NoSQL & MongoDB

Sinziana Gafitanu

	SQL Databases	NOSQL Databases
Types	One type (SQL database) with minor variations	Many different types including key-value stores, document databases, wide-column stores, and graph databases
Development History	Developed in 1970s to deal with first wave of data storage applications	Developed in late 2000s to deal with limitations of SQL databases, especially scalability, multi-structured data, geo-distribution and agile development sprints
Examples	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Data Storage Model	Individual records (e.g., 'employees') are stored as rows in tables, with each column storing a specific piece of data about that record (e.g., 'manager,' 'date hired,' etc.), much like a spreadsheet. Related data is stored in separate tables, and then joined together when more complex queries are executed. For example, 'offices' might be stored in one table, and 'employees' in another. When a user wants to find the work address of an employee, the database engine joins the 'employee' and 'office' tables together to get all the information necessary.	Varies based on database type. For example, key-value stores function similarly to SQL databases, but have only two columns ('key' and 'value'), with more complex information sometimes stored as BLOBs within the 'value' columns. Document databases do away with the table-and-row model altogether, storing all relevant data together in single 'document' in JSON, XML, or another format, which can nest values hierarchically.

Source: https://www.mongodb.com/nosql-explained

Schemas	Structure and data types are fixed in advance. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.	Typically dynamic, with some enforcing data validation rules. Applications can add new fields on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary. For some databases (e.g., wide-column stores), it is somewhat more challenging to add new fields dynamically.
Scaling	Vertically, meaning a single server must be made increasingly powerful in order to deal with increased demand. It is possible to spread SQL databases over many servers, but significant additional engineering is generally required, and core relational features such as JOINs, referential integrity and transactions are typically lost.	Horizontally, meaning that to add capacity, a database administrator can simply add more commodity servers or cloud instances. The database automatically spreads data across servers as necessary.
Development Model	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
Supports Transactions	Yes, updates can be configured to complete entirely or not at all	In certain circumstances and at certain levels (e.g., document level vs. database level)

Source: https://www.mongodb.com/nosql-explained

Data Manipulation	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs
Consistency	Can be configured for strong consistency	Depends on product. Some provide strong consistency (e.g., MongoDB, with tunable consistency for reads) whereas others offer eventual consistency (e.g., Cassandra).

Source: https://www.mongodb.com/nosql-explained

NoSql Database Types

- **Key-value stores** are the simplest. Every item in the database is stored as an attribute name (or "key") together with its value. Riak, Voldemort, and Redis are the most well-known in this category.
- **Wide-column stores** store data together as columns instead of rows and are optimized for queries over large datasets. The most popular are Cassandra and HBase.
- **Document databases** pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents. MongoDB is the most popular of these databases.
- **Graph databases** are used to store information about networks, such as social connections. Examples are Neo4J and HyperGraphDB.

Source: https://www.mongodb.com/scale/types-of-nosql-databases

NoSql

- **Column**: Accumulo, Cassandra, Druid, HBase, Vertica, SAP HANA
- **Document**: Apache CouchDB, ArangoDB, Clusterpoint, Couchbase, DocumentDB, HyperDex, IBM Domino, MarkLogic, MongoDB, OrientDB, Qizx, RethinkDB
- **Key-value**: Aerospike, ArangoDB, Couchbase, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, InfinityDB, MemcacheDB, MUMPS, Oracle NoSQL Database, OrientDB, Redis, Riak, Berkeley DB
- **Graph**: AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso, Stardog
- **Multi-model**: Alchemy Database, ArangoDB, CortexDB, Couchbase, FoundationDB, InfinityDB, MarkLogic, OrientDB

Source: https://en.wikipedia.org/wiki/NoSQL

MongoDB

- NoSQL
- Open Source
- Cross Platform
- Document Database

Start playing with MongoDB

1. Install Docker
2. Run:

   ```
   docker run --rm --name mongo -p 27017:27017 -d mongo:3.4
   ```

Using MongoDB + NodeJS

Install:

```
npm install mongodb --save
```

Connecting to MongoDB from Node:

```
MongoClient.connect('mongodb://127.0.0.1:27017/testing', function(err, db) {
    if(err)
        throw err;
    console.log("connected tp MongoDB !");
});
```

Tip: Use Monk for simple use cases

Monk is a tiny wrapper for MongoDB in NodeJS that helps manage the connection to the database.

Monk: https://github.com/Automattic/monk

Demo