Базы данных

Лекция 12

Нереляционные БД: Redis, ClickHouse, MongoDB

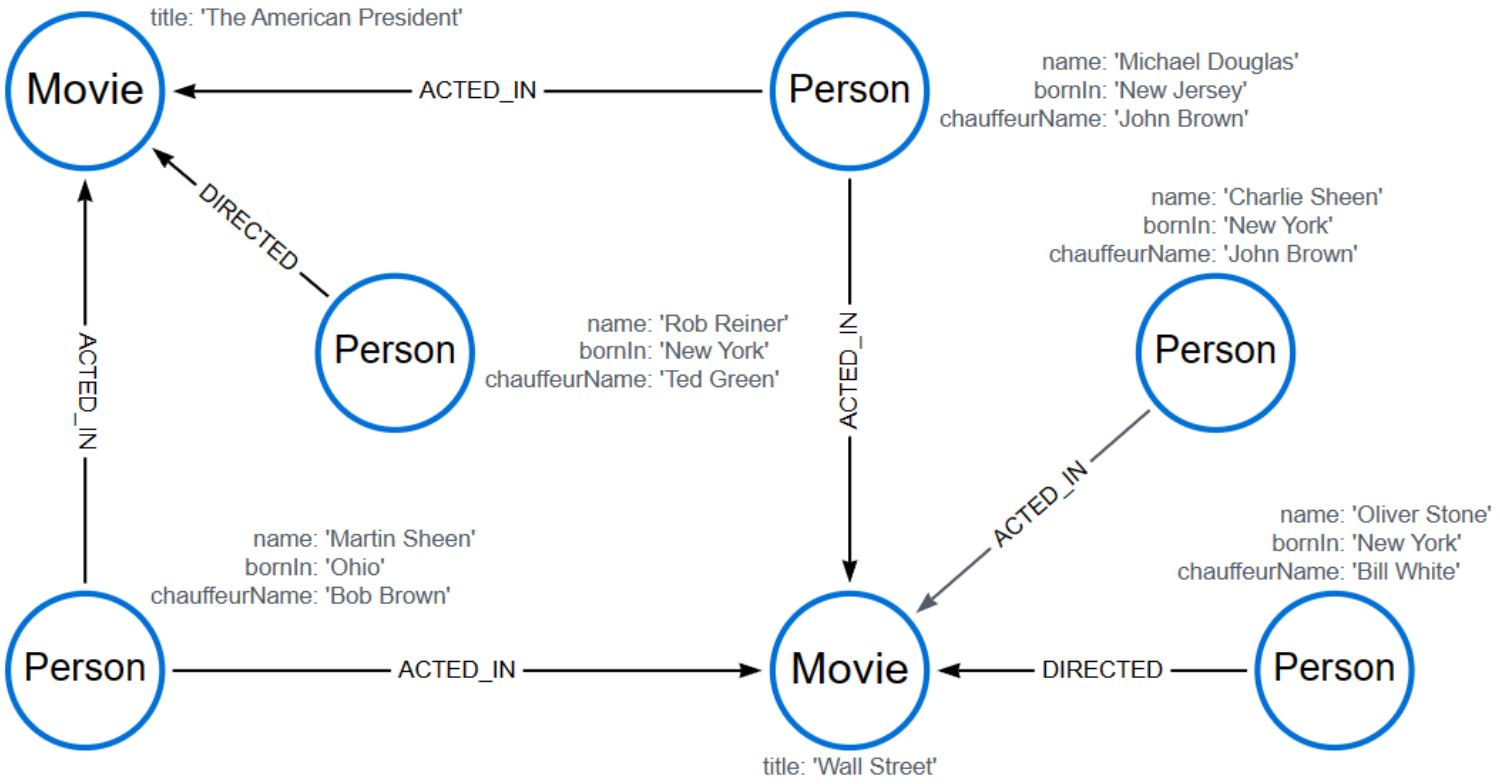
МФТИ, 2024

Игорь Шевченко

@igorshvch

0. Несколько дополнительных слов о Neo4j

- Представим, что мы импортируем данные из реляционной БД в по Neo4j
- В Neo4j рациональнее распределять записи из таблицы по отдельным узлам. Таким образом, каждый узел в своем заголовке будет содержать имя отношения из РБД, а в своих свойствах – одну запись из таблицы. Таким образом допускается дублирование узлов с одинаковыми заголовками (labels)



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

I. Короткое дополнение о лицензиях

- На данный момент не все программные продукты одинаково доступны. При выборе того или иного решения обязательно обращайте внимание на лицензии проекта и «редакцию» поставки
- Самый простой способ найти данные о лицензии в первом приближении – англоязычная статья в википедии об используемом продукте.
- Обратите внимание на среду, в которой планируется развертывание сервиса. Например, в облаке Яндекса поддерживаются кластеры серверов MongoDB 5.0 и 6.0 в редакции Enterprise

II. Reddis

Redis

- Remote Dictionary Server
- Это хранилище ключ-значение в оперативной памяти
- Возможно сохранение на диск
- Поддерживает репликацию по схеме «master-slave». Есть основной узел, с которым идет работа – master, а также есть slave, на который идет репликация.
- Поддерживает событийный механизм по схеме «publish-subscribe»
- Написан на C, доступны библиотеки под многие языки
- Работает на Unix-подобных ОС
- Есть порт для Windows, который работает с оговорками, но пригоден для разработки
- Открытый исходный код есть на Github

Redis

- Архитектура:
  - redis-server – процесс, отвечающий за обработку и хранение данных в оперативной памяти (по умолчанию процесс слушает 127.0.0.1:6379)
  - redis-cli – интерактивная утилита для терминала
- Некоторые команды:
  - redis-cli -h 127.0.0.1 -p 6379 – подключение к БД
  - Добавление данных, получение данных:
    - SET bike:1 "Process 134"
    - GET bike:1
  - Добавление данных в виде словаря и получение значения поля:
    - HSET bike:1 model Deimos brand Ergonom type 'Enduro bikes' price 4972
    - HGET bike:1 model

    ->"Deimos"

Redis

- Зачем вообще использовать Redis?
- Первый вариант – использовать как кэш
  - Почему не держать данные в памяти приложения? Ответ здесь довольно простой – данные в памяти приложения будут постоянно чиститься при остановке приложения. Таким образом Redis дает устойчивость кэша.
- Возможность масштабировать систему
  - Если кэш в памяти приложения, то его нельзя будет использовать в нескольких его экземплярах.

III. ClickHouse

ClickHouse

- ClickHouse — «столбцовая система управления базами данных (СУБД) для онлайн-обработки аналитических запросов (OLAP)»
- Особенности:
  - «По-настоящему столбцовая СУБД» ☺
  - Сжатие данных
  - Хранение данных на диске
  - Параллельная обработка запроса на многих процессорных ядрах
    - Большие запросы естественным образом распараллеливаются, используя все необходимые ресурсы из доступных на сервере
  - Поддержка SQL
    - ClickHouse поддерживает декларативный язык запросов на основе SQL и во многих случаях совпадающий с SQL-стандартом.
    - Поддерживаются GROUP BY, ORDER BY, подзапросы в секциях FROM, IN, JOIN, функции window, а также скалярные подзапросы.
    - Зависимые подзапросы не поддерживаются, но могут стать доступными в будущем.

ClickHouse

- Потенциальные недостатки:
  - Отсутствие полноценных транзакций
  - Возможность изменять или удалять ранее записанные данные с низкими задержками и высокой частотой запросов не предоставляется. Есть массовое удаление и изменение данных для очистки более не нужного
  - Разреженный индекс делает ClickHouse плохо пригодным для точечных чтений одиночных строк по своим ключам

ClickHouse

- Некоторые команды:
  - Подлкючение: clickhouse-client -d example_db
  - Создание и наполнение таблицы:

```
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

- Некоторые команды:
  - ClickHouse не поддерживает традиционное удаление строк с помощью DELETE, вместо этого используется фильтрация данных через механизмы альтерации таблицы. Однако можно использовать ALTER TABLE для удаления строк на основе условий:

```
ALTER TABLE example_table DELETE WHERE age > 30;
```

IV. MongoDB

MongoDB

- Данные хранятся в виде документов в формате, похожем на JSON (BSON). Каждый документ может иметь разную структуру. Поля в документах могут различаться, что позволяет легко менять структуру данных.
- Оптимальная поддержка горизонтального масштабирования: MongoDB поддерживает шардинг - данные могут быть распределены по нескольким серверам для горизонтального масштабирования.
- Доступна поддержка транзакций: начиная с версии 4.0, MongoDB поддерживает мульти-документные ACID транзакции

MongoDB

- Некоторые примеры команд:
  - mongo – подключение к процессу
  - use <DB_name>– переключение контекста или создание БД
  - db.users.insert({"name": "John", "age": 30, "email": "john@example.com"}) - добавление данных в коллекцию users
  - db.users.remove({"name": "John"}) – удаление
  - db.users.findOne({"name": "John"}) – поиск одного документа по критерию
  - db.users.find({"age": {"$gt": 20}}) – поиск всех документов по критерию
  - db.users.createIndex({"email": 1}) – создание индекса по полю
  - db.users.aggregate([

    {"$group" : {"_id": "$age", "count": {"$sum": 1}}}

    ]) - агрегация для группировки данных по возрасту и подсчета количества пользователей в каждой группе