NoSQL Databases

Mashenkov Timofei
mashfeii

March 22, 2025

1. Types of Databases

- **NoSQL**
  - Document Stores
  - Key-Value Stores
  - Column-Family Stores
  - Graph Databases
- **NewSQL**
  1. Google Spanner
  2. CockroachDB
  3. VoltDB
- **Graph**
  - Neo4j
  - JanusGraph
- **Time Series**
  - InfluxDB
  - TimescaleDB
  - OpenTSDB
- **In-Memory**
  - Redis
  - Memcached
- **Object Storage**
  - Amazon S3
  - Google Cloud Storage

1.1. NoSQL Databases

1.1.1. Document Stores

- **MongoDB -** Document Store
- **CouchDB -** Stores data as JSON documents and supports RESTful API
- **ArangoDB -** Multi-model database
- **RethinkDB -** Real-time database, pushes data updates to connected clients

1.1.2. Column-Family Stores

- **Apache Cassandra** - Distributed database system
- **Apache HBase** - Open-source implementation of Google BigTable (almost unused nowadays)
- **ScyllaDB** - High-performance NoSQL database

2. Why are there so many types of databases?

Let's try as an example build a simple social network, initially we contain the following entities:

- Users
- Posts
- Comments

2.1. SQL Databases: PostgreSQL

Advantages:

- The data is normalized
- The application is working
- Fixes are made with standard DB toolkit

Disadvantages:

- To get the data we had to write a lot of SQL queries
- Since it gets a lot of data, it gets very slow
- Front-end fixes are done with hands and often with downtime

2.1.1. NoSQL Databases: MongoDB

Advantages:

- There is no complex joins
- Flexible schema - we can add whatever we want
- Faster reading
- Horizontal scaling

Disadvantages:

- No ACID transactions
- Complex connections are hard and expensive to make

2.1.2. How to Split the Data into Collections?

Nested:

- Data goes together in requests
- Object always lives in the parent's context
- Nested objects weight limited

Linked/Separate:

- Data is used separately
- Object could have many links to other objects
- Avoid data duplication

3. Analytics

3.1. Analytical Queries in Production

Advantages:

- Data is always up-to-date
- Just write queries

Disadvantages:

- Production is getting slow and can be down
- There are laws regulating the user data

3.2. OLTP - Online Transaction Processing

- **Goal** - fast transaction processing
- **Example** - web-shops, bank systems, CRM, ERP
- **Data Structure** - normalized (3NF) databases
- **Data Volume** - small to medium
- **Frequency of Operations** - high (many small transactions per second)
- **Query Types** - simple `SELECT`, `INSERT`, `UPDATE`, `DELETE`
- **Consistency** - high, since data should be consistent in real-time

3.3. OLAP - Online Analytical Processing

- **Goal** - data analysis, reports making, identifying trends/patterns
- **Example** - data warehouses, business intelligence systems
- **Data Structure** - denormalized schemas (star, snowflake) for fast access
- **Data Volume** - large
- **Frequency of Operations** - low (few complex queries per hour)
- **Query Types** - aggregations (`SUM`, `AVG`, `COUNT`), `GROUP BY`
- **Consistency** - eventual, since data is not required to be up-to-date

4. Graph Databases

The main idea is to store connections between entities. For example, we need to store common friends between users.

When to use?

- You need to store complex connections between objects (one-to-many, many-to-many)
- If queries are connected with some depth (e.g. "Who is a friend of a friend through a friend?")
- If data is too heterogeneous (different types of objects and connections)

Caution:

- Avoid super-nodes (nodes with a lot of connections)
- Use directed connections
- Use right granularity (not too many connections) for spreading the data