Basics of Databases/SQL

Sarah Hegarty
CAS Code Review
27/04/2018

Key points

What is a database?
What is a a *relational* database?
Quick overview:
- DB/tables/rows/datatypes
- Writing queries

Practical example:
- Build a db and create a table
- Insert data
- Try some queries
- Interpreting the output

Postgres in brief

What is a database?

- A structured collection of data

What is a database?

- A structured collection of data

[Virgo, M87, galaxy, 12h 30m 49.42338s, +12° 23′ 28.0439″, Virgo cluster]

What is a database?

- A structured collection of data

Unstructured data!

~~*[Virgo, M87, galaxy, 12h 30m 49.42338s, +12° 23′ 28.0439″, Virgo cluster]*~~

What is a database?

- A structured collection of data
- **Hierarchical, network, relational, object-oriented……**

~~*[Virgo, M87, galaxy, 12h 30m 49.42338s, +12° 23′ 28.0439″, Virgo cluster]*~~

What is a database?

- A structured collection of data
- Hierarchical, network, **relational**, object-oriented……

~~*[Virgo, M87, galaxy, 12h 30m 49.42338s, +12° 23′ 28.0439″, Virgo cluster]*~~

What is a relational database?

What is a relational database?

- A collection of data which uses a **table** structure

What is a relational database?

- A collection of data which uses a table structure

Name	ID	Type	RA	DEC	Group
Virgo	M87	Elliptical	187.7059	+12.3911	Virgo cluster

What is a relational database?

- A collection of data which uses a table structure

Columns/Fields

Name	ID	Type	RA	DEC	Group
Virgo	M87	Elliptical	187.7059	+12.3911	Virgo cluster

What is a relational database?

- A collection of data which uses a table structure

Name	ID	Type	RA	DEC	Group
Virgo	M87	Elliptical	187.7059	+12.3911	Virgo cluster

Rows/records

What is a relational database?

- A collection of data which uses a table structure
- Predefined data types

Name	ID	Type	RA	DEC	Group
Virgo	M87	Elliptical	187.7059	+12.3911	Virgo cluster

What is a relational database?

- A collection of data which uses a table structure
- Predefined data types

Name	ID	Type	RA	DEC	Group
Virgo	M87	Elliptical	187.7059	+12.3911	Virgo cluster

“….kind of seems like just a spreadsheet. What’s the big deal?”

Why bother with a relational database?

Why bother with a relational database?

- Efficiency

Why bother with a relational database?

- Efficiency
- Mysterious table-based powers

Why bother with a relational database?

- Efficiency
- Mysterious table-based powers

….I will come back to this

Why bother with a relational database?

- Efficiency
- Mysterious table-based powers
- **Incredibly powerful, elegant query capabilities**

SQL: Structured Query Language

SQL: Structured Query Language

- Relational databases are always associated with a *Relational Database Management System (RDBMS).*

SQL: Structured Query Language

- Relational databases are always associated with a *Relational Database Management System (RDBMS).*
- A set of programs/OS that creates/maintains the DB, and makes **queries** possible

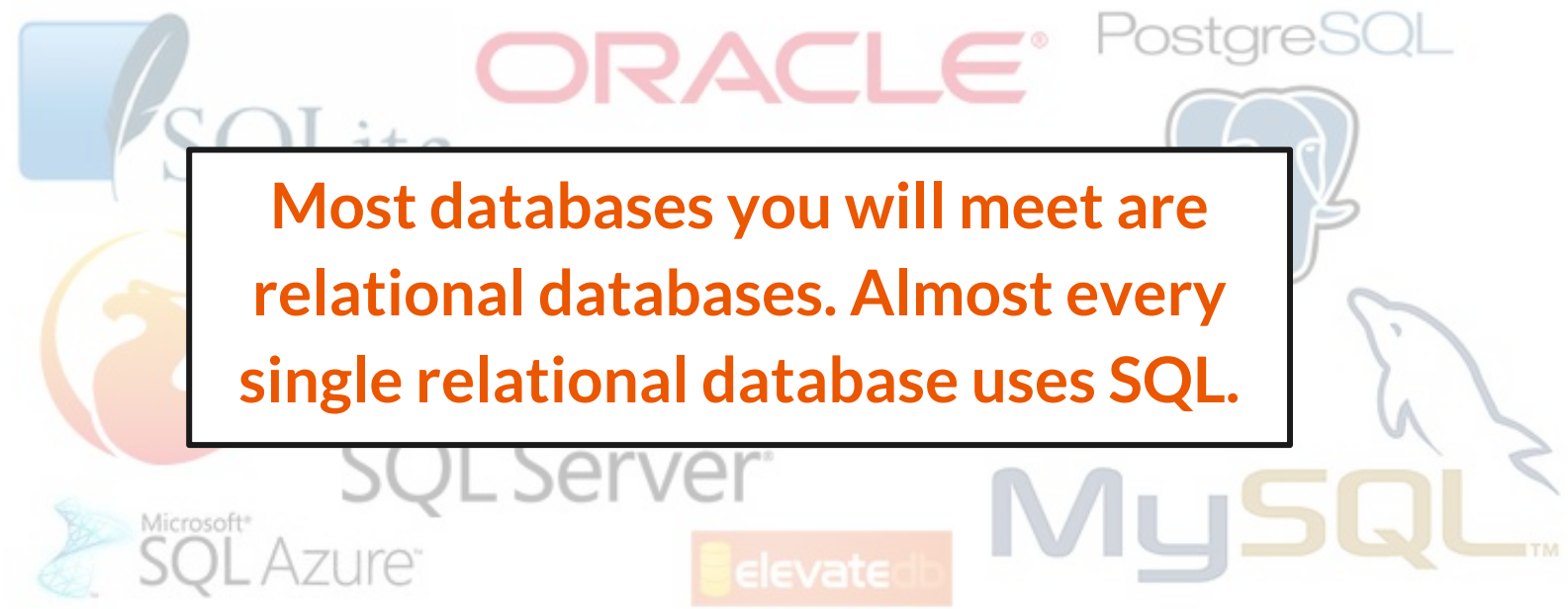
SQL: Structured Query Language

- Relational databases are always associated with a *Relational Database Management System (RDBMS).*
- A set of programs/OS that creates/maintains the DB, and makes queries possible
- **Overwhelmingly, the language of the DBMS is SQL**

SQL: Structured Query Language

Image: http://www.virtutech.com.au

SQL: Structured Query Language



Most databases you will meet are relational databases. Almost every single relational database uses SQL.

Image: http://www.virtutech.com.au

Great! I want to learn SQL!

Great! I want to learn SQL!

- Let’s start with SQLite

Great! I want to learn SQL!

- Let’s start with SQLite
- It is straightforward, and readily transferable

Great! I want to learn SQLite!

- In python 2:
  - import sqlite3 as sql
- At the command line on, eg, gstar:
  - Module load sqlite

Creating a database

- In python 2:

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()
```

Connect to a database if it exists; create it if it does not.

- At the command line
  $ sqlite3 **test_database**

Using a cursor

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()
```

Using a cursor

- A cursor is a “temporary work area created in the system memory when a SQL statement is executed” (codeproject.com)

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()
```

Using a cursor

- A cursor is a “temporary work area created in the system memory when a SQL statement is executed” (codeproject.com)

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()
```

...ie: in python, the cursor is where you are going to **execute** all your **queries - all your interactions with the database**

Query 1: CREATE TABLE in your new database

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

Ok, we want to execute an SQL query!

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

THIS STRING IS THE QUERY
WE WANT TO EXECUTE

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

Type any query directly at the sqlite command prompt to achieve the same thing

```
sqlite> .database
seq  name             file
---  ---------------  ----------------------------------------------------------
0    main             /mnt/home/shegarty/test_database
sqlite> .tables
sqlite> CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text);
sqlite> .tables
my_galaxies
```

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

Here, we declare what the fields in our new table will be

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

Here, we declare what the fields in our new table will be

We have to declare both the name and the data type of each field

Query 1: CREATE TABLE in your new database

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE my_galaxies (id text, name text, type text, ra float, dec float, group_name text)")

cur.close()
conn.close()
```

Here, we declare what the fields in our new table will be

We have to declare both the name and the data type of each field.

Different SQL implementations use slightly different types; check the documentation

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster"
cur.commit()

cur.close()
conn.close()
```

We are going to execute an INSERT statement

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster"
cur.commit()

cur.close()
conn.close()
```

Always have to indicate the table that we want to use!

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster"
cur.commit()

cur.close()
conn.close()
```

This is a very basic syntax for inserting data

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster"
cur.commit()

cur.close()
conn.close()
```

This is a very basic syntax for inserting data

These ?'s in the statement show we are going to place a variable in each of our six columns...

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster
cur.commit()

cur.close()
conn.close()
```

And finally, commit our changes...

Query 2: INSERTing data to your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster
cur.commit()

cur.close()
conn.close()
```

Note: we can alter this data later using the UPDATE statement

Query 3: SELECTing data from your new table

Query 3: SELECTing data from your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster")
cur.commit()

cur.execute("SELECT * FROM my_galaxies")

my_selection = cur.fetchall()

print my_selection
```

We are going to execute a SELECT statement

Query 3: SELECTing data from your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster")
cur.commit()

cur.execute("SELECT * FROM my_galaxies")

my_selection = cur.fetchall()

print my_selection
```

Again, have to indicate which table we want to operate on

Query 3: SELECTing data from your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster")
cur.commit()

cur.execute("SELECT * FROM my_galaxies")

my_selection = cur.fetchall()

print my_selection
```

SELECT * loads EVERYTHING from that table into our cursor workspace

Query 3: SELECTing data from your new table

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster")
cur.commit()

cur.execute("SELECT * FROM my_galaxies")

my_selection = cur.fetchall()

print my_selection
```

Then, we fetch it from that memory, and can work with it.

```
import sqlite3 as sql

conn = sql.connect('test_database.db')

cur = conn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS my_galaxies (gal_id text, name text, type text, ra float, dec float, group_name text)")

cur.execute("INSERT INTO my_galaxies VALUES(?, ?, ?, ?, ?, ?)",("M87", "Virgo", "Elliptical", 187.7059, 12.3911, "Virgo Cluster")
cur.commit()

cur.execute("SELECT * FROM my_galaxies")

my_selection = cur.fetchall()

print my_selection

cur.execute("SELECT name, ra, dec FROM my_galaxies")

my_selection_by_columns = cur.fetchall()

print my_selection_by_columns

cur.close()
conn.close()
```

We can also select the columns we are interested in

Writing complex queries: examples from SkyServer

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Writing complex queries: examples from SkyServer

```
-- This sample query find all objects with spectra
-- classified as stars. The query also checks that zWarning has no bits set,
-- meaning that there are no known problems with the spectra.

-- Other possible values with of class are 'QSO', 'GALAXY' and 'UNKNOWN'.

SELECT TOP 100 specObjID
FROM SpecObj
WHERE class = 'STAR' AND zWarning = 0
```

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Writing complex queries: examples from SkyServer

```
-- This sample query find all objects with spectra
-- classified as stars. The query also checks that zWarning has no bits set,
-- meaning that there are no known problems with the spectra.

-- Other possible values with of class are 'QSO', 'GALAXY' and 'UNKNOWN'.

SELECT TOP 100 specObjID
FROM SpecObj
WHERE class = 'STAR' AND zWarning = 0
```

```
SELECT TOP 1000 objID
FROM Galaxy
WHERE
        (r - extinction_r) < 22
```

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Writing complex queries: examples from SkyServer

```
-- This sample query finds unique objects in an RA/Dec box.
-- For a more efficient way to find objects by position, see the next query,
-- Searching around a sky position.

SELECT TOP 100
      objID, ra ,dec                    -- Get the unique object ID and coordinates
FROM
      PhotoPrimary                      -- From the table containing photometric data for unique objects
WHERE
      ra > 185 and ra < 185.1
      AND dec > 15 and dec < 15.1       -- that matches our criteria
```

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Writing complex queries: examples from SkyServer

```
-- Provide a list of moving objects consistent with an asteroid.
-- This sample query uses the 'as' syntax, which allows you to
-- give your own names to columns in the results.

SELECT TOP 10
      objID, ra, dec,
      sqrt( power(rowv,2) + power(colv, 2) ) as velocity
FROM PhotoObj
WHERE
      (power(rowv,2) + power(colv, 2)) > 50
      AND rowv != -9999 and colv != -9999
```

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Why bother with a relational database?

- Efficiency
- Mysterious table-based powers

….I will come back to this

Why bother with a relational database?

- Efficiency
- Mysterious table-based powers

….I will come back to this

Why bother with a relational database?

- Efficiency
- Incredibly powerful, elegant query capabilities
- **You can use these queries across *and between* multiple tables**

Why bother with a relational database?

- Efficiency
- Incredibly powerful, elegant query capabilities
- You can use these queries across and between multiple tables
- **JOIN or UNION commands let you do this**

```
-- When you need to search for data in two or more tables, you must "join" the tables
-- together for the purposes of your query by using a JOIN...ON statement.
-- JOIN...ON allows you to search for data and/or constraints in both tables.

-- The syntax of the statement is:
-- JOIN [the second table] ON [a variable the tables have in common].
-- The variable the tables have in common is called the "key" (think of it
-- as the key that unlocks your ability to search two tables).
-- The key variable(s) in each table are shown on the About the Database page.
-- Find the two tables you want to join and look for a key variable they have in common.

-- The sample query looks for spectra of quasars and shows the date and time at which
-- each spectrum was taken.

SELECT TOP 100
       sp.objID,
       sp.ra,
       sp.dec,
       sp.mjd,
       px.taiBegin,
       px.taiEnd,
       sp.fiberID,
       sp.z
FROM specPhoto AS sp
JOIN plateX AS px
       ON sp.plateID = px.plateID
WHERE
       (sp.class='QSO')
       AND sp.z > 3
```

See http://skyserver.sdss.org/dr8/en/help/docs/realquery.asp

Summary

- Databases let you store your data in a structured way
- Relational databases, which structure your data into tables, are particularly useful
- SQL lets us interact with relational databases
- You can get started with SQL relatively quickly - and learn to write very powerful queries!