Database

Определение и описание БД

База данных (далее “БД”) - это место для хранения данных. Используется в клиент-серверной архитектуре, например интернет-магазинах, сайтах кинотеатров или авиабилетов и т. д. При оформлении заказа ваши данные сохраняются в ту самую БД.

Если речь идет о небольшом приложении, то необходимости в БД нет, но для его роста (при появлении большого количества новых клиентов, например) есть смысл сохранять информацию о клиентах не в файлы или на девайс, а в базу данных, для структурирования информации и избежания некорректной работы девайса. Ведь чем больше памяти заполнено на девайсе, тем медленнее он будет работать.

Как выглядит БД? Примерно как табличка Excel с колонками, заголовками и информацией внутри. Это называется ***реляционная*** БД, то есть, набор таблиц, хранящихся в одном **пространстве**.

Чтобы понять о каком **пространстве** идет речь, следует представить данные, которые мы обычно храним в таблице. Можно все эти данные вставить в одну огромную таблицу, а можно поделить: информация о клиентах, заказах, адреса и т. д. То есть у нас несколько разных табличек с информацией в отдельной папке (как правило). Пространство внутри базы данных - это точно такая же папка, место, куда мы складываем все необходимые нам таблички с информацией, чтобы они все были в одном месте.

Это помогает структурировать информацию и избежать путаницы.

Также есть варианты нереляционных БД, где информация хранится не в таблицах, по типу excel, а отдельными объектами. Иногда информация может хранится в текстовых файлах.

Для получения информации из БД используется понятный для нее язык под названием SQL (Structured Query Language). В нем присутствуют необходимые слова, которые помогут получить нужную вам информацию:

select - выбрать такие-то колонки

from - из такой-то таблицы

where - такую-то информацию

Пример: Если нам нужно получить информацию о клиенте “Иванов Иван Иванович”

select * **from** *clients* **where name** = ‘*Иванов Иван Иванович*’

Если бы мы искали ту же информацию в таблице excel, то нам все равно необходимо было бы знать название таблицы, где лежат данные.

Иногда необходимо достать данные из разных таблиц, и в Excel это сделать сложнее, так как необходимо открыть все нужные нам таблицы. В БД внутри запроса можно указать какие именно колонки из каких таблиц нужны. В результате такого запроса, вы получаете данные скомпанованные из разных таблиц.

Из чего состоит БД? Есть различные уровни работы с данными:

- ***Слой доступа к данным***, который удобно использовать для языков программирования.
- ***Слой хранения***. Это отдельный слой, потому что обычно удобно хранить данные другими способами, чем использовать те же самы данные. То есть, схема удобная для хранения не всегда удобны для использования и наоборот.
- ***Железо*** - слой, где лежат данные, причем там они организованы еще третьим способом, потому что дисками управляет операционная система, и общаются они только через драйвер.

Для ***слоя доступа к данным*** есть требования, чтобы удобно было работать:

1. **Универсальность** - чтобы с помощью любой технологии можно было запрашивать данные.
2. **Оптимальность** запроса - метод доступа к данным должен быть такой, чтобы было удобно доставать данные из базы.
3. **Параллелизм** - так как все масштабируются, разные серверы обращаются к базе за одними и теми же данными. Нужно приложить максимально усилий к тому, чтобы быстрее обрабатывать данные.

Для ***слоя хранения*** важна сохранность и целостность данных. То есть, если мы что-то записали в БД, то мы должны быть уверены, что мы это получим обратно.

Для ***Железа*** важна надежность.

Какая информация хранится в БД?

Связанные таблицы.
Данные могут быть организованы по-разному в зависимости от типа базы. Чаще всего речь идет о реляционных БД — базах данных, где информация представлена в виде связанных друг с другом таблиц. Такие СУБД управляются с помощью языка запросов SQL и обычно хранят структурированные данные, между которыми есть жесткие связи.

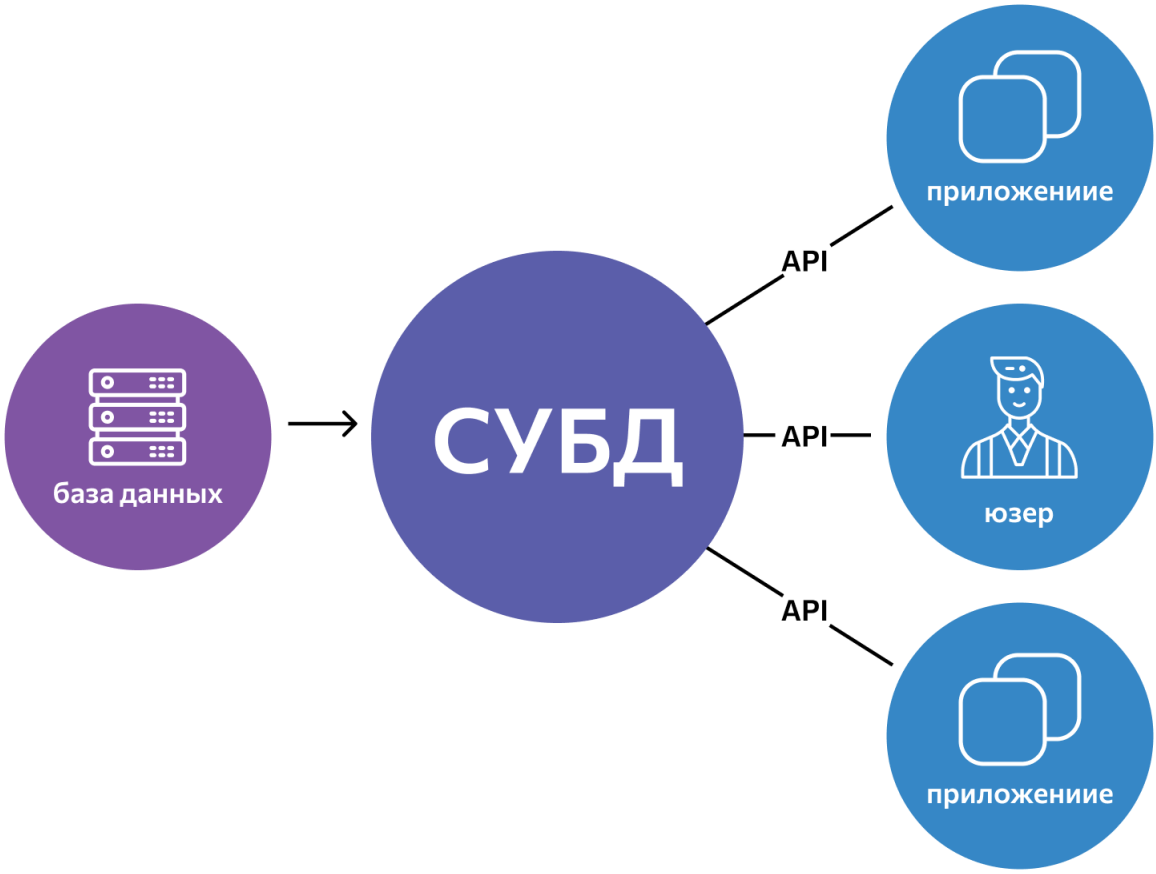
Объекты.
Объектные и объектно-реляционные БД представляют блоки информации как объект — сложную сущность с рядом свойств и методов. Объектная модель дает больше возможностей при работе с данными: у объектов есть наследование и другие свойства, которых нет у реляционных таблиц.

Древовидные структуры.
Еще один вариант системы хранения информации — иерархический. В нем данные хранятся в виде древовидной структуры. Его расширение — сетевой тип: он отличается от иерархического тем, что данные могут иметь больше одного «предка».

Иногда частным подвидом иеархического типа называют документно-ориентированную модель, при которой данные представлены в виде JSON-подобных документов. Она более гибкая и хорошо подходит для информации, не связанной между собой. Но для жестко связанных данных такой способ не подойдет.

СУБД - Система управления базами данных - сложное программное обеспечение, которое требуется, чтобы создавать базы данных, изменять их, получать из них информацию и контролировать версии.

Отличия БД от СУБД - База данных - это хранилище для информации. Она может принадлежать сайту, приложению, и любой программе. Там будут находится сведения, связанные с работой проекта. СУБД - это программное обеспечение, которое помогает администрировать базу, защищать ее целостность и конфиденциальность сведений, хранящихся в ней.



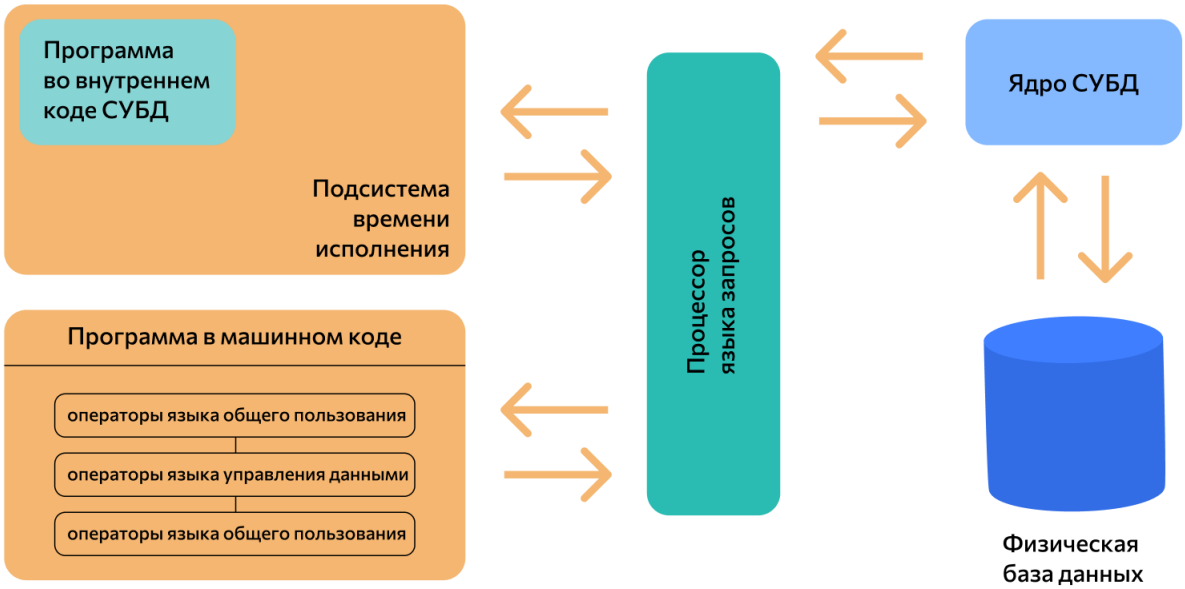
Кто пользуется СУБД?

- Бэкенд-разработчики, которые часто взаимодействуют с БД для получения данных.
- Разработчики локальных приложений, которые тоже могут хранить собственные данные.
- Администраторы БД - если продукт сложный, то для обслуживания БД, как правило, необходим администратор. Такие сотрудники обычно специализируются на конкретной СУБД.
- А также аналитики, DevOps-инженеры или специалисты по Big Data.

Из чего состоит СУБД?

СУБД имеет сложное устройство. Ядро СУБД отвечает за главные операции: хранение базы, ее обслуживание, документирование изменений. Это основная часть системы. Процессор языка или компилятор обрабатывает запросы. Обычно СУБД реляционного, объектно-ориентированного и объектно-реляционного типа поддерживают язык SQL и внутренние языки запросов.

Набор утилит предназначен для различных сервисных функций: их может быть очень много, а некоторые СУБД могут расширяться с помощью пользовательских модулей.



Виды СУБД по способу доступа

Системы по-разному обеспечивают хранение и доступ к данным. Существуют три вида архитектуры.

Клиент-серверная. База данных находится на сервере, СУБД располагается там же. К базе могут обращаться различные клиенты — конечные устройства. Например, пользователи запрашивают информацию на конкретном сайте.

Клиент-серверная архитектура подразумевает, что прямой доступ к базе есть только у сервера — он обрабатывает обращения клиентов. Сами клиенты не обязаны иметь специальное ПО для взаимодействия с базами данных. Так для доступа к сайту не нужно устанавливать программы, которые будут обрабатывать запросы, — все сделает сервер, жестко отделенный от клиентской части.

Такие базы надежны и обычно имеют высокую доступность. Ими пользуются чаще всего.

Файл-серверная. Тут все иначе: база хранится на файл-сервере, вот СУБД — на каждом клиентском компьютере. Доступ к базе данных могут получить только устройства, на которых установлена и настроена система.

Сейчас такие системы используются очень редко, в основном во внутренних приложениях, которые работают в локальных сетях. В крупных проектах файл-серверные СУБД не применяют.

Встраиваемая.
Это маленькая локальная СУБД, которая используется для хранения данных отдельной программы. Такие системы не функционируют как самостоятельные единицы, а встраиваются в программный продукт как модуль. Они нужны при разработке локальных приложений, целиком размещаются на одном устройстве и обычно очень мало весят.

Что такое NoSQL системы?

Не все БД управляются SQL. Есть системы, которые не подразумевают использования SQL. Отсюда и название NoSQL системы.

К СУБД NoSQL относят любые нереляционные системы — те, где не поддерживается реляционная модель представления информации. Некоторые нужны для хранения больших данных, другие — для ведения логов, третьи — для хранения данных с огромным количеством связей. Например, документно-ориентированные СУБД тоже относятся к NoSQL.

Вместо SQL применяются внутренние языки запросов, часто основанные на тех или иных языках программирования. Иногда они схожи с SQL, а иногда вместо внутреннего языка система использует JavaScript или иной язык программирования.

Примеры современных СУБД

- Oracle Database — объектно-реляционная клиент-серверная СУБД, одна из первых и самых популярных в мире. Платная, сложная, подходит для больших проектов.
- PostgreSQL — объектно-реляционная СУБД клиент-серверного типа, которую иногда называют бесплатным аналогом Oracle. Масштабная, рассчитана на высоконагруженные проекты, содержит огромное количество функций и распространяется бесплатно.
- MySQL — реляционная клиент-серверная СУБД. Популярный выбор для проектов небольшого и среднего размера. Легкая, гибкая и довольно простая в использовании. Она бесплатная, хорошо подходит для обучения и веб-проектов.
- MongoDB — документно-ориентированная NoSQL-СУБД, где данные хранятся в JSON-подобных файлах. Тоже бесплатная, а внутренний язык

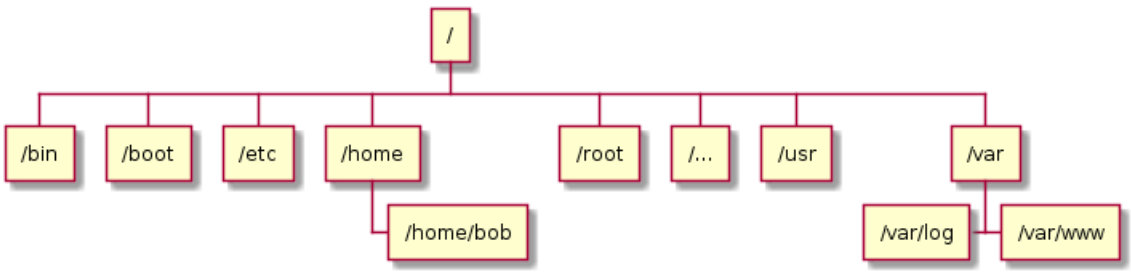
запросов основан на JavaScript.

- SQLite — маленькая и легкая встраиваемая СУБД, которая активно применяется в локальных проектах.

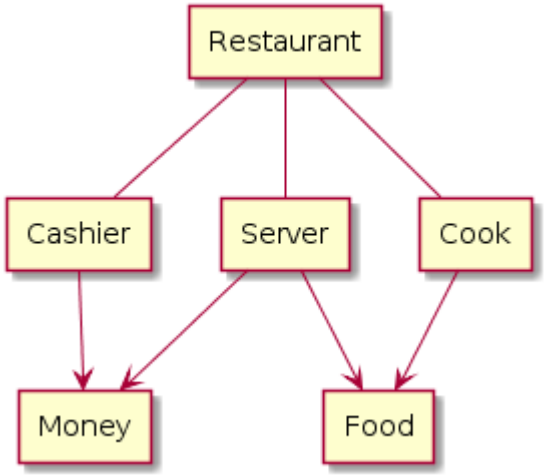
Виды БД

Простейшие типы БД

1. *Простые структуры дынных*
   Первый и простейший способ хранения данных – текстовые файлы. Метод применяется и сегодня для работы с небольшими объёмами информации. Для разделения полей используется специальный символ: запятая или точка с запятой в csv-файлах датасетов, двоеточие или пробел в *nix-подобных системах.
   - ограничен тип и уровень сложности хранимой информации;
   - трудно установить связи между компонентами данных;
   - отсутствие функций параллелизма;
   - практичны только для систем с небольшими требованиями к чтению и записи;
   - используются для хранения конфигурационных данных;
   - нет необходимости в стороннем программном обеспечении.
2. *Иерархические БД*
   В отличие от текстовых таблиц, в следующем типе БД появляются связи между объектами. В иерархических базах данных каждая запись имеет одного «родителя». Это создаёт древовидную структуру, в которой записи классифицируются по их отношениям с цепочкой родительских записей.
   - Информация организована в виде древовидной структуры с отношениями предок-потомок.
   - Каждая запись может иметь не более одного родителя.
   - Связи между записями выполнены в виде физических указателей.
   - Невозможно реализовать отношения “многих-ко-многим”.
   Применяются такие БД в файловых системах, DNS и LDAP.

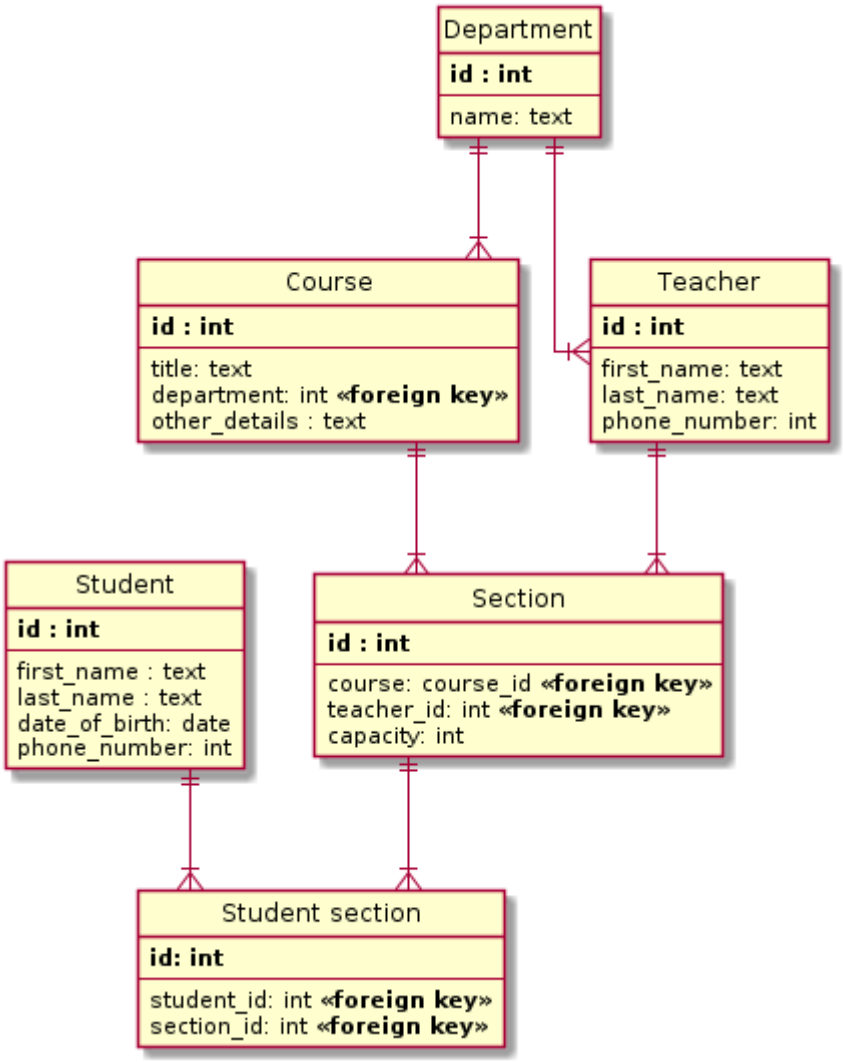


3. *Сетевые БД*
   Сетевые базы данных расширяют функциональность иерархических: записи могут иметь более одного родителя. А значит, можно моделировать сложные отношения.
   - Сетевые БД представляются не деревом, а общим графом.
   - Ограничены теми же шаблонами доступа, что иерархические БД.



Реляционные БД

4. *SQL БД*
   Реляционные БД - это старейший тип до сих пор широко используемых БД общего назначения. Данные и связи между данными организованы с помощью таблиц. Каждый столбец в таблице имеет имя и тип. Каждая строка представляет отдельную запись или элемент данных в таблице, который содержит значения для каждого из столбцов.
   - поле в таблице, называемое внешним ключом, может содержать ссылки на столбцы в других таблицах, что позволяет их соединять.
   - высокоорганизованная структура и гибкость делает реляционные БД мощными и адаптируемыми ко различным типам данных.
   - для доступа к данным используется язык структурированных запросов (SQL).
   - надежный выбор для многих приложений.
   Примеры: MySQL, MatiaDB, PostgreSQL, SQLite.



NoSQL БД - группа типов БД, предлагающих подходы, отличные от стандартного реляционного шаблона. Говоря NoSQL, подразумевают либо «не-SQL», либо «не только SQL», чтобы уточнить, что иногда допускается SQL-подобный запрос.

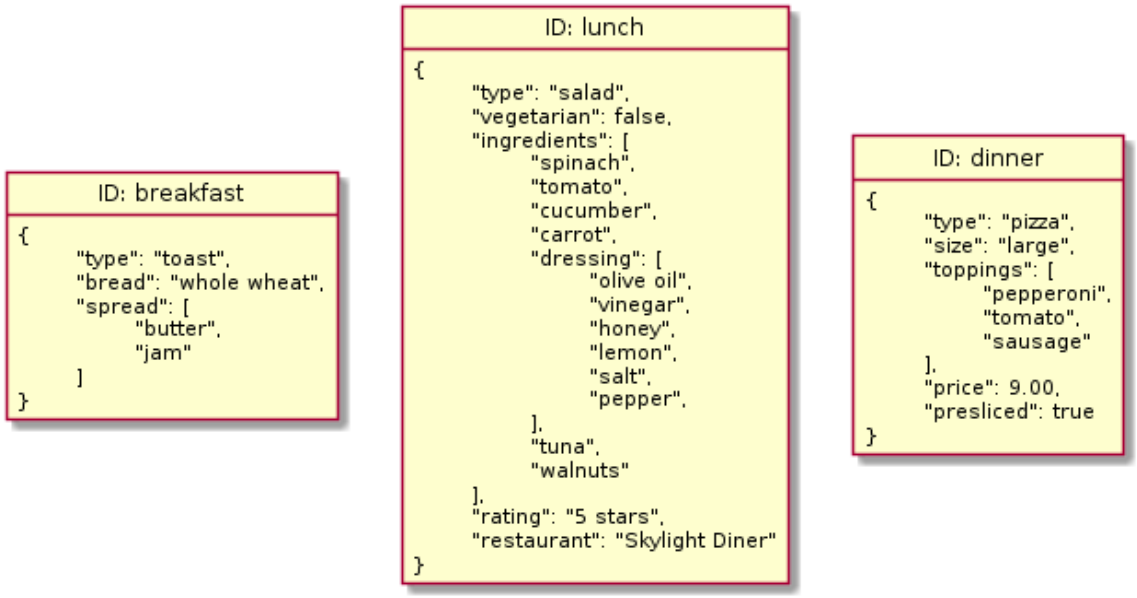
5. *Базы дынных “ключ-значение”*
   В базах данных «ключ-значение» для хранения информации вы предоставляте ключ и объект данных, который нужно сохранить.
   - хранилища обеспечивают быстрый и малозатратный доступ.
   - часто хранят данные конфигураций и информацию о состоянии данных, представленных словарями или хэшем.
   - нет жёсткой схемы отношения между данными, поэтому в таких БД часто хранят одновременно различные типы данных.
   - разработчик отвечает за определение схемы именования ключей и за то, чтобы значение имело соответствующий тип/формат.

6. *Документная БД*

   Совместно используют базовую семантику доступа и поиска хранилищ ключей и значений. Такие БД также используют ключ для уникальной идентификации данных. Разница между хранилищами «ключ-значение» и документными БД заключается в том, что вместо хранения blob-объектов, документоориентированные базы хранят данные в структурированных форматах – JSON, BSON или XML.

   - База данных не предписывает определенную схему или формат.
   - Каждый документ может иметь свою внутреннюю структуру.
   - Документные БД являются хорошим выбором для быстрой разработки.
   - В любой момент можно менять свойства данных, не изменяя структуру или сами данные.

   Примеры: MongoDB, RethinkDB



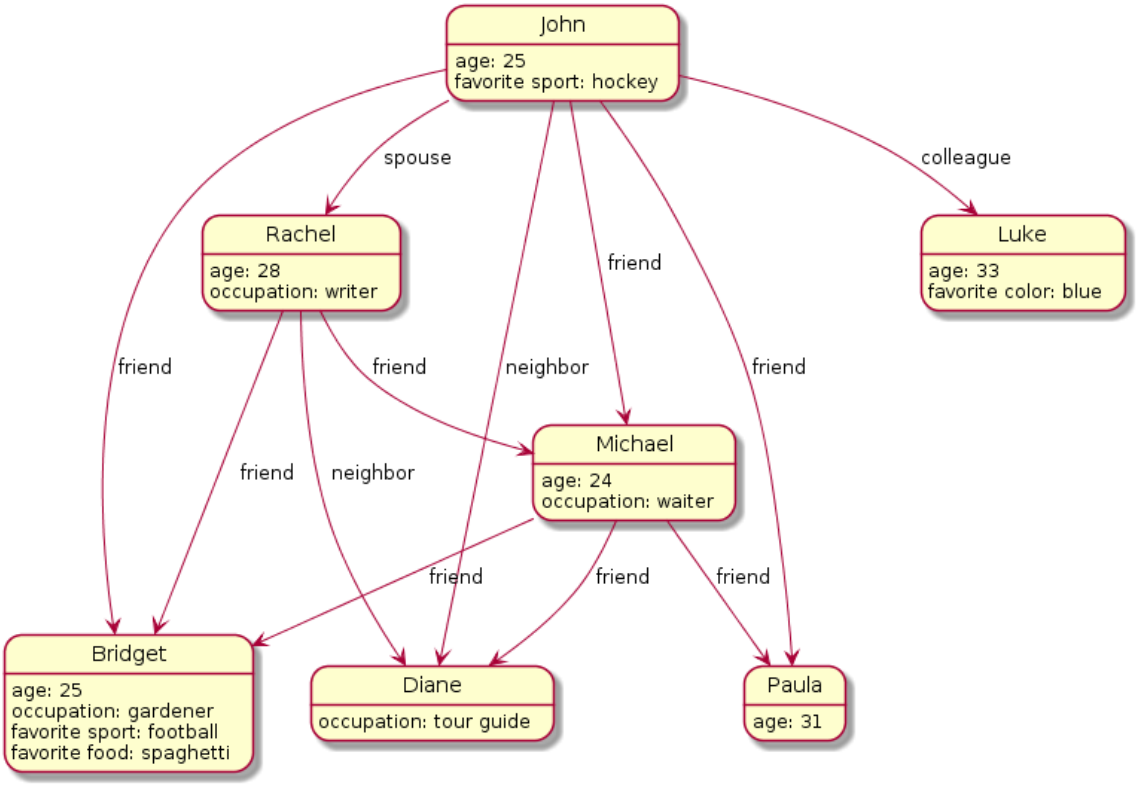
7. **Графовая БД**

   Вместо сопоставления связей с таблицами и внешними ключами, графовые БД устанавливают связи, используя узлы, ребра и свойства.

   Данный вид БД представляет данные в виде отдельных узлов, которые могут иметь любое количество связанных с ними свойств.

   - Выглядит аналогично сетевым.
   - Фокусируются на связях между элементами.
   - Явно отображает связи между типами данных.
   - не требуют пошагового обхода для перемещения между элементами.
   - нет ограничений в типах представляемых связей.

   Примеры: Neo4j, JanusGraph, Dgraph.



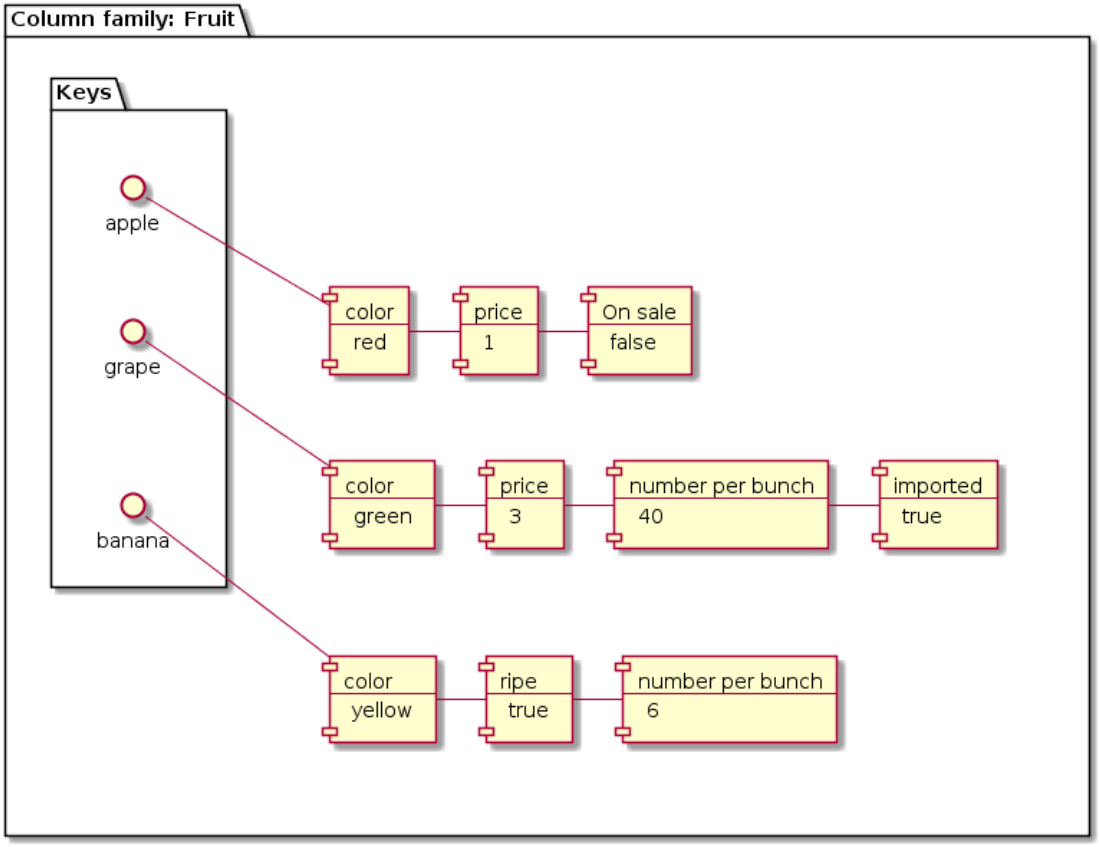
8. **Колоночные БД**

   Колоночные базы данных (также нереляционные колоночные хранилища или базы данных с широкими столбцами) принадлежат к семейству NoSQL БД, но внешне похож на реляционные БД. Как и реляционные, колоночные БД хранят данные, используя строки и столбцы, но с иной связью между элементами.

   В реляционных БД все строки должны соответствовать фиксированной схеме. Схема определяет, какие столбцы будут в таблице, типы данных и другие критерии. В колоночных базах вместо таблиц имеются структуры – «колоночные семейства». Семейства содержат строки, каждая из которых определяет собственный формат. Строка состоит из уникального идентификатора, используемого для поиска, за которым следуют наборы имён и значений столбцов.

   - БД удобны при работе с приложениями, требующими высокой производительности.
   - Данные и метаданные записи доступны по одному идентификатору.
   - Гарантировано размещение всех данных из строки в одном кластере, что упрощает сегментацию и масштабирование данных.

   Примеры: Cassandra, HBase



9. **БД временных рядов**

   Базы данных временны́х рядов созданы для сбора и управления элементами, меняющимися с течением времени. Большинство таких БД организованы в структуры, которые записывают значения для одного элемента. Например, можно создать таблицу для отслеживания температуры процессора. Внутри каждое значение будет состоять из временной метки и показателя температуры. В таблице может быть несколько метрик.

   - Ориентированы на запись.
   - Предназначены для обработки постоянного потока входных данных.
   - Производительность зависит от количества отслеживаемых элементов, интервала опроса между записью новых значений и фактической полезной нагрузкой данных.

   Примеры: OpenTSDB, Prometheus, InfluxDB, TimescaleDB.

Time	CPU Temp	System Load	Memory Usage %
2019-10-31T03:48:05+00:00	37	0.85	92
2019-10-31T03:48:10+00:00	42	0.87	90
2019-10-31T03:48:15+00:00	33	0.74	87
2019-10-31T03:48:20+00:00	34	0.72	77
2019-10-31T03:48:25+00:00	40	0.88	81
2019-10-31T03:48:30+00:00	42	0.89	82
2019-10-31T03:48:35+00:00	41	0.88	82

Комбинированные типы БД

NewSQL и многомодельные БД являются разными типами баз данных, но решают одну группу проблем, вызванных полярными подходами SQL или NoSQL-стратегии.

10. **NewSQL БД**
    NewSQL базы данных наследуют реляционную структуру и семантику, но построены с использованием более современных, масштабируемых конструкций. Цель – обеспечить большую масштабируемость, нежели реляционные БД, и более высокие гарантии согласованности, чем в NoSQL. Компромисс между согласованностью и доступностью является фундаментальной проблемой распределённых баз данных, описываемой теоремой CAP.
    - возможность горизонтального масштабирования.
    - высокая доступность.
    - большая производительность и репликация.
    - небольшой функционал и гибкость.
    - немалое потребление ресурсов и необходимость специализированных знаний для работы с БД.
    Примеры: MemSQL, VoltDB, Spanner, Calvin, CockroachDB, FaunaDB, yugabyteDB.

11. **Многомодельные БД**
    Многомодельные базы данных – базы, объединяющие функциональные возможности нескольких видов БД. Преимущества такого подхода очевидны – одна и та же система может использовать различные представления для разных типов данных.
    Совместное размещение данных из нескольких типов БД в одной системе позволяет выполнять новые операции, которые в противном случае были бы затруднены или невозможны. Например, многомодельные базы могут позволить юзерам получить доступ к данным, хранящимся в разных типах БД, и управлять ими в рамках одного запроса, а также поддерживают

согласованность данных при выполнении операций, изменяющих информацию сразу в нескольких системах.
- помогают уменьшить нагрузку на СУБД.
- позволяют расширяться до новых моделей по мере изменения потребностей без внесения изменений в базовую инфраструктуру;
- обеспечивают непрерывный доступ и простое распределение данных;
- имеют линейную масштабируемость и просты для разработки.
Примеры: ArangoDB, OrientDB, Couchbase.

SQL

DB Testing