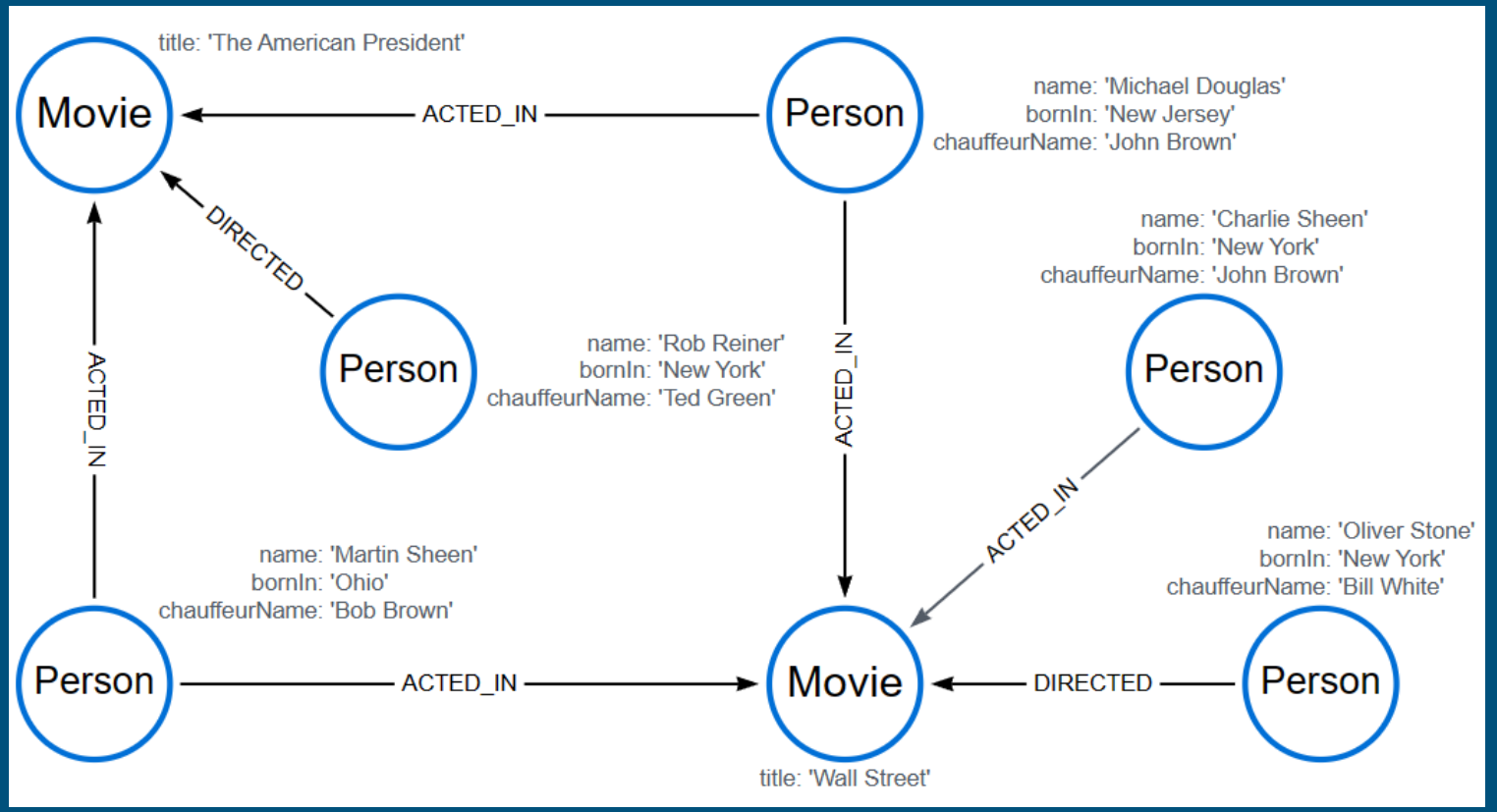
Databases

Lecture 12

Non-relational databases: Redis, ClickHouse, MongoDB

0. Some extra words about Neo4j

- Let's imagine that we are importing data from a relational database into Neo4j
- In Neo4j, it is more rational to distribute records from the table across separate nodes. Thus, each node in its header will contain the name of the relationship from the RDB, and in its properties - one record from the table. This allows duplication of nodes with the same headers (labels)



Документация по языку запросов Cypher - [Cypher Cheat Sheet - Neo4j Community Edition]

I. A short addition about licenses

- At the moment, not all software products are equally accessible. When choosing a particular solution, be sure to pay attention to the project licenses and the “edition” of the delivery
- The easiest way to find license information as a first approximation is an English-language Wikipedia article about the product you are using.
- Consider the environment in which you plan to deploy the service. For example, the Yandex cloud supports MongoDB 5.0 and 6.0 server clusters in the Enterprise edition

II. Reddis

Redis = Remote Dictionary Server

- This is a key-value store in RAM
- Possibility of saving to disk
- Supports replication according to the master-slave scheme. There is a main node with which work is carried out - master, and there is also a slave to which replication occurs.
- Supports the event mechanism according to the “publish-subscribe” scheme
- Written in C, libraries available for many languages
- Runs on Unix-like OS
- There is a port for Windows, which works with reservations, but is suitable for development
- The code is open source on Github

Redis

- Architecture:
  - redis-server is a process responsible for processing and storing data in RAM (by default the process listens to 127.0.0.1:6379)
  - redis-cli – interactive terminal utility
- Some commands:
  - redis-cli -h 127.0.0.1 -p 6379 – connection to the database
  - Adding data, getting data:
    - SET bike:1 "Process 134"
    - GET bike:1
  - Adding data as a dictionary and getting the field value:
    - HSET bike:1 model Deimos brand Ergonom type 'Enduro bikes' price 4972
    - HGET bike:1 model
  - ->"Deimos"

Redis

Why do we need Redis?

- The first option is to use it as a cache
- Why not keep the data in application memory? The answer here is quite simple - the data in the application memory will be constantly cleared when the application is stopped. This way Redis gives cache resiliency.
- Ability to scale the system
- If the cache is in the application's memory, it cannot be used across multiple instances of the application.

III. ClickHouse

ClickHouse

ClickHouse is a “columnar database management system (DBMS) for online analytical query processing (OLAP)”

- Properties:
  - “A truly columnar DBMS” ☺
  - Data compression
  - Storing data on disk
  - Parallel request processing on many processor cores
  - Large queries are naturally parallelized, using all the necessary resources available on the server
- SQL support
  - ClickHouse supports a declarative query language based on SQL and in many cases aligned with the SQL standard.
  - GROUP BY, ORDER BY, subqueries in FROM, IN, JOIN sections, window functions, as well as scalar subqueries are supported.
- Dependent subqueries are not supported but may become available in the future.

ClickHouse

Potential disadvantages:

- Lack of full-fledged transactions
- There is no ability to modify or delete previously recorded data with low latency and high request rates. There is a mass deletion and modification of data to clear what is no longer needed
- The sparse index makes ClickHouse poorly suited for point reads of single rows against its keys

ClickHouse

Some commands:

- Connection: clickhouse-client -d example_db
- Creating and filling a table:
  - CREATE TABLE example_table (
  - id Uint32,
  - name String,
  - age Uint32
  -) ENGINE = MergeTree() ORDER BY id;

  - INSERT INTO example_table (id, name, age) VALUES (1, 'Alice', 25), (2, 'Bob', 30), (3, 'Charlie', 35);

ClickHouse

Some commands:

- ClickHouse does not support traditional row deletion using DELETE; instead, data filtering is used through table alteration mechanisms. However, you can use ALTER TABLE to delete rows based on conditions:
- ALTER TABLE example_table DELETE WHERE age > 30;

IV. MongoDB

MongoDB

- Data is stored as documents in a JSON-like (BSON) format. Each document may have a different structure. Fields in documents can vary, making it easy to change the data structure.
- Optimal support for horizontal scaling: MongoDB supports sharding - data can be distributed across multiple servers for horizontal scaling.
- Transaction support available: Since version 4.0, MongoDB supports multi-document ACID transactions

MongoDB

Some example commands:

- mongo – connection to a process
- use <DB_name>– context switching or creating a database
- db.users.insert({"name": "John", "age": 30, "email": "john@example.com"}) - adding data to the users collection
- db.users.remove({"name": "John"}) – removal
- db.users.findOne({"name": "John"}) – search for one document by criterion
- db.users.find({"age": {"$gt": 20}}) – search for all documents by criterion
- db.users.createIndex({"email": 1}) – creating an index on a field
- db.users.aggregate([{"$group" : {"_id": "$age", "count": {"$sum": 1}}}]) - aggregation for grouping data by age and counting the number of users in each group