Databases for Data Science

Lecture 12 · 2022-10-12

Today

- Discussion of other databases
    - SQLite
- PostgreSQL support for JSON
- Miscellaneous topics
- Extra practice

All sorts of SQLs

Flavor	Developer	FOSS
PostgreSQL	UC Berkeley → Open-source contributors	✅
MySQL	Oracle	❌
MariaDB	MySQL fork → Open-source contributors	✅
SQL Server	Microsoft	❌
IBM Db2	IBM	❌
SQLite	Open-source contributors	✅

SQLite

PostgreSQL and most other database engines have a standalone **server process**.

This runs separately from the clients, like the `psql` console and `psycopg2`, that connect to it.

SQLite, instead, is a **library** that works in the same process as your own code.

some properties:

	PostgreSQL	SQLite
Database engine runs in…	A standalone server process	The same process as your code
Storage	Multiple files on-disk	One `.db` file (or in-memory)
When you run a query…	Your client sends an HTTP request to the server	Library code is called directly
Write-concurrency	Isolated transactions	One writer at a time

SQLite

Why use SQLite?

SQLite

Why use SQLite?

- Easier to setup and use
- Runs in restricted environments (phones, embedded)
- Instances are isolated by default

So, why use a client/server system like Postgres?

SQLite

Why use SQLite?

- Easier to setup and use
- Runs in restricted environments (phones, embedded)
- Instances are isolated by default

So, why use a client/server system like Postgres?

- Works with remote clients
- Better concurrency features
- Handles practically unlimited amounts of data

further reading: https://www.sqlite.org/whentouse.html

SQLite: Practice

On CS1:

```
// copy the `orders.csv` file to your working directory
cp /usr/share/databases/orders.csv .

// start SQLite with a new database file
sqlite3 lec12.db
```

Next, in the SQLite console:

```
.import orders.csv raw_data

SELECT * FROM raw_data;
```

Exercise: In SQLite, split this data into `orders` and `customers` tables.
Each order should have a foreign key to a customer.

PostgreSQL JSON features

PostgreSQL added JSON support in v9.2 (2012).

- (SQLite followed suit in v3.38.0 - earlier this year!)

It's still a relational database, so how does this work?

- Tables can have `json`- and `jsonb`-typed columns
  - `json` is stored as plaintext, while `jsonb` is more efficient
- We have new operators and functions for working with JSON data in SQL.

JSON in PSQL

```sql
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    title TEXT,
    info JSONB
);
INSERT INTO products (title, info)
VALUES
    (
        'Chalk', -- We use single quotes to start/end the JSON and double quotes inside it
        '{ "manufacturer": "Crayola", "sizes": [12,24,48], "unit_price": 0.20 }'
    ),
    (
        'USB Cable',
        '{
            "manufacturer": "Anker",
            "skus": [
                {"length": 1.5, "color": "black"},
                {"length": 6.0, "color": "white"}
            ]
        }'
    );
```

Exercise: On your own computer, create a `products` table and insert some records with JSON.

JSON in PSQL : Queries

The arrow operator allows us to extract nested properties from the JSON.

```sql
SELECT
    title,
    info -> 'manufacturer' AS manufacturer,
    info -> 'skus' AS skus,
    info -> 'skus' -> 0 AS sku_0,
FROM products;
```

Properties that are missing will be mapped to `NULL` values.

Exercise
Insert a product with a deeply nested property, then write a query to select that property.

Exercise
Insert some products with `color` and `price` fields, e.g. `'{"color": "blue", "price": 12.99 }'`
Take the average price grouped by color.

JSON in PSQL : Operators

Two operators:

- `->` returns a JSON data structure
- `->>` converts its output to a text string

```
WITH step1 AS (
    SELECT
        title,
        info ->  'manufacturer' AS mfg_json,
        info ->> 'manufacturer' AS mfg_text
    FROM products
)
SELECT
    title,
    mfg_json,
    pg_typeof(mfg_json),
    mfg_text,
    pg_typeof(mfg_text)
FROM step1;
```

JSON in PSQL : Arrays

We have multiple options for dealing with arrays.

Using specific indices:

```
SELECT
    title,
    info -> 'manufacturer' AS manufacturer,

    -- Select array elements by index
    info -> 'skus' -> 0 AS sku_0,
    info -> 'skus' -> 1 AS sku_1,
FROM products;
```

Using functions to transform the array:

```
SELECT
    title,
    info -> 'manufacturer' AS manufacturer,

    -- Unroll the array into separate rows
    json_array_elements(info -> 'skus') as sku,

    -- We can then process the individual array elements
    json_array_elements(info -> 'skus') -> 'color' as color

FROM products;
```

Exercise

Create a new SKU table with columns `sku_id SERIAL PRIMARY KEY, color TEXT, info JSONB`.
Write a query that selects SKUs from `products` and inserts them into the new table.

PostgreSQL vs MongoDB

Postgres is competitive with Mongo in performance.

So, why use MongoDB?

PostgreSQL vs MongoDB

Postgres is competitive with Mongo in performance.

So, why use MongoDB?

- Preference for its query language / workflows
- Working with highly unstructured data
- Better support for *sharding*
  - Distributing data across a wide set of computers

further reading: https://www.mongodb.com/compare/mongodb-postgresql

Other NoSQL options

In general, more diversity than in the SQL world.

Flavor	Developer
MongoDB	Document store
CouchDB	Document store
HBase	Column-oriented
Neo4j	Graph database
DynamoDB	Key/value store (cloud)
Redis	Key/value store (in-memory)

Note: The CAP Theorem

> “ Consistency, availability, and partition tolerance - pick two. ”

- Consistency: All replicas appear to have the same state.
- Availability: Operations are always possible.
- Partition Tolerance: The system remains consistent if replicas are disconnected.

This is true of distributed systems in general.

Open Q&A / Exercises

thank you!