Базы данных

...

и что это такое

Базы данных и СУБД

База данных - набор данных, которые организованы определенным образом.

Системы управления базами данных (СУБД) - специальные приложения (или библиотеки) для управления базами данных, различных форм и размеров.

В современном мире сама база данных бесполезна, если с ней не связана СУБД для доступа к ее данным.

Чаще всего термин “База данных” используется как сокращение от “Система управления базами данных”

CRUD

- Create (создать) новые данные
- Read (прочитать) и получить информацию
- Update (обновить) уже существующие данные с использованием новых значений
- Delete (удалить) существующие данные, которые нам больше не нужны

Эти четыре основные операции сокращенно обозначаются *CRUD*. Этот термин встречается довольно часто когда мы работаем с системами баз данных или разрабатываем бэкенд системы.

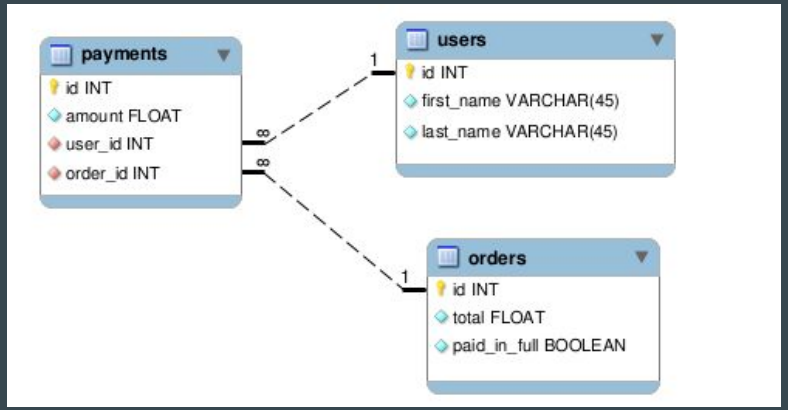
Различные парадигмы баз данных

Существуют различные типы баз данных, такие как: ключ-значение, документо-ориентированные или реляционные базы данных. Они различаются в основном тем, как хранят информацию и извлекают ее из памяти.

В дальнейшем мы сосредоточимся в основном на модели реляционных баз данных.

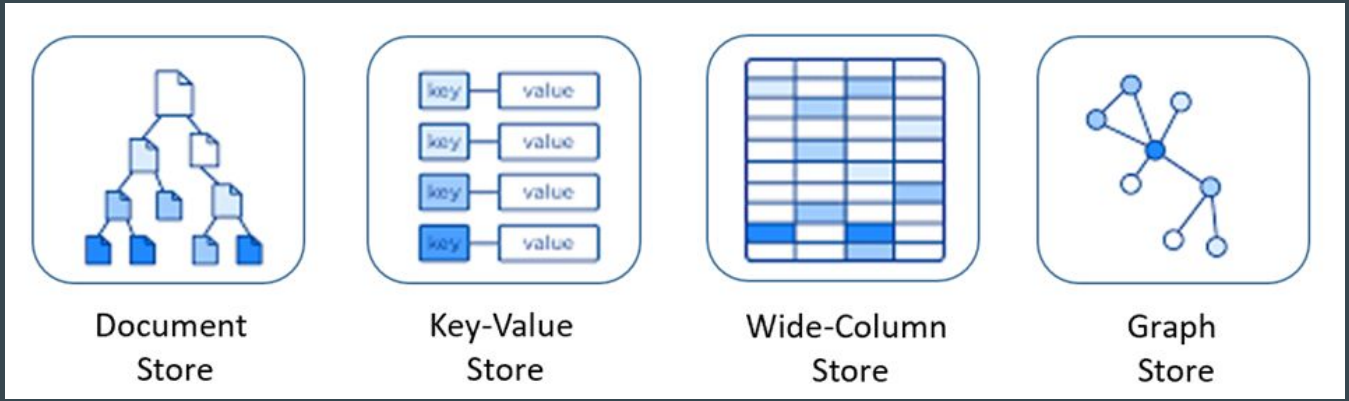
Реляционная база данных

Реляционная часть в названии этой парадигмы относится к нашей способности создавать сущности, которые имеют отношения друг с другом. Это дает нашим данным логическую структуру связи.



NoSQL

Нереляционная база данных - это база данных, в которой в отличие от большинства традиционных систем баз данных не используется табличная схема строк и столбцов.



CAP теорема (Eric Brewer, 2000)

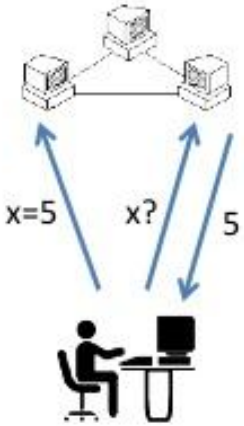
Согласованность (Consistency) - все рабочие узлы содержат одинаковую информацию.

Доступность (Availability) - возможность доступа к кластеру, даже если узел в кластере выходит из строя.

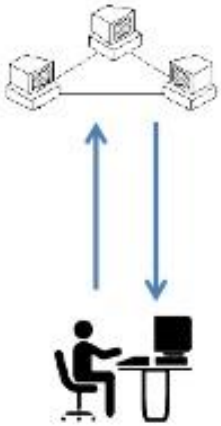
Терпимость к разделению сети (Partition Tolerance) - независимо от сбоев в работе сети узлы продолжают функционировать.

CAP Theorem

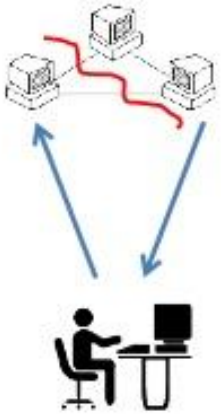
Consistency



Availability



Partition tolerance

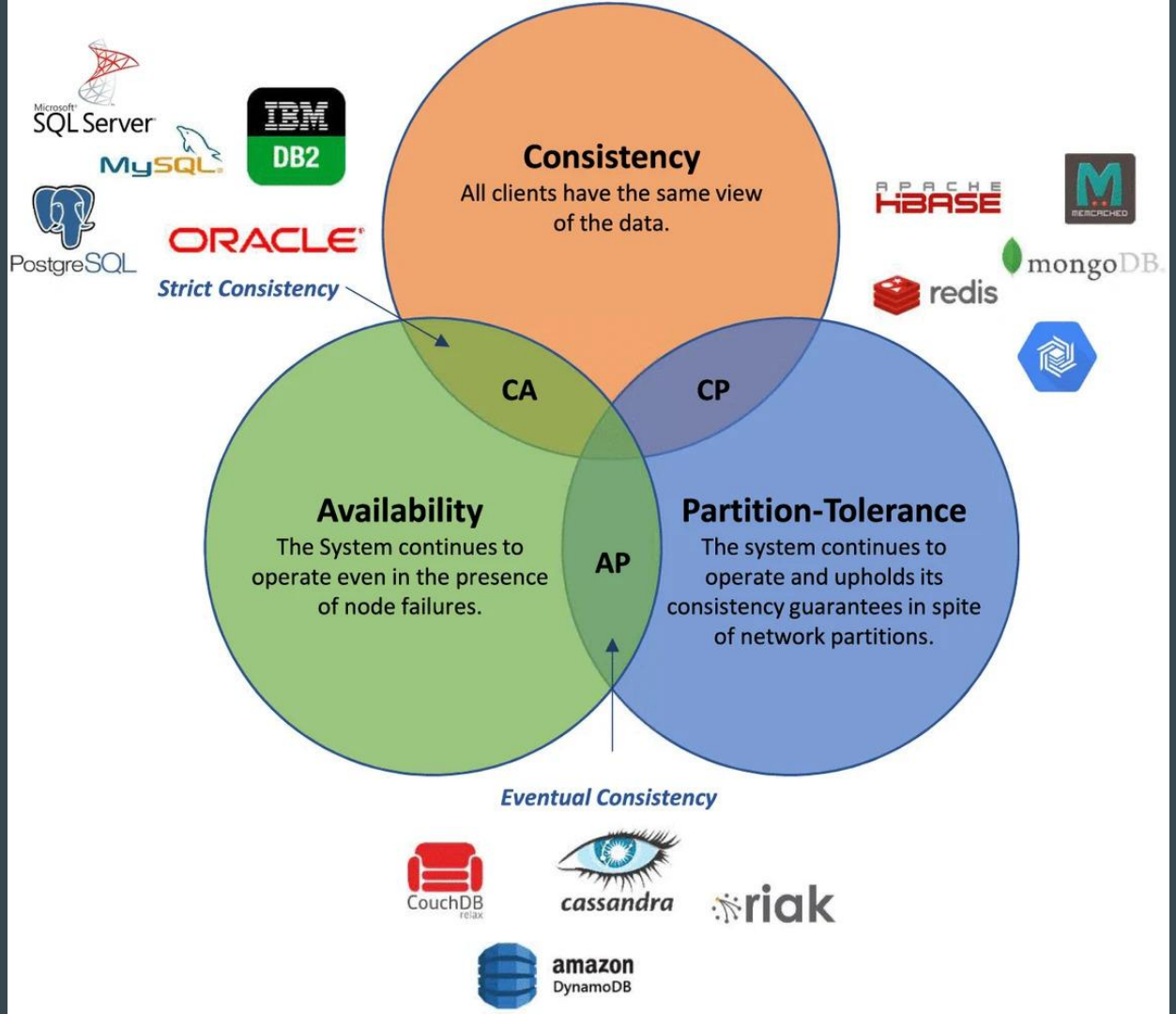


CAP теорема на примере «позвони - напомним»

1. У меня есть записная книжка и телефон - **нет разделимости.**
2. Не справляюсь с потоком - **нет доступности**. Беру в помощь еще одного человека.
3. а) У него своя записная книжка, но клиент звонит то мне, то ему - **нет согласованности.**

   б) Договариваемся вносить изменения в обе книжки.

Много времени уходит на синхронизацию, падает скорость приема звонков - **нет доступности.**


Microsoft SQL Server
MySQL
IBM DB2
PostgreSQL
ORACLE
Strict Consistency
Consistency
All clients have the same view of the data.
APACHE HBASE
MEMCACHED
mongoDB.
redis
CA
CP
Availability
The System continues to operate even in the presence of node failures.
AP
Partition-Tolerance
The system continues to operate and upholds its consistency guarantees in spite of network partitions.
Eventual Consistency
CouchDB
cassandra
riak
amazon DynamoDB

Минусы CAP теоремы

- Теорема описывает системы слишком упрощенно.
- Каждое понятие возведено в абсолют.
- Невозможно достичь идеально CAP для всех операций, но можно выбрать, где какой параметр важнее.

SQL VS NoSQL

- Структура
- Масштабируемость
- Для чего больше подходят

Структура реляционных БД

В реляционных СУБД данные представлены в виде таблиц, которые имеют заранее заданную жесткую схему данных.

Реляционные БД используют структурированный язык запросов(Structured Query Language, SQL) для определения и обработки данных, это один из наиболее гибких и распространенных языков запросов.

Структура нереляционных БД

Нереляционные базы данных, в свою очередь, предлагают динамическую структуру данных, которые могут храниться несколькими способами: ориентированно по колонкам, документо-ориентированно, в виде графов или на основе пар «ключ-значение».

Такая гибкость означает следующее:

- Вы можете создавать документы, не задавая их структуру заранее
- Каждый документ может обладать собственной структурой
- У каждой БД может быть собственный синтаксис
- Вы можете добавлять поля прямо во время работы с данными

Масштабируемость

В большинстве случаев SQL базы данных вертикально масштабируемые, то есть вы можете увеличивать нагрузку на отдельно взятый сервер, наращивая мощность центральных процессоров, объёмы ОЗУ или системы хранения данных. А NoSQL базы данных горизонтально масштабируемые. Это означает, что вы можете увеличивать трафик, распределяя его или добавляя больше серверов к вашей СУБД.

Для чего подходят

SQL — верный выбор для любого проекта, который может положиться на предопределенную структуру и заданные схемы. С другой стороны, NoSql — отличный вариант для быстрорастущих проектов без определённой схемы данных. В особенности если вы не можете определить схему для своей базы данных, вам не подходит ни одна из предлагаемых другими СУБД или в вашем проекте она постоянно меняется, как, например, в случае с мобильными приложениями, системами аналитики в реальном времени или контент-менеджмента.

Postgresql

...

Основные понятия

PostgreSQL - это реляционная система управления базами данных (РСУБД). Это система управления данными, представленными в виде отношений (relation).

Каждая таблица представляется как совокупность строк и столбцов, где строки соответствуют экземпляру объекта, конкретному событию или явлению, а столбцы - атрибутам (признакам, характеристикам, параметрам) объекта, события, явления.

При практической разработке БД таблицы - сущности называют таблицами, строки -экземпляры - записями, столбцы - атрибуты - полями.

Типы данных

- Числовые
- Символьные (строковые)
- Дата/время
- Логические
- JSON
- ...

подробнее: https://postgrespro.ru/docs/postgresql/9.6/datatype

Первичный ключ

В каждой таблице существует первичный ключ - это поле или набор полей, однозначно (уникально) идентифицирующих запись.

Первичный ключ должен быть минимально достаточным: в нем не должно быть полей, удаление которых из первичного ключа не отразится на его уникальности.

Отношения между таблицами

Существуют три разновидности связей между таблицами:

- один - к - одному
- один - ко - многим
- многие - ко - многим

Отношение один к одному

Отношение один - к - одному имеет место, когда одной записи в родительской таблице может соответствовать одна запись в дочерней таблице.

Отношение один ко многим

Отношение один - ко - многим имеет место, когда одной записи в родительской таблице может соответствовать несколько записей в дочерней таблице

Отношение многие ко многим

Отношение многие - ко - многим имеет место, когда:

- записи в родительской таблице может соответствовать больше одной записи в дочерней таблице
- записи в дочерней таблице может соответствовать больше одной записи в родительской таблице

Язык SQL, DDL, DML

SQL - это язык структурированных запросов (Structured Query Language), позволяющий хранить, манипулировать и извлекать данные из реляционных баз данных.

DDL - Data Definition Language (язык описания данных)

DML - Data Manipulation Language (язык манипулирования данными)

Примеры DDL и DML

```
CREATE TABLE person(
    name VARCHAR(32) NOT NULL,
     age INTEGER,
    CHECK(age >= 0 AND  age < 200)
);

INSERT INTO person (name, age)
    VALUES (‘Вольдемар', 22);

UPDATE person SET age = 17 WHERE name = ‘Вольдемар';

DELETE FROM person WHERE age < 18;
```

Нормализация

Нормализация - это удаление избыточности данных.

Избыточность - состояние БД, при котором в таблицах присутствуют лишние данные. Это приводит к увеличению занимаемого места на диске, а также к аномалиям включения, обновления, удаления.

Устранение избыточности производится, как правило, за счет декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты(т.е. факты, не выводимые из других хранимых фактов).

Нормализация. Первая нормальная форма.

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж (строка таблицы) содержит только одно значение для каждого из атрибутов(колонка таблицы).

Простыми словами

- в каждой колонке должно быть только одно значение
- не должно быть повторяющихся строк

1НФ до нормализации

name	hobbies
Суворов Алексей	Коньки, Мотоцикл
Окулов Ярослав	ClickHouse
Созинов Илья	Портреты, PlayStation
Белянин Алексей	Фотография, Велосипеды, Аудиотехника, Ножи
Збруев Роман	Гитара, Радиотехника

1НФ после нормализации

name	hobbie
Суворов Алексей	Коньки
Суворов Алексей	Мотоцикл
Окулов Ярослав	ClickHouse
Созинов Илья	Портреты
...	...
Збруев Роман	Радиотехника

Нормализация. Вторая нормальная форма.

Переменная отношения находится во второй нормальной форме (2НФ) тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо(функционально полно) зависит от ее потенциального ключа.

Простыми словами

- таблица в 1НФ
- есть первичный ключ
- все атрибуты зависят от первичного ключа целиком, а не от какой - то его части

2НФ до нормализации

name	project	project_client
Суворов Алексей	КайзерДом	Сергей И.
Суворов Алексей	Юнитраст	Иван С.
Окулов Ярослав	Доска почета	Аркадий Г.
Созинов Илья	КайзерДом	Сергей И.
Збруев Роман	КайзерДом	Сергей И.
Збруев Роман	Юнитраст	Иван С.

2НФ добавим первичный ключ

worker_id	project_id	name	project	project_client
1	1	Суворов Алексей	КайзерДом	Сергей И.
1	2	Суворов Алексей	Юнитраст	Иван С.
2	3	Окулов Ярослав	Доска почета	Аркадий Г.
3	1	Созинов Илья	КайзерДом	Сергей И.
4	1	Збруев Роман	КайзерДом	Сергей И.
4	2	Збруев Роман	Юнитраст	Иван С.

2НФ после нормализации(декомпозиции)

project_id	project	project_client
1	КайзерДом	Сергей И.
2	Юнитраст	Иван С.
3	Доска почета	Аркадий Г.

worker_id	project_id
1	1
1	2
2	3
3	1
4	1
4	2

worker_id	name
1	Суворов Алексей
2	Окулов Ярослав
3	Созинов Илья
4	Збруев Роман

Нормализация. Третья нормальная форма.

Переменная отношения находится в третьей нормальной форме (3НФ) тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

Простыми словами

- таблица в 2НФ
- все атрибуты зависят только от первичного ключа, но не от других атрибутов

3НФ до нормализации

tyre_id	tyre_name	supplier	supplier_phone
1	Nokian Hakka	ИП Шлепаков Л.	+79233213123
2	Nokian Nordman	ООО Русский Север	+79242343423
3	Continental ContiIce	ООО Автомакс	+79434234234
4	Matador	ИП Кузнецов П.	+79989493248
5	Nordmaster	ИП Иванов П.	+79646564345

3НФ после нормализации

tyre_id	tyre_name	supplier_id
1	Nokian Hakka	1
2	Nokian Nordman	2
3	Continental ContiIce	3
4	Matador	4
5	Nordmaster	5

id	supplier	supplier_phone
1	ИП Шлепаков Л.	+79233213123
2	ООО Русский Север	+79242343423
3	ООО Автомакс	+79434234234
4	ИП Кузнецов П.	+79989493248
5	ИП Иванов П.	+79646564345

Нормализация. Нормальная форма Бойса-Кодда(не бойся кода).

Переменная отношения находится в нормальной форме Бойса - Кодда(иначе в усиленной третьей нормальной форме) тогда и только тогда, когда каждая ее нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ.

Простыми словами

- таблица в 3НФ
- ключевые атрибуты не должны зависеть от неключевых

Нормализация. Нормальная форма Бойса-Кодда (пример)

Дано:

- Каждый сотрудник может курировать только ту работу для которой он квалифицирован
  - Максим - маркетинг
  - Рома - программирование
  - Илья - дизайн
  - ...
- Есть куча проектов над которыми они работают
- Для каждого из проектов могут быть выполнены и разработка, и дизайн и маркетинг
- Куратор по каждому из направлений у проекта может быть только один (для того чтобы не было неразберихи)

Нормализация. Нормальная форма Бойса-Кодда (пример)

project_id	task	resposible
1	Разработка	Рома
2	Маркетинг	Максим
2	Дизайн	Илья
1	Дизайн	Илья
3	Маркетинг	Диана
3	Разработка	Миша

Нормализация. Нормальная форма Бойса-Кодда (пример)

- первичный ключ - составной = проект + задача
- но проявляется зависимость части первичного ключа (задачи) от ответственного
- зная кем является ответственный можно четко сказать какую задачу он выполняет в проекте

Нормализация. Нормальная форма Бойса-Кодда (пример)

id	name	skill
1	Рома	Разработка
2	Максим	Маркетинг
3	Илья	Дизайн
4	Илья	Дизайн
5	Диана	Маркетинг
6	Миша	Разработка

workers

project_id	responsible_id
1	1
2	2
2	3
1	4
3	5
3	6

projects

Нормализация. Четвертая нормальная форма.

Переменная отношения находится в четвертой нормальной форме (4НФ) тогда и только тогда, когда она находится нормальной форме Бойса - Кодда и не содержит нетривиальных многозначных зависимостей.,

Простыми словами

- таблица в НФ БК
- устраняются многозначные зависимости

Нормализация. Четвертая нормальная форма (пример)

Для одного сотрудника есть множество проектов и множество увлечений

worker_id	project	hobbie
1	Сантехник	Радиотехника
1	КайзерДом	Гитара
2	FabioRoss	Футбол
3	КайзерДом	Хоккей
3	Доска Почета	Гитара

Нормализация. Четвертая нормальная форма (пример)

Первый вопрос - а что будет если увлечений у данного сотрудника больше чем проектов?

worker_id	project	hobbie
1	Сантехник	Радиотехника
1	КайзерДом	Гитара
2	FabioRoss	Футбол
3	КайзерДом	Хоккей
3	Доска Почета	Гитара

Нормализация. Четвертая нормальная форма (пример)

worker_id	project
1	Сантехник
1	КайзерДом
2	FabioRoss
3	КайзерДом
3	Доска Почета

workers_projects

worker_id	hobbie
1	Радиотехника
1	Гитара
2	Футбол
3	Хоккей
3	Гитара

workers_hobbies

Нормализация. Пятая нормальная форма.

Переменная отношения находится в пятой нормальной форме (иначе - в проекционно-соединительной нормальной форме) тогда и только тогда, когда каждая нетривиальная зависимость соединения в ней определяется потенциальным ключом (ключами) этого отношения.

Простыми словами

- таблица в 4НФ
- устраняются нетривиальные зависимости
- (декомпозиция без потерь)

Нормализация. Пятая нормальная форма (пример)

- Миша делает только Frontend
- Рома наоборот, делает все, кроме Frontend
- Миша участвует в большом количестве проектов, Ярик только в одном

worker	project	part
Миша	КайзерДом	Frontend
Рома	КайзерДом	Backend
Рома	Сантехник	API Integration
Ярик	Доска почета	Backend
Миша	Доска Почета	Frontend
Миша	Резина	Frontend

Нормализация. Пятая нормальная форма (пример)

Декомпозируем таким образом, чтобы при объединении декомпозированных таблиц мы получили исходную таблицу

worker	project	part
Миша	КайзерДом	Frontend
Рома	КайзерДом	Backend
Рома	Сантехник	API Integration
Ярик	Доска почета	Backend
Миша	Доска Почета	Frontend
Миша	Резина	Frontend

Нормализация. Пятая нормальная форма (пример)

worker	project
Миша	КайзерДом
Рома	КайзерДом
Рома	Сантехник
Ярик	Доска почета
Миша	Доска Почета
Миша	Резина

worker	part
Миша	Frontend
Рома	Backend
Рома	API Integration
Ярик	Backend

project	part
КайзерДом	Frontend
КайзерДом	Backend
Сантехник	API Integration
Доска почета	Backend
Доска Почета	Frontend
Резина	Frontend