**[Нормализация данных](https://www.youtube.com/watch?v=zqQxWdTpSIA&ab_channel=ListenIT) –** процессорганизации структуры хранения данных, который позволяет привести БД к минимальной избыточности (повторяемые данные). Избыточность, как правило, устраняется за счет декомпозиции отношений (разбиение одной таблицы на несколько). Направлена на устранение аномалий путем устранения избыточного дублирования и повышение производительности.

Избыточность данных создает предпосылки для возникновения аномалий (некоторые примеры):

1) Дублирующая информация – ненормализованная таблица может содержать множество одинаковых значений одного типа, что увеличивает объем данных для хранения

2) Изменение значения типа - Если значение типа изменится, то нужно изменить все строки, содержащие это значение

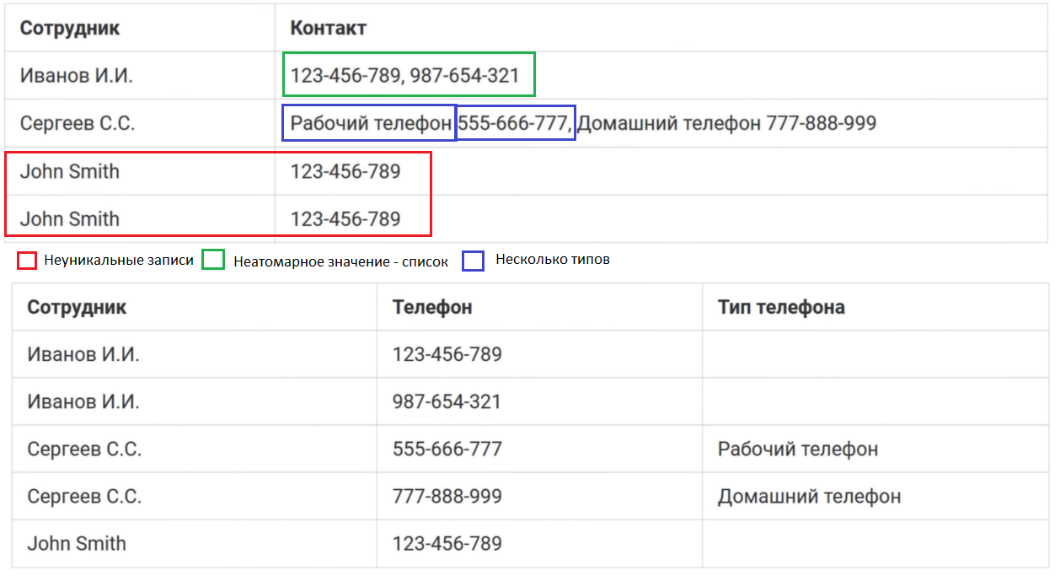
3) Потеря данных - При заполнении таблицы данными типа он может быть записан неверно, что приведет к невозможности включения этих данных в нужный тип.

4) Потеря типа - Информация о типах хранится в таблице представляющей сущности. Если тип представлен одной сущностью в таблице, то удаление этой сущности приведет к потере типа.

***Нормальные формы*** *–* Набор правил и критериев, которым должна отвечать БД для соответствия той или иной форме БД. Каждая последующая форма подразумевает, что БД должна соответствовать всем предыдущим нормальным формам.

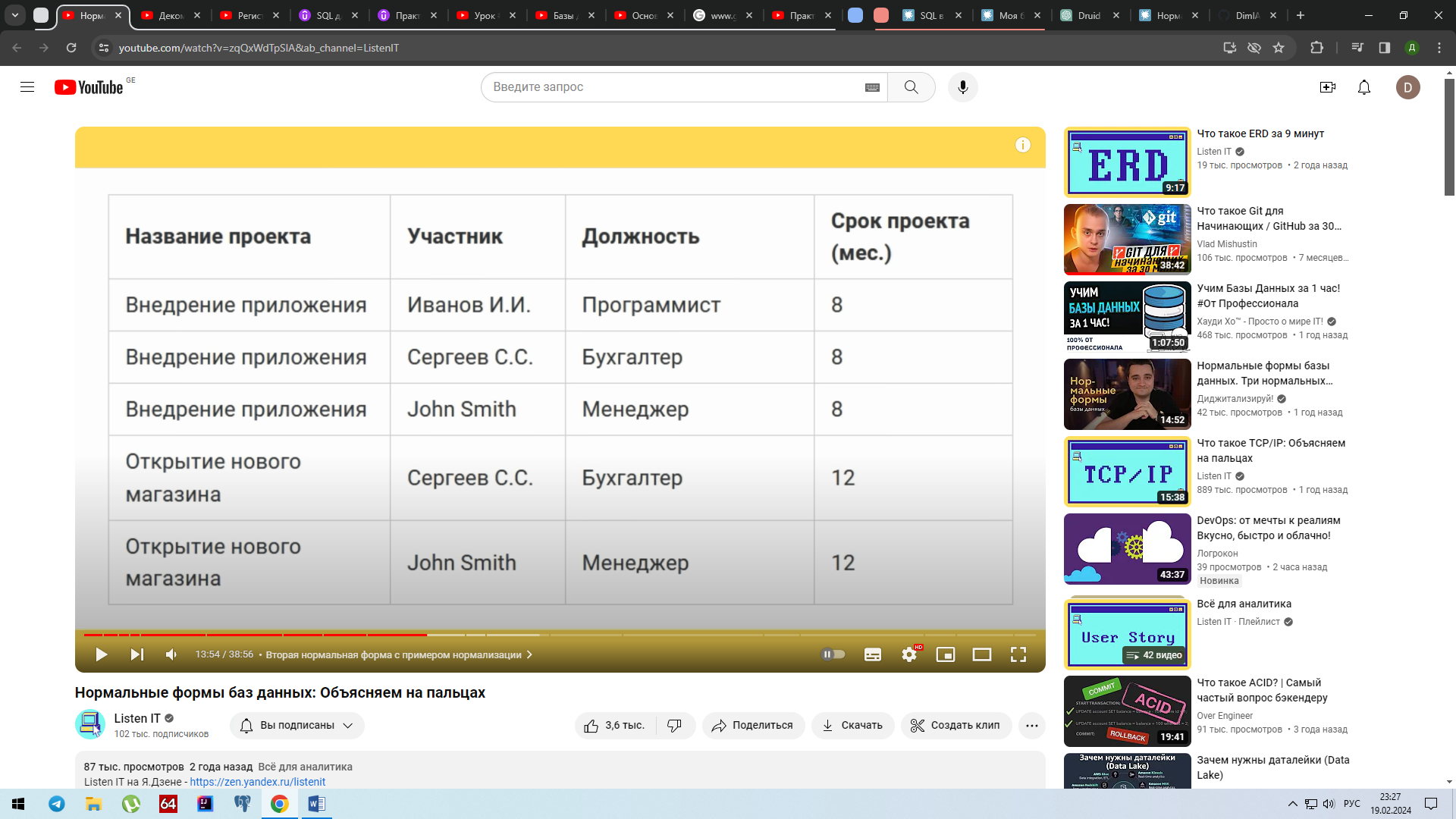
0НФ – Приведение БД к виду, соответствующему базовым принципам реляционной теории. Строки в таблицах не должны быть пронумерованы (порядок строк и столбцов не имеет значения).

*1НФ –* Строка – уникальна (нет полностью идентичных строк), ячейка - атомарна (хранит одно неделимое по смыслу значение - не должно быть массивов и списков значений), столбец – хранит данные только одного типа.



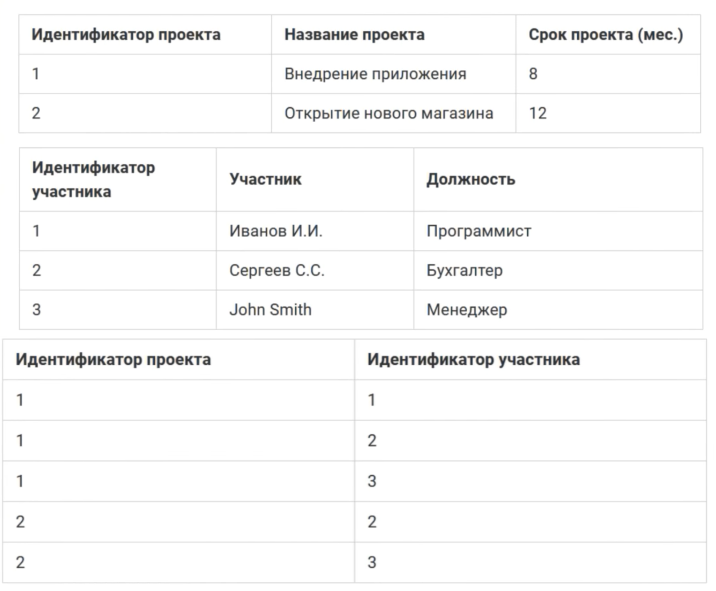
*2НФ* – Таблица должна иметь первичный ключ, для возможности однозначно идентифицировать каждый кортеж (ключ может быть простой или составной). В случае, если необходим составной ключ, то все неключевые атрибуты таблицы должны зависеть от полного ключа, т.е. невозможно однозначно определить запись в таблице, зная только часть ключа.

Если таблица выражена отношением многие ко многим, то для однозначной идентификации записи потребуется использование составного ключа. То есть, имея только название проекта, невозможно однозначно определить строку. Поэтому нужно выбрать несколько атрибутов, которые будут обеспечивать свойство уникальности (название проекта, участник).



Для обеспечения второго свойства, необходимо проверить, можно ли получить должность или срок проекта, зная только часть ключа. В данном случае, зная только название можно определить срок, а зная участника – определить должность – второе свойство не выполняется. Значит нужно прибегнуть к декомпозиции.

Объединить связанные типы в сущности – участник и должность, проект и срок. И присвоить им простые первичные ключи, так же создать третью отношение, отражающее связь проектов и участников.

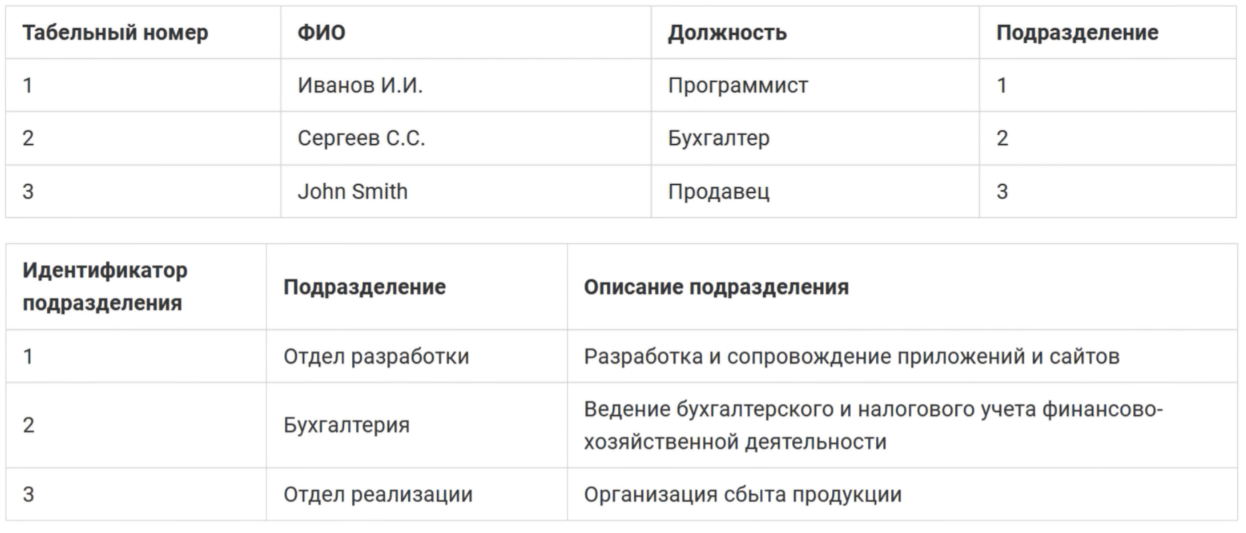


*3НФ* – В таблице должна отсутствовать транзитивная (непрямая) зависимость. То есть неключевые атрибуты напрямую зависят только от первичного ключа, но не от других атрибутов. Неключевые столбцы не должны играть роль ключа.

Описание подразделения относится к подразделению, а не к первичному ключу, то есть подразделение (неключевой атрибут) может играть роль первичного ключа.

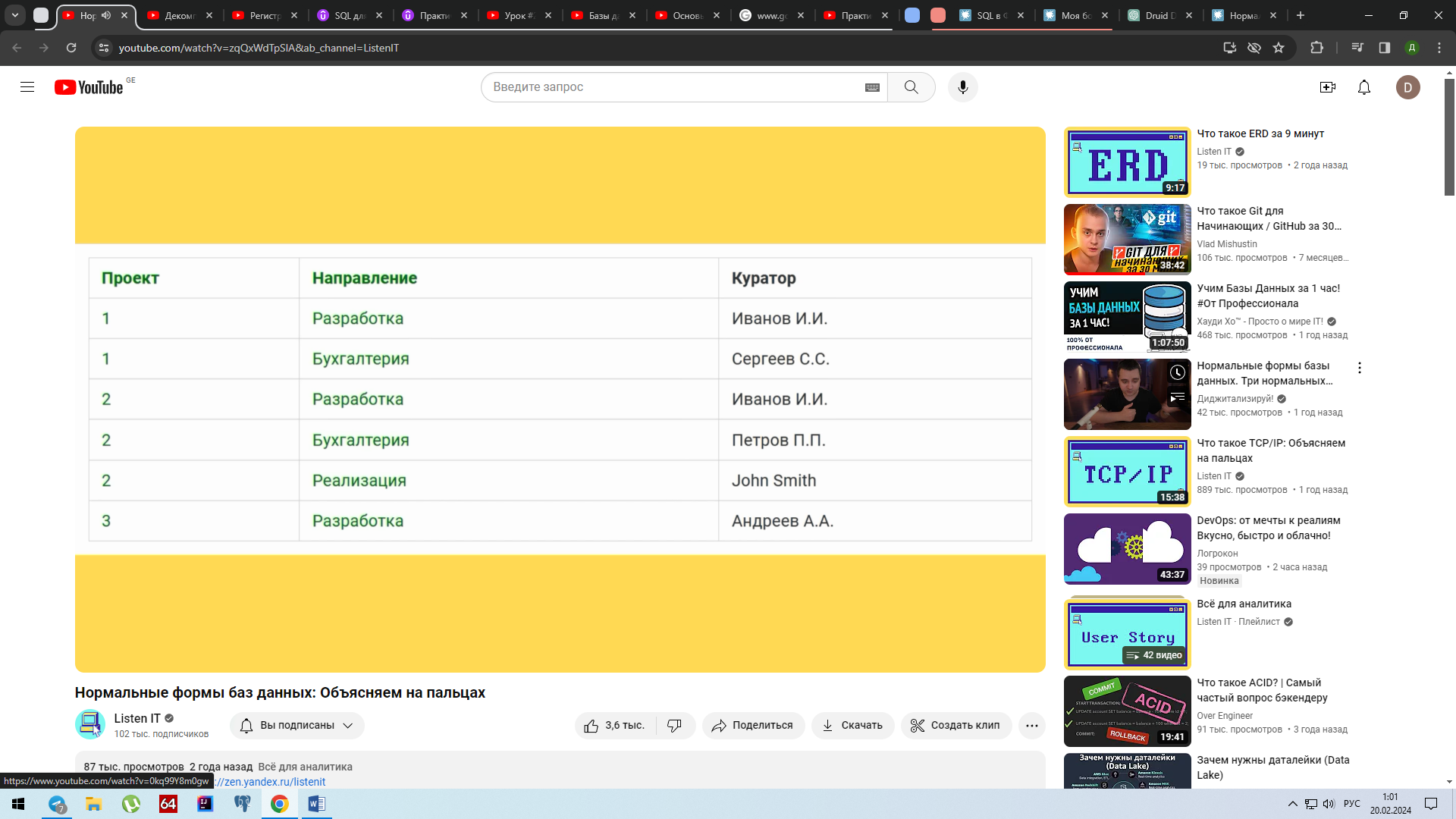


Для приведения к 3НФ – декомпозиция и создание связи через внешние ключи:



Нормальная форма Бойса-Кодда (усиленная 3НФ) – Ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов. То есть данная форма актуальна только для отношений, имеющих составной первичный ключ.

Выбрав неключевой атрибут (куратора) можно точно определить часть составного ключевого атрибута (направление)



Решается также декомпозицией



Все последующие формы нормализации практически не встречаются.

*4НФ* – в таблице не должно быть многозначных зависимостей. Наличие таких зависимостей приводит невозможности редактирования одной зависимости без вмешательства в другую

Пример многозначной зависимости – Есть аргументы A, B, C. B и C не связаны между собой, но связаны по отдельности зависят от A. То есть для каждого A есть множество значений B и множество значений C.

Таблица с расписанием курсов – Преподаватель и Аудитория между собой независимы, но по отдельности каждый из них зависит от типа Курс.



Удаление преподавателя Иванов приведет к потере типа Аудитория – 101 и 203;

Так же при составлении расписания аудитория может быть известна, а преподаватель не определен, в таком случае тип преподаватель будет равен null или default, что так же является аномалией.



Список хобби студентов, которые посещают курс SQL выводит неверный результат, так как Иванов посещает курс SQL и имеет хобби Футбол и Хоккей, чего нет в результирующей выборке.

*5НФ* – Таблица соответствует 5НФ, если возможна ее декомпозиция без потерь - разбиение таблицы на несколько и возможность их последующего соединения в точно такую же таблицу (без потерь строк и возникновения новых).

Таблица будет находиться в 5НФ, если она не содержит зависимостей соединения – существование связи между двумя атрибутами, через третий атрибут, который не является ключом

Невозможно точно определить, находится ли таблица в 5НФ, точно не зная предметной области и всех возможных зависимостей.

*Так же есть доменно-ключевая и 6 нормальные формы.*

Минусы сильной нормализации – теряется понятность данных в таблице и нужно писать более сложные запросы к таблице, снижение производительности СУБД – до 4НФ производительность растет, если использовать 4НФ и последующие, то у такой БД будет проседать производительность, но при этом уменьшаться кол-во аномалий.

Исходя из этого, нужно искать баланс между удобством работы, ее производительностью и допустимым кол-вом аномалий.

*Денормализация* - пренебрежение формами нормализации для увеличения производительности и упрощения написания запросов.