Инструментальные средства информационных систем

БИН-19-1

Лекция 3 Виды СУБД

Содержание

1 Введение

Сегодня мы рассмотрим с вами не то чтобы верхушку айсберга под названием СУБД (системы управления базами данных, DBMS – Database Management System), но, скажем так, рассмотрим три пикселя с фотографии верхушки айсберга. Изучению СУБД нужно посвятить как минимум два семестра университета и, как бы мне не хотелось запихнуть всё в одну лекцию, это просто невозможно. Поэтому поговорим о СУБД только как об инструменте.

? Где применяются БД?

Да везде! В сервисах с хранением данных, в фронтенд разработке, в мобильных приложениях, в онлайн играх и так далее.

? А как вы думаете, сколько лет базам данных в их современном понимании?

Базы данных появились в 1960-хх, SQL – в 1974, а Oracle – в 1977, т.е. очень давно.

? Знаете, к чему я клоню?

Конечно, базы данных существуют давно, и до сих пор безусловно актуальны и, почти на всех собеседованиях есть вопросы по SQL. Так что знать основы БД необходимо.

2 Виды БД

База данных – это данные, организованные по некоторым правилам и которыми можно управлять по некоторым правилам.

СУБД – это программное средство, которое позволяет обеспечить создание и использование баз данных.

Существует множество видов БД, вот основные из них:

2.1 Иерархическая БД

Можно рассматривать структуру папок и файлов в компьютере как иерархическую БД, свойства:

– простая реализация,

– понятная структура,

– существует ограничение на связи узлов.

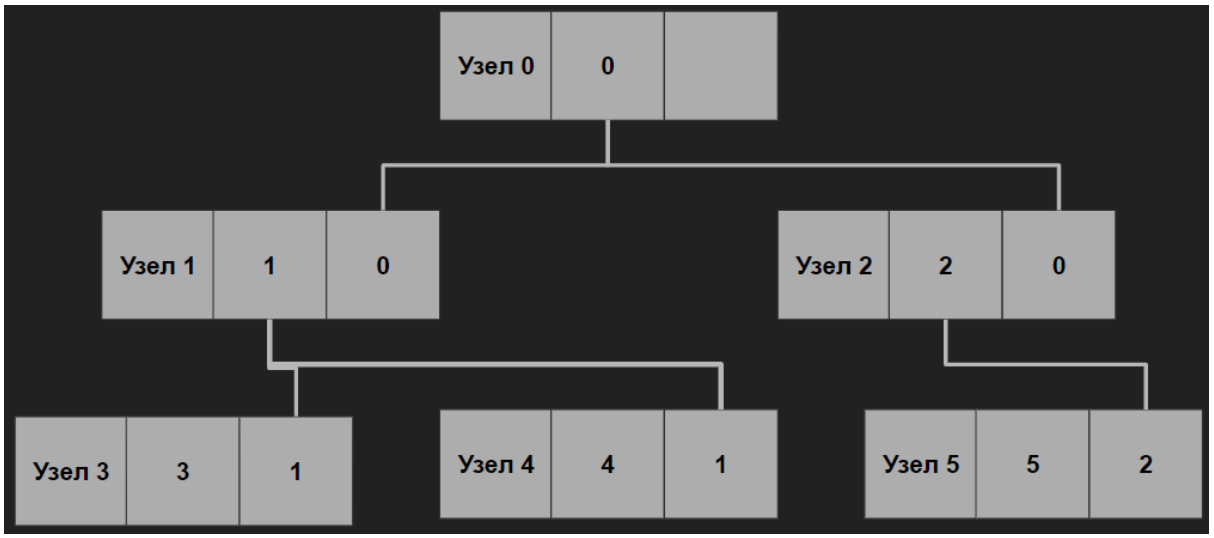

Рисунок 1 – Иерархическая БД

2.2 Графовые БД

В графовых БД основные элементами являются узлы и связи, свойства таких БД:

– нет ограничения на связи узлов,

– сложно поддерживать целостность,

– чем больше становится данных, тем больше возрастает сложность структуры.

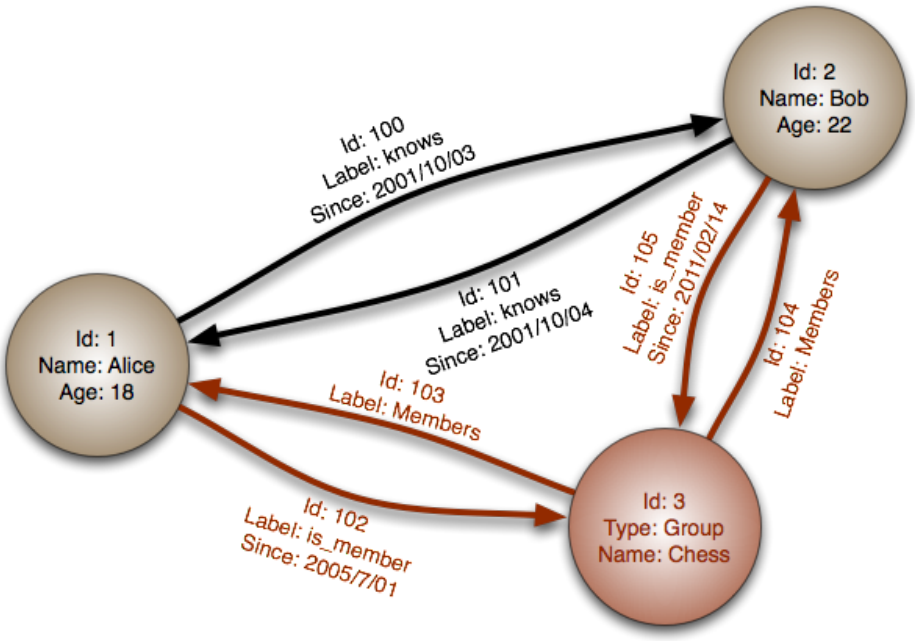

Рисунок 2 – Графовая БД

(Картинка из википедии. Стараюсь не брать данные из википедии, но картинка очень понятная.)

2.3 Реляционная БД

В реляционных БД данные хранятся в двумерных таблицах, связанных между собой (рисунок 3), свойства таких БД:

– гибкость проектирования БД , так как всё хранится в плоских таблицах и одна таблица содержит объекты одного типа,

– простая структура, ссылки на другие объекты являются атрибутами объектов.

К реляционной модели данных относится понятие **нормализации**. Это действия направленные на преобразование способа хранения данных так, чтобы они соответствовали некоторым требованиям, нормальным формам.

Принципы нормализации мы воспринимаем как само собой разумеющееся. Например, что по ID мы получим одну запись или кортеж. Что в атрибуте записано одно значение и т.д.

Отношение находится в **первой нормальной форме (1НФ)** тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение в каждом его атрибуте.

Т.е. на пересечении одной строки и столбца находится одно значение. Все последующие нормальные формы накладываются на предыдущие.

Таблица находится во **второй нормальной форме (2НФ)** тогда и только тогда, когда находится в 1НФ и каждый не ключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.

Т.е. выбирая запись по этому потенциальному ключу, мы всегда получим одну запись, все атрибуты которой относятся к одной записи, связанной с этим ключом.

Денормализация отношений нужна, если мы хотим создать иную модель данных. Например PostgreSQL называют постреляционной БД, так как в ее отношениях нарушается 1НФ, потому что в поле может быть записан массив или JSON документ.

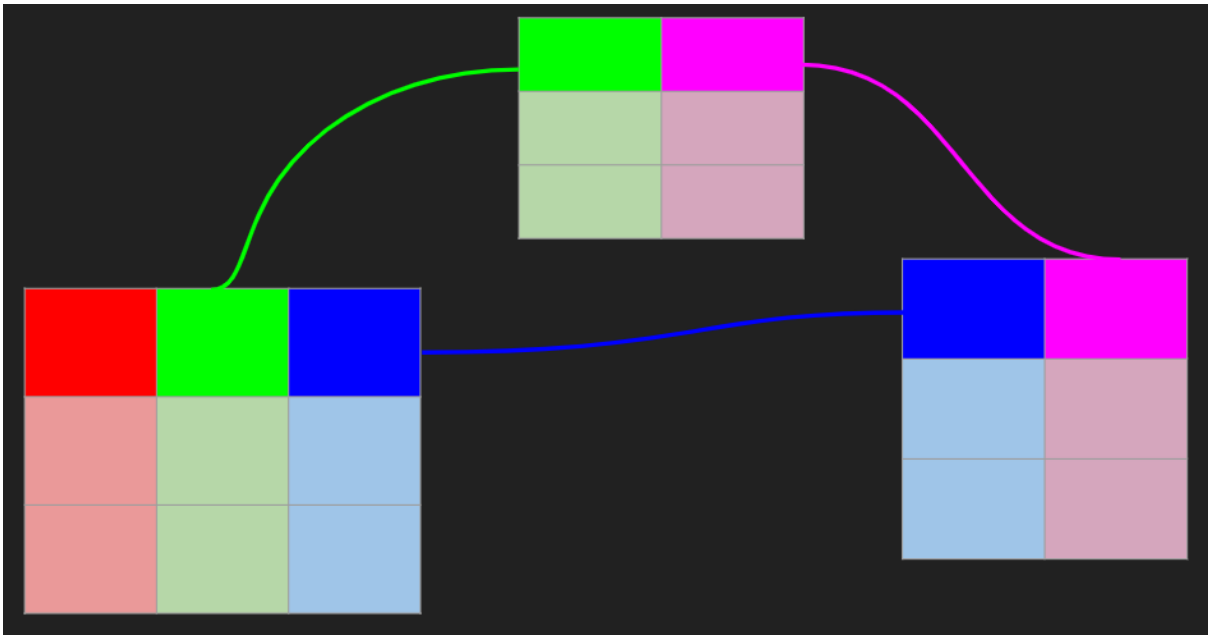

Рисунок 3 – Реляционная БД

Термины реляционных БД:

1. Таблица – Отношение
2. Поле – Колонка – Атрибут
3. Запись – Строка – Кортеж

С реляционной моделью данных тесно связано понятие транзакции, практически все реляционные БД поддерживают транзакции.

(*?Сталкивались с транзакциями?*)

Транзакция – это логическая единица работы с данными, основное свойство которой атомарность, т.е. операция должна выполняться целиком или не выполняться целиком.

Требования к работе транзакций описываются аббревиатурой **ACID**.

Atomicity – Атомарность – транзакция либо выполняется целиком, либо не выполняется целиком.

Consistency – Согласованность – БД остается в корректном состоянии.

Isolation – Изолированность – транзакции не мешают друг другу, т.е. во время работы транзакции над данными, эти данные не доступны другим транзакциям.

Durability – Прочность – результаты транзакций фиксируются в БД.

Реляционные БД не являются панацеей, если нам нужно хранить какие-то документы, например, договоры. Разбивать договоры на части и записывать их в разные таблицы будет очень сложно, кроме того не все договоры будут иметь одинаковый набор атрибутов. Такие объекты не структурированы.

Для работы со слабоструктурированными данными необходимы другие подходы, таким образом появились объектно-ориентированные БД.

2.3 Объектно-ориентированные БД

Объекты в объектно-ориентированной БД отражают объекты реального мира как можно точнее, как они есть. Свойства:

– сложные объекты, но гибкие по структуре,

– частный случай подобных БД: документо-ориентированные БД.

Договоры можно хранить как объекты какого-то набора или коллекции. Внутри каждого объекта может быть организована структура, например JSON документ. (Как это сделано в MongoDB.)

3 SQL, NoSQL и NotOnlySQL

3.1 SQL

Именно для работы с реляционными БД появился SQL.

SQL (Structured Query Language) - декларативный “язык структурированных запросов”.

Запрос на SQL не содержит в себе каких-либо инструкций, например, как получить данные или какую-то последовательность действий, он просто описывает, что мы хотим получить.

MySQL

MySQL очень прост в установке и настройке. (Первый выпуск в 1995.) Он является ведущей СУБД для веб-разработки.

Преимущества:

– СУБД для веб разработки по умолчанию, входит в LAMP, набор (стек) серверного ПО, включающее LINUX + Apache + MySQL + PHP.

– Хорош для старта, есть потенциал масштабирования, можно дописывать, когда сервис станет высоконагруженным.

На самом деле MySQL это платформа для других движков.

InnoDB – высокопроизводительный, поддержка транзакций с точками сохранения. Медленный, но универсальный.

MyISAM (+Aria) – оптимизирован под интенсивное чтение, но не запись. Не поддерживают транзакции и внешние ключи.

MyRocks – оптимизирует место на диске, оптимизирован под интенсивную запись. Написан Facebook.

В 2010 году от MySQL, когда он был куплен Oracle, отпочковалась (создали fork) MariaDB:

– Форк, совместимый в большинстве случаем, включая движки.

– Нет функциональности последних версий (например, работы с геоданными).

– Полностью свободен. Не Oracle.

ORACLE

Компания основана в 1977. Oracle можно описать следующим образом:

– Топ-1 среди СУБД для больших корпораций.

– Имеет экосистему для разработки приложений и сервисов.

– Обладает хорошей масштабируемость, гибкой настройкой оптимизаций, ориентирован на большие данные.

– Платная (проприетарная), дорогая, закрытая. И они могут себе это позволить, так как для БД с большим объёмом данных Oracle является безальтернативной СУБД.

– Предоставляет множество инструментов для анализа данных.

Oracle постоянно анализирует какие запросы выполняются в системе, он может предлагать создать какие-то индексы, а может в автоматическом режиме создавать их сам в фоновом режиме.

Oracle имеет свой язык программирования PL/SQL

– Высокоуровневый язык программирования с оптимизацией SQL.

– Множество типов, возможность структурирования кода.

– Поддерживает ООП.

– Можно разрабатывать веб-приложения.

В некоторых компаниях на PL/SQL написана вся бизнес-логика компании.

PostgreSQL

Первый выпуск в 1996. Преимущества и особенности:

– Хорошая масштабируемость.

– Гибкая, но непростая настройка.

– Удобство работы JSON/XML.

– Вложенные транзакции с точками сохранения.

– Гибкая система блокировок (lock).

– Расширение PostGIS.

– Свободный код, популярная, много расширений.

(*?Несколько раз упоминалось масштабируемость, а кто может сказать что это значит?*)

DDL и администрирование несколько сложнее чем в MySQL. Если чтобы разобраться с MySQL достаточно прочитать документацию, для ORACLE придётся нанять сертифицированных специалистов, то для PostgreSQL придётся нанять специалиста, который разбирается с внутренностями PostgreSQL. Настройка его может показаться не простой и документации не достаточно.

У PostgreSQL тоже есть процедурный язык PL/pgSQL.

MS SQL

Первый выпуск 24 апреля 1989. Особенности и область применения:

– Хорошо интегрируется с продуктами Microsoft (Active Directory, Azure, PowerBI (это расширение Excel и Access)).

– Основная ниша – инфраструктура банков.

– Отличная производительность и аналитические функции.

– Разработка приложений, отчетов и т.д.

– Использует диалект T-SQL.

3.2 NoSQL

NoSQL – совсем не SQL. В нереляционных БД применяется другой подход работы с данными, при этом мы отказываемся от некоторых преимуществ SQL, но взамен получаем другие, необходимые для решения конкретной задачи.

NoSQL это целый класс СУБД, которые решают задачи не по силам обычной реляционной СУБД (РСУБД).

Недостатки РСУБД

- Необходимость описывать схему хранения данных.
- Ограниченность быстродействия архитектурой СУБД и концепциями РСУБД.

Рассмотрим виды и примеры NoSQL БД.

Документо-ориентированные БД

Ключевая особенность – отсутствие схемы, возможность складывать сколь угодно сложные объекты.

Назначение: слабоструктурированные данные, для хранения того, что в реальном мире похоже на документ.

MongoDB

Первый выпуск в 2009. Характеристики:

- Документо-ориентированная СУБД, можно сказать, основная документо-ориентированная СУБД.
- Подходит для хранения и анализа слабоструктурированных объектов.
- Хорошо масштабируется.
- Хранимые процедуры создаются на JavaScript.
- Данные для MongoDB представлены в виде коллекций, в виде JSON документов.

Преимущества:

- не использует схемы данных, лучшее schemaless-решение
- имеет экосистему для работы с данными, мониторинга и администрирования

Недостатки:

- нет ACID-транзакций

- нет JOIN (*?Знаете что такое?*)

Названия сущностей в MongoDB можно соотнести к аналогичным сущностям SQL следующим образом:

- Таблица – Коллекция
- Строка – Документ
- Колонка – Поле

Ключ-значение БД (key-value)

Ключевая особенность – по значению ключа быстро получаем некоторый объект, в таких системах зачастую всё что мешает быстродействию отключено. Такую базу данных нетрудно написать свою для своей какой-то конкретной задачи.

Назначение: оптимизация скорости доступа к данным.

Redis

Первый выпуск в 2009. Характеристики:

- СУБД ключ–значение.
- Основная СУБД для высокочастотных операций.
- Создаются под узкую задачу.
- Хранит все данные в оперативной памяти, поэтому очень быстрая.
- Хорошо масштабируется.
- Поддерживает транзакции.
- Поддерживает временные записи (данные сами удаляются из БД).
- Поддерживает различные структуры данных.

Преимущества:

- пока нет более быстрее
- надежная и популярная СУБД

Недостатки:

- сложность восприятия схемы данных
- масштабируемость требует значительных затрат на железо.

Графовые БД

В графовых БД связаны объекты, а не таблицы, наилучшим образом подходят для поиска транзитивных связей.

3.3 NotOnlySQL

Поддерживают запросы на SQL, но очень далеки от стандарта.

Колоночные БД

В колоночной БД данные хранятся в одной большой таблице, т.е. это фактически некоторый материализованное представление (*?Знают ли студенты?*), в котором данные уже соединены, данные могут дублироваться и такая таблица будет занимать много места, при использовании таких БД приходится думать об оптимизации дискового пространства.

Основная идея: убрать из РСУБД всё, что мешает легкому масштабированию (транзакции, join).

Apache Cassandra, ClickHouse от Яндекс:

- СУБД типа семейство столбцов,
- аналитические БД с большим объёмом данных,
- для обработки больших данных,
- решают задачи аналитиков, Data Scientist.

Постреляционные БД

Сюда же (к NotOnlySQL) отнесем постреляционные БД, о типичном примере которых мы уже говорили – PostgreSQL, в таблице в ячейке которой можно хранить в качестве атрибута большие сложные структуры данных, например, JSON документы, списки.

3.4 Сравнительная таблица SQL vs NoSQL

При сравнении можно рассмотреть три параметра: схем данных, язык запросов, наличие транзакций или отказ от них ради других преимуществ.

	SQL	NoSQL
Структура данных	Схема данных	Без схемы данных Динамическая схема данных
Запросы	SQL	Свой язык
Транзакции	ACID	CAP-теорема

CAP-теорема

Используя СУБД на одной машине, мы можем упереться в производительность этой машины, тогда нам придется переходить на использование распределенной СУБД. В данном случае мы столкнемся с задачей о сохранении доступности данных или их консистентности (т.е. правильности и непротиворечивости). Данную ситуацию описывает CAP-теорема.

CAP- теорема – это утверждение, что распределенная система может удовлетворять только двум из перечисленных свойств:

1. согласованность – нет противоречащих данных, (англ. consistency),
2. доступность – корректность отклика на запрос, (англ. availability),
3. устойчивость к разделению, (англ. partition tolerance).

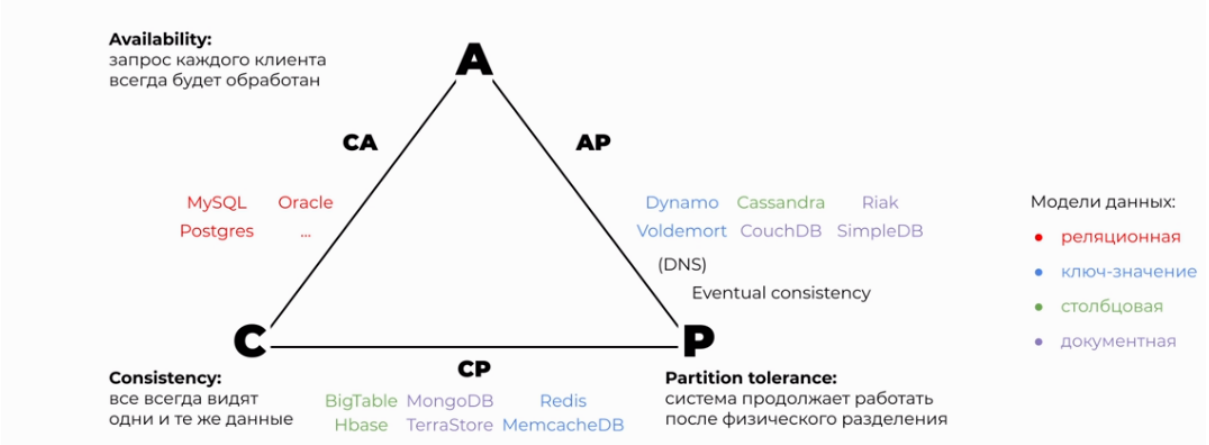

Рисунок 4 – CAP-теорема

3.5 Выбор СУБД

Теперь переходим к самому главному, как выбрать СУБД для работы. Чтобы выбрать СУБД для своего проекта потребуется ответить на ряд вопросов.

Начать нужно с описания, что мы собираемся хранить, какие сущности, какие у них будут связи, это называется составление ER-диаграммы (*ER – entity relationship*).

Затем определим для наших сущностей и связей подходит строгая схема данных или наши данные слабоструктурированные.

Как будут использоваться данные? Будут ли это активные одновременные операции чтения-записи множества пользователей (т.е. нам нужны транзакции)

или это будет БД для аналитики (где запись осуществляется, например, раз в сутки, а в остальное время происходит только чтение больших объемов информации)?

Далее смотрим на технические требования. И на то можем ли мы её обслуживать.

Ниже приведен пример дерева выбора СУБД:

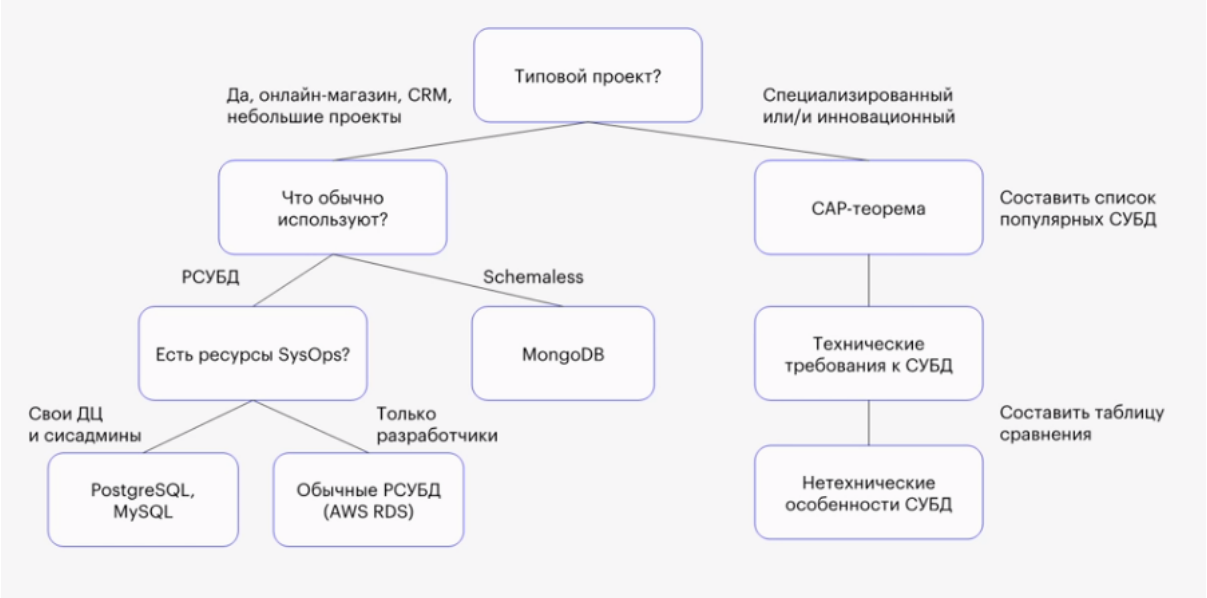

Рисунок 5 – Выбор СУБД

Не делайте типичных ошибок, не пытайтесь использовать что-то редкое и новое для стандартных задач, или наоборот не пытайтесь решить специализированную задачу с помощью универсальной СУБД.

Можно использовать несколько СУБД в одном проекте, например, основную задачу будет выполнять какая-то реляционная БД (MySQL), а быстрый доступ к некоторым заранее рассчитанным данным можно реализовать с помощью БД ключ-значение (Redis).

4 Запросы

Рассмотрим как выглядят запросы в различных видах БД.

4.1 Запросы MySQL

Сокращенно запрос MySQL выглядит примерно так:

```
[WITH подзапрос]
SELECT
    [атрибут]
    [выражение]
    [подзапрос]
    [FROM таблица, [подзапрос]
    [WHERE условия]
    [GROUP BY {атрибут | выражение | позиция}, ... ]
    [HAVING условие]
    [ORDER BY {атрибут | выражение | позиция} [ASC | DESC]]
    [LIMIT {количество | количество OFFSET смещение}]
```

Select – ключевое слово начала запроса, после него указываем что мы хотим получить, перед этим можно сформировать с помощью конструкции with некоторые выборки, которые пригодятся в основном запросе. Например, далее у нас будет сложный запрос, где мы несколько раз будем отбирать одно и тоже, но для разных обработок, поэтому целесообразно вынести повторяющийся отбор в with.

Далее после ключевого слова From указываем из какой/каких таблиц или подзапросов отобрать данные.

После ключевого слова Where следуют условия отбора, т.е. набор условий соединенных логическими операторами. Условие – это выражение, которое возвращает true, если запись таблицы подходит под него, и false, если запись не удовлетворяет условию. В условиях мы также можем использовать подзапросы.

Три предыдущие операции над данными можно назвать основой запроса. Далее возможно произвести некоторые действия над полученными результатами. Результаты можно сгруппировать описав правила группировки в Group by, при этом в select мы должны указать какие-либо агрегирующие функции (*?Знаете что такое агрегирующие функции? И как работает группировка?*) . Полученные группы мы также можем отфильтровать, указав условие в Having.

Полученные данные можно отсортировать, указав каким образом и по какому полю это сделать в Order by.

И в конце можно ограничить количество результатов в итоговой выборке в Limit, где мы можем указать не только сколько первых записей взять, но и с какой начать.

4.2 Запросы MongoDB

В MongoDB используется JSON формат документов и синтаксис языка JavaScript.

База данных является объектом и на текущую базу данных ссылается переменная *db*. У объекта есть методы и объекты. У *db* объектами являются коллекции, пусть наша коллекция называется *users*.

Тогда чтобы получить какие-то данные нужно использовать метод find():

db.users.find()

– где find() это метод объекта users.

В общем виде запрос выглядит следующим образом:

```
db.collection.find(query, projection, options)
```

query – запрос, задаем параметры с помощью операторов запроса (такие как операторы сравнения, логические операторы и прочие).

projection – указываем, какие поля должны возвращаться по запросу, если параметр отсутствует получим все поля.

options – дополнительные параметры запроса.

Примеры.

В MongoDB Существует зарезервированный ключ для идентификаторов документов: *_id*. Пусть у нас в JSON документах _id будет адрес электронной почты пользователя.

db.users.find({"_id": "mail@example.ru"})

– ищем документ по id равному “mail@example.ru"

Все ключевые слова вводятся символом *$*, а для отбора строк можно использовать регулярные выражения , например:

db.users.find({"karma": {$lt: 10}, "first_name":/.*ale.*/})

– ищем пользователей, у которых карма меньше 10 и имя содержит “an”

$lt – less than, меньше чем.

/.*ale.*/ – регулярное выражение, вхождение подстроки “ale”.

Можно ограничить количество получаемых результатов, для этого используется следующий метод *limit()*:

db.users.find({"karma": {$lt: 10}, "first_name":/.*ale.*/}).limit(1)

– отобрать одну запись.

С запросами на агрегацию в MongoDB связано понятие **pipeline**, т.е. трубопровод или по другому конвейер. Когда данные проходят по конвейеру обрастая всё новыми результатами. На первом шаге мы делаем какой-то запрос, затем передаем полученный результат на следующий шаг, и т.д.

4.3 Запросы Redis

С Redis (remote dictionary server) мы работаем не запросами, а командами. Рассмотрим основные команды.

Команды

set

– записать значение по ключу.

set hello world,

где hello - ключ, world - значение.

get

– получить значение.

get hello

Принято разделять части ключа знаками пунктуации:

set user:10:name Ivan

exists

– проверка ключа на существование.

exists user:10

del

– удаление ключа

Можно создавать временные ключи.

set hello world ex 60

– этот ключ исчезнет через 60 секунд.

ttl

– проверка сколько осталось жить ключу.

persist

– изменение времени жизни ключа на постоянный.

append

– дозапись значения по ключу

append hello "!!!"

rename

– переименование ключа

Обычно Redis используется для хранения ключей в пределах сотен тысяч.

Внутри key-value хранилища используется hash-таблица, несколько модифицированная.