**Базы данных и SQL**

База данных (БД) – структурированное хранилище информации, организованные таким образом, чтобы обеспечить хранение, управление и доступ к информации. Чаще всего эти данные взаимосвязаны. Сама по себе БД – это просто хранилище данных, остальные свойства базы данных обеспечиваются за счет системы управления базой данных.

Система управления базой данных (СУБД) – программные средства для управления данными. Отвечает за поддержку языка БД (обеспечивает интерфейс для взаимодействия с данными), за механизм хранения и извлечения данных, за оптимизацию процессов извлечения данных.

**Виды БД:**

*Без использования СУБД*

|  |  |
| --- | --- |
| Файловые | Данные хранятся в виде текста в файле. |
| Иерархические | Имеет древовидную структуру (узел может иметь только одного родителя) Пример: файловая система. |
| Сетевые | Имеет структуру графа (узел может иметь более одного родителя) |

*С использованием СУБД*

|  |  |
| --- | --- |
| Реляционные | Структура в виде таблиц, связанных между собой |
| NoSQL | Общее название для нереляционных БД. |

**Типы СУБД:**

NoSQL (not only SQL) – Нереляционные СУБД, в которых данные могут быть организованы в виде иных структур, отличных от таблиц, ориентированных под свои задачи.

Используется язык отличный от SQL.

Отсутствие жесткой схемы данных (Приятное следствие отсутствия схемы — эффективность работы с разреженными (sparse) данными. Если в одном документе есть поле date\_published, а во втором — нет, значит никакого пустого поля date\_published для второго создано не будет)

Отказ от атомарности и согласованности данных в пользу горизонтального масштабирования.

1) Ключ-значение – база данных организована в виде словаря. Некая таблица, имеющая уникальный ключ и соответствующее ему значение. Подходит для кэширования. [Redis]

Высокая скорость, легкое масштабирование. Не подходит для сложных структур (сущности с разными типами данных) и сложных запросов.

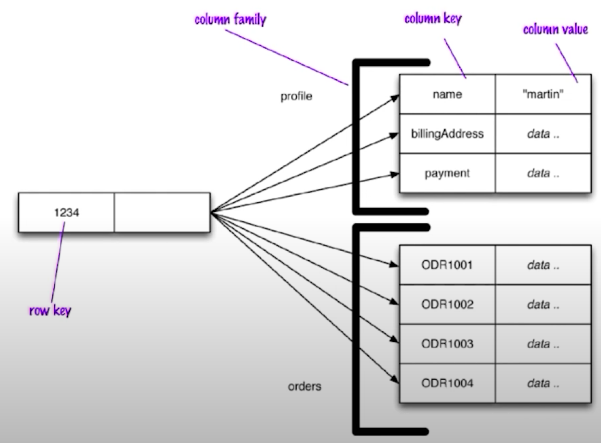
2) Колоночные – Данные организованы в виде таблиц так же, как и в реляционной СУБД, но не построчно (для чтения колонки придется вычитать всю строку до нужной колонки), а «поколоночно», т.е. колонка — это отдельная структура, в которой содержатся данные для всех строк по определенному ключу.

Подходит для аналитических запросов на больших объемах данных (эффективна при запросах для более 100 миллионов строк) – OLAP. [Apache druid]

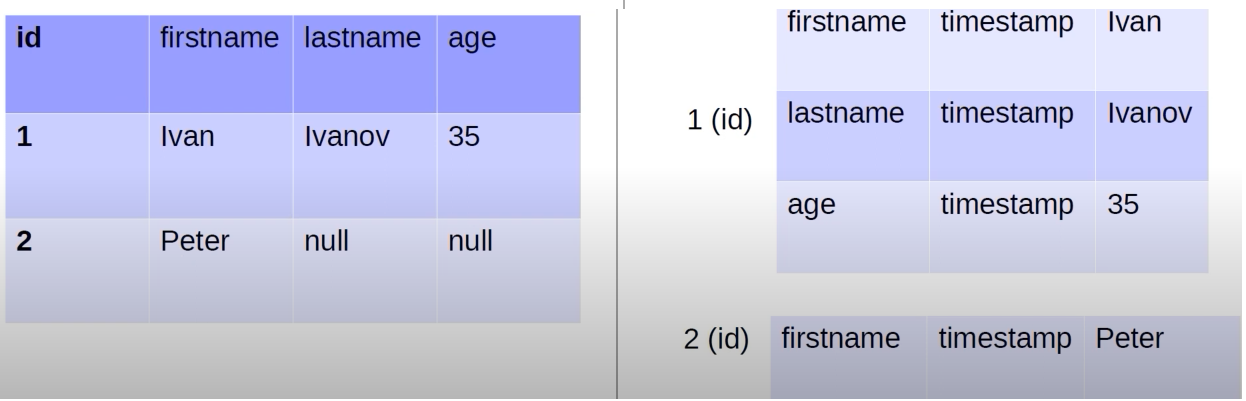
3) Ширококолоночные (Wide Column Stores) – Каждому ключу соответствует набор column family – логическая группировка колонок, каждая из которых состоит из множества сущностей ключ (названия колонки) – значение.

В отличие от реляционных не имеют жесткой схемы (в случае добавления или отсутствия значений не нужно хранить null) (см. рис.2), лучшее сжатие, горизонтально масштабируема [HBase, Cassandra]

Общая структура:



Реляционная СУБД Колоночная СУБД



4) Документо-ориентированная – Данные организованы в виде коллекций (аналог таблицы), которые в свою очередь содержат документы (аналог строки в таблице). Документ представлен парами ключ-значение, имеет формат JSON или BSON

Гибкий – не требует схем данных, данные хранятся в одной сущности – не требует соединений, как в реляционной, горизонтально масштабируемый

5) Графовые – имеют структуру графа, необходимы в ситуации, когда нужно отразить большое кол-во связей между сущностями. [Neo4j, AllegroGraph]

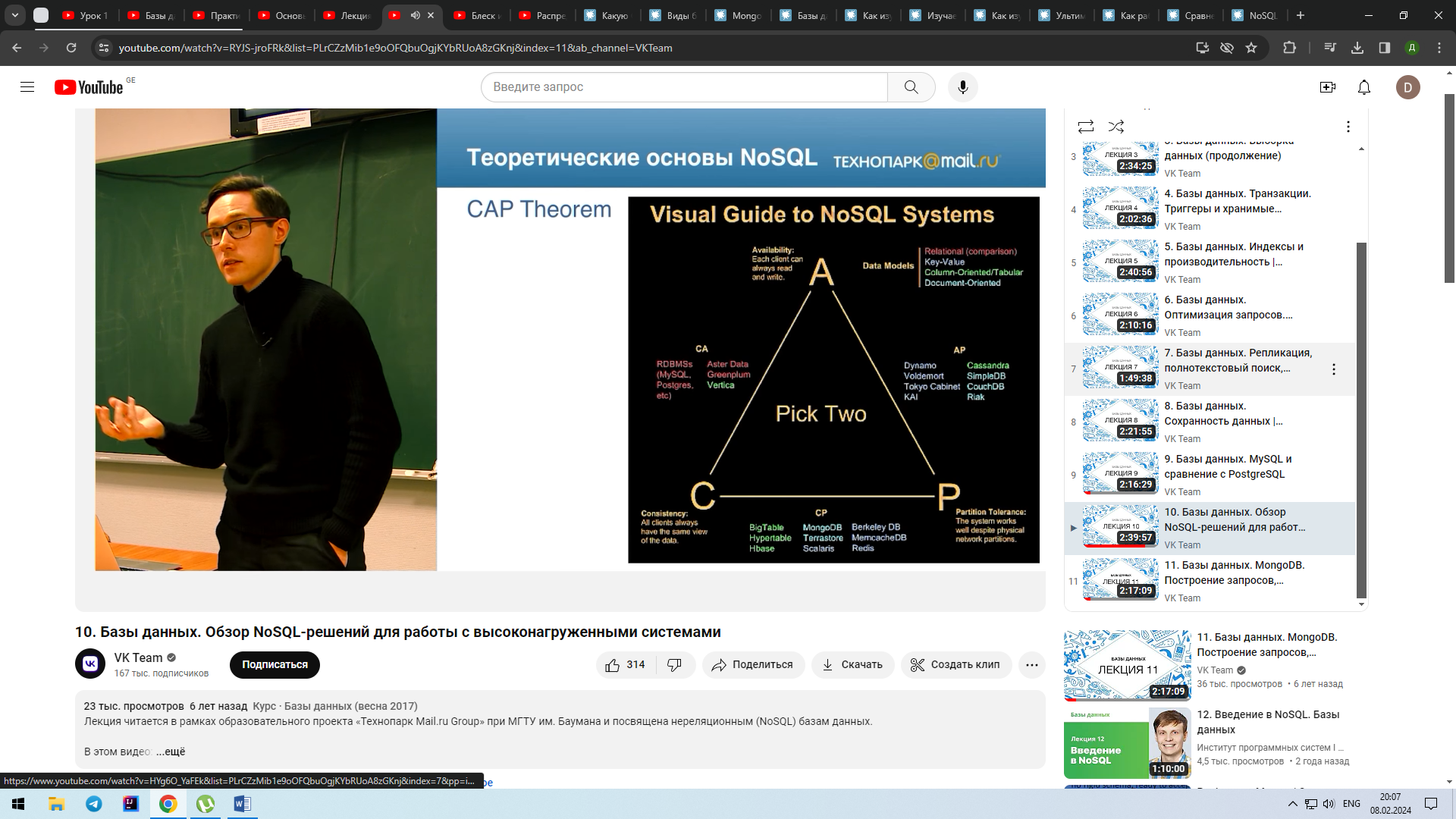
6) Реляционные – Данные организованы в виде отношений (таблиц) которые имеют жесткую схему организации, систему операций над которыми определяет реляционная алгебра, операции выражаются языком SQL. Согласованность данных в приоритете, заточена под транзакции, которые поддерживают свойства ACID. Построчное хранение данных – быстрая вставка, но накладные расходы при чтении данных.

**CAP теорема –** распределенная система не может в полной мере обеспечивать свойства consistency (согласованность), availability (доступность), partition tolerance (устойчивость к разделению)

Consistency – Гарантирует, что запрос к системе выдаст последнюю версию изменений, даже если изменение было произведено на другом узле – актуальность данных.

Availability - Каждый запрос завершается успешным ответом.

Partition tolerance – Система продолжает функционировать при наличии разделения/разрыва сети между узлами

CA системы – такая система подразумевает, что если выходит из строя один из компонентов системы, то она перестает работать

CP – система не гарантирует, что ответ от нее будет получен в разумные сроки (пока запрашиваемые данные не будут синхронизованы на всех узлах, клиент будет ждать ответ на запрос)

AP – система не гарантирует консистентность данных, такие системы не подразумевают того, что данные будут всегда противоречивы, они говорят о том, что данные не будут констстентны в каждый момент времени

Данная «теорема» является условной, не является теоремой в математическом понимании и содержит много противоречий [1](https://habr.com/ru/articles/231703/), [2](https://habr.com/ru/articles/322276/), [3](https://habr.com/ru/articles/328792/), [4](https://habr.com/ru/articles/258145/)

Для описания свойств SQL и NoSQL систем используются ACID и BASE

**ACID –** свойство транзакций, направленные на обеспечение надежности и целостности данных.

*Atomicity* (Атомарность) – Транзакция не может быть зафиксирована частично, то есть не может иметь промежуточных состояний, она либо выполнена полностью, либо не выполнена вовсе.

*Consistency* (Согласованность) - свойство, гарантирующее, что каждая успешная транзакция зафиксирует только допустимые результаты. То есть, это гарантия того, что при успешной транзакции будут выполнены все правила, ограничения, которые предъявляет система к конкретным данным, иначе транзакция не будет выполнена и данные в системе вернутся к прежнему состоянию.

*Isolation* (Изолированность) – Параллельные транзакции (транзакции, работающие с одними и теми же данными) не должны влиять на результаты выполнения друг друга.

Durability (Надежность) – Если клиент получил подтверждение от системы подтверждение завершения транзакции, то эти изменения не будут отменены из-за сбоя.

***Уровни изоляции для ACID -***

**BASE** – альтернативный подход к построению распределенных систем или свойства распределенных систем, в которых Consistency и Availability выполнены не в полной мере.

Basically Available (Базовая доступность) – на любой запрос будет дан ответ, но он может отражать неконсистентные данные.

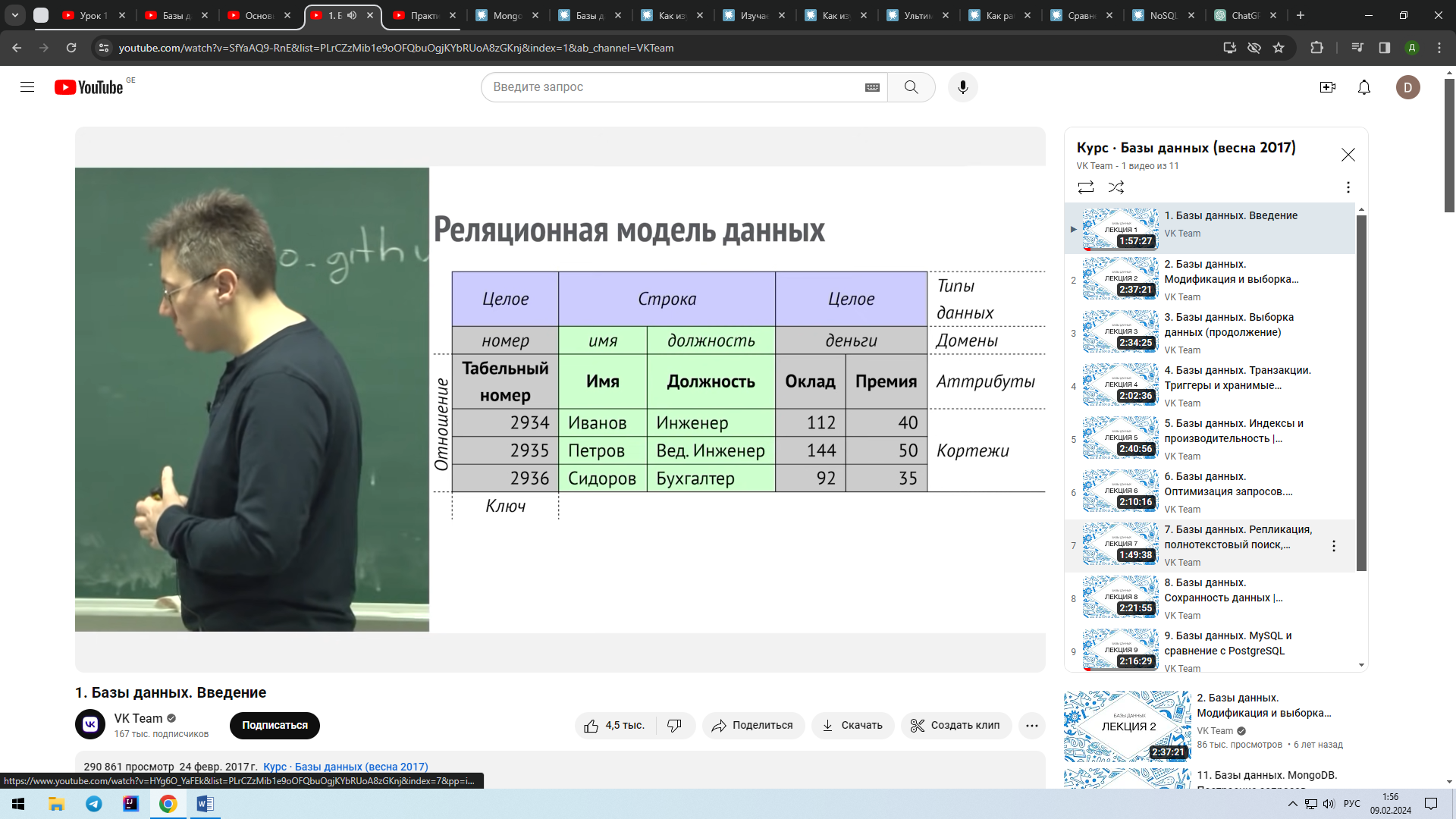
Soft state (Неустойчивое состояние) – данные в системе могут меняться, даже если изменений в системе нет

Eventual consistency (согласованность в конечном итоге) – при отсутствии изменений в системе, в конечном итоге, все данные будут согласованы.

**OLAP/OLTP**

**Подробнее про реляционные СУБД**

*Реляционная модель данных:*



*Атрибут (столбец) –* Имеет Тип данных, Домен, Имя

*Домен –* некое условие для типа данных (ограничение)

*Ключ* – атрибут, который однозначно идентифицирует каждую запись в таблице (должен обладать свойством уникальности и иметь значение отличное от Null)

*Строка/кортеж* – одна запись в таблице, т.е. набор значений каждого атрибута для определенного ключа

*Отношение -* таблица

*Сущность* – Объект предметной области, который может быть описан

Результирующий набор – результат запроса на языке SQL

**Реляционная алгебра и SQL –** любой запрос на языке SQL представляет из себя набор из перечисленных операций из реляционной алгебры.

|  |  |  |  |
| --- | --- | --- | --- |
| Выборка | Извлекает только те строки, которые удовлетворяют заданному предикату (условию). | SELECT \* FROM “Люди” WHERE “Возраст” >=34 |  |
| Проекция | Получить только указанные атрибуты | SELECT “Возраст”, ”Вес” FROM “Люди” |  |
| Объединение | Возвращает отношение, содержащее все уникальные кортежи, которые встречаются в отношениях. | SELECT \* FROM “Люди” UNION SELECT \* FROM “Персонажи” |  |
| Пересечение | Возвращает только совпадающие кортежи | SELECT \* FROM “Люди” NATURAL JOIN “Персонажи” |  |
| Разность | Возвращает отношение, где первое отношение берется как основа и повторяющиеся значения из второго отношения не включаются в результирующее | SELECT \* FROM “Люди” NATURAL LEFT JOIN “Персонажи” WHERE “Персонажи” IS NULL |  |
| Произведение | Каждая запись из первого отношения комбинируется с каждой записью из отношения (декартово произведение) | SELECT \* FROM “Люди”, “Работники” |  |
| Деление | Из первого отношения берутся записи, которые включают все комбинации из второго отношения |  |  |
| Соединение | Записи из отношений соединяются по ключу | **SELECT \* FROM “asd” JOIN “DSA” ON** |  |

Объединение пример: При операции объединения двух отношений с разными атрибутами на выходе получаем отношение, содержащее все атрибуты из обоих отношений, если атрибуты отношений совпадают, то в конечном отношении будут содержаться только уникальные строки.

|  |  |
| --- | --- |
|  |  |