Introduction to databases

Gianluca Campanella

Contents

Databases

Databases manage...

- Storage of information
- Querying and retrieval

$\rightarrow$ Structured Query Language (SQL)

- Consistency and access rights (permissions)

Relational databases

Relational databases

- Organised in tables (‘entities’ or ‘concepts’)
- Each table has a schema describing data types and constraints
- In addition, tables have keys that serve as…
  - Identifiers (primary key)
  - 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

- 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 or recently accessed items

Examples

- Apache Cassandra
- Oracle NoSQL Database

NoSQL databases

A NoSQL database…

- Organises data in ‘entities’ that allow for nesting
- Typically describes data using JSON

Examples

- Apache CouchDB
- MongoDB

Structured Query Language

Selecting data

Syntax

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

Notes

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

Selecting data

Example

```sql
SELECT store, sales
FROM global_sales
WHERE country == 'UK'
```

Grouping

Syntax

```
SELECT STATISTIC(<column>), ...
FROM <table>
...
GROUP BY <indices>
```

Notes

- Usually paired with a `STATISTIC` such as **COUNT**, **SUM**, **AVG**, **MIN**, or **MAX** computed within groups
- **GROUP BY** can be omitted to aggregate over all rows

Grouping

Example

```sql
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**ending order
- Can also **ORDER BY** multiple columns

Ordering

Example 1

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

Example 2

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

Joining

Syntax

```
SELECT <columns>
FROM <table>
JOIN <table> ON <conditions>
...
```

Notes

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

Joining

Example

```sql
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
```