SELECT Lab 2 - Solutions

DB Practice CSC 330 Spring 2022

README

- This is a notebook for you to review SQL using the SQLite database management system using a small but fully formed database 'foods'.
- This notebook also introduces you to some aspects of the Entity Relational Model which underlies the design of relational DB
- In another notebook we'll look at JOINs using this database.

Download sample data

- `[ ]` Download the sample data `foods.sql` from GDrive using your browser.
- `[ ]` Open a terminal and go to the download directory - most likely `C:\Downloads` - with the command `cd C:\Downloads`.
- `[ ]`

  Create an SQLite database by typing at the terminal prompt:

  ```
  sqlite3 foods.db < foods.sql
  ```

  This runs SQLite on the SQLite data and generates a binary database file.

- `[ ]`

  Check your progress by looking up the tables in `foods.db`.

  ```
  .tables
  ```

  ```
  episodes        food_types      foods           foods_episodes
  ```

- There should be four tables: `episodes`, `food_types`, `foods`, and `foods_episodes`.

Getting to know the data

The data

- The database contains all episode titles of the popular TV show "Seinfeld" together with the foods shown in each episode, and the types of food.
- For an overview of the variables contained in each table, see the schema printed in 1.

Entity relationship diagrams

- `[ ]`

You'll learn more about the data later. Here is an Entity Relationship Diagram that shows the four different tables together with their attributes:

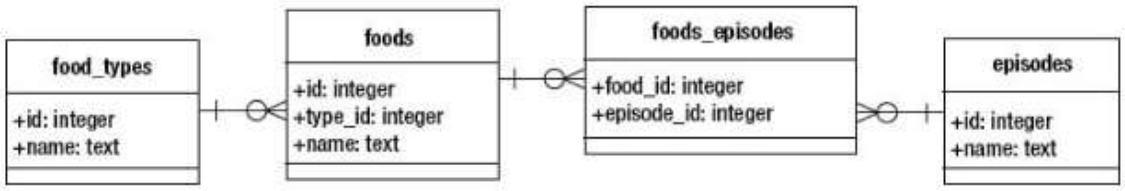

Figure 1: Entity-Relationship Diagram for foods.db

- You can create such diagrams yourself in an Object-Relationship Mapper (ORM) application like *ponyorm*. This is more than a drawing program because it generates DDL code for different RDBMS automatically. See foods database in ponyorm.

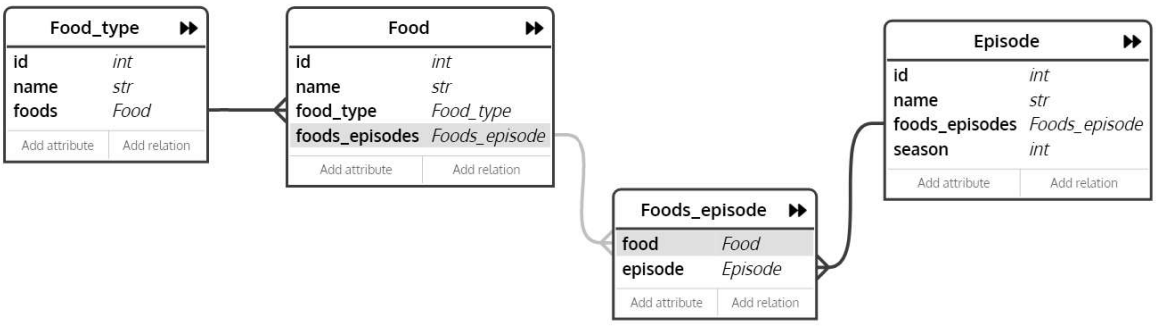

Figure 2: Object-relationship map for foods (ponyorm.com)

- [ ]

  In the next code blocks list the commands that are needed to import the ORM schema, stored in a file `orm.sql` into SQLite.

  - Create a database orm.db (in 1)
  - Open SQLite on orm.db (in 1)
  - Check if the tables are there (in 1)
  - Check the data definitions (in 1)

```
sqlite3 orm.db < orm.sql ## creates foods_orm.db
sqlite3 orm.db           ## open sqlite3 on orm.db
```

```
.tables
.schema
```

```
Episode        Food           Food_type      Foods_episode
CREATE TABLE IF NOT EXISTS "Episode" (
  "id" INTEGER PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL,
  "season" INTEGER
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE IF NOT EXISTS "Food_type" (
  "id" INTEGER PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL
);
CREATE TABLE IF NOT EXISTS "Food" (
  "id" INTEGER PRIMARY KEY AUTOINCREMENT,
  "name" TEXT NOT NULL,
  "food_type" INTEGER NOT NULL REFERENCES "Food_type" ("id") ON DELETE CASCADE
);
CREATE INDEX "idx_food__food_type" ON "Food" ("food_type");
CREATE TABLE IF NOT EXISTS "Foods_episode" (
  "food" INTEGER NOT NULL REFERENCES "Food" ("id") ON DELETE CASCADE,
  "episode" INTEGER NOT NULL REFERENCES "Episode" ("id") ON DELETE CASCADE,
  PRIMARY KEY ("food", "episode")
);
CREATE INDEX "idx_foods_episode__episode" ON "Foods_episode" ("episode");
```

Simple queries

- Let's get started with two simple queries.
- [ ]

  How many food types are there?

  ```
  SELECT COUNT(type_id) FROM foods;
  ```

  COUNT(type_id)

  412

- [ ]

  Run the previous query again, but this time show what the result means by creating a meaningful header.

  ```
  SELECT COUNT(type_id) AS "Total food types" from foods;
  ```

  Total food types

  412

- [ ]

  Let's use another function: what is the (alphabetically) last episode?

```
SELECT MAX(name) FROM episodes;
```

```
MAX(name)
-------------
The Yada Yada
```

Translate SQL query into natural language

- SQL is a declarative, natural language. You are able to speak commands easily while writing efficient code. Example: the DDL command 1 would read:

  > "Create a table `tbl` with only one column named `id`, which is an `integer` number, and which cannot be empty or unknown (i.e. a value has to be assigned to it."

```
CREATE TABLE tbl (id INTEGER NOT NULL);
```

- `[ ]`

  Your turn! Turn the following SQL query into a normal sentence.

```
SELECT foods.name
FROM foods
WHERE foods.name LIKE 'M%';
```

---

name

Marble Ryes

Mini Ritz

Muffin

Muffin Tops

Muffin Stumps

Maple Syrup

Mustard (fancy)

Morning Thunder (with caffine)

Mellow Yellow soda

Merlot

Milk

Milkshake

Molotov Cocktail

Morning	Thunder	(with	caffein)
Macinaw	peaches		
Mangos			
Melons			
M	and	M's	
Macadamia	Nuts		
Mints			
Mr.	Goodbar		
Meat	Loaf		
Meatball	Sandwich		
Mutton			
Macaroni			
Motzah	Ball		
Medium	Turkey	Chili	
Mulligatawny	soup		
Mushroom	Barley		

"Print the `name` of all foods in the table `foods` that begin with the letter `M`."

Operator Types

- `[X]`

  The SELECT command can execute many operators. How many operators can you distinguish in the code block 1? List them below in 1.

```
SELECT (-1) +
(SELECT 1=1) *
(SELECT AVG(foods.id)
FROM foods
WHERE foods.id < 100);
```

```
(-1) +
(SELECT 1=1) *
(SELECT AVG(foods.id)
FROM foods
WHERE foods.id < 100)
-----------------------------------------------------------------------------
49.0
```

```
(-1) +
(SELECT 1==1) *
```

```
(SELECT AVG(foods.id)
FROM foods
WHERE foods.id < 100)
-------------------------------------------------------------------------
49.0
```

Answer:

1. - (unary arithmetic)
2. + (binary arithmetic)
3. == (binary logical)
4. < (binary relational)
5. . (binary selectional)
6. AVG (unary aggregate)

- [ ] Do you think this code will run? Try it. If you cannot understand what's going on, dissect the command and run it in parts.

Restrict output rows

- [ ]

Write a query to get all names of Seinfeld episodes, and restrict the output to the episodes 5-15 only.

```
SELECT episodes.id, episodes.name
FROM episodes
LIMIT 11 OFFSET 5;
```

```
id  name
--  ---------------------
5   The Ex-Girlfriend
6   The Pony Remark
7   The Busboy
8   The Baby Shower
9   The Jacket
10  The Chinese Restaurant
11  The Phone Message
12  The Apartment
13  The Stranded
14  The Statue
15  The Heart Attack
```

Filter patterns

- [ ]

Write a query that returns the names of all foods in the table foods, which begin with the letter P. Print the first 5 entries of that list, and name the column "P foods".

```
SELECT foods.name AS "P foods"
FROM foods
WHERE foods.name LIKE 'P%'
LIMIT 5;
```

```
P foods
--------------------------------
Pastries (of the Gods)
Peach Muffin
Peppridge Farms Cookies (Milanos)
Pizza Bagels
Pie
```

Ordering rows

- [ ]

  Write a query that returns the `season` and the episode `name` for all episodes in the seasons 5 to 10, whose name contains the letters 'ng'. Print the output so that the later seasons are displayed first.

```
SELECT season, name
FROM episodes
WHERE season BETWEEN 5 and 10 AND name LIKE '%ng%'
ORDER BY season DESC;
```

```
season  name
------  ----------------------
9       The Strongbox
9       The Burning
8       The English Patient
7       The Engagement
7       The Sponge
5       The Mango
5       The Sniffing Accountant
```

Functions

- [ ]

  How many letters does the Seinfeld episode with the shortest `name` have? Write one query to find it and print both length of the episode title (as 'Length') and the title (as 'Title').

```
SELECT episodes.name AS 'Title',
MIN(LENGTH(episodes.name)) AS 'Length'
FROM episodes;
```

```
Title    Length
-------  ------
The Dog  7
```

Grouping rows

- [ ]

How many episodes did each of the 10 seasons of the Seinfeld show have? Write a query that returns the episode count for each `season`, and call the output 'Episode Count'.

```
SELECT COUNT(*) AS 'Episode Count'
FROM episodes
GROUP BY season;
```

```
Episode Count
-------------
2
4
13
22
24
22
24
24
22
24
```

Eliminate duplicates

- [ ]

How many different episode counts per season did the Seinfeld show have? Write a query that eliminates duplicate entries from the previous query 1.

```
SELECT DISTINCT COUNT(season) AS 'Episode Count'
FROM episodes
GROUP BY season;
```

```
Episode Count
-------------
0
4
13
22
24
```

Coercion

- [ ]

Explain the difference between the results of these three queries!

```
SELECT LENGTH("2==3");
SELECT LENGTH("2=3");
SELECT LENGTH(2==3);
```

```
LENGTH("2==3")
--------------
```

```
4
LENGTH("2=3")
-------------
3
LENGTH(2==3)
------------
1
```

```
1) LENGTH("2==3") is the length of the string "2==3"
2) LENGTH("2=3") is the length of the string "2=3"
3) LENGTH(2==3) is the length of the result of 2==3 (FALSE = 0)
```

References

- Kreibich (2010). Using SQLite. O'Reilly.
- Allen/Owens (2010). The Definitive Guide to SQLite. APress.

Author: Marcus Birkenkrahe
Created: 2022-03-31 Thu 12:34
Validate