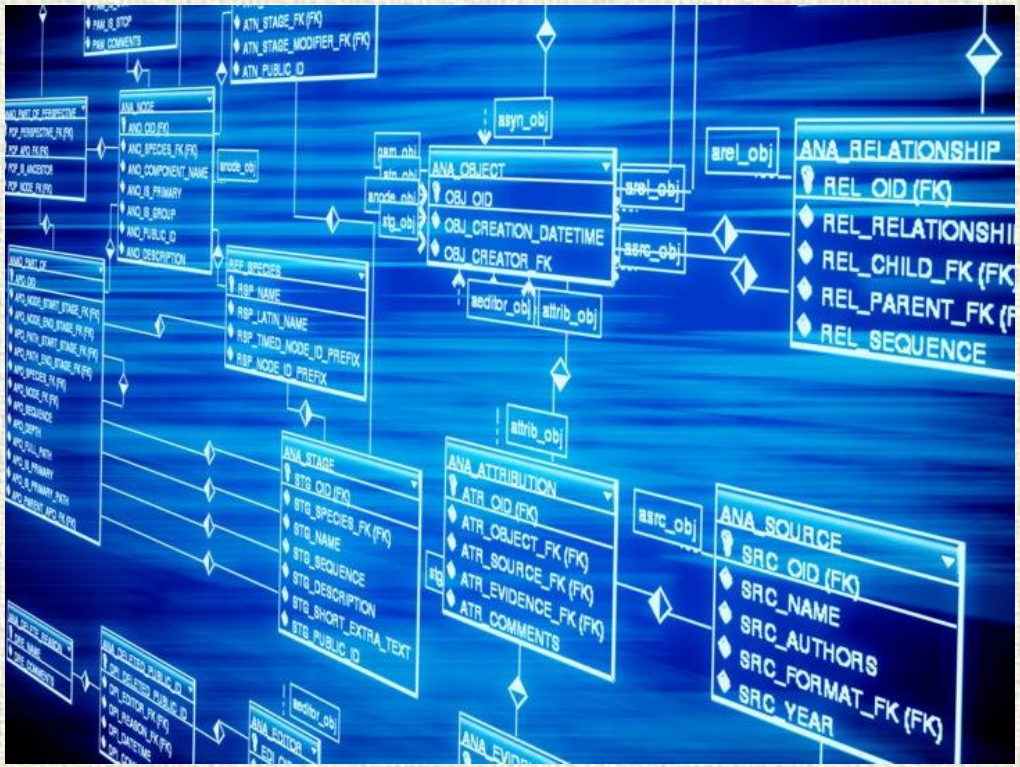
Базы данных

Что это?

База данных – совокупность структурированных данных разного типа, взаимодействие с которыми происходит по определенным правилам.

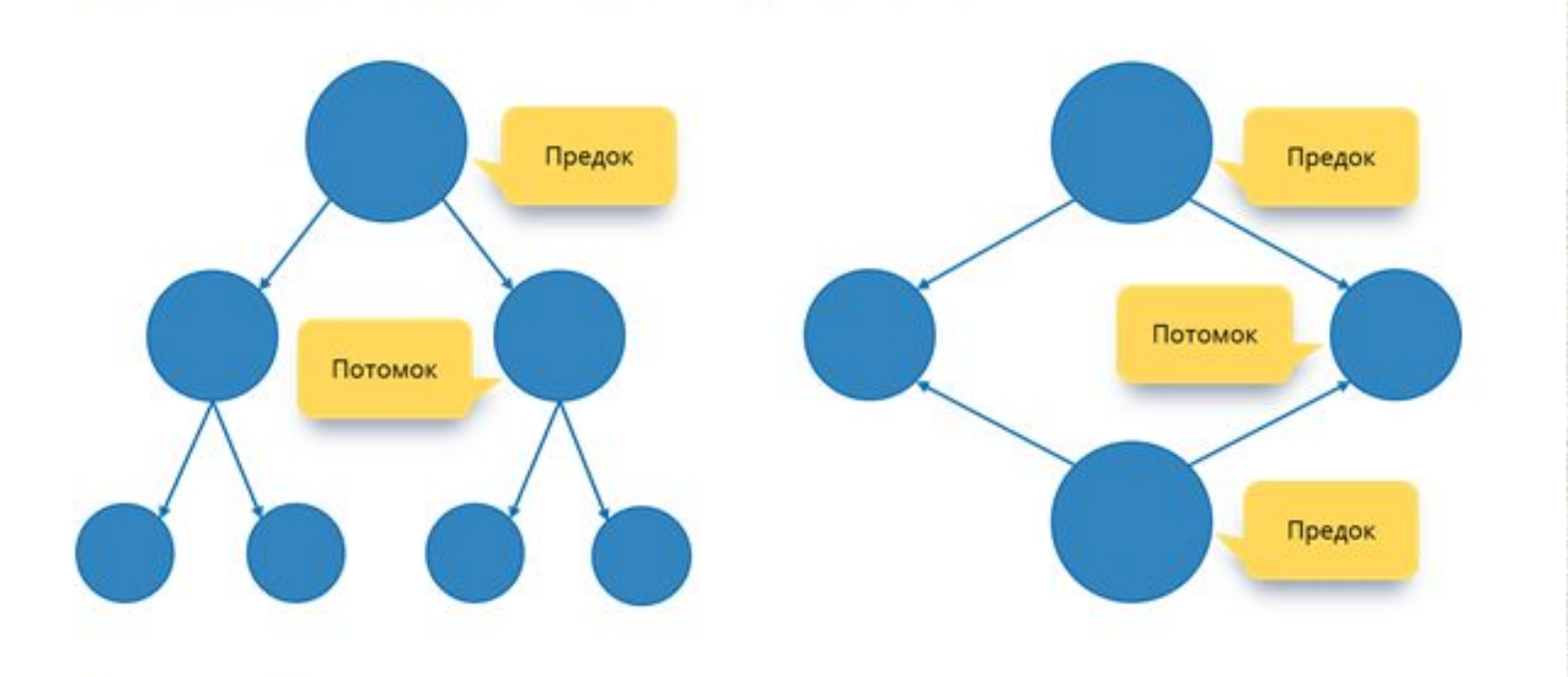
Действие с данными:

1)Добавление
2)Поиск и фильтрация
3)Редактирование
4)Удаление



Типы БД

1. Иерархическая – многоуровневая подчинённая структура, где у каждого потомка 1 предок.
2. Сетевая - многоуровневая подчинённая структура, где у каждого потомка может быть несколько предков.

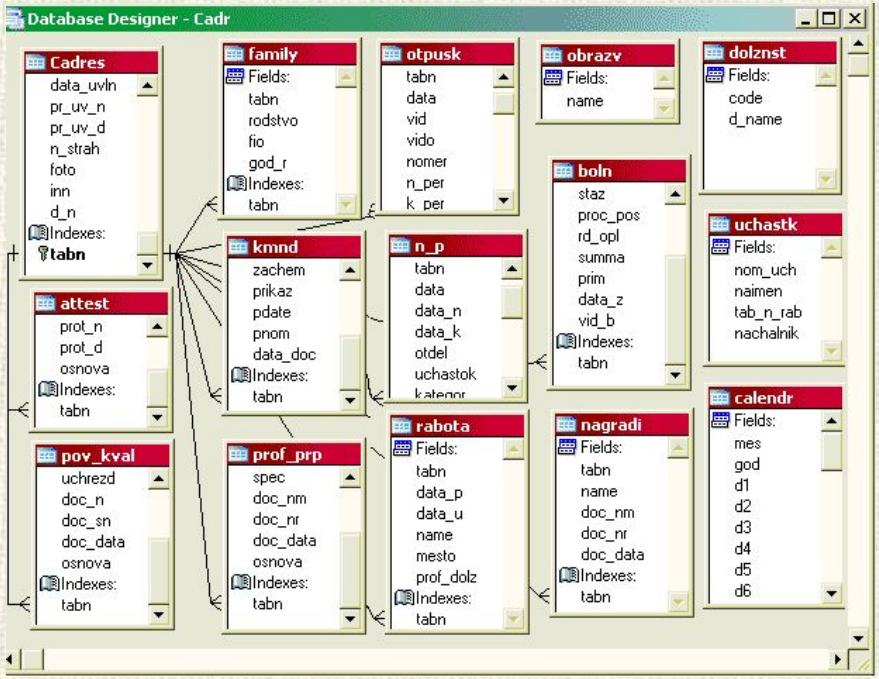


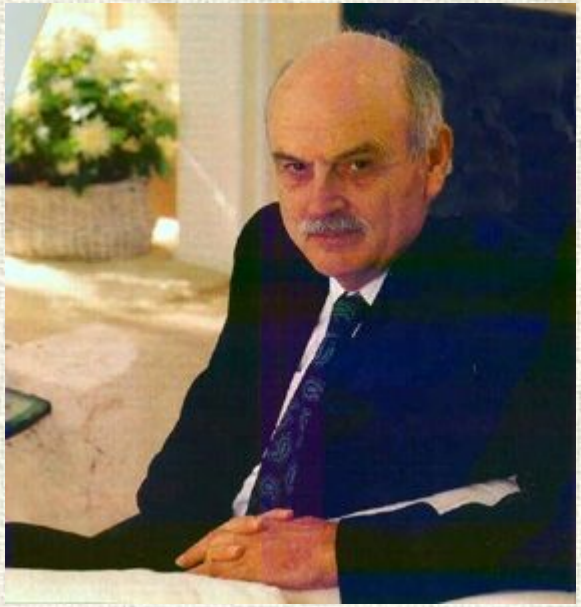
Типы БД

3. Реляционная* – структура данных в виде связанных двумерных таблиц.

* В теории множеств совокупности данных, которые можно отобразить в виде таблицы соответствует термин «отношение».

Реляционные базы данных подробно описываются в математике (благодаря Кодду), следовательно имеют надежную программную реализацию.





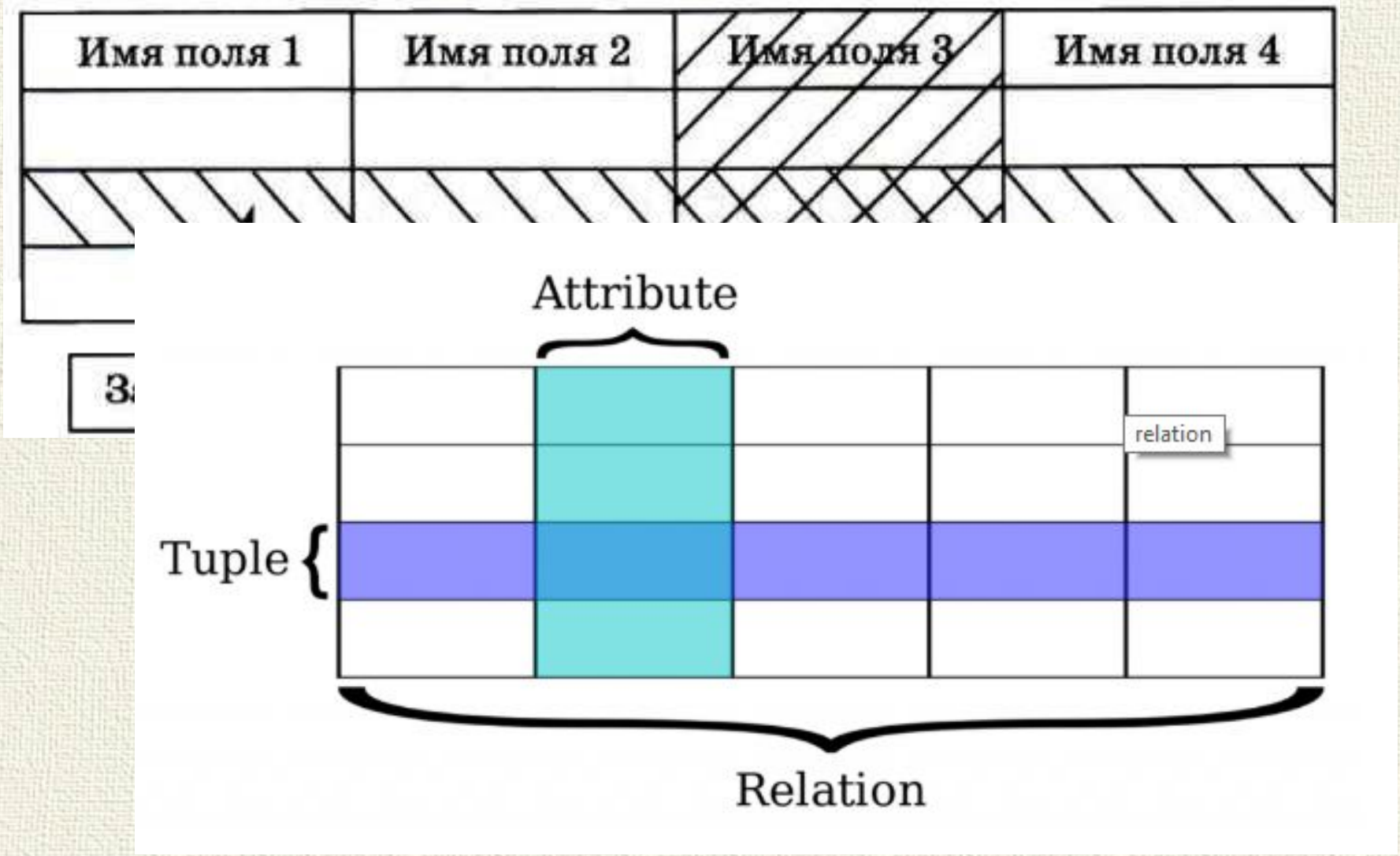
Эдгар Франк Кодд

Важные понятия

- Сущность (entity) – описание типа объектов, хранимых в базе (класс)
- Объект – конкретный экземпляр сущности
- Атрибут - свойства объекта (название столбца в таблице)
- Запись (кортеж) - строка в таблице (набор конкретных значений объекта)
- Поле – значение конкретного атрибута (ячейка)

Для однозначности обращения каждая запись должна быть однозначно идентифицирована.

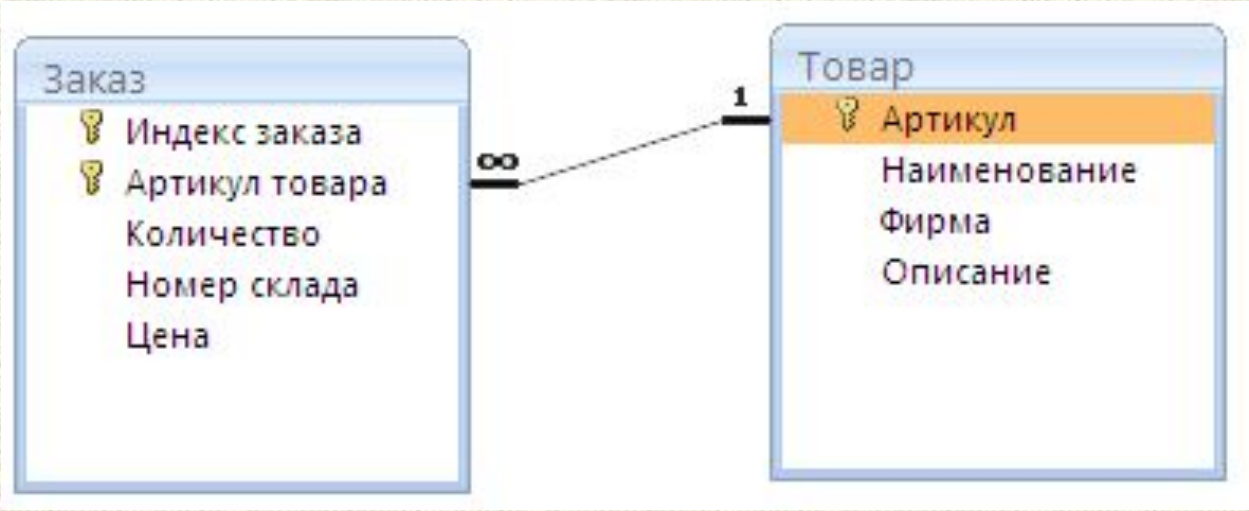
Реляционная БД



Ключи

Первичный ключ – поле (или набор полей), позволяющие однозначно идентифицировать запись БД (чаще всего используют поле-идентификатор).

Вторичный (внешний) ключ – поле (или набор полей), используемые для связи нескольких таблиц БД. Не может включать первичный ключ.



Таким образом достигается то, что не ключевые атрибуты таблиц могут изменяться независимо от связанных сущностей, не порождая при этом противоречий и коллизий.

СУБД

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств, позволяющих работать с базами данных.

Назначение:

1) Управление созданием и использованием БД
2) Контроль доступа к данным
3) Проверка ограничений проектирования и учет особенностей модели БД

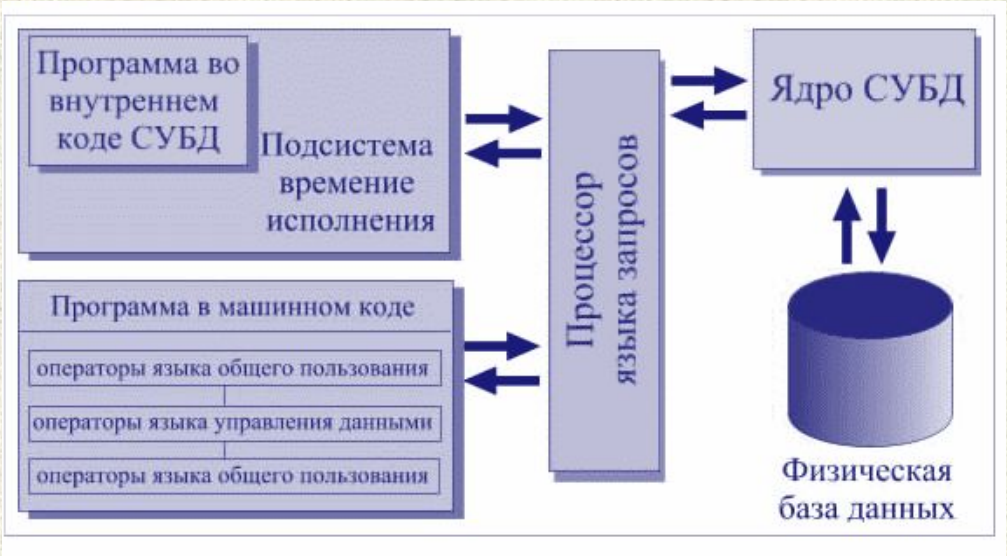
Где используют?

- Банки, картотеки, любые крупные системы, в которых нужно хранить и изменять данные, а также иметь доступ к ним с разными ролями. (системы, где уже не обойтись xls-файликом)
- Веб-сайты - сервер использует БД для удобства управления информацией и взаимодействия с пользователем.
- Приложения (мобильные и десктопные) используют локальные базы для удобства хранения данных по некоторым правилам.
- Любой программный продукт, подразумевающий отделение бизнес-логики и уровня хранения данных.

Архитектура СУБД

Основные компоненты:

- Ядро - процессы, сеть, память, файловая система и т.д.
- Диспетчер данных - транзакции, кэш
- Диспетчер запросов - парсер запроса, оптимизатор, исполнитель
- Набор инструментов для служебных операций – резервное копирование, восстановление, мониторинг



Реляционные СУБД

Преимущества:

- Простая схема данных для пользователя.
- Логическая и физическая независимость от данных.
- Целостность и защищенность данных.

Недостатки:

- Относительно низкая скорость доступа к данным.
- Не универсальное решение для любой предметной области.
- Недостаточная гибкость при добавлении своих типов данных и операций.

SQL

SQL (англ. structured query language — «язык структурированных запросов») — декларативный язык программирования, применяемый для работы с данными в реляционных БД (которые могут управляться различными СУБД).

Типы SQL СУБД:

- Oracle
- MySQL
- PostgreSQL
- MS SQL
- SQLite
- Access

Что может?

Возможности SQL:

- Создавать таблицы БД
- Изменять таблицы БД
- Удалять таблицы БД
- Вставлять записи (строки) в таблицы БД
- Редактировать записи в таблицах БД
- Извлекать выборочную информацию из таблиц БД
- Удалять выборочную информацию из БД
- И т.д.

Язык SQL не зависит от регистра

Программа представляет собой набор инструкций для СУБД

В конце каждой инструкции ставится **;**

Команды SQL

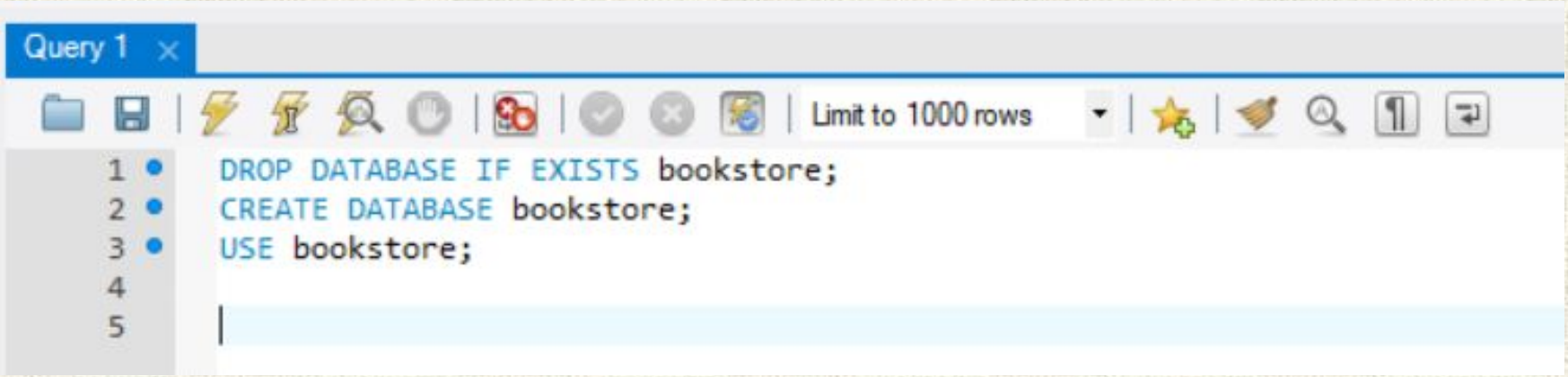
Работа с БД:

CREATE DATABASE testDB; - создание БД

DROP DATABASE testDB; - удаление БД

DROP DATABASE IF EXISTS testDB; - удалить БД, если существует

USE testDB; - использовать БД в скрипте по умолчанию



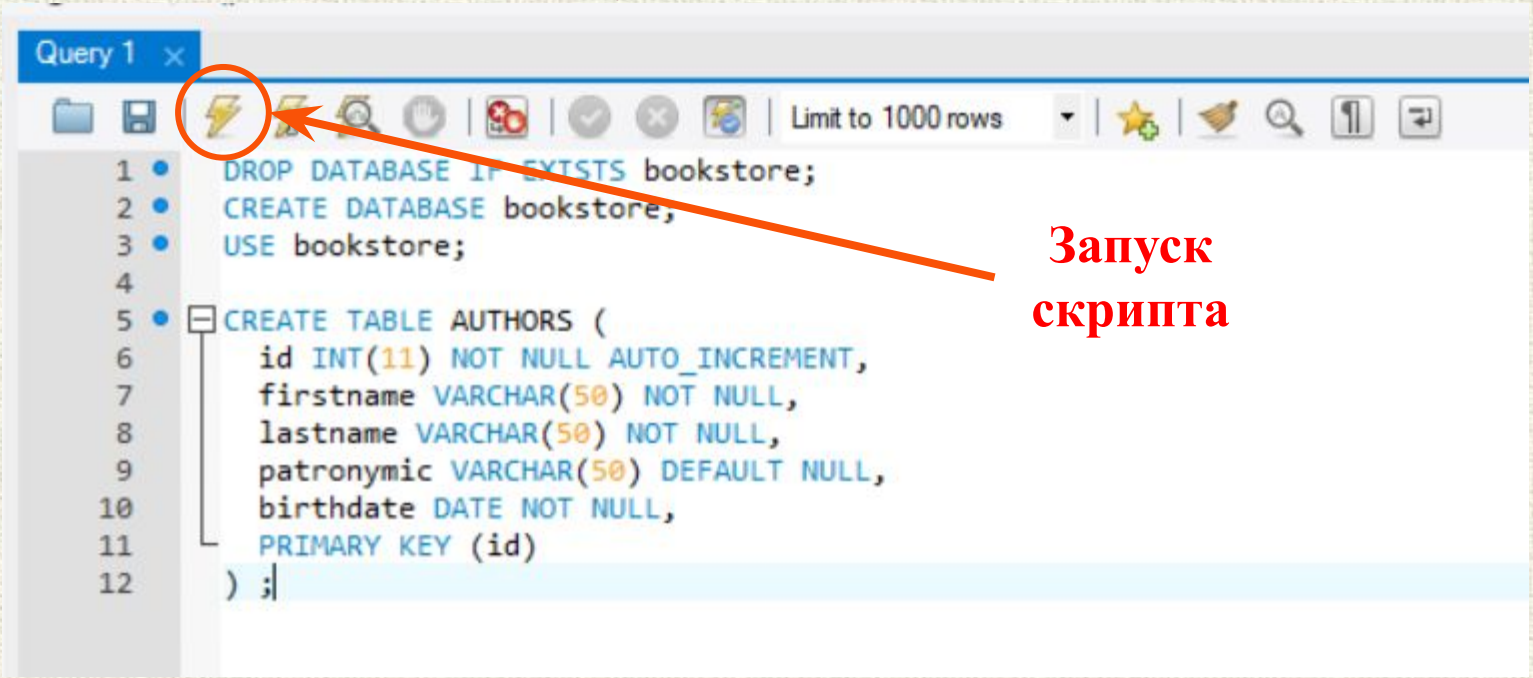
Команды SQL

Создание таблицы:

```
CREATE TABLE ИМЯ (
  имя_поля1 тип1 значение_по_умолчанию,
  имя_поля2 тип2 значение_по_умолчанию,
PRIMARY KEY (имя_поля) # первичный ключ
) ;

CREATE TABLE AUTHORS (
  id INT(11) NOT NULL AUTO_INCREMENT,
  firstname VARCHAR(50) NOT NULL,
  lastname VARCHAR(50) NOT NULL,
  patronymic VARCHAR(50) DEFAULT NULL,
  birthdate DATE NOT NULL,
  PRIMARY KEY (id)
) ;
```

Команды SQL



Помимо числовых и строковых типов можно использовать множества, дату, время и т.д.

Подробнее о типах

NOT NULL - не может быть пустым.
DEFAULT NULL - по умолчанию пустое.
AUTO_INCREMENT - автоматически увеличивается на 1.

Команды SQL

Добавление записи:
INSERT INTO

```
INSERT INTO AUTHORS VALUES (NULL,"Вася","Пупкин", NULL, "2000-1-1");
```

или

```
INSERT INTO AUTHORS (lastname, firstname, birthdate) VALUES("Иванов","ИВАН","1980-7-1");
```

В таблице SQL в принципе нет начала и конца. Есть большая куча строк. А порядок в этой куче появляется только когда вы явно указываете сортировку.

Без сортировки база может возвращать строки в любом порядке, в каком захочет.

Команды SQL

Выборка данных (одна или несколько записей частично или целиком):
SELECT

Выбрать все записи со всеми полями из таблицы AUTHORS
SELECT * FROM AUTHORS;

Число записей в таблице
SELECT COUNT(*) FROM AUTHORS;

Выбрать все записи, но только поля firstname и lastname
SELECT firstname, lastname FROM AUTHORS;

Команды SQL

Добавление условий в выборке:
WHERE

Выбрать все записи, удовлетворяющие условию, с выбранными полями

```
SELECT firstname, lastname FROM AUTHORS WHERE firstname = "Петр";
SELECT firstname, lastname FROM AUTHORS WHERE patronymic IS NULL;
SELECT firstname, lastname FROM AUTHORS WHERE patronymic IS NOT NULL;
```

Маски – позволяют заменить один или несколько символов при запросе.

% - несколько символов (включая 0)

_ - ровно один символ

```
SELECT firstname, lastname FROM AUTHORS WHERE lastname LIKE "Иван%";
SELECT firstname, lastname FROM AUTHORS WHERE lastname LIKE "%ов";
SELECT firstname, lastname FROM AUTHORS WHERE lastname LIKE "Иван__";
```

Команды SQL

Сложные условия

SELECT firstname, patronymic , birthdate FROM AUTHORS WHERE patronymic IS NULL AND lastname LIKE "Иван__";

SELECT firstname, patronymic, birthdate FROM AUTHORS WHERE patronymic IS NULL OR lastname LIKE "Иван__";

SELECT firstname, lastname, birthdate, patronymic FROM AUTHORS WHERE patronymic IS NULL OR (lastname LIKE "Иван%" AND birthdate > "1990-1-1");

Команды SQL

Сортировка выборки:
ORDER BY

Сортировать вывод по полю birthdate
SELECT * FROM AUTHORS ORDER BY birthdate;

В обратном порядке
SELECT * FROM AUTHORS ORDER BY birthdate DESC;

Условие и сортировка
SELECT * FROM AUTHORS WHERE lastname LIKE "%ов" ORDER BY birthdate;

Сортировка по нескольким полям
SELECT * FROM AUTHORS ORDER BY firstname, lastname;

Команды SQL

Редактировать записи:
UPDATE

Обновить все записи, установив lastname в значение "Сидоров"
UPDATE AUTHORS SET lastname = "Сидоров";

Обновить записи, где firstname = "Петр"
UPDATE AUTHORS SET lastname = "Петров" WHERE firstname = "Петр";

Обновить все записи, где firstname = "Петр" или firstname = "Иван"
UPDATE AUTHORS SET lastname = "Петров" WHERE firstname = "Петр" OR firstname = "Иван";

Команды SQL

Изменить структур таблицы:
ALTER TABLE

Добавить поле
ALTER TABLE AUTHORS ADD COLUMN sex CHAR(1);

Обновить параметры поля
ALTER TABLE AUTHORS CHANGE sex gender TINYINT(1);

Удалить поле
ALTER TABLE AUTHORS DROP COLUMN gender ;

Переименовать таблицу
ALTER TABLE AUTHORS RENAME WRITERS;

Команды SQL

Удалить записи:
DELETE

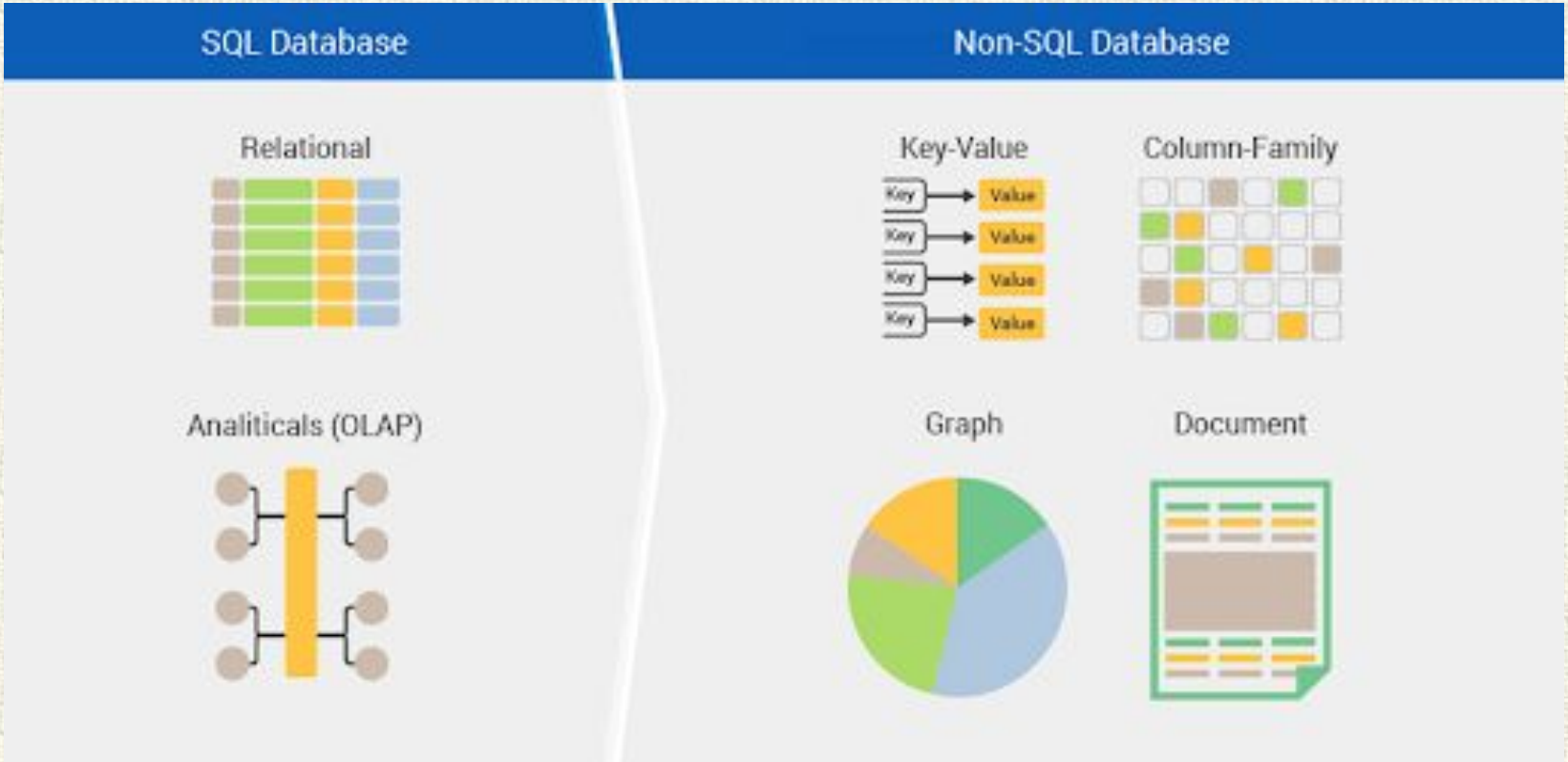
Удалить записи, удовлетворяющие условию DELETE FROM AUTHORS WHERE firstname = "Петр" OR firstname = "Иван";

Удалить все записи из таблицы AUTHORS
DELETE FROM AUTHORS;

Базы данных NoSQL

Что это?

NoSQL (от англ. not only SQL - не только SQL) – модели баз данных, решающие проблемы масштабируемости и целостности данных. По структуре отличаются от реляционных (доступ к которым происходит при помощи SQL).

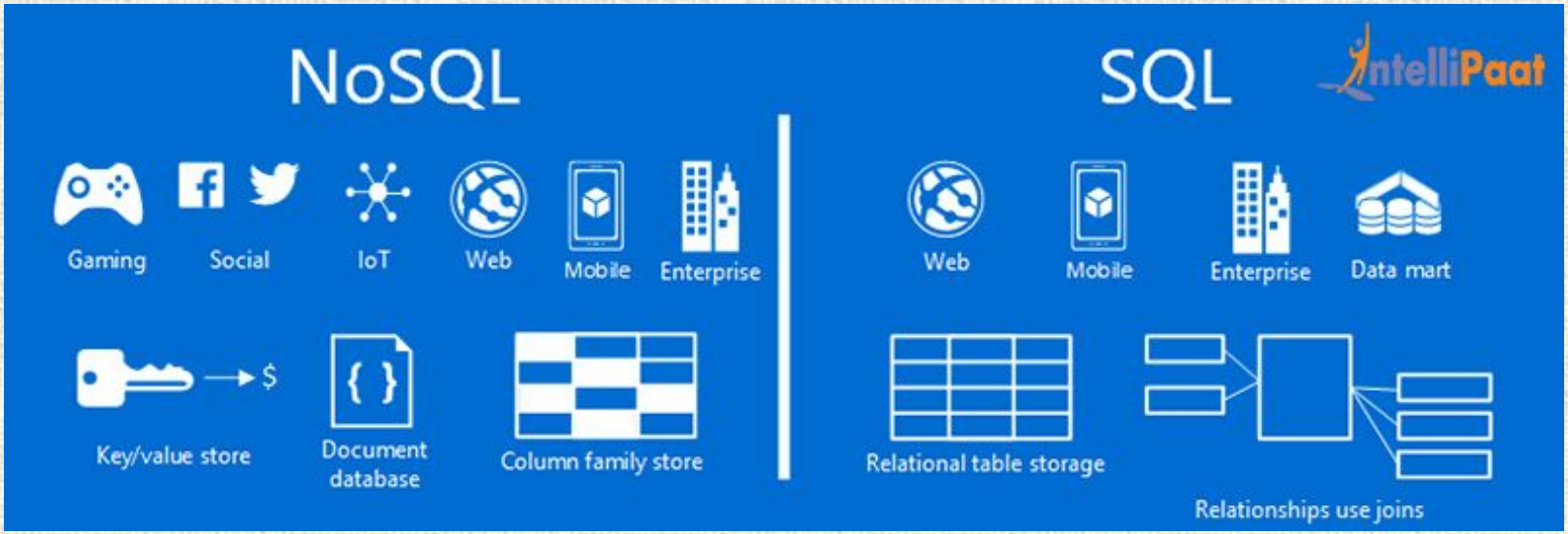


Типы NoSQL БД

Описание схемы данных в случае использования NoSQL-решений осуществляется через использование различных структур данных: хеш-таблиц, деревьев и т.п.

В зависимости от модели данных и способов их обработки различают четыре основных типа NoSQL БД:

- «ключ-значение» (key-value store)
- хранилища семейств колонок (column database)
- графовые базы данных (graph database)
- документно-ориентированные (document store)



Ключ - значение

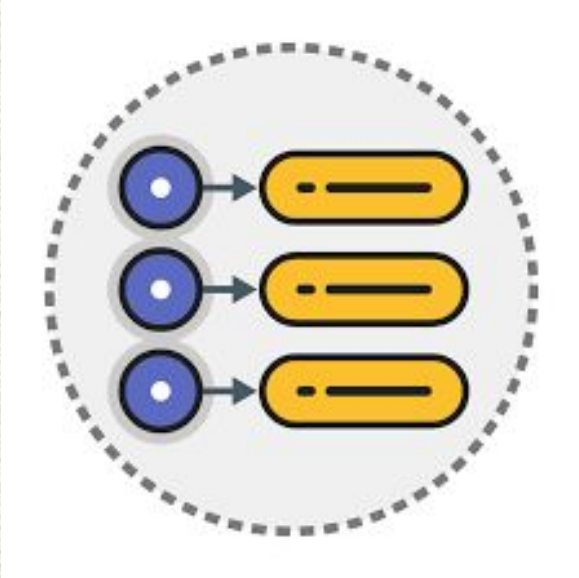
Является простейшим хранилищем данных, использующим ключ для доступа к значению.

Используются для:

- хранения изображений,
- создания специализированных файловых систем,
- в качестве кэшей для объектов

Примеры:

- Berkeley DB,
- MemcacheDB,
- Redis,
- Riak,
- Amazon DynamoDB.



Хранилища семейств колонок

Гибрид между NoSQL и реляционной БД. В этом хранилище данные хранятся в виде разреженной матрицы, строки и столбцы которой являются ключами.

Используются для:

- систем управления контентом
- блогов
- регистрации событий

Примеры:

- Apache HBase,
- Apache Cassandra,
- Hypertable
- SimpleDB

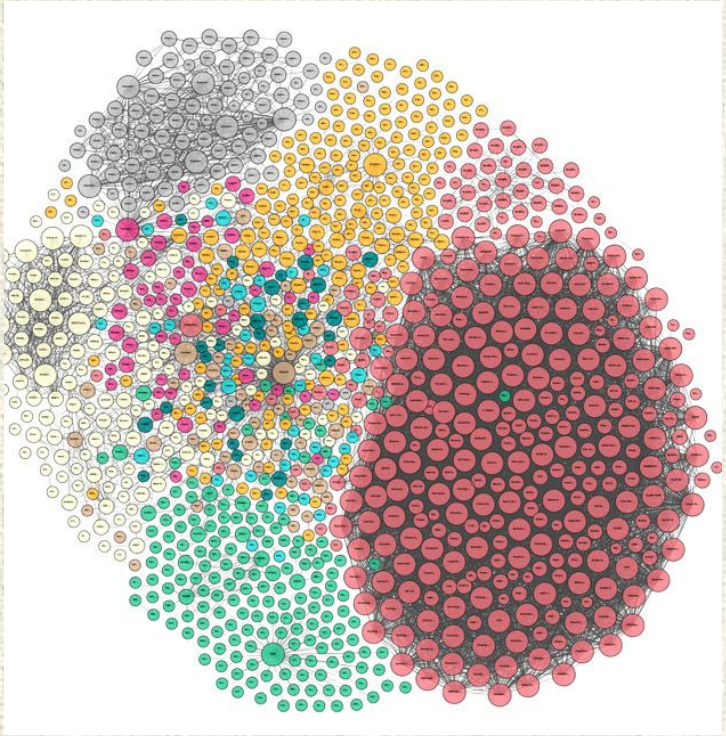
Графовые БД

Применяются для задач, в которых данные имеют большое количество связей.

Так как рёбра графа материализованы (являются хранимыми), то обход графа не требует дополнительных вычислений (как в SQL), но для нахождения начальной вершины обхода требуется наличие индексов.

Используются для:

- Систем выявления мошенничества
- Социальных сетей
- Сервисов рекомендаций



Примеры:

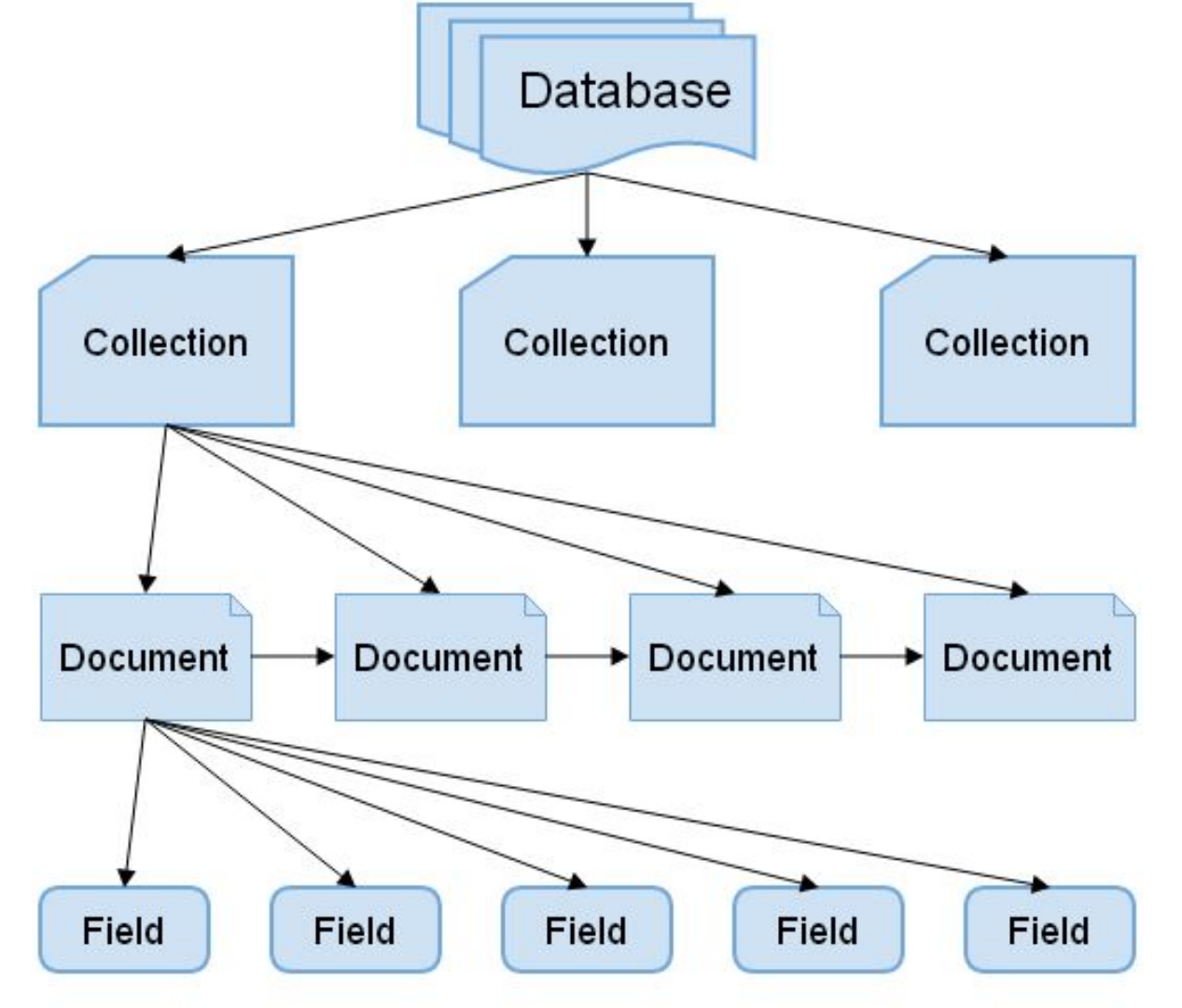
- Neo4j
- OrientDB
- InfiniteGraph
- Titan

Документоориентированные БД

- В основе БД лежат документные хранилища (document store), имеющие структуру дерева (иногда леса).
- Структура дерева начинается с корневого узла и может содержать несколько внутренних и листовых узлов.
- Листовые узлы содержат данные, которые при добавлении документа заносятся в индексы.
- Документы могут быть организованы (сгруппированы) в коллекции.
- Коллекции могут содержать документы любого вида, однако принято в коллекции объединять похожие по структуре документы.
- Коллекции могут содержать другие коллекции.

Документоориентированные БД

Примеры:
- CouchDB
- MarkLogic
- MongoDB
- eXist
- Berkeley DB



Так что выбрать?

Сегодня у большинства компаний просто нет таких объемов данных и других специфических условий работы, в которых NoSQL решения были бы выгодны в качестве основной базы данных.

NoSQL хранилища показывают себя с очень хорошей стороны в симбиозе с реляционными базами данных. Например, в системах, где основной объем информации хранит SQL, а за кэш отвечает NoSQL.

Нормализация данных	Данные в виде агрегатов
• Целостность информации при обновлении (меняем запись в одной таблице, а не в нескольких) • Ориентированность на широкий спектр запросов к данных	• Оптимизация только под определенный вид запросов • Сложности при обновлении денормализованных данных
• Неэффективна в распределенной среде • Низкая скорость чтения при использовании объединений (joins) • Несоответствие объектной модели приложения физической структуре данных (impedance mismatch, решается с помощью Hibernate etc.)	• Лучший способ добиться большой скорости на чтение в распределенной среде • Возможность хранить физически объекты в том виде, в каком с ними работает приложение (легче кодировать и меньше ошибок при преобразовании) • Родная (native) поддержка атомарности на уровне записей

MongoDB

MongoDB (от англ. humongous - огромный) - это документоориентированная СУБД, использующая JSON-подобные документы и схему базы данных.

Где применяют?

- Хранение и регистрация событий
- Для систем управления документами и контентом
- Электронная коммерция
- Игры
- Мобильные приложения
- Хранилище операционных данных веб-страницы
- Хранение комментариев
  - ✔ Рейтинги
  - ✔ Регистрация пользователя
  - ✔ Сеансы данных
  - ✔ и т.д.
- Проекты, использующие итеративные или гибкие методологии разработки
- Управление статистикой в реальном времени

JSON

JSON (англ. JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript.

JSON-объект:

1) Набор пар “ключ: значение”.

Ключ - только строка, значение - любая форма (объект, словарь, запись, структура и т.д.)

2) Упорядоченный набор значений (массив, список, строка, число и т.д.)

JSON

```
{
    "firstName": "Иван",
    "lastName": "Иванов",
    "address": {
        "streetAddress": "Московское ш., 101, кв.101",
        "city": "Ленинград",
        "postalCode": 101101
    },
    "phoneNumbers": [
        "812 123-1234",
        "916 123-4567"
    ]
}
```

MongoDB основные понятия

Документ (объект) - структура, хранящая данные.

Ключ - строковый уникальный идентификатор в документе, метка, с которой ассоциирована определенная часть данных.

Если какому-то ключу не сопоставлено значение, то этот ключ просто опускается в документе и не употребляется.

Значение – данные, хранящиеся по некоторому ключу в документе. Типы данных.

Коллекция - множество документов, может содержать самые разные документы, имеющие различную структуру и различный набор ключей.

Идентификатор документа - уникальный ключ “_id”, который может быть автоматически сгенерирован (бинарное значение размером 12 байт).

MongoDB пример документа

```
{
"_id": ObjectId("3b64c116ec8cf5adb508cd05"),
"name": "Ivan",
"age": 23,
"weight": 78.2,
"married": true,
"birthday": ISODate("1995-04-21T14:54:04.691Z"),
"company": {
      "companyId" : ObjectId("5adb5163d053b64c116ec8d3"),
      "position" : "seller",
      "since": ISODate("2010-12-05T09:00:00.001Z")
},
"wife": {
       "wifeId": ObjectId("053b64c116ec8d35adb5163d")
},
"children": [],
"updatedAt" : ISODate("2018-04-21T14:54:04.691Z"),
  "createdAt" : ISODate("2018-04-21T14:54:04.691Z")
}
```

Команды MongoDB

Работа с БД:

- **use** имя_БД - выбрать БД (если ее нет, создается автоматически).
- **db.stats()** - статистика по БД
- **db.dropDatabase()** - удаление БД

Команды MongoDB

Добавление документа:

insert()

может добавлять как один, так и несколько документов

```
db.users.insert({"name": "Tom", "age": 28, languages: ["english", "spanish"]})
```

Можно определить документ как переменную:

```
document=({"name": "Bill", "age": 32, languages: ["english", "french"]})
db.users.insert(document)
```

insertOne(): добавляет один документ

insertMany(): добавляет несколько документов

Команды MongoDB

Добавление или редактирование документа: save()

Если метод находит документ с указанным значением _id, то документ обновляется.
Если же с подобным _id нет документов, то документ добавляется.
Если параметр _id не указан, то документ добавляется, а параметр _id генерируется автоматически.

```
db.users.save({name: "Eugene", age : 29, languages: ["english", "german", "spanish"]})
```

Команды MongoDB

Выборка данных:

find()

db.users.find()

db.users.find({name: "Tom"})

pretty() - форматный вывод

```
> db.users.find({name: "Tom"}).pretty()
{
        "_id" : ObjectId("5aef68393a6f54f3394871c4"),
        "name" : "Tom",
        "age" : 28,
        "languages" : [
                "english",
                "spanish"
        ]
}
```

Команды MongoDB

по двум ключам
db.users.find({name: "Tom", age: 32})

в массиве есть элемент
db.users.find({languages: "english"})

в массиве несколько элементов
db.users.find({languages: ["english", "german"]})

1й элемент массива равен “english”
db.users.find({"languages.0": "english"})

Доступ к ключам вложенных объектов осуществляется через точку:
db.users.find({"company.name": "microsoft"})

```
{
        "_id" : ObjectId("5aef6ff63a6f54f3394871c9"),
        "name" : "Alex",
        "age" : 28,
        "company" : {
                "name" : "microsoft",
                "country" : "USA"
        }
} _
```

Команды MongoDB

Сложные условия:

db.users.find ({age: {$lt : 30}})
db.users.find ({age: {$gt : 30, $lt: 50}})
db.users.find ({age: {$eq : 22}}) ~ db.users.find ({age: 22})

db.users.find ({age: {$ne : 22}})
db.users.find ({age: {$in : [22, 32]}})

db.users.find ({name: "Tom", $or : [{age: 22}, {languages: "german"}]})
db.users.find ({$and : [{name: "Tom"}, {age: 32}]})

Подробнее про условные операторы:

Команды MongoDB

Редактирование документа:
update(query, objNew, options)

- query: принимает запрос на выборку документа, который надо обновить;
- objNew: представляет документ с новой информацией, который заместит старый при обновлении;
- options: дополнительные параметры при обновлении документов. Может быть: upsert и multi.

upsert: true - mongodb будет обновлять документ, если он найден, и создавать новый, если такого документа нет
multi указывает, должен ли обновляться первый элемент в выборке (используется по умолчанию) или же должны обновляться все документы в выборке.

```
db.users.update({name : "Tom"}, {name: "Tom", age : 25}, {upsert: true})
```

Команды MongoDB

db.users.update({name : "Tom"}, {$inc: {age:2}}) - увеличить значение на 2

оператор $set - обновление отдельного поля, если поля нет - оно добавляется.
db.users.update({name : "Tom", age: 29}, {$set: {age : 30}})
db.users.update({name : "Tom", age: 29}, {$set: {salary : 300}})
db.users.update({name : "Tom"}, {$set: {name: "Tom", age : 25}}, {multi:true})

оператор $unset - для удаления отдельного ключа.
db.users.update({name : "Tom"}, {$unset: {salary: 1}})
db.users.update({name : "Tom"}, {$unset: {salary: 1, age: 1}})

Команды MongoDB

Обновление массивов

$push - добавить еще одно значение к уже существующему:
db.users.updateOne({name : "Tom"}, {$push: {languages: "russian"}})

$each - добавить сразу несколько значений:
db.users.update({name : "Tom"}, {$push: {languages: {$each: ["russian", "spanish", "italian"]}}})

$addToSet - добавить объекты в массив, если их еще нет в массиве:
db.users.update({name : "Tom"}, {$addToSet: {languages: "russian"}})

Команды MongoDB

Обновление массивов

$pop - удалить элемент из массива:

db.users.update({name : "Tom"}, {$pop: {languages: 1}})

Указывая для ключа languages значение 1, мы удаляем первый элемент ***с конца****. Чтобы удалить первый элемент сначала массива, надо передать отрицательное значение:*

db.users.update({name : "Tom"}, {$pop: {languages: -1}})

$pull - удалить каждое вхождение элемента в массив:

db.users.update({name : "Tom"}, {$pull: {languages: "english"}})

$pullAll - удалить каждое вхождение для нескольких элементов:

db.users.update({name : "Tom"}, {$pullAll: {languages: ["english", "german", "french"]}})

Команды MongoDB

Удаление документа:
remove()

удалить все объекты с именем Tom
db.users.remove({name : "Tom"})

удалить только первый объект из выборки
db.users.remove({name : "Tom"}, true)

удалить все документы в коллекции
db.users.remove({})

Команды MongoDB

Работа с коллекциями:

- db.createCollection(“users”) - явно создать коллекцию
- db.getCollectionNames() - получить имена всех коллекций
- db.users.renameCollection("accounts") - переименовать коллекцию
- db.users.drop() - удалить коллекцию
- db.users.count() - число документов в коллекции

- db.users.find({name: "Tom"}).count() - число документов в выборке
- db.users.distinct("name") - получить уникальные значения по ключу “name”