Databases

Dr Gianluca Campanella

28th June 2016

Contents

Databases

Databases manage…

- **Storage** of information

- **Querying** of data

→ Structured Query Language (SQL)

- **Management** of datasets (verification/consistency) and user access rights (permissions)

Relational databases

Relational databases

- Organised in **tables** (‘entities’ or ‘concepts’)
- Each table is like a `DataFrame`, and has a **schema** describing data types and constraints
- In addition, tables have **keys** that serve as identifiers (**primary key**) or indices (**secondary keys**)

Database management systems (DBMS)

Open source

- MySQL and derivatives
- PostgreSQL

Commercial

- Microsoft SQL Server
- Oracle

Normalisation

A **normalised** database has:

- One table per entity
- Many foreign keys and/or associative tables

Normalisation

A **normalised** database has:

- One table per entity
- Many foreign keys and/or associative tables

Pros

- Minimal data duplication
- Saves storage space

Cons

- Data is split across different tables
- Requires joins to ‘reconstruct’

Other database types

Key-value stores

A **key-value store**...

- Is like a Python **dictionary**, but not limited to available memory
- Uses **caching** strategies to ensure quick access to commonly/recently accessed items

Examples

- Apache Cassandra
- Oracle NoSQL Database

NoSQL databases

A **NoSQL database**…

- Organises data in (partly normalised) ‘entities’ that allow for **nesting**
- Typically describes data using **JSON**

Examples

- Apache CouchDB
- MongoDB

Structured Query Language (SQL)

`SELECT`ing data

Syntax

```
SELECT <columns>
FROM <table>
WHERE <condition>
```

Notes

- `SELECT *` will select **all columns**
- `WHERE` can be omitted to retrieve **all rows**

SELECTing data

Example

```
SELECT store, sales
FROM global_sales
WHERE country == ‘UK’
```

GROUPing

Syntax

```
SELECT STATISTIC(<column>), …
FROM <table>
…
GROUP BY <index>
```

Notes

- `GROUP BY` must be paired with a `STATISTIC` such as `COUNT(*)`, `SUM`, `AVG`, `MIN` and `MAX`
- `GROUP BY` can be omitted to aggregate over **all rows**

GROUPing

Example

```
SELECT store, SUM(sales)
FROM global_sales
WHERE country == ‘UK’
GROUP BY store
```

ORDERing

Syntax

```
SELECT <columns>
FROM <table>
…
ORDER BY <indices> [DESC]
```

Notes

- Default sorting is in `ASC` order
- Can also `ORDER BY` multiple columns

ORDERing

Example 1

```
SELECT country, city, store
FROM global_sales
ORDER BY country, city
```

Example 2

```
SELECT store, SUM(sales) AS total_sales
FROM global_sales
ORDER BY total_sales DESC
```

`JOIN`ing

Syntax

```
SELECT <columns>
FROM <table>
JOIN <table>
ON <condition>
…
```

Notes

- Performs an inner join (matching both tables)
- Outer joins (`LEFT`, `RIGHT`, or `FULL`) can also be performed

JOINing

Example

```
SELECT st.city, st.store, SUM(sa.sales) AS total_sales
FROM stores AS st
JOIN global_sales AS sa
ON st.store == sa.store
WHERE st.country == ‘UK’
GROUP BY st.country
ORDER BY total_sales DESC
```