the Actor table has information about actors that star in films. This query asks for the first and last names of the actors:

```
sqlite> SELECT first_name, last_name FROM actor;

first_name  last_name
----------  ----------
Penelope    Guiness
Nick        Wahlberg
Ed          Chase
Jennifer    Davis
...
```

SQL's `select` statement implements the project (π) operator of relational algebra

Exploring the Database

Before you can type a SELECT statement you need to know

- table names
- names of columns in a table

When exploring you probably want just a few rows

- this DB has 200 actors, 4500 inventory records, 16000 rental transactions

What Tables Are in This DB?

In SQLite:

```
.tables
```

If you type this command in a session with `sakila211.db` this is what you'll see:

```
sqlite> .tables
actor              film               payment
address            film_actor         rental
category           film_category      sales_by_film_category
city               film_list          sales_by_store
country            film_text          staff
customer           inventory          staff_list
customer_list      language           store
```

What's in a Table? (Method 1)

"Official" way to learn about table:

```
.schema table-name
```

On the first slide: the database schema defines the collection of tables and how they are related

The system will print the "create table" command that made the table, including names and types of each column

```
sqlite> .schema actor
CREATE TABLE actor (
  actor_id numeric NOT NULL ,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  last_update TIMESTAMP NOT NULL,
  PRIMARY KEY  (actor_id)
  );
```

"varchar" is short for "variable length character field", i.e. "string"

Much more info than you need, just look for column names...

What's in a Table? (Method 2)

Another way to find out what's in a table is to just print the contents.

First, use this query to find out how many records are in the table (this example is for the actor table):

```
SELECT count(*) FROM actor;
```

Then, if there are lots of records, add a "limit N" to the end of a query that selects all columns:

```
SELECT * FROM actor limit 5;
```

*A * in a query means "all columns"*

Example:

```
sqlite> select * from actor limit 5;
actor_id    first_name  last_name   last_update
----------  ----------  ----------  --------------------
1           Penelope    Guiness     2015-02-13 15:52:43
2           Nick        Wahlberg    2015-02-13 15:52:43
3           Ed          Chase       2015-02-13 15:52:43
4           Jennifer    Davis       2015-02-13 15:52:43
5           Johnny      Lollobrigi  2015-02-13 15:52:43
```

[demo]

What should I type to learn about film and customer tables?

WHERE Clause

We usually want to narrow a search for data

In a SELECT statement we can specify attributes of the rows we want

Example: the customer table has column named active; 0 means “account closed”. To get names of inactive customers:

```
SELECT last_name FROM customer WHERE active = 0;
```

Note that COUNT can be combined with WHERE:

```
SELECT count(last_name) FROM customer WHERE active = 0;
```

SQL’s `select` *statement also implements the select (σ) operator of relational algebra*

Boolean Expressions

Values in a WHERE clause can be combined with Boolean operators

The SQL operators are AND, OR, NOT

```
SELECT title FROM film WHERE rental_duration = 4 AND
    rental_rate < 2.99;

SELECT rating, title, description FROM film WHERE
    rating = 'G' OR rating = 'PG';
```

[demo]

which films rated G or PG cost less than $1 to rent?

how many films were released in 2006?

which films have a replacement cost less than three times the rental rate?

Pattern Matching

Queries can see if a text field matches a pattern

Use the LIKE operator:

```
SELECT title, description FROM film WHERE
   description LIKE "%Scientist%";
```

NOTE: you need a wildcard (%) to make a partial match

Aside: Is Pattern Matching Case Sensitive?

String comparisons are case sensitive. Try this:

```
SELECT 2 * 5;
SELECT 'a' = 'A';
SELECT 'A' = 'A';
```

But pattern matching with the LIKE operator is not case sensitive:

```
SELECT 'aloha' like 'A%';
```

Groups

Use “GROUP BY” to collapse rows into groups that have the same value for specific column(s):

```
SELECT rental_duration, count(title) FROM film GROUP BY
rental_duration;
```

How this works:

- internally SQLite does a select to make a temporary table, and then it sorts the table according to rental duration
- rows that have the same duration are merged into a single row
- summary functions (count, max, etc) can be applied to the group

[demo]

How do I write these queries using groups?

print a table that shows how many films were released each year

print a table that shows how many customers are active or inactive

how many copies of film 1 are in each store? [use inventory table]

Some more queries, using all of the above:

how many film categories are there?

how many films are horror movies (category 11)? [use film_category table]

how many films include trailers?

how many NC-17 films include deleted scenes?