

ОСНОВЫ SQL

Зачем нужны базы данных

- Программам нужно где-то хранить свои данные
- В простых случаях можно хранить данные в виде файлов
- Но если данных становится слишком много, то возникают проблемы с производительностью
- Кроме того, каждый раз приходится писать свой код, чтобы обновлять/удалять/добавлять данные в эти файлы
- И файлы ненадежны – если в середине обработки файла программа или ОС упадет, то файл останется в том состоянии, в котором был на тот момент
- Чтобы избавиться от этих проблем, появились **базы данных**

База данных

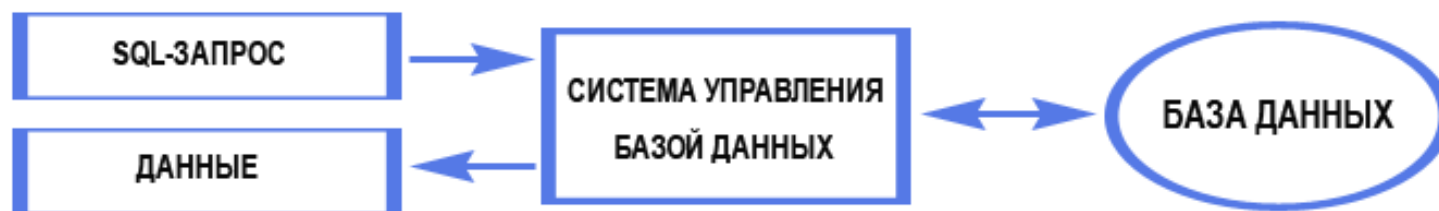
- **База данных (БД)** – это набор сведений, хранящихся некоторым упорядоченным образом
- Т.е. это хранилище данных
- На англ. база данных - **database (DB)**
- Для работы с базами данных используют **системы управления базами данных (СУБД)**
- На англ. – **Database Management System (DBMS)**

Системы управления базами данных

- **Система управления базами данных (СУБД)** – это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д.
- Т.е. СУБД позволяют создавать базы и работать с ними и их данными
- И СУБД предоставляют некоторый язык для работы с БД
- Во многих СУБД (но не во всех) в качестве языка используется **SQL (Structured Query Language – язык структурированных запросов)**

SQL

- **SQL** – язык запросов, позволяющий создавать/изменять/удалять базы данных, а также искать в них данные
- Примерно так выглядит взаимодействие пользователей с базой данных



Что дают базы данных

- Хранилище данных с высокой производительностью и сжатием данных
- Возможность настройки прав доступа к данным
- Транзакционность – в случае ошибки данные откатываются к предыдущему корректному сохраненному состоянию
- Журналирование (логирование) – все операции над БД записываются в лог-файл, который можно просматривать, чтобы находить ошибки, подозрительные операции и т.д.
- Удобные языки. Во многих СУБД используется стандартизованный язык SQL для работы с базами данных – не нужно реализовать свою логику вставки, редактирования, удаления данных
- Множество различных проверок и валидаций при работе с данными
- Множество удобных программ и библиотек для работы с БД

Виды баз данных

- В БД данные хранятся в структурированном виде, но структурировать данные можно по-разному
- В зависимости от структуры хранения данных СУБД делятся на несколько типов:
 - **Реляционные** – данные хранятся в виде таблиц
 - **Сетевые** – данные хранятся в виде графа (связанных между собой узлов)
 - **Объектно-ориентированные** – данные хранятся в виде объектов
 - И др.

Реляционные СУБД

- В **реляционных** СУБД данные хранятся в виде таблиц
- Реляционные СУБД наиболее распространены и являются инструментом общего назначения
- В курсе будем рассматривать именно реляционные СУБД

Популярные реляционные СУБД:

- Oracle
- MySQL
- MS SQL
- PostgreSQL
- SQLite

Будем использовать MySQL, потому что она очень распространена и бесплатна

Все эти СУБД используют SQL, поэтому изучить другие потом будет несложно

Классификация по способу доступа к БД

- **Клиент-серверная СУБД:**
 - Сама СУБД установлена на сервере, и БД находятся на сервере
 - Клиенты делают запросы к серверу СУБД
 - Примеры: **Oracle, MySQL, MS SQL, PostgreSQL**
- **Встраиваемая СУБД:**
 - Не требуется отдельный сервер и отдельная установка – СУБД просто является частью приложения, где она используется
 - Обычно является подключаемой библиотекой
 - Предназначена для локального хранения данных
 - Пример: **SQLite**

- **NoSQL** – общее название для нереляционных СУБД
- Сюда относятся очень разные СУБД, с совершенно разными концепциями и возможностями
- Такие СУБД заточены на определенный круг задач и не подходят или плохо подходят за пределами своей области применения
- **Популярные NoSQL СУБД:**
 - MongoDB
 - Redis
 - Neo4j

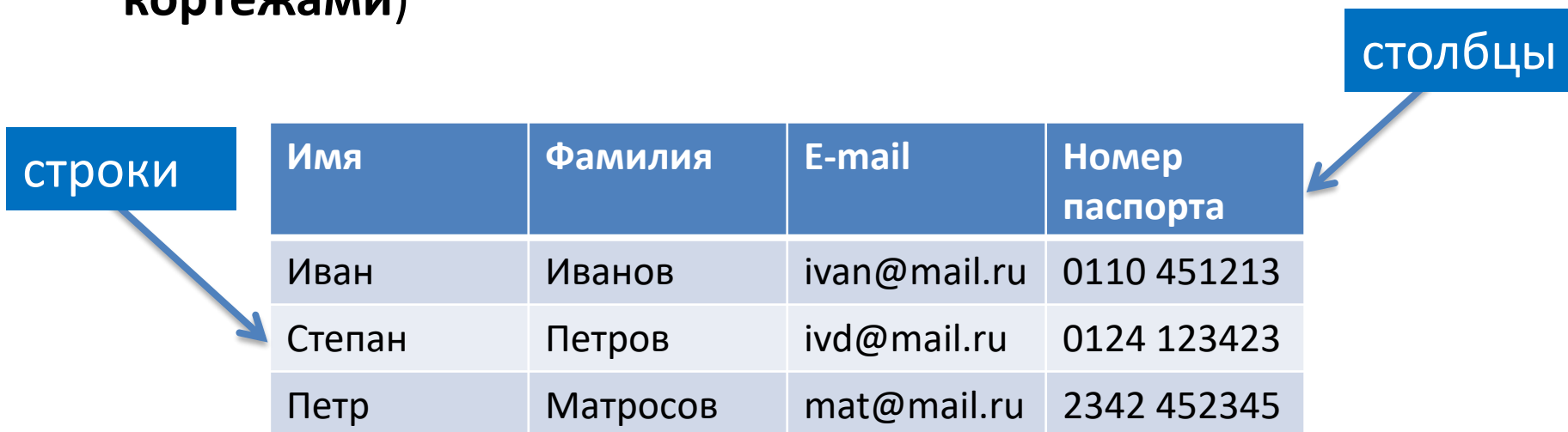
Статьи по NoSQL

- <https://aws.amazon.com/ru/nosql/>
- <https://mcs.mail.ru/blog/sravnenie-sql-i-nosql-kak-vybrat-sistemu-hraneniya-dannyh>

Теория по реляционным СУБД

Таблица

- Ключевое понятие реляционных баз данных – **таблица**
- **Таблица** – это хранилище сущностей (данных) одного вида
- Таблица состоит из **столбцов** (их еще называют **поля** и **атрибуты**) и **строк** (их еще называют **записями** и **кортежами**)



Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

The diagram illustrates the structure of a table. A blue box labeled 'строки' (rows) has an arrow pointing to the first column of the table. Another blue box labeled 'столбцы' (columns) has an arrow pointing to the first row of the table.

Столбцы

столбцы

строки

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

- **Столбец** имеет название и тип данных
- Название каждого **столбца** уникально внутри таблицы, каждый столбец хранит значения одного типа
- **Столбцы** отвечают за структуру данных в таблице – т.е. какие параметры есть у сущностей (данных), которые хранятся в таблице

Строки

столбцы

строки

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

- **Строки** отвечают за конкретные сущности (данные), которые хранятся в таблице
- Например, в первой строке в примере содержится информация о человеке Иван Иванов и т.д.

Аналогия с ООП

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

- Аналогия с ООП – таблица это класс, столбцы – это поля, а строки – это конкретные объекты класса

Таблица

столбцы

строки

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

- В таблице запрещено иметь полностью одинаковые строки
- В таблице может не быть ни одной строки, но должен быть хотя бы один столбец

Порядок столбцов

столбцы

строки

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

- Столбцы таблицы находятся в некотором порядке
- Обычно порядок фиксируется при создании таблицы
- Но на этот порядок не следует полагаться, всегда стоит обращаться к столбцам по именам

Первичный ключ

- Таблица должна иметь некоторое поле (или совокупность полей), по которым можно однозначно найти любую строку
- Такое поле (или совокупность полей) называется **первичным ключом (primary key)**
- Например, в этой таблице первичным ключом можно выбрать номер паспорта, т.к. по любому известному значению паспорта мы однозначно найдем всю строку

Имя	Фамилия	E-mail	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110 451213
Степан	Петров	ivd@mail.ru	0124 123423
Петр	Матросов	mat@mail.ru	2342 452345

Составной первичный ключ

- Если первичный ключ состоит из нескольких столбцов, то такой первичный ключ называют **составным**
- Например, если мы разобьем столбец «**Номер паспорта**» на 2 отдельных столбца – «**Серия паспорта**» и «**Номер паспорта**», то это будет составной первичный ключ
- По-отдельности серия и номер паспорта не уникальны, а в совокупности - уникальны

Имя	Фамилия	E-mail	Серия паспорта	Номер паспорта
Иван	Иванов	ivan@mail.ru	0110	451213
Степан	Петров	ivd@mail.ru	0124	123423
Петр	Матросов	mat@mail.ru	2342	452345

Суррогатный первичный ключ

- Но обычно в качестве первичного ключа не берут какие-то осмысленные данные, а вводят отдельный числовой столбец, который обычно называют **идентификатором (ID)**
- Такой первичный ключ называют **суррогатным**, потому что он не несет информационного смысла

ID	Имя	Фамилия	E-mail	Номер паспорта
1	Иван	Иванов	ivan@mail.ru	0110 451213
2	Степан	Петров	ivd@mail.ru	0124 123423
3	Петр	Матросов	mat@mail.ru	2342 452345

Суррогатный первичный ключ

- ID не является порядковым номером в полном смысле этого слова
- Это просто некоторое уникальное число для каждой строки
- Обычно оно заполняется по порядку при вставке строк в таблицу

ID	Имя	Фамилия	E-mail	Номер паспорта
1	Иван	Иванов	ivan@mail.ru	0110 451213
2	Степан	Петров	ivd@mail.ru	0124 123423
3	Петр	Матросов	mat@mail.ru	2342 452345

Уникальность


- Значения первичного ключа должны быть **уникальными**, т.е. не могут повторяться в пределах таблицы
- Потому что иначе поле не было бы первичным ключом – строка находилась бы не однозначно

ID	Имя	Фамилия	E-mail	Номер паспорта
1	Иван	Иванов	ivan@mail.ru	0110 451213
2	Степан	Петров	ivd@mail.ru	0124 123423
3	Петр	Матросов	mat@mail.ru	2342 452345

Связь между таблицами

- Допустим, мы хотим хранить в базе данных пользователей форума и их сообщения
- Мы хотели бы хранить их так – в столбце сообщения хранить перечень сообщений, т.е. как бы иметь вложенную таблицу

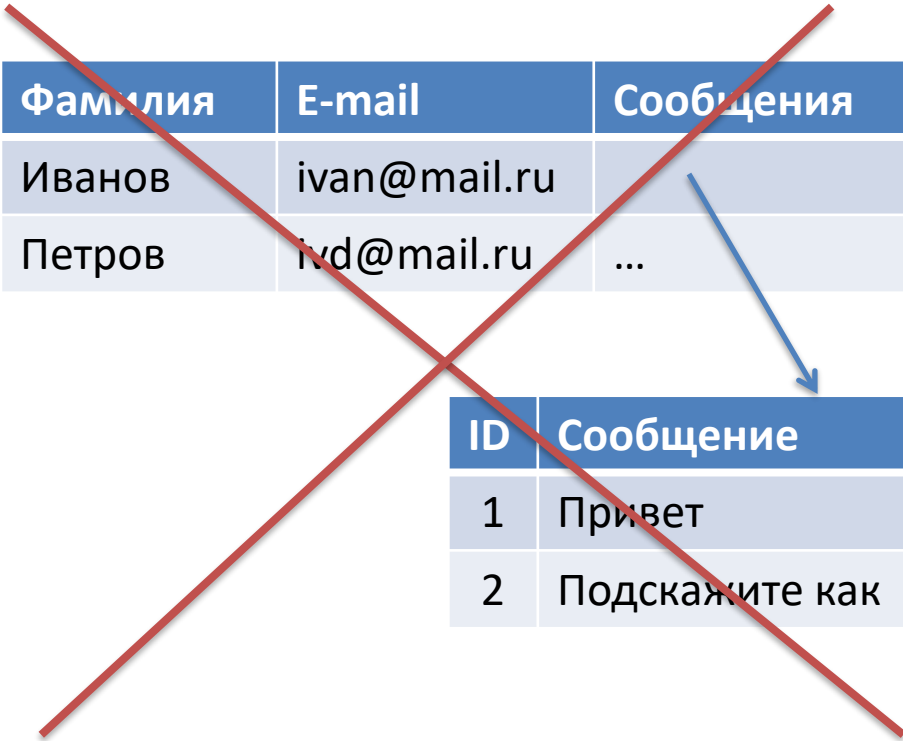
ID	Имя	Фамилия	E-mail	Сообщения
1	Иван	Иванов	ivan@mail.ru	
2	Степан	Петров	ivd@mail.ru	...



ID	Сообщение	Время
1	Привет	20:01
2	Подскажите как	20:02

Связь между таблицами

- Но в реляционных СУБД такое невозможно
- Ячейка таблицы может содержать только одно значение, которое принадлежит типу столбца



ID	Имя	Фамилия	E-mail	Сообщения
1	Иван	Иванов	ivan@mail.ru	
2	Степан	Петров	ivd@mail.ru	...

ID	Сообщение	Время
1	Привет	20:01
2	Подскажите как	20:02

Связь между таблицами

- Чтобы задать связь, сначала делаются две отдельные таблицы
- Заметим, что в таблице пользователей теперь нет поля «Сообщения»

ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

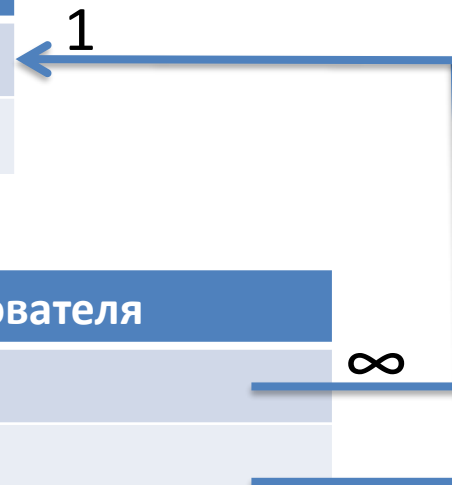
ID	Сообщение	Время
1	Привет	20:01
2	Подскажите как	20:02

Связь между таблицами

- Затем в таблице сообщений добавляют поле «ID пользователя»
- У сообщений пользователя это поля заполняется идентификатором пользователя

ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

ID	Сообщение	Время	ID пользователя
1	Привет	20:01	1
2	Подскажите как	20:02	1



Связь между таблицами

- В таблице сообщений имеется поле ID, которое содержит первичный ключ таблицы пользователей
- Так как первичный ключ позволяет однозначно найти строку в таблице, то по этому номеру мы всегда можем найти пользователя

ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

ID	Сообщение	Время	ID пользователя
1	Привет	20:01	1
2	Подскажите как	20:02	1

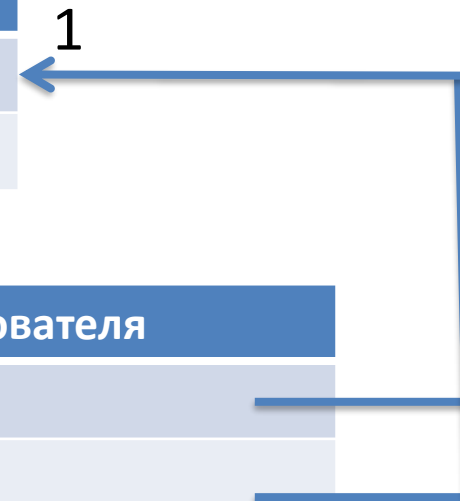


Внешний ключ

- Если одна таблица содержит столбец (или совокупность столбцов), который является первичным ключом в другой таблице, то этот столбец (или совокупность столбцов) называется **внешним ключом (foreign key)**

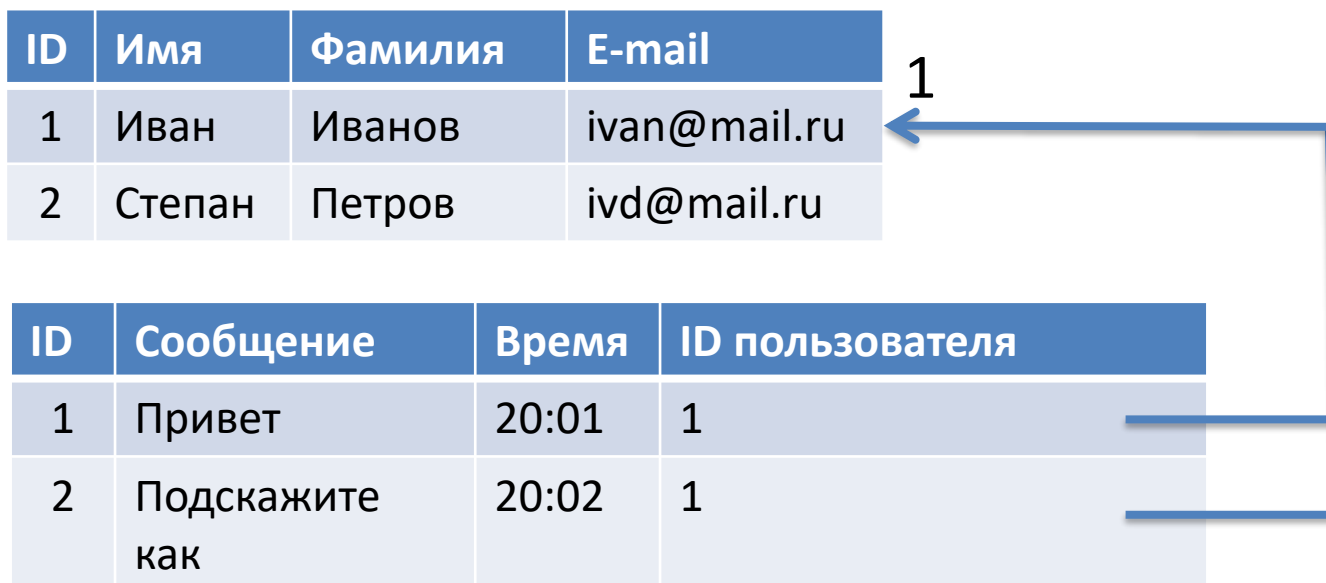
ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

ID	Сообщение	Время	ID пользователя
1	Привет	20:01	1
2	Подскажите как	20:02	1



Виды связей

- В данном примере связь между таблицами называется **один ко многим** (или **многие к одному**)
- Связь имеет такое название, потому что каждой строке одной таблицы может соответствовать много строк в другой таблице



Связь многие ко многим

- Что делать если нужно сделать связь многие ко многим?
- Например, отношения между таблицами **Автор** и **Книга**
- У книги может быть много авторов, а автор мог написать много книг

ID	Имя	Фамилия
1	Лев	Толстой
2	Брайен	Керниган
3	Деннис	Ричи

ID	Название	Год
1	Война и мир	1869
2	Анна Каренина	1876
3	Язык C	1978

Связь многие ко многим

- Чтобы задать такую связь, нужна вспомогательная таблица
- Она будет содержать записи, состоящие из пары внешних ключей
- И в этой таблице еще может быть свой ID (иногда его не делают)



Связь один ко многим

- Связь один ко многим также может быть выполнена при помощи вспомогательной таблицы
- Но при этом нужно навесить ограничение уникальности на столбец «ID сообщения», иначе получится многие ко многим

ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

ID	Сообщение	Время
1	Привет	20:01
2	Подскажите	20:02

ID	ID польз	ID сообщ
1	1	1
2	1	2



Сравнение способов для 1 ко многим

- **Способ с внешним ключом:**
 - Проще в реализации
 - Работает быстрее
- **Способ с вспомогательной таблицей:**
 - Сложнее в реализации
 - Работает медленнее
 - Но зато мы не меняем данные в таблицах, которые связываем – связи хранятся в отдельной таблице
 - В целом, этот способ применяется редко

Связь один к одному

- Иногда встречается связь **один к одному**
- Например, связь между пользователем и его настройками
 - Тут предполагаем, что все настройки одного пользователя хранятся в одной строке таблицы, каждая настройка – отдельный столбец этой таблицы

Связь один к одному – реализация

- Способы реализации:
 1. Сделать первичный ключ одной из таблиц внешним ключом на другую таблицу
 - В итоге у таблиц будут связаны записи с одинаковым значением первичного ключа
 2. Добавить внешний ключ в любую из таблиц.
Но при этом на него нужно будет навесить ограничение уникальности, иначе получится 1 ко многим
 3. Через вспомогательную таблицу.
Но нужно навесить ограничение уникальности на оба внешних ключа из этой таблицы, иначе будет многие ко многим

Нормальные формы

Пояснение к разделу

- Эта тема содержит довольно много теории и математических терминов
- Тестировщиков эту тему на собеседовании, как правило, не спрашивают, но знать это полезно для кругозора
- Разработчиков на эту тему могут спросить, но не обязательно знать все это сильно глубоко, достаточно общего понимания и объяснения своими словами
- Не обязательно заучивать все эти термины

Нормальные формы

- **Нормальная форма (НФ)** – это требование к структуре таблиц реляционной БД для устранения избыточных **функциональных зависимостей** между полями таблиц
- **Функциональная зависимость** – ситуация, когда некоторое поле (или совокупность полей) напрямую зависит от другого поля (или совокупности полей)
 - Более формально – если в разных строках у некоторого поля (или совокупности полей) задано некоторое одинаковое значение **X**, то у другого поля (или совокупности полей) в этих строках будет одинаковое значение **Y**
 - Например, если в некоторой таблице в поле страна указана «Россия», то поле столица всегда «Москва»

Цель нормализации

- Нормальные формы направлены на то, чтобы устранить избыточность данных - чтобы каждая единица данных хранилась всего 1 раз
- Это снижает вероятность ошибок в данных
- Если данные не дублируются, то не может быть такого, что вы изменили/удалили некоторую информацию в одном месте, а в других местах забыли
- Т.е. не будет расхождений и противоречий в данных

Нормальные формы

- Всего выделяют 6 НФ
- Каждая последующая НФ включает в себя предыдущие
- Часто используют 3 НФ
- Материалы про нормальные формы:
- <https://habr.com/ru/post/254773/>
- <https://site-do.ru/db/db5.php>

- Требования:
 - Все поля являются **атомарными**, т.е. неделимыми
 - Нет одинаковых строк в таблице
- С некоторой точки зрения реляционные СУБД сами по себе уже удовлетворяют 1 НФ – одинаковые строки запрещены, а поля хранят одно значение
- Но с логической точки зрения 1 НФ может нарушаться

1 НФ

- Допустим, у нас есть таблица людей в таком виде:

ID	ФИО	E-mail
1	Иван Иванов	ivan@mail.ru
2	Степан Петров	ivd@mail.ru

- А нам в программе нужно будет использовать фамилию и имя по отдельности, сейчас это будет неудобно – придется отделять фамилию от имени
- Т.е. нарушена **атомарность** (неделимость). Исправляем:

ID	Имя	Фамилия	E-mail
1	Иван	Иванов	ivan@mail.ru
2	Степан	Петров	ivd@mail.ru

- Требования:
 - Соблюдена 1 НФ
 - Каждое неключевое поле **неприводимо (функционально полно)** зависит от первичного ключа
- **Неприводимость** означает, что неключевое поле зависит от всего ключа в целом, а не только от его части
- Если у таблицы первичный ключ состоит из 1 поля, то это требование автоматически выполнено
- Поэтому рассмотрим что это означает для таблиц с **составным первичным ключом** (т.е. состоящим из нескольких полей)

2 НФ

- Допустим, у нас есть такая таблица автомобилей:

<u>Модель</u>	<u>Фирма</u>	Цена	Скидка
M5	BMW	5500000	5
M1	BMW	2500000	5
GT-R	Nissan	5000000	10

- Первичный ключ у нас состоит из полей **Модель** и **Фирма**
- Но по логике этой базы цена зависит от модели и фирмы, а скидка – только от фирмы
- Поэтому данная таблица нарушает 2 НФ – скидка зависит не от ключа в целом, а от его части – от фирмы

Приведение к 2 НФ

- Чтобы привести эту таблицу к НФ, ее придется разбить на 2 таблицы

<u>Модель</u>	<u>Фирма</u>	Цена
M5	BMW	5500000
M1	BMW	2500000
GT-R	Nissan	5000000

<u>Фирма</u>	Скидка
BMW	5
Nissan	10

- Требования:
 - Соблюдена 2 НФ
 - Каждое неключевое поле **нетранзитивно зависит** от первичного ключа
- Т.е. имеется в виду, что каждое неключевое поле зависит именно от первичного ключа, а не от какого-то другого неключевого поля
- На примере будет понятнее

3 НФ

- Допустим, у нас есть такая таблица магазинов и фирм:

<u>Фирма</u>	Магазин	Телефон
BMW	Риал-авто	12-11-11
Audi	Риал-авто	12-11-11
Nissan	Некст-авто	43-33-33

- Пусть первичный ключ у нас – это поле **Фирма**
- Считается, что магазин работает только с автомобилями одной фирмы
- Понятно, что телефон указан не для фирмы, а для магазина
- Т.е. логически телефон зависит от магазина, а не от фирмы, а это нарушение 3 НФ

Приведение к 3 НФ

- Для исправления нужно вынести магазины в отдельную таблицу:

<u>Фирма</u>	Магазин
BMW	Риал-авто
Audi	Риал-авто
Nissan	Некст-авто

<u>Магазин</u>	Телефон
Риал-авто	12-11-11
Некст-авто	43-33-33

Краткий итог

- НФ нужны, чтобы избавиться от дублирования данных и позволяют избежать ошибок с данными
- Обычно используют 3 НФ
- Общая идея **нормализации**:
 - Максимально разбивать каждый столбец на отдельные столбцы, если это возможно (например, **ФИО** на столбцы **Фамилия**, **Имя** и **Отчество**)
 - Стараться делать первичные ключи из 1 поля (тогда автоматически выполняется 2 НФ)
 - Максимально разбивать все сущности на отдельные таблицы, чтобы каждая единица данных хранилась всего в 1 месте и не дублировалась

Материалы и домашняя работа

Материалы для изучения SQL

- **Хороший курс по основам БД:**

- <http://site-do.ru/db/db.php>

- **Задачи:**

- <http://www.sql-ex.ru/>

- **Еще курс:**

- <http://www.sql-tutorial.ru/>

Домашнее задание

- Установить MySQL
- Посмотреть тестовые базы данных через Workbench
- Запомнить пройденные понятия
- Прочитать «Основы баз данных» - уроки 1-5:
<http://site-do.ru/db/db.php>