

Lecture 25 - Introduction to SQL with SQLite

DSE 511

Drew Schmidt
2022-11-22

Announcements

- Schedule:
 - Nov 22 - databases
 - Nov 24 - No class for US Thanksgiving 🦃🥧🍁
 - Nov 29 and Dec 1 - more databases
 - Dec 6 - course wrapup
- New homework (last one)
 - Assigned now
 - Due Mon Dec 5
 - No homework on last module (databases)
- Questions?

Content

- SQL
- SQLite

SQL

What Is a Database?

- "A database is an organized collection of structured information, or data, typically stored electronically in a computer system" - Oracle
- Usually referring to a DBMS + its stored data
- Data is usually "tabular", but can form complicated hierarchies



Types of Databases

Relational (tables - SQL)

- MySQL
- PostgreSQL
- SQLite
- Oracle

Non-Relational ("NoSQL")

- MongoDB (document-oriented)
- Redis (key/value)
- Apache Cassandra (columnar)

How Do We Interact with a DB



- Structured Query Language
- S-Q-L or "sequel"
- A DSL
- Differences across implementations
- Simple queries are easy to understand: `SELECT x FROM y WHERE z`

The General Process

- "Attach" to DB (open connection)
- Execute SQL query/queries
- "Detach" from DB (close connection)

SQL Writing

- Writing to an existing table is easy
- You will (probably) use a high-level API (e.g. R, Pandas)
- Setting up the tables is the hard part!
- Efficiency
 - Indexing matters a lot
 - Selection criteria (`WHERE` clauses)
 - Data engineers know more about this
- We'll treat this as (mostly) someone else's problem

SQL Reading

- Mostly straight-forward
- Can become unbearably complicated if you have a goofy table structure
- This will be our main focus today

Some SQL Commands

- `SELECT` - extract data
- `UPDATE` - update data
- `DELETE` - delete data
- `CREATE INDEX` - create an index (for searching)
- `DROP INDEX` - delete an index

Some SQL Conventions

- We'll use UPPER CASE for the SQL bits
- We'll use lower case for our data
- This is a common convention; there are others

Some Basic SQL

- Read all the data from `my_table`

```
SELECT * FROM my_table;
```

- Select specified columns and all rows from `my_table`

```
SELECT col1 col2 col3 FROM my_table;
```

- Read data matching a 'filter' criterion

```
SELECT * FROM my_table WHERE group_var = 'gp1';  
-- Another example <-- also a comment  
SELECT * FROM my_table WHERE group_var = 'gp1' AND col1 >= 0;
```

SQLite

- We'll be using this for examples/demos
- Not a daemon/service
- The "db" is a file!
- Strictly speaking: a library
- Accessible via R and Python



SQLite Example

1. Using R

1. Generate data
2. Write data to DB (SQLite file - sub for your DB of choice!)
3. Inspect
4. Read examples

2. Using Python

1. Open DB we just wrote
2. Inspect
3. Read examples

Packages

- sqlite - `sudo apt install sqlite3 libsqlite3-dev`
- R
 - glue (optional) - `install.packages("glue")`
 - DBI - `install.packages("DBI")`
 - RSQLite - `install.packages("RSQLite")`
- Python
 - sqlite3 - `pip install sqlite3`
 - pandas - `pip install pandas`

SQLite: Data Generation

Generate fake data:

```
set.seed(1234)
n = 100
big_tbl = data.frame(
  ind = 1:n,
  x = runif(n),
  y = rnorm(n)
)

big_tbl |> head(n = 3)
```

	ind	x	y
1	1	0.1137034	-1.8060313
2	2	0.6222994	-0.5820759
3	3	0.6092747	-1.1088896

SQLite: Writing

Write table to disk:

```
library(glue)
library(DBI)
library(RSQLite)

table_name = "big_tbl"
db_file = "/tmp/db.sqlite"

db = DBI::dbConnect(RSQLite::SQLite(), db_file)
DBI::dbWriteTable(db, table_name, big_tbl)
```

SQLite: Inspection

```
# Warning! SQLite-ism!  
query = glue("pragma table_info({table_name});")  
query
```

```
pragma table_info(big_tbl);
```

```
DBI::dbGetQuery(db, query)
```

	cid	name	type	notnull	dflt_value	pk
1	0	ind	INTEGER	0	NA	0
2	1	x	REAL	0	NA	0
3	2	y	REAL	0	NA	0

SQLite: Inspection

```
query = glue("SELECT COUNT(*) from {table_name};")
query
```

```
SELECT COUNT(*) from big_tbl;
```

```
res = DBI::dbGetQuery(db, query)
res
```

```
  COUNT(*)
1      100
```

```
DBI::dbGetQuery(db, glue("SELECT COUNT(ind) from {table_name};")) |> unname()
```

```
1 100
```

```
nrow(big_tbl)
```

```
[1] 100
```

SQLite: Reading

```
query = glue("SELECT * FROM {table_name};")
tbl = DBI::dbGetQuery(db, query)
nrow(tbl)
```

```
[1] 100
```

```
cbind(big_tbl[1:3, ], tbl[1:3, ])
```

	ind		x		y	ind		x		y
1	1	0.1137034	-1.8060313	1	0.1137034	-1.8060313				
2	2	0.6222994	-0.5820759	2	0.6222994	-0.5820759				
3	3	0.6092747	-1.1088896	3	0.6092747	-1.1088896				

```
all.equal(big_tbl, tbl)
```

```
[1] TRUE
```

SQLite: Reading

```
ind_low = 3
ind_high = 5
query = glue("SELECT * FROM {table_name} WHERE ind >= {ind_low} AND ind <= {ind_high};")
query
```

```
SELECT * FROM big_tbl WHERE ind >= 3 AND ind <= 5;
```

```
sub_tbl = DBI::dbGetQuery(db, query)
sub_tbl
```

	ind	x	y
1	3	0.6092747	-1.1088896
2	4	0.6233794	-1.0149620
3	5	0.8609154	-0.1623095

```
all.equal(big_tbl[ind_low:ind_high, ], sub_tbl, check.attributes = FALSE)
```

```
[1] TRUE
```


SQLite: Closing the Connection

```
db
```

```
<SQLiteConnection>  
  Path: /tmp/db.sqlite  
  Extensions: TRUE
```

```
DBI::dbDisconnect(db)  
db
```

```
<SQLiteConnection>  
  DISCONNECTED
```

- Now we'll swap over to Python
- All the queries we ran before still work
- Interface is a bit different
 - Attach to DB like before
 - Create "cursor" object
 - Execute queries on this object
 - But mostly we'll interact via Pandas

SQLite: Setup

```
import sqlite3
import pandas as pd

table_name = "big_tbl"
db_file = "/tmp/db.sqlite"

db = sqlite3.connect(db_file)
db
```

<sqlite3.Connection object at 0x7fb31c99f340>

SQLite: Inspection

```
query = f'pragma table_info({table_name});'
```

```
'pragma table_info(big_tbl);'
```

```
cur = db.cursor()  
cur.execute(query)
```

```
<sqlite3.Cursor object at 0x7fb31c807ac0>
```

```
cur.fetchall()
```

```
[(0, 'ind', 'INTEGER', 0, None, 0), (1, 'x', 'REAL', 0, None, 0), (2, 'y', 'REAL', 0, None, 0)]
```

SQLite: Inspection

```
query = f'SELECT COUNT(*) from {table_name};'  
cur.execute(query).fetchall()
```

```
[(100,)]
```

SQLite: Reading

```
query = f'SELECT * from {table_name}'  
df = pd.read_sql_query(query, db)  
print(df.head())
```

	ind	x	y
0	1	0.113703	-1.806031
1	2	0.622299	-0.582076
2	3	0.609275	-1.108890
3	4	0.623379	-1.014962
4	5	0.860915	-0.162310

SQLite: Reading

```
ind_low = 3
ind_high = 5
query = f'SELECT * FROM {table_name} WHERE ind >= {ind_low} AND ind <= {ind_high}'
query
```

```
'SELECT * FROM big_tbl WHERE ind >= 3 AND ind <= 5'
```

```
df = pd.read_sql_query(query, db)
df
```

	ind	x	y
0	3	0.609275	-1.108890
1	4	0.623379	-1.014962
2	5	0.860915	-0.162310

Wrapup

Wrapup

- SQL is a convention for managing DB data.
- Database management is really hard - beyond the scope of the course.
- Appending to a table and extracting from a table are pretty easy (most of the time...).
- Next time: more SQL and SQLite
- Have a great weekend!

Questions?

