SQL в работе с базами данных. Архитектура БД и оптимизация запросов.

Типы СУБД:

- Файл-серверные: Microsoft Access
- Клиент-серверные: MySQL, PostgreSQL...
- Встраиваемые: SQLite

Основные клиент-серверные СУБД:

- MySQL
- PostgreSQL
- Oracle
- MS SQL
- MariaDB

Почему мы будем использовать Postgres:

- Free (Open Source)
- Лучший выбор для изучения: проинсталлировал и «понеслась»!
- «Взрослая» СУБД, хорошо поддерживающая транзакционность из коробки
- Весьма развитой диалект SQL
- В сравнении с MySQL есть свои плюсы и минусы
- В любом случае, 90% возможностей диалекта SQL поддерживаемого PostgreSQL можно без изменений использовать и в др. СУБД

НЕ реляционные СУБД:

- Не реляционные СУБД существовали задолго до реляционных (иерархические, например)
- На смену пришли реляционные (начиная с 1970-х)
- И спустя несколько декад, не реляционные СУБД возродились (расцвет в 2010-х)
- Появилось **NoSQL** движение
- mongoDB – одна из наиболее популярных NoSQL-СУБД
- Реляционные СУБД не хуже и не лучше не реляционных СУБД
- У всех видов есть свои преимущества и недостатки

Типы NoSQL Баз Данных:

- Реляционные базы данных
- Key-value базы данных
- Документно-ориентированные базы данных
- Графовые базы данных
- Колоночные базы данных

1. Key-Value БД:

В такой базе хранят данные, которые удобно представить в виде пары ключ-значение. Основное преимущество таких баз – это очень быстрый поиск значения по ключу. При этом значение может содержать какие угодно типы данных.

1. Документно-ориентированные

В документно-ориентированной базе данных единицей хранения является документ (который может быть в формате json, или xml, или в каком-нибудь еще формате). Удобство таких баз в том, что в них быстро и легко записывать любые типы данных, при этом эти данные не обязаны обладать четкой структурой. Минус таких баз в том, что данные в них неудобно анализировать.

1. Графовые

Как следует из названия, в графовой базе данных данные хранятся в виде графов. Данный тип баз удобен, когда надо находить информацию не только о каком-то объекте, но и доставать информации о связях этого объекта с другими.

1. Колоночные

В реляционных базах данных данные записаны в виде строк. Что же касается колоночных баз данных, то тут данные записываются в виде столбцов. Потому поиск данных в колоночной базе данных осуществляется не перебором всех строк, как это происходит в реляционной базе данных, а поиском необходимого значения в тех столбцах таблицы, которые нас интересуют.
Преимущество колоночных баз данных в том, что они могут быстро находить определенные значения в столбцах, которые нас интересуют

Реляционная модель включается в себя несколько важных понятий:

- Сущность - например, клиенты, продажи и т.д.
- Таблица - отношение
- Отношения между таблицами
- Столбец - атрибут или поле
- Строка/запись - кортеж
- Индекс
- Результирующий набор - результат запроса SQL

Отношения между таблицами определяются с помощью primary key и foreign-key

Primary key – это столбец (или группа столбцов) таблицы, который содержит уникальные значения для каждой строки. На примере выше primary key каждой таблицы я выделила зеленым цветом. То есть, например, в таблице с заказами каждая строка будет описывать отдельный заказ. Не будет 2 строк, которые описывают один и тот же заказ, потому ID заказа будет разный для каждой строки.

Foreign key – это столбец в таблице, который содержит primary key другой таблицы. То есть, таблица с заказами содержит ID клиента, который является primary key в таблице с клиентами, но в таблице с заказами он будет foreign key.

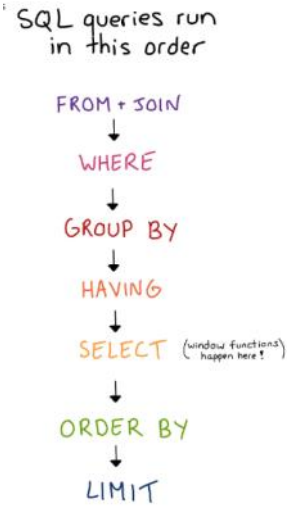
SQL:

- SQL (Structured Query Language, язык структурных запросов) – стандартный язык запросов к реляционным базам данных.
- SQL основан на реляционной алгебре.
- Результатом SQL запроса является результирующий набор (как правило – таблица)
- DDL (Data Definition Language) – CREATE, ALTER, DROP
- DML (Data Manipulation Language) – SELECT, INSERT, UPDATE, DELETE
- TCL (Transaction Control Language) – COMMIT, ROLLBACK, SAVEPOINT
- DCL (Data Control Language) – GRANT, REVOKE, DENY
- ANSI SQL-92
- Различия в процедурных расширениях:
  PL/pgSQL в PostgreSQL, **PL/SQL** в Oracle, **T-SQL** в MS SQL

Общая структура запросов:

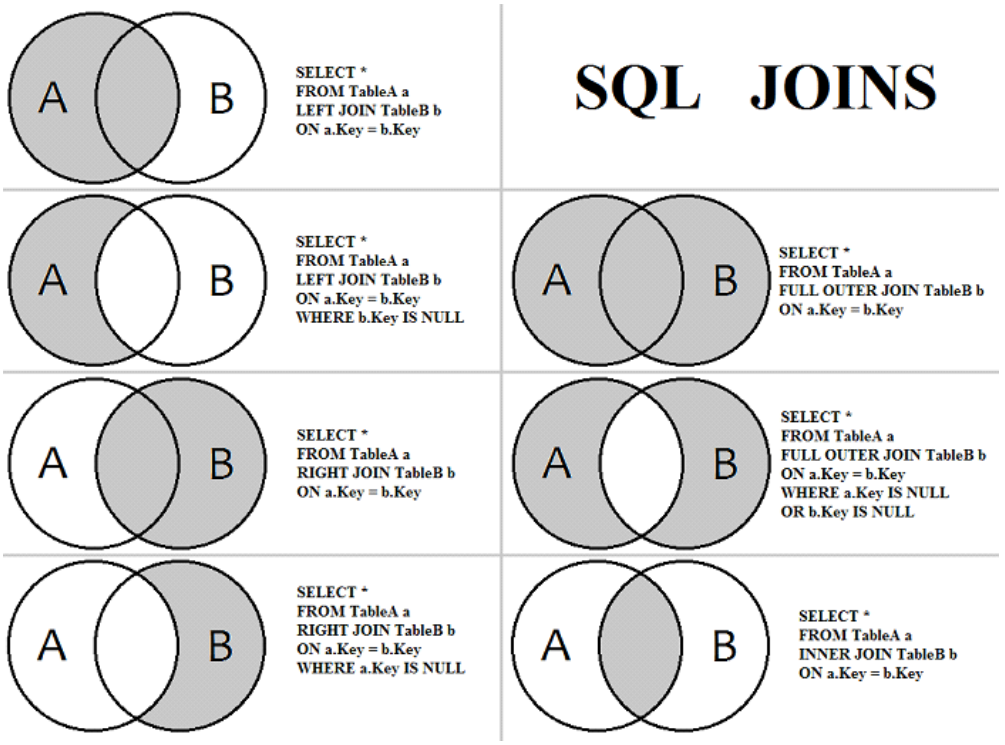
```
SELECT ('столбцы или * для выбора всех столбцов; обязательно')
FROM ('таблица; обязательно')
WHERE ('условие/фильтрация, например, city = 'Moscow'; необязательно')
GROUP BY ('столбец, по которому хотим сгруппировать данные; необязательно')
HAVING ('условие/фильтрация на уровне сгруппированных данных; необязательно')
ORDER BY ('столбец, по которому хотим отсортировать вывод; необязательно')
```

Порядок выполнения SQL запрос:



Как видно, в начале выполняется полный JOIN и выборка все строк из запроса, после которого уже идет отбор и последующие операции

Типы JOIN:



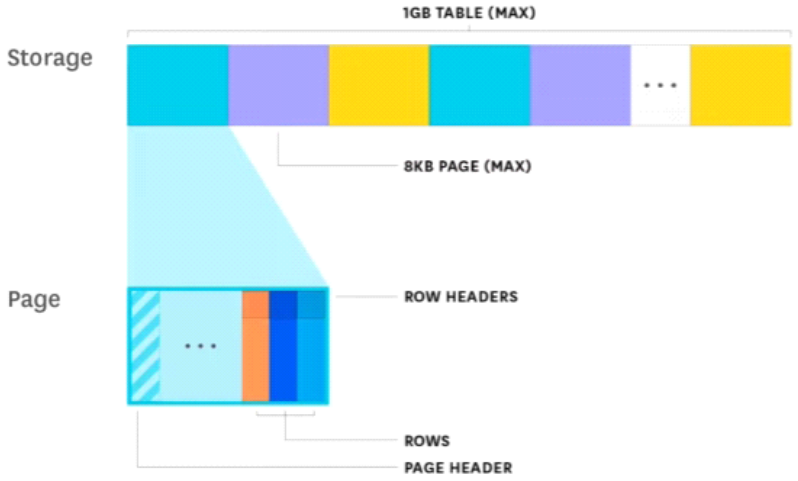
Устройство БД:

Вот главные элементы, которые есть в каждой СУБД, и их функции:

- **Ядро**. Это основа всей системы, которая отвечает за хранение и обработку баз данных. В ядре фиксируются все изменения: добавление, удаление или исправление целых баз и отдельных ячеек.
-**Процессор, или компилятор.** Обрабатывает запросы к базам данных на внутренних языках и SQL, преобразуя их в нужные команды и передавая результаты.
-**Программные средства, или утилиты**. С их помощью пользователи вводят запросы, а администраторы баз данных настраивают доступ и другие параметры.
-**Базы данных.** То, где хранятся данные, организованные особым образом, иногда — в зашифрованном виде.

- database cluster – одна и более БД, управляемые из под одной инстанции сервера
- Файлы данных кластера лежат в директории data (часто называемой PGDATA)
- Для каждой БД есть своя подпапка в PGDATA/base
- Для каждой таблицы и индекса выделяется отдельный файл

- Таблица состоит из массива страниц (блоков, размером 8кб)
- Файл таблицы называется Heap File -
  содержат списки неупорядоченных записей различной длины



- Таблица размером 1 ГБ
- Состоит из страниц по 8 КБ
- Каждая из которых содержит:
  заголовок страницы, строки с их заголовками
- Страница содержит ссылки на строки (CTID)

- Рядом с файлом таблицы лежит файл FSM (free space map)
- FSM не обновляется при каждом обновлении или удалении строк
- VACUUM [FULL] – команда для очистки «дохлых» версий строк
- Рядом с файлом таблицы лежит файл VM (visibility map)

Vacuum:

- Если вообще не обслуживать БД, то фрагментация данных будет нарастать
- VACUUM – команда для очистки «дохлых» версий строк
- VACUUM FULL – полный «компактинг» таблицы
- Необходим периодический запуск VACUUM
- Активно обновляемым БД рекомендуется проходить VACUUM каждую ночь
- VACUUM FULL только если удалили много данных
- VACUUM ANALYZE – комбинация двух операций

Индексы:
Индекс - структура данных, ускоряющая выборку данных из таблицы за счет дополнительных операций записи и пространства на диске, используемых для хранения структуры данных и поддержания ее в актуальном состоянии.

Преимущества:

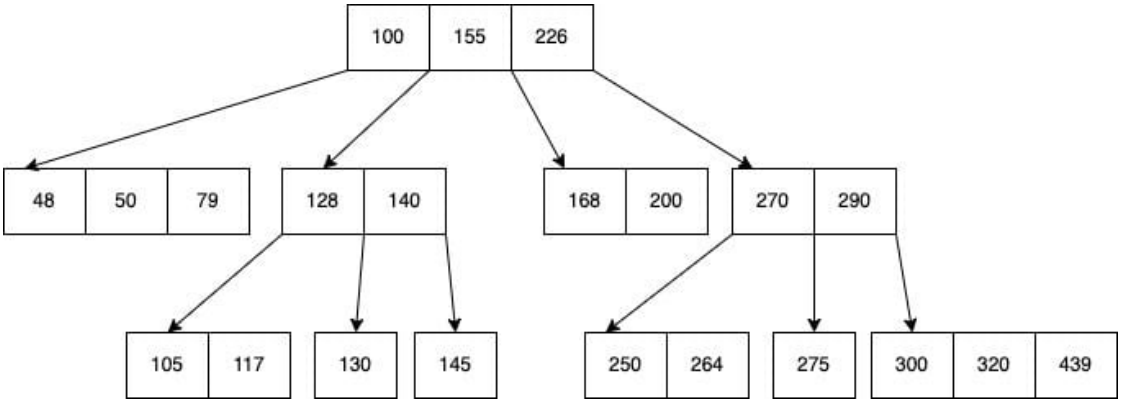
- Позволяет искать значения без полного перебора
- Оптимизация выборки «небольшого» числа записей
- «Небольшое» число - число относительное кол-ва записей
- По PRIMARY KEY и UNIQUE столбцам индекс создаётся автоматически
- Индексы не бесплатны

SELECT amname FROM pg_am;

- B-tree (сбалансированное дерево)
- Хеш-индекс
- GiST (обобщённое дерево поиска)
- GIN (обобщённый обратный)
- SP-GiST (GiST с двоичным разбиением пространства)
- BRIN (блочно-диапазонный)

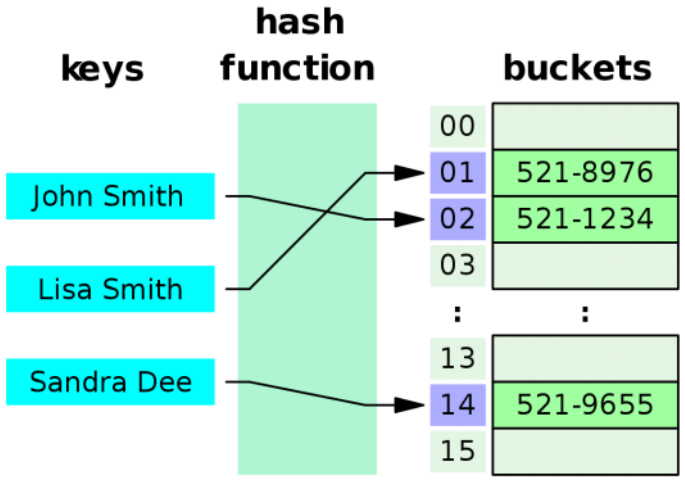
1.B-TREE:

- Создаётся по умолчанию (CREATE INDEX *index_name* ON *table_name* (*column_name*))
- Поддерживает операции:
  <, >, <=, >=, =
- Поддерживает LIKE ‘abc%’ (но не ‘%abc’)
- Индексирует NULL
- Сложность поиска O(logN)



2.HASH:

- CREATE INDEX *index_name* ON *table_name* USING HASH (*column_name*);
- Поддерживает только операцию “=“
- Не отражается в журнале предзаписи (WAL)
- Не рекомендуется к применению (в общем и целом)
- Сложность поиска O(1)



Методы Сканирования данных:

- Индексное (index scan)
- Исключительно индексное сканирование (index only scan)
- Сканирование по битовой карте (bitmap scan)
- Последовательное сканирование (sequential scan)

EXPLAIN

- Если есть проблема с производительностью – надо понять откуда «растут ноги»
- EXPLAIN *query* позволяет посмотреть на план выполнения запроса
- EXPLAIN ANALYZE *query* прогоняет запрос, показывает план и реальность

ANALYZE

- Собирает статистику по данным таблицы
- Планировщик смотрит на статистику при построении плана
- ANALYZE [*table_name* [*(column1, column2…)*]]
- Запускать как минимум один раз в день
- autovacuum (если включен) в том числе запускает ANALYZE