Министерство науки и высшего образования Российской Федерации

Университетский колледж
федерального государственного бюджетного образовательного учреждения
высшего образования
«Оренбургский государственный университет»

Отделение информационных технологий

# БАЗЫ ДАННЫХ:

# Теория нормализации

Методические указания

Составители: Н.А. Кривошеева, М.Г. Таспаева

Рекомендовано к изданию редакционно-издательским советом федерального государственного бюджетного образовательного учреждения высшего образования «Оренбургский государственный университет» для обучающихся по программам среднего профессионального образования по специальностям 09.02.01 Компьютерные системы и комплексы, 10.02.05 Обеспечение информационной безопасности автоматизированных систем

Оренбург
2021

УДК 004(075.32)
ББК 32.97я723
Б17

Рецензент – доцент кафедры программного обеспечения вычислительной техники и автоматизированных систем И.А. Щудро

Б17 **Базы данных: теория нормализации**: методические указания/ составители Н.А. Кривошеева, М.Г. Таспаева; Оренбургский гос. ун-т. – Оренбург: ОГУ, 2021. – 48 с.

Методические указания предназначены для изучения теории нормализации реляционных баз данных при изучении дисциплины «Базы данных» в Университетском колледже ОГУ для обучающихся второго курса специальностям 09.02.01 Компьютерные системы и комплексы, 10.02.05 Обеспечение информационной безопасности автоматизированных систем.

Методические указания составлены с учетом Федерального государственного образовательного стандарта среднего профессионального образования.

УДК 004(075.32)
ББК 32.97я723

© Кривошеева Н.А.,
Таспаева М.Г., составление, 2021
© ОГУ, 2021

# Содержание

# Введение

Нормализация – процесс организации данных в базе данных, включающий создание таблиц и установку отношений между этими таблицами в соответствии с правилами, предназначенными для защиты данных и обеспечения большей гибкости базы данных за счет исключения избыточности и несогласованности зависимости. Избыточные данные занимают место на диске и создают проблемы с обслуживанием. Существуют некоторые правила нормализации баз данных, которые позволяют решить обозначенные проблемы.

В методических указаниях описывается терминология нормализации баз данных. Основные сведения, полученные обучающимися в процессе изучения данной темы, будут полезны при обсуждении структуры реляционной базы данных и могут быть использованы при изучение профессиональных дисциплин, связанных с разработкой и применением баз данных.

В методических указаниях приведены теоретические сведения по нормализации базы данных, рассмотрены примеры практической реализации данного материала в понятной и удобной для изучения форме, рассмотрены вопросы целостности данных.

Методические указания могут быть использованы при преподавании дисциплин, связанных с информационными системами и технологиями.

# 1 Реляционная база данных

В целом под базой данных можно понимать любой набор информации, которую можно найти в этой базе данных и воспользоваться ей, однако если говорить в контексте SQL, то речь будет идти, конечно, о реляционных базах данных.

**Реляционная база данных** – это упорядоченная информация, связанная между собой определёнными отношениями [1].

Логически такая база данных представлена в виде таблиц, в которых и лежит вся эта информация. В реляционных базах данных есть такое понятия, как «Нормализация».

**Нормализация** – это процесс удаления избыточных данных.

Также нормализацию можно рассматривать и с позиции проектирования базы данных, в таком случае мы можем сформулировать определение нормализации следующим образом.

**Нормализация** – это метод проектирования базы данных, который позволяет привести базу данных к минимальной избыточности.

Избыточность устраняется, как правило, за счёт декомпозиции отношений (таблиц), то есть разбиения одной таблицы на несколько.

Может возникнуть вопрос: «Зачем вообще нормализовать базу данных и бороться с этой избыточностью?».

Дело в том, что избыточность данных создает предпосылки для появления различных аномалий, снижает производительность, и делает управление данными не гибким и не очень удобным. Отсюда можно сделать вывод, что нормализация нужна для:

- устранения аномалий;
- повышения производительности;
- повышения удобства управления данными.

**Избыточность данных** появляется когда одни и те же данные хранятся в базе в нескольких местах, именно это и приводит к аномалиям.

Так как в этом случае необходимо добавлять, изменять или удалять одни и те же данные в нескольких местах. Например, если не выполнить операцию в каком-нибудь одном месте, то возникает ситуация, когда одни данные не соответствуют вроде как точно таким же данным в другом месте [2].

Рассмотрим пример: допустим, у нас есть таблица 1, которая хранит информацию о предметах мебели, в частности наименование предмета и материал, из которого изготовлен этот предмет [3].

Таблица 1 – Информация о предметах мебели

| Идентификатор предмета | Наименование предмета | Материал |
|---|---|---|
| 1 | Стул | Металл |
| 2 | Стол | Массив дерева |
| 3 | Кровать | ЛДСП |
| 4 | Шкаф | Массив дерева |
| 5 | Комод | ЛДСП |

Теперь допустим, что у нас возникла необходимость подкорректировать название материала, вместо «Массив дерева» нужно написать «Натуральное дерево», и чтобы это сделать необходимо будет внести изменения сразу в несколько строк, так как предметов, изготовленных из массива дерева, несколько: стол и шкаф.

Далее представим, что по каким-то причинам мы внесли изменения только в одну строку, в итоге в нашей таблице 2 будет и «Массив дерева», и «Натуральное дерево».

Таблица 2 – Добавление материала «Натуральное дерево»

| Идентификатор предмета | Наименование предмета | Материал |
|---|---|---|
| 1 | Стул | Металл |
| 2 | Стол | **Натуральное дерево** |
| 3 | Кровать | ЛДСП |
| 4 | Шкаф | **Массив дерева** |
| 5 | Комод | ЛДСП |

Как теперь определить какое из двух названий в таблице 2 будет правильным? Если также представить, что мы можем внести еще какое-то новое значение при добавлении новых записей, например, просто «Дерево», то в этом случае в таблице 3 будет и «Массив дерева», и «Натуральное дерево», и просто «Дерево», и вообще, что угодно, ведь это просто текст.

Таблица 3 – Добавление материала «Дерево»

| Идентификатор предмета | Наименование предмета | Материал |
|---|---|---|
| 1 | Стул | Металл |
| 2 | Стол | **Натуральное дерево** |
| 3 | Кровать | ЛДСП |
| 4 | Шкаф | **Массив дерева** |
| 5 | Комод | ЛДСП |
| 6 | Тумба | **Дерево** |

Однако по своей сути это один и тот же материал, мы просто решили или подкорректировать его название, или ошиблись при добавлении новой записи. Это и есть аномалия, когда одни данные в одном месте не соответствуют вроде

как точно таким же данным в другом месте. Это всего лишь один вид аномалии, однако, в процессе добавления, изменения и удаления данных может возникать много других противоречивых ситуаций, т.е. аномалий.

При этом, обязательно стоит отметить, что в наших таблицах всего пять записей, а теперь представьте, что их миллион!

Именно поэтому мы должны устранять избыточность данных в базе, то есть проводить так называемую нормализацию базы данных.

В данном конкретном случае мы должны название материала, из которого изготовлены предметы мебели, вынести в отдельную таблицу 4, а в таблице 5 с предметами сделать всего лишь ссылку на нужный материал, тем самым, соотнеся эту ссылку с исходной записью, мы будем понимать, из какого материала сделан тот или иной предмет.

Таблица 4 - Материалы, из которых изготовлены предметы мебели

| Идентификатор материала | Материал |
|---|---|
| 1 | Массив дерева |
| 2 | Металл |
| 3 | ЛДСП |

Таблица 5 - Предметы мебели с идентификатором материала, из которых изготовлены

| Идентификатор предмета | Наименование предмета | Идентификатор материала |
|---|---|---|
| 1 | Стул | 2 |
| 2 | Стол | 1 |
| 3 | Кровать | 3 |
| 4 | Шкаф | 1 |
| 5 | Комод | 3 |

Теперь когда нам потребуется изменить название материала, мы будем вносить изменение только в одном месте, то есть править только одну строку в таблице 4, а в таблице 5 они будут меняться автоматически при правильной настройке связей между соответствующими таблицами.

Таким образом, представляя материалы в виде отдельной сущности и создавая для нее отдельную таблицу, мы устраняем описанную выше аномалию.

Другими словами, каждая сущность должна храниться отдельно, а в случае необходимости использования этой сущности в другой таблице на нее делается всего лишь ссылка, то есть выстраивается связь.

## 2 Нормальные формы базы данных

В целом процесс нормализации базы данных выглядит следующим образом: мы, следуя определённым правилам и соблюдая определенные требования, проектируем таблицы в базе данных.

При этом все эти правила и требования можно сгруппировать в несколько наборов, и если спроектировать базу данных с соблюдением всех правил и требований, которые включаются в тот или иной набор, то база данных будет находиться в определённом состоянии, то есть форме, и такая форма называется нормальная форма базы данных.

Иными словами, следуя определённым правилам и соблюдая определенные требования, мы приводим базу данных к определенной нормальной форме.

**Нормальная форма базы данных** – это набор правил и критериев, которым должна отвечать база данных.

Каждая следующая нормальная форма содержит более строгие правила и критерии, тем самым приводя базу данных к определённой нормальной форме, мы устраняем определённый набор аномалий.

Отсюда можно сделать вывод, что чем выше нормальная форма, тем меньше аномалий в базе будет.

**Процесс нормализации** – это последовательный процесс приведения базы данных к эталонному виду, то есть переход от одной нормальной формы к следующей.

Иными словами, процесс перехода от одной нормальной формы к следующей – это усовершенствование базы данных. Так как если база данных находится в какой-то определённой нормальной форме – это означает, что в базе данных отсутствует определенный вид аномалий.

Существует пять основных нормальных форм базы данных:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);
- четвертая нормальная форма (4NF);
- пятая нормальная форма (5NF).

Однако выделяют еще дополнительные нормальные формы:

- ненормализованная форма или нулевая нормальная форма (UNF)
- нормальная форма Бойса-Кодда (BCNF);
- доменно-ключевая нормальная форма (DKNF);
- шестая нормальная форма (6NF).

Если объединить оба этих списка и упорядочить нормальные формы от менее нормализованной до самой нормализованной, то есть начиная с формы, при которой база данных по своей сути не является нормализованной, и заканчивая самой строгой нормальной формой, то мы получим следующий перечень:

1) ненормализованная форма или нулевая нормальная форма (UNF)
2) первая нормальная форма (1NF);
3) вторая нормальная форма (2NF);
4) третья нормальная форма (3NF);
5) нормальная форма Бойса-Кодда (BCNF);
6) четвертая нормальная форма (4NF);

7) пятая нормальная форма (5NF);
8) доменно-ключевая нормальная форма (DKNF);
9) шестая нормальная форма (6NF).

База данных считается нормализованной, если она находится как минимум в третьей нормальной форме (3NF). В реальном мире нормализация до третьей нормальной формы (3NF) является обычной, стандартной практикой, так как 3NF устраняет достаточное количество аномалий, при этом производительность базы данных, а также удобство ее использования не снижается, что нельзя сказать о всех последующих формах.

Ситуации, при которых требуется нормализовать базу данных до четвертой нормальной формы (4NF), в реальном мире встречаются достаточно редко.

Если говорить о всех последующих нормальных формах (5NF, DKNF, 6NF), то в реальной жизни трудно даже представить ситуации, при которых потребуется нормализовать базу данных до этих форм.

Иными словами, 5NF, DKNF, 6NF – это в большей степени теоретические нормальные формы, немного отстраненные от реального мира.

Стоит отметить, что приведение базы данных к какой-то конкретной нормальной форме, обязательно требует, чтобы эта база данных уже находилась в предыдущей нормальной форме. Другими словами, если Вы хотите нормализовать базу данных до третьей нормальной формы, то база уже должна находиться во второй нормальной форме, т.е. нельзя нормализовать базу данных до третьей формы, если она еще не нормализована до второй [4].

# 3 Описание нормальных форм базы данных

## 3.1 Ненормализованная форма или нулевая нормальная форма (UNF)

Обычно данную форму как отдельную нормальную форму базы данных не выделяют, но мы ее рассмотрим, так как это поможет понять, какие таблицы являются реляционными, а какие нет.

Итак, как Вы знаете, существует несколько так называемых нормальных форм базы данных и процесс нормализации заключается в том, что мы последовательно приводим базу данных к определенной нормальной формы, то есть переходим от одной нормальной формы к следующей.

Однако перед тем как переходить к процессу нормализации и приведению базы данных к определённой нормальной форме, необходимо привести базу данных к табличному виду, но так, чтобы он отвечал базовым принципам реляционной теории, так как мы говорим о реляционных базах данных, и только после этого задумываться о процессе нормализации.

По реляционной теории строки в таблицах не должны быть пронумерованы, то есть порядок строк не имеет значения, так же как не имеет значения порядок столбцов. Например, если мы поменяем порядок столбцов, или порядок строк, ничего измениться не должно, это не должно ни на что повлиять. Таким образом, по реляционной теории мы не можем обратиться к определённой строке или столбцу по ее номеру.

И если Ваши таблицы соблюдают эти принципы, то можно переходить к нормализации базы данных. Рассмотрим небольшой пример [3]. Достаточно часто в Excel можно встретить таблицы следующего вида (таблица 6).

Таблица 6 – Пример оформления таблиц в Excel

| Порядковый номер строки | A | B |
|---|---|---|
| 1 | Иван | Иванов |
| 2 | Сергей | Сергеев |
| 3 | John | Smith |
| 4 | Иван | Иванов |

Однако, к сожалению, подобные таблицы нельзя назвать реляционными, так как это пронумерованные двумерные массивы данных. И если мы поменяем местами строки, то наша нумерация просто нарушится.

По реляционной теории данные в таблицах никак не упорядочены, и мы не можем сказать, что нам нужно получить данные из строки с порядковым номером 2 или из второго столбца.

Поэтому чтобы приступить к нормализации нашей таблицы, нам необходимо как минимум удалить столбец с порядковым номером и не учитывать порядок столбцов, например, задав им более корректные имена (таблица 7).

Таблица 7 – Приведение таблицы к реляционной форме

| first_name | last_name |
|---|---|
| Иван | Иванов |
| Сергей | Сергеев |
| John | Smith |
| Иван | Иванов |

Теперь можно сказать, что эта таблица соблюдает базовые принципы реляционной теории и мы можем начинать приводить ее к той или иной нормальной форме, например, к первой нормальной форме.

Что такое ненормализованная форма или нулевая нормальная форма (UNF), мы рассмотрели и можем переходить к рассмотрению нормальных форм базы данных.

### 3.2 Первая нормальная форма (1NF)

Требование первой нормальной формы (1NF) очень простое и оно заключается в том, чтобы таблицы соответствовали реляционной модели данных и соблюдали определённые реляционные принципы.

Таким образом, чтобы база данных находилась в 1NF, необходимо чтобы ее таблицы соблюдали следующие реляционные принципы:

- в таблице не должно быть дублирующих строк;
- в каждой ячейке таблицы хранится атомарное значение (одно не составное значение);
- в столбце хранятся данные одного типа;
- отсутствуют массивы и списки в любом виде.

Рассмотрим пример приведения таблицы к первой нормальной форме.

Таблица 8 не находится даже в первой нормальной форме, так как у нас есть дублирующие строки, а в некоторых ячейках хранятся списки значений (каждый номер телефона — это одно значение).

Таблица 8 - Список сотрудников в ненормализованном виде

| Сотрудник | Контакт |
|---|---|
| 1 | 2 |
| Иванов И.И. | 123-456-789, 987-654-321 |
| Сергеев С.С. | Рабочий телефон 555-666-777,<br>Домашний телефон 777-888-999 |

Продолжение таблицы 8

| 1 | 2 |
|---|---|
| John Smith | 123-456-789 |
| John Smith | 123-456-789 |

Чтобы привести эту таблицу к первой нормальной форме, необходимо удалить дублирующие строки, в ячейках хранить один номер телефона, а не список, а тип телефона (домашний или рабочий) вынести в отдельный столбец, так как столбцы хранят структурную информацию (таблица 9).

Таблица 9 – Список сотрудников в первой нормальной форме

| Сотрудник | Телефон | Тип телефона |
|---|---|---|
| Иванов И.И. | 123-456-789 | |
| Иванов И.И. | 987-654-321 | |
| Сергеев С.С. | 555-666-777 | Рабочий телефон |
| Сергеев С.С. | 777-888-999 | Домашний телефон |
| John Smith | 123-456-789 | |

Таким образом, главное правило первой нормальной формы звучит следующим образом - строки, столбцы и ячейки в таблицах необходимо использовать строго по назначению:

- назначение строк – хранить данные;
- назначение столбцов – хранить структурную информацию;
- назначение ячеек – хранить атомарное значение.

Если ячейка таблицы по реляционной теории должна хранить одно атомарное значение, не нужно записывать в ячейку какой-то список значений или составное значение. Также не нужно создавать строки, которые уже есть в таблице и хранить в столбце значения разных типов данных.

На основе всего вышеизложенного можно сделать следующий вывод:

- если таблица создана с соблюдением всех реляционных принципов, значит, она уже находится в первой нормальной форме, таким образом, по сути абсолютно все реляционные таблицы находятся в первой нормальной форме;

- если таблица создана без учета реляционных принципов, значит эта таблица не является реляционной.

После того как мы привели таблицы базы данных к первой нормальной форме, мы можем переходить к приведению таблиц до второй нормальной формы (2NF) [5].

### 3.3 Вторая нормальная форма (2NF)

Перед тем как переходить к процессу приведения таблиц базы данных до второй нормальной формы, необходимо чтобы эти таблицы уже находились в первой нормальной форме, подробно процесс приведения таблиц базы данных до первой нормальной формы, а также все требования, предъявляемые к первой нормальной форме были рассмотрены в п.3.2. После того как таблицы базы данных находятся в первой нормальной форме, мы можем начинать приводить базу данных ко второй нормальной форме и рассматривать соответствующие требования.

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

- таблица должна находиться в первой нормальной форме;
- таблица должна иметь ключ;
- все неключевые столбцы таблицы должны зависеть от полного ключа (в случае если он составной);

**Ключ** – это столбец или набор столбцов, по которым гарантировано можно отличить строки друг от друга, то есть ключ идентифицирует каждую

строку таблицы. По ключу мы можем обратиться к конкретной строке данных в таблице.

Если ключ составной (состоит из нескольких столбцов), то все остальные неключевые столбцы должны зависеть от всего ключа (от всех столбцов в этом ключе). Если какой-то атрибут (столбец) зависит только от одного столбца в ключе, значит, база данных не находится во второй нормальной форме.

Иными словами, в таблице не должно быть данных, которые можно получить, зная только половину ключа, то есть только один столбец из составного ключа.

Главное правило второй нормальной формы (2NF) звучит следующим образом: таблица должна иметь правильный ключ, по которому можно идентифицировать каждую строку.

Рассмотрим пример приведения таблицы ко второй нормальной форме [3].

Представим, что нам нужно хранить список сотрудников организации, и для этого мы создали таблицу 10.

Таблица 10 – Список сотрудников в первой нормальной форме

| ФИО | Должность | Подразделение | Описание подразделения |
|---|---|---|---|
| Иванов И.И. | Программист | Отдел разработки | Разработка и сопровождение приложений и сайтов |
| Сергеев С.С. | Бухгалтер | Бухгалтерия | Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности |
| John Smith | Продавец | Отдел реализации | Организация сбыта продукции |

Мы видим, что таблица 10 удовлетворяет условиям первой нормальной формы, так как в ней нет дублирующих строк и все значения атомарны.

Теперь мы можем начать процесс нормализации этой таблицы до второй нормальной формы. Что для этого нам нужно сделать? Нам нужно внедрить первичный ключ, по которому мы сможем распознать каждую запись однозначно.

Поработав немного с предметной областью, мы выясняем, что в этой организации каждому сотруднику присваивается уникальный табельный номер, который никогда не будет изменен.

Поэтому очевидно, что для таблицы, которая будет хранить список сотрудников, первичным ключом может выступать табельный номер, зная который мы можем четко идентифицировать каждого сотрудника, т.е. каждую строку нашей таблицы. Если бы такого табельного номера у нас не было или в рамках организации он мог повторяться (например, сотрудник уволился, и спустя время его номер присвоили новому сотруднику), то для первичного ключа мы могли бы создать искусственный ключ с целочисленным типом данных, который автоматически увеличивался бы в случае добавления новых записей в таблицу. Тем самым мы бы точно также четко идентифицировали каждую строку в таблице.

Таким образом, чтобы привести эту таблицу ко второй нормальной форме, мы должны добавить в нее еще один атрибут, в нашем случае это столбец с табельным номером (таблица 11).

В результате, так как наш первичный ключ является простым, а не составным, наша таблица автоматически переходит во вторую нормальную форму.

Иными словами, если первичный ключ простой ( состоящий из одного столбца), второе требование, которое предъявляется к таблицам для перехода во вторую нормальную форму, выполнять не требуется, так как оно относится только к таблицам, у которых первичный ключ составной.

Таблица 11 - Список сотрудников во второй нормальной форме с простым первичным ключом

| Табельный номер | ФИО | Должность | Подразделение | Описание подразделения |
|---|---|---|---|---|
| 1 | Иванов И.И. | Программист | Отдел разработки | Разработка и сопровождение приложений и сайтов |
| 2 | Сергеев С.С. | Бухгалтер | Бухгалтерия | Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности |
| 3 | John Smith | Продавец | Отдел реализации | Организация сбыта продукции |

А теперь давайте рассмотрим другую ситуацию, в которой первичный ключ у нас будет составным.

Представим, что наша организация выполняет несколько проектов, в которых может быть задействовано несколько участников, и нам необходимо хранить информацию об этих проектах. В частности мы хотим знать, кто участвует в каждом из проектов, продолжительность этого проекта, ну и возможно какие-то другие сведения. При этом мы понимаем, что отдельно взятый сотрудник может участвовать в нескольких проектах.

Для хранения таких данных мы создали таблицу 12.

Таблица 12 – Список проектов организации в первой нормальной форме

| Название проекта | Участник | Должность | Срок проекта (мес.) |
|---|---|---|---|
| Внедрение приложения | Иванов И.И. | Программист | 8 |
| Внедрение приложения | Сергеев С.С. | Бухгалтер | 8 |
| Внедрение приложения | John Smith | Менеджер | 8 |
| Открытие нового магазина | Сергеев С.С. | Бухгалтер | 12 |
| Открытие нового магазина | John Smith | Менеджер | 12 |

Как видим, таблица 12 находится в первой нормальной форме, значит, мы можем пытаться приводить ее ко второй нормальной форме. Вспомним, что для того, чтобы привести таблицу ко второй нормальной форме, необходимо определить для нее первичный ключ.

Посмотрев на таблицу 12, мы понимаем, что четко идентифицировать каждую строку мы можем только с помощью комбинации столбцов, например, «Название проекта» + «Участник», иными словами, зная «Название проекта» и «Участника», мы можем четко определить конкретную запись в таблице, то есть каждое сочетание значений этих столбцов является уникальным.

Таким образом, мы определили первичный ключ и он у нас составной, (состоящий их двух столбцов) (таблица 13).

Таблица 13 - Список проектов организации с внедренным составным первичным ключом

| Название проекта | Участник | Должность | Срок проекта (мес.) |
|---|---|---|---|
| Внедрение приложения | Иванов И.И. | Программист | 8 |
| Внедрение приложения | Сергеев С.С. | Бухгалтер | 8 |
| Внедрение приложения | John Smith | Менеджер | 8 |
| Открытие нового магазина | Сергеев С.С. | Бухгалтер | 12 |
| Открытие нового магазина | John Smith | Менеджер | 12 |

Так как первичный ключ составной, нам необходимо проверить еще и второе требование, которое гласит, что «Все неключевые столбцы таблицы должны зависеть от полного ключа».

Другими словами, остальные столбцы, которые не входят в первичный ключ, должны зависеть от всего первичного ключа, то есть от всех столбцов, а не от какого-то одного. Чтобы это проверить, мы можем задать себе несколько вопросов.

Можем ли мы определить «Должность», зная только название проекта? Нет. Для этого нам необходимо знать и участника. Значит, пока все хорошо, по этой части ключа мы не можем четко определить значение неключевого столбца. Идем дальше и проверяем другую часть ключа.

Можем ли мы определить «Должность» зная только участника? Да, можем. Значит наш первичный ключ плохой, и требование второй нормальной формы не выполняется. Что делать в этом случае?

В этом случае мы будем выполнять действие, которое выполняется, наверное, в 99 % случаев на протяжении всего процесса нормализации базы данных – это декомпозиция.

**Декомпозиция** – это процесс разбиения одного отношения (таблицы) на несколько.

Чтобы декомпозировать нашу таблицу и привести базу данных к нормализованной форме, мы должны создать таблицы 14, 15, 16.

Таблица 14 - Проекты

| Идентификатор проекта | Название проекта | Срок проекта (мес.) |
|---|---|---|
| 1 | Внедрение приложения | 8 |
| 2 | Открытие нового магазина | 12 |

Таблица 15 - Участники

| Идентификатор участника | Участник | Должность |
|---|---|---|
| 1 | Иванов И.И. | Программист |
| 2 | Сергеев С.С. | Бухгалтер |
| 3 | John Smith | Менеджер |

Таблица 16 - Связь проектов и участников этих проектов

| Идентификатор проекта | Идентификатор участника |
|---|---|
| *1* | *2* |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |

Продолжение таблицы 16

| *1* | *2* |
|---|---|
| 2 | 2 |
| 2 | 3 |

На данном этапе мы создали три таблицы:

1) таблица «Проекты» в которую мы добавили искусственный первичный ключ;

2) таблица «Участники» в которую мы также добавили искусственный первичный ключ;

3) таблица «Связь между проектами и участниками» нужна для реализации связи «Многие ко многим», так как между этими таблицами связь именно такая.

После того как мы привели таблицы базы данных ко второй нормальной форме, мы можем переходить к приведению таблиц до третьей нормальной формы (3NF).

### 3.4 Третья нормальная форма (3NF)

Перед тем как переходить к процессу приведения таблиц базы данных до третьей нормальной формы, необходимо чтобы эти таблицы уже находились во второй нормальной форме, подробно процесс приведения таблиц базы данных до второй нормальной формы, а также все требования, предъявляемые ко второй нормальной форме, мы рассмотрели в п. 3.3. После того как таблицы базы данных находятся во второй нормальной форме, мы можем начинать приводить базу данных к третьей нормальной форме и рассматривать соответствующие требования.

Требование третьей нормальной формы (3NF) заключается в том, чтобы в таблицах отсутствовала транзитивная зависимость.

**Транзитивная зависимость** – это когда неключевые столбцы зависят от значений других неключевых столбцов.

Если в первой нормальной форме наше внимание было нацелено на соблюдение реляционных принципов, во второй нормальной форме в центре нашего внимания был первичный ключ, то в третьей нормальной форме все наше внимание уделено столбцам, которые не являются первичным ключом, то есть неключевым столбцам.

Чтобы нормализовать базу данных до третьей нормальной формы, необходимо сделать так, чтобы в таблицах отсутствовали неключевые столбцы, которые зависят от других неключевых столбцов.

Иными словами, неключевые столбцы не должны пытаться играть роль ключа в таблице, так как они действительно должны быть неключевыми столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Главное правило третьей нормальной форме (3NF) звучит следующим образом: таблица должна содержать правильные неключевые столбцы.

Рассмотрим пример приведения таблиц базы данных к третьей нормальной форме [3]. Для рассмотрения примера возьмем таблицу 11 с сотрудниками, которую в предыдущем разделе мы привели ко второй нормальной форме путем добавления в нее дополнительного атрибута «Табельный номер», который в результате стал первичным ключом.

Чтобы определить, находится ли эта таблица в третьей нормальной форме, мы должны проверить все неключевые столбцы, каждый из них должен зависеть только от первичного ключа, и никаким образом к другим неключевым столбцам он не должен относиться.

Однако, в результате проверки мы выясняем, что столбец «Описание подразделения» не зависит напрямую от первичного ключа. Мы это выяснили,

когда задали себе один вопрос «Каким образом описание подразделения связано с сотрудником?». И наш ответ звучит следующим образом: «Атрибут описание подразделения содержит детальные сведения того подразделения, в котором работает сотрудник».

Отсюда следует, что столбец «Описание подразделения» не связан на прямую с сотрудником, он связан напрямую со столбцом «Подразделение», который напрямую связан с сотрудником, ведь сотрудник работает в каком-то конкретном подразделении. Это и есть транзитивная зависимость, когда один неключевой столбец связан с первичным ключом через другой неключевой столбец.

Чтобы привести эту таблицу к третьей нормальной форме, мы должны выполнить декомпозицию. Другими словами - мы должны эту таблицу разбить на две: в первой хранить сотрудников (таблица 17), а во второй подразделения (таблица 18). А для реализации связи в таблице сотрудников создать ссылку на таблицу подразделений, то есть добавить внешний ключ.

Таблица 17 - Список сотрудников в третьей нормальной форме

| Табельный номер | ФИО | Должность | Подразделение |
|---|---|---|---|
| 1 | Иванов И.И. | Программист | 1 |
| 2 | Сергеев С.С. | Бухгалтер | 2 |
| 3 | John Smith | Продавец | 3 |

Таблица 18 - Список подразделений в третьей нормальной форме

| Идентификатор подразделения | Подразделение | Описание подразделения |
|---|---|---|
| 1 | Отдел разработки | Разработка и сопровождение приложений и сайтов |
| 2 | Бухгалтерия | Ведение бухгалтерского и налогового учета финансово-хозяйственной деятельности |
| 3 | Отдел реализации | Организация сбыта продукции |

Таким образом, в наших таблицах отсутствует транзитивная зависимость, и они находятся в третьей нормальной форме.

После того как мы привели таблицы базы данных к третьей нормальной форме, мы можем переходить к приведению таблиц до следующей нормальной формы, в частности до нормальной формы Бойса-Кодда (BCNF).

## 3.5 Нормальная форма Бойса-Кодда (BCNF)

Между 3NF и 4NF есть еще и промежуточная нормальная форма, она называется – Нормальная форма Бойса-Кодда (BCNF). Иногда ее еще называют «Усиленная третья нормальная форма». Промежуточной или усиленной третьей нормальной формой ее называют потому, что ситуации, в которых могут предъявляться требования нормальной формы Бойса-Кодда, возникают не всегда, то есть это некий частный случай, именно поэтому данная форма не включена в основную градацию. Однако во всех источниках эта форма рассматривается, поэтому и мы ее тоже рассмотрим.

Перед тем как переходить к процессу приведения таблиц базы данных до нормальной формы Бойса-Кодда, необходимо, чтобы эти таблицы уже находились в третьей нормальной форме, подробно процесс приведения таблиц базы данных до третьей нормальной формы, а также все требования, предъявляемые к третьей нормальной форме были рассмотрены в п.3.4.

После того как таблицы базы данных находятся в третьей нормальной форме, мы можем начинать приводить базу данных к нормальной форме Бойса-Кодда и рассматривать соответствующие требования.

Требования нормальной формы Бойса-Кодда следующие:

– таблица должна находиться в третьей нормальной форме (первое требование, как и во всех предыдущих случаях, заключается в том, чтобы таблица находилась в предыдущей нормальной форме, в данном случае в третьей нормальной форме);

– ключевые атрибуты составного ключа не должны зависеть от неключевых атрибутов.

Отсюда следует, что требования нормальной формы Бойса-Кодда предъявляются только к таблицам, у которых первичный ключ составной. Таблицы, у которых первичный ключ простой, и они находятся в третьей нормальной форме, автоматически находятся и в нормальной форме Бойса-Кодда.

Главное правило нормальной формы Бойса-Кодда (BCNF) звучит следующим образом: часть составного первичного ключа не должна зависеть от неключевого столбца.

Рассмотрим пример приведения таблиц базы данных к нормальной форме Бойса-Кодда [3]. Представим, что у нас есть организация, которая реализует множество различных проектов. При этом в каждом проекте работа ведётся по нескольким функциональным направлениям, в каждом из которых есть свой куратор. Сотрудник может быть куратором только того направления, на котором он специализируется, т.е. если сотрудник программист, он не может курировать в проекте направление, связанное с бухгалтерией.

Допустим, что нам нужно хранить информацию о кураторах всех проектов по каждому направлению.

В итоге мы реализуем следующую таблицу, в которой первичный ключ составной «Проект + Направление», так как в каждом проекте есть несколько направлений работы и поэтому, зная только проект, мы не можем определить куратора направления, так же как зная только направление, мы не сможем определить куратора, нам нужно знать и проект и направление, чтобы определить куратора этого направления в этом проекте (таблица 19).

Таблица 19 - Проекты и кураторы

| Проект | Направление | Куратор |
|---|---|---|
| 1 | Разработка | Иванов И.И. |
| 1 | Бухгалтерия | Сергеев С.С. |
| 2 | Разработка | Иванов И.И. |
| 2 | Бухгалтерия | Петров П.П. |
| 2 | Реализация | John Smith |
| 3 | Разработка | Андреев А.А. |

Таблица 19 находится в третьей нормальной форме, так как у нас есть первичный ключ, а неключевой столбец зависит от всего ключа, а не от какой-то его части.

В данном случае таблица 19 не находится в нормальной форме Бойса-Кодда, дело в том, что зная куратора, мы можем четко определить, какое направление он курирует, иными словами, часть составного ключа, то есть «Направление», зависит от неключевого атрибута, то есть «Куратора».

Чтобы привести данную таблицу к нормальной форме Бойса-Кодда, необходимо, как всегда сделать декомпозицию данного отношения, иными словами разбить эту таблицу на несколько таблиц (таблицы 20, 21).

Таблица 20 - Кураторы

| Идентификатор куратора | ФИО | Направление |
|---|---|---|
| 1 | Иванов И.И. | Разработка |
| 2 | Сергеев С.С. | Бухгалтерия |
| 3 | Петров П.П. | Бухгалтерия |
| 4 | John Smith | Реализация |
| 5 | Андреев А.А. | Разработка |

Таблица 21 - Связи кураторов и проектов

| Проект | Идентификатор куратора |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 2 | 3 |
| 2 | 4 |
| 3 | 5 |

Таким образом, в таблице кураторов у нас хранится список кураторов и их специализация, то есть направление, которое они могут курировать, а в таблице связи кураторов и проектов отражается связь проектов и кураторов.

После того как мы привели таблицы базы данных к нормальной форме Бойса-Кодда (BCNF), мы можем переходить к приведению таблиц до следующей нормальной формы, в частности до четвертой нормальной формы (4NF).

### 3.6 Четвертая нормальная форма (4NF)

Перед тем как переходить к процессу приведения таблиц базы данных до четвертой нормальной формы, необходимо чтобы эти таблицы уже находились в третьей нормальной форме или, если первичный ключ составной, в нормальной форме Бойса-Кодда, подробно процесс приведения таблиц базы данных до этих форм мы рассматривали в п. 3.4 и 3.5.

После того как таблицы базы данных находятся в третьей нормальной форме или, если первичный ключ составной, в нормальной форме Бойса-Кодда, мы можем начинать приводить базу данных к четвертой нормальной форме и рассматривать соответствующие требования.

Требование четвертой нормальной формы (4NF) заключается в том, чтобы в таблицах отсутствовали нетривиальные многозначные зависимости.

Рассмотрим, как выглядит в таблицах многозначная зависимость.

Начнем с того, что таблица должна иметь как минимум три столбца, допустим A, B и C, при этом B и C между собой никак не связаны и не зависят друг от друга, но по отдельности зависят от A, и для каждого значения A есть множество значений B, а также множество значений C.

В данном случае многозначная зависимость обозначается вот так:

**A → B**

**A →C**

Если подобная многозначная зависимость есть в таблице, то она не соответствует четвертой нормальной форме.

Рассмотрим пример приведения таблиц базы данных к четвертой нормальной форме [3]. Представим, что мы работаем в каком-то учебном заведении, где есть курсы, которые изучают студенты, преподаватели, которые читают эти курсы, и аудитории, в которых преподаватели проводят занятия по курсам (таблицы 22, 23, 24).

Таблица 22 - Курсы

| Идентификатор курса | Название курса |
|---|---|
| 1 | SQL |
| 2 | Python |
| 3 | JavaScript |

Таблица 23 - Преподаватели

| Идентификатор преподавателя | ФИО |
|---|---|
| 1 | Иванов И.И. |
| 2 | Сергеев С.С. |
| 3 | John Smith |

Таблица 24 - Аудитории

| Идентификатор аудитории | Название аудитории |
|---|---|
| 1 | 101 |
| 2 | 203 |
| 3 | 305 |
| 4 | 407 |
| 5 | 502 |

При этом мы понимаем, что один и тот же курс могут преподавать разные преподаватели, и необязательно в какой-то одной аудитории, один раз курс может читаться в одной аудитории, а в другой раз совсем в другой аудитории, например, на курс записалось гораздо меньше студентов и чтобы не занимать

аудиторию большого размера, под этот поток могут выделить аудиторию меньшего размера.

Также стоит отметить, что под каждый курс подходит только определенный набор аудиторий, например, те, которые оснащены необходимым оборудованием, или те, которые имеют соответствующую вместимость для конкретно этого курса.

В учебном заведении, конечно же, постоянно возникают вопросы с составлением расписания, однако для того чтобы его составлять, необходимо предварительно знать возможности этого учебного заведения. Иными словами, какие преподаватели могут преподавать тот или иной курс, а также в каких аудиториях тот или иной курс может читаться.

Для этого нам необходимо соединить эти три сущности в одной таблице 25. В итоге у нас получается следующая таблица (для наглядности здесь представлены текстовые значения, а не идентификаторы).

Таблица 25 - Связь курсов, преподавателей и аудиторий

| Курс | Преподаватель | Аудитория |
|---|---|---|
| SQL | Иванов И.И. | 101 |
| SQL | Иванов И.И. | 203 |
| SQL | Сергеев С.С. | 305 |
| SQL | Сергеев С.С. | 407 |
| Python | John Smith | 502 |
| Python | John Smith | 305 |

В данном случае первичный ключ здесь состоит из всех трех столбцов, поэтому эта таблица автоматически находится в третьей нормальной форме и нормальной форме Бойса-Кодда. Однако она не находится в четвертой нормальной форме, так как здесь есть многозначная зависимость

**Курс →→ Преподаватель**

**Курс →→ Аудитория**

Читается такая запись таким образом, что для каждого курса в этой таблице может быть несколько преподавателей, а также несколько аудиторий.

При этом, понимаем, что преподавателю без разницы, в какой аудитории читать лекцию, так же как и самой аудитории без разницы, какой преподаватель в ней будет работать. Иными словами, эти два атрибута «Преподаватель» и «Аудитория» никак не зависят друг от друга, но они оба по отдельности зависят от курса. Но что же плохого в этой таблице и в этой многозначной зависимости? Чтобы ответить на этот вопрос, мы можем задать себе несколько других вопросов.

Что будет если, например, преподаватель «Иванов И.И.» уволился? Нам нужно будет удалить две строки из этой таблицы, но удалив эти строки, мы удалим всю информацию и об аудиториях 101 и 203. Но они на самом-то деле есть и должны участвовать в планировании расписания. Это аномалия, и это плохо.

Или другая ситуация, что будет, если курсу назначен преподаватель, но аудитория еще не определена? Или наоборот, с аудиторией уже определились, а вот преподаватель еще не известен.

Мы должны создать записи либо с NULL либо со значениями по умолчанию, и это также является аномалией.

Многозначные зависимости плохи как раз тем, что их нельзя независимо друг от друга редактировать. Иными словами, чтобы внести изменения в одну зависимость, мы неизбежно должны затронуть другую зависимость.

Поэтому главное правило четвертой нормальной формы звучит следующим образом: в таблице не должно быть многозначных зависимостей.

Решение в данном случае как всегда – **декомпозиция.**

Мы должны вынести каждую многозначную зависимость в отдельную таблицу, то есть разнести независимые друг от друга атрибуты, в нашем случае «Преподаватель» и «Аудитория», по разным таблицам 26, 27.

Таблица 26 - Связь курсов и преподавателей

| Курс | Преподаватель |
|---|---|
| SQL | Иванов И.И. |
| SQL | Сергеев С.С. |
| Python | John Smith |

Таблица 27 - Связь курсов и аудиторий

| Курс | Аудитория |
|---|---|
| SQL | 101 |
| SQL | 203 |
| SQL | 305 |
| SQL | 407 |
| Python | 502 |
| Python | 305 |

Чтобы стало еще понятней, давайте закрепим знания и рассмотрим классический пример, который обычно используется в литературе для пояснения четвертой нормальной формы (таблица 28).

Таблица 28 - Связи студентов, курсов и хобби

| Студент | Курс | Хобби |
|---|---|---|
| 1 | 2 | 3 |
| Иванов И.И. | SQL | Футбол |
| Иванов И.И. | Java | Хоккей |
| Сергеев С.С. | SQL | Волейбол |
| Сергеев С.С. | SQL | Теннис |

Продолжение таблицы 28

| 1 | 2 | 3 |
|---|---|---|
| John Smith | Python | Футбол |
| John Smith | Java | Теннис |

Данная таблица хранит информацию о студентах, в частности здесь хранятся курсы, которые посещает студент, и увлечения этого студента, то есть хобби. Отсюда следует, что каждый студент может посещать несколько курсов и иметь несколько увлечений.

Первичный ключ здесь составной и состоит он из всех трех столбцов. При этом мы можем заметить, что курс и хобби никак не связаны и не зависят друг от друга, но по отдельности зависят от студента.

Таким образом, мы можем наблюдать в этой таблице нетривиальную многозначную зависимость

**Студент →→ Курс**

**Студент →→ Хобби**

Поэтому таблица 28 не находится в четвертой нормальной форме.

Кроме всех тех аномалий, связанных с редактированием данных, которые мы уже рассмотрели на предыдущем примере, в данном случае еще продемонстрирована проблема неоднозначной выборки данных.

Допустим, нам необходимо получить информацию о хобби студентов, которые посещают курс по SQL. Очевидным действием станет выборка с условием Курс = SQL, в результате мы получим 3 хобби: футбол, волейбол и теннис (таблица 29).

Таблица 29 - Результат выборки. Хобби студентов, которые посещают курс по SQL

| Студент | Курс | Хобби |
|---|---|---|
| Иванов И.И. | SQL | Футбол |
| Сергеев С.С. | SQL | Волейбол |
| Сергеев С.С. | SQL | Теннис |

Однако, если мы заглянем в исходную таблицу, то мы четко увидим, что «Иванов И.И.» посещает курс по SQL и имеет хобби «Хоккей», но в нашей выборке этого хобби нет.

Чтобы нормализовать эту таблицу, мы должны точно так же, как и в предыдущем примере, разбить ее на две (таблица 30, 31).

Таблица 30 - Связь студентов и курсов

| Студент | Курс |
|---|---|
| Иванов И.И. | SQL |
| Иванов И.И. | Java |
| Сергеев С.С. | SQL |
| John Smith | Python |
| John Smith | Java |

Таблица 31 - Связь студентов и хобби

| Студент | Хобби |
|---|---|
| 1 | 2 |
| Иванов И.И. | Футбол |
| Иванов И.И. | Хоккей |

Продолжение таблицы 31

| 1 | 2 |
|---|---|
| Сергеев С.С. | Волейбол |
| Сергеев С.С. | Теннис |
| John Smith | Футбол |
| John Smith | Теннис |

Однако в реальности такую ситуацию и такую таблицу вряд ли можно встретить, так как следуя здравому смыслу такие абсолютно не связанные друг с другом данные никто не будет хранить в одной таблице. Поэтому этот пример чисто теоретический и приводится для демонстрации принципов четвертой нормальной формы.

И если говорить о реальных данных, то нормализация до четвертой нормальной формы, как и до всех последующих, в современном мире практически не встречается. Если четвертую нормальную форму еще как-то можно представить и даже встретить данные, нормализованные до этой формы, то встретить данные, нормализованные до 5NF или 6NF, практически невозможно.

Вы можете спросить, а почему не нормализуют данные до 5 или 6 нормальной формы? Ведь каждая нормальная форма устраняет определенные аномалии, и если сделать полностью нормализованную базу данных, то, по сути, она будет идеальная, не содержащая ни одной аномалии, это же хорошо.

Да, совершенно верно, база данных не будет содержать аномалий, но давайте вспомним, какие преимущества нам дает нормализация.

Обычно во всех источниках приводится два основных глобальных преимущества:

- устранение аномалий;
- повышение производительности.

Если с устранением аномалий все ясно, то есть в полностью нормализованной базе данных их не будет и это хорошо, то с повышением производительности не все так однозначно.

Также нормализация повышает производительность, но только где-то до 3NF. Начиная с 4NF производительность увеличиваться не будет, более того, с каждой новой формой производительность будет значительно снижаться, не говоря уже о том, что с нормализованной базой данных до 5NF или 6NF будет крайне сложно и неудобно работать и сопровождать ее, ведь с каждой новой формой мы значительно увеличиваем количество таблиц в базе данных.

Поэтому процесс нормализации не является строго обязательным, то есть не нужно нормализовать базу данных, только для того чтобы она была нормализована. В процессе проектирования базы данных необходимо следовать здравому смыслу и найти баланс между отсутствием аномалий и приемлемой производительностью.

Полностью нормализованная база данных – это плохая база данных. Хорошая база данных – это база, которая достаточно нормализована, чтобы не создавать аномалии для пользователей этой базы данных, и в то же время она имеет хорошую производительность.

После того как мы привели таблицы базы данных к четвертой нормальной форме, мы можем переходить к приведению таблиц до пятой нормальной формы (5NF).

### 3.7 Пятая нормальная форма (5NF)

Перед тем как переходить к процессу приведения таблиц базы данных к пятой нормальной форме, необходимо чтобы эти таблицы уже находились в четвертой нормальной форме, подробно процесс приведения таблиц базы дан-

ных до четвертой нормальной формы, а также все требования, предъявляемые к четвертой нормальной форме, мы рассматривали в п. 3.6.

После того как таблицы базы данных находятся в четвертой нормальной форме, мы можем начинать приводить базу данных к пятой нормальной форме и рассматривать соответствующие требования.

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. Это стандартное определение для пятой нормальной формы. К сожалению, более простыми словами сформулировать определение для пятой нормальной формы достаточно сложно.

Однако на основе этого определения мы можем сделать следующий вывод: требование пятой нормальной формы (5NF) заключается в том, чтобы в таблице каждая нетривиальная зависимость соединения определялась потенциальным ключом этой таблицы. Здесь вводится новое понятие «Зависимость соединения».

До 5NF мы осуществляли декомпозицию таблиц и не задумывались ни о какой потере данных, ведь у нас такой потери данных просто не было. Однако существуют таблицы, которые не получится декомпозировать на две таблицы без потери данных, то есть какие-то данные мы потеряем при соединении двух итоговых, полученных после декомпозиции, таблиц. Но, если декомпозировать такую таблицу не на две, а на три таблицы, то потери данных можно избежать.

И таблица будет находиться в пятой нормальной форме, если при соединении этих трех таблиц, которые были получены в результате декомпозиции, будут формироваться ровно те же самые данные, что и в исходной таблице до декомпозиции. Однако если этого происходить не будет, то есть данные будут отличаться, например, какие-то строки были потеряны, или созданы новые, то в этом случае возникает так называемая **зависимость соединения,** то есть часть данных одного столбца зависит от части данных другого столбца.

Таким образом, таблица будет находиться в пятой нормальной форме, если она не будет содержать зависимости соединения. И здесь вводится еще одно новое понятие «Декомпозиция без потерь».

**Декомпозиция без потерь** – процесс разбиения одной таблицы на несколько, при условии, что в случае соединения таблиц, которые были получены в результате декомпозиции, будет формироваться ровно та же самая информация, что и в исходной таблице до декомпозиции. Иными словами, чтобы выполнить требование пятой нормальной формы, необходимо осуществить декомпозицию таблицы без потери данных.

Схематично это выглядит примерно следующим образом.

Допустим, существует таблица **T (C1, C2, C3)** где C1, C2, C3 – столбцы и вместе они являются составным первичным ключом. Таблица находится в четвертой нормальной форме. В соответствии с требованиями предметной области у нас проявляется зависимость соединения: **{C1, C2}, {C1, C3}, {C2, C3}**.

Чтобы привести данную таблицу к пятой нормальной форме, необходимо декомпозировать ее на следующие три таблицы:

- T1 (C1, C2);
- T2 (C1, C3) ;
- T3 (C2, C3) ;

При этом, если мы соединим три новые таблицы (T1, T2, T3) и получим исходную таблицу (T), то это будет означать, что декомпозицию мы выполнили без потерь.

Рассмотрим пример приведения таблиц базы данных к пятой нормальной форме [3]. Представим, что у нас есть таблица, которая хранит информацию о связи сотрудников с проектами и направлениями работы сотрудников в этих проектах.

В случае таблицей 32 мы не можем сказать, находится ли она в 5NF или нет, так как нам сначала необходимо разобраться в предметной областью и определить ограничения.

Таблица 32 - Связь сотрудников с проектами и направлениями работы в проектах

| Сотрудник | Проект | Направление |
|---|---|---|
| Иванов И.И. | Интернет магазин | Разработка |
| Сергеев С.С. | Интернет магазин | Бухгалтерия |
| Сергеев С.С. | Новый офис | Реализация |
| John Smith | Личный кабинет | Бухгалтерия |
| Иванов И.И. | Личный кабинет | Разработка |
| Иванов И.И. | Информационная система | Разработка |

Поработав с предметной областью, мы выясняем, что:

– Иванов И.И. может работать только в направлении «Разработка»;

– Сергеев С.С. может работать в любом направлении, за исключением «Разработка»;

– Иванов И.И. может участвовать в большом количестве проектов;

– John Smith может участвовать только в одном проекте.

Если придерживаться этих требований, то в нашу таблицу можно очень легко внести некорректные данные, и у нас точно так же, как и в случае с четвертой нормальной формой, будут возникать аномалии при добавлении, изменении и удалении данных.

Таблица 32 находится в четвертой нормальной форме, так как у нас отсутствует многозначная зависимость, ведь у нас нет таких атрибутов, которые зависели бы от другого атрибута.

Однако принимая во внимание наши требования, мы понимаем, что часть данных каждого из столбцов зависит от части данных другого столбца, то есть существуют некие зависимости, и эти зависимости определяются не целым потенциальным ключом, а только его частью.

Поэтому, чтобы устранить возможность внесения некорректных данных, мы можем попытаться выполнить декомпозицию без потерь, и тем самым привести таблицу к пятой нормальной форме.

Чтобы выполнить декомпозицию без потерь, нам нужно разбить данную таблицу на три проекции: **{Сотрудник, Проект}, {Сотрудник, Направление}, {Проект, Направление}** с условием, что в случае обратного соединения, мы получим те же самые данные, что у нас были и до декомпозиции (таблицы 33, 34, 35).

Если это нам удастся сделать, то мы устраним нетривиальные зависимости соединения и нормализуем наши таблицы до пятой нормальной формы.

Таблица 33 - Связь сотрудников и проектов

| Сотрудник | Проект |
|---|---|
| Иванов И.И. | Интернет магазин |
| Сергеев С.С. | Интернет магазин |
| Сергеев С.С. | Новый офис |
| John Smith | Личный кабинет |
| Иванов И.И. | Личный кабинет |
| Иванов И.И. | Информационная система |

Таблица 34 - Связь сотрудников и направлений

| Сотрудник | Направление |
|---|---|
| Иванов И.И. | Разработка |
| Сергеев С.С. | Бухгалтерия |
| Сергеев С.С. | Реализация |
| John Smith | Бухгалтерия |

Таблица 35 - Связь проектов и направлений

| Проект | Направление |
|---|---|
| Интернет магазин | Разработка |
| Интернет магазин | Бухгалтерия |
| Новый офис | Реализация |
| Личный кабинет | Бухгалтерия |
| Личный кабинет | Разработка |
| Информационная система | Разработка |

Таблицы созданы, теперь если мы выполним следующий запрос, который соединяет эти три таблицы, и он вернет нам точно такие же данные, что и в исходной таблице, то зависимости соединения у нас нет, и наши таблицы находятся в 5NF (рисунки 1, 2).

```
SELECT СП.Сотрудник, ПН.Проект, СН.Направление
FROM СотрудникПроект СП
JOIN ПроектНаправление ПН ON СП.Проект = ПН.Проект
JOIN СотрудникНаправление СН ON СП.Сотрудник = СН.Сотрудник
AND ПН.Направление = СН.Направление
```

Рисунок 1 – Запрос, соединяющий три таблицы

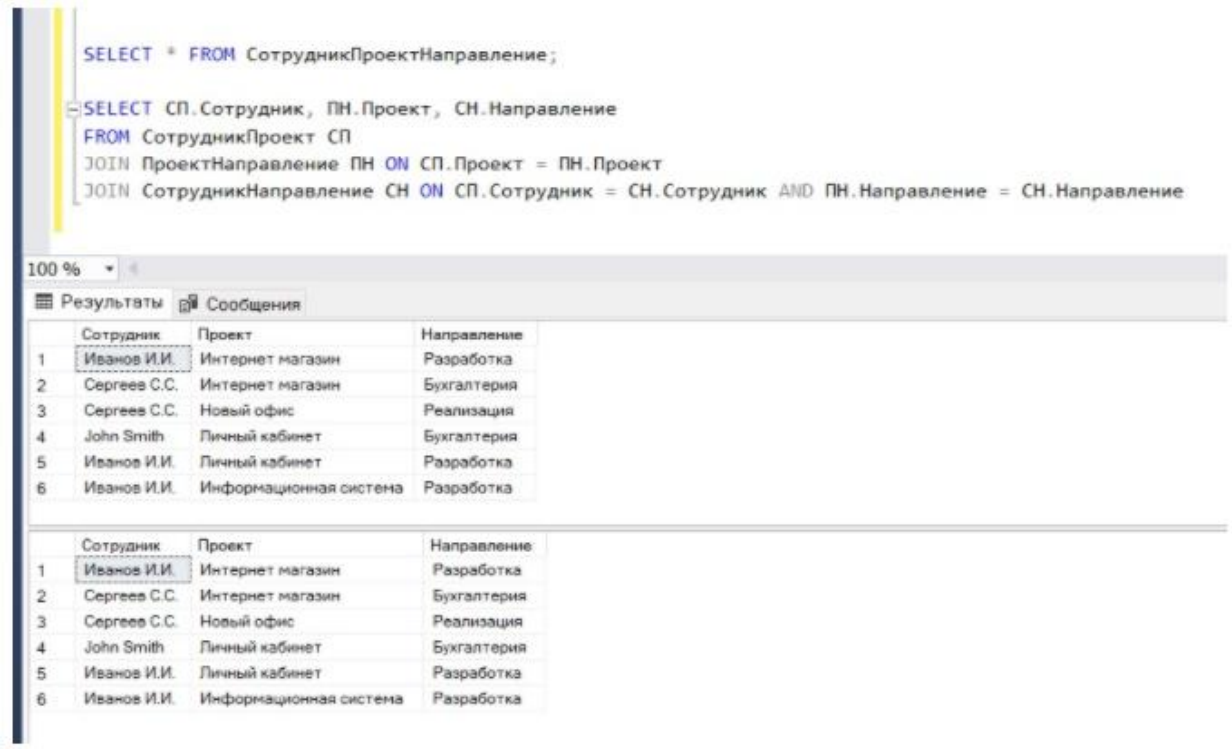


Рисунок 2 – Результат выполнения запроса

Как видим, данные точно такие же. Можно сделать вывод, что таблицы находятся в пятой нормальной форме.

Обязательно стоит отметить, что пятая нормальная форма является окончательной нормальной формой по отношению к операциям разбиения таблиц на проекции и их соединения, именно поэтому ее альтернативное название – проекционно-соединительная нормальная форма. Таким образом, если таблица находится в 5NF, то гарантируется, что она не содержит аномалий, которые могут быть исключены посредством ее разбиения на проекции.

Также стоит отметить, что таблицы, которые необходимо нормализовать до пятой нормальной формы, встречаются крайне редко. Более того, такие таблицы являются не совсем удачными с точки зрения проектирования. Кроме всего прочего, чтобы привести таблицу к пятой нормальной форме, Вы должны очень хорошо разбираться в предметной области, чтобы определить зависимости соединения, ведь это действительно очень сложно. Иными словами, если

Вам удастся определить эти зависимости соединения, то только в этом случае Вы сможете привести таблицу к пятой нормальной форме.

### 3.8 Доменно-ключевая нормальная форма (DKNF)

Пятая нормальная форма является окончательной нормальной формой по отношению к операциям разбиения таблиц на проекции и их соединения. Однако существуют и другие нормальные формы, например, доменно-ключевая нормальная форма (DKNF), которая, в отличие от рассмотренных раннее нормальных форм, не определяется в терминах функциональных зависимостей, многозначных зависимостей или зависимостей соединения. Вместо этого в фокусе внимания в этой нормальной форме стоят ограничения доменов и ограничения ключей.

**Ограничение домена** – это ограничение, предписывающее использование для определенного атрибута значений только из некоторого заданного домена (набора значений).

**Ограничение ключа** – это ограничение, утверждающее, что некоторый атрибут или комбинация атрибутов представляет собой потенциальный ключ.

Таким образом, требование доменно-ключевой нормальной формы заключается в том, чтобы каждое наложенное ограничение на таблицу являлось логическим следствием ограничений доменов и ограничений ключей, которые накладываются на данную таблицу [6].

Таблица, находящаяся в доменно-ключевой нормальной форме, обязательно находится в 5NF, и соответственно, в 4NF и т.д. Однако, стоит отметить, что не всегда возможно привести таблицу к доменно-ключевой нормальной форме, более того, не всегда возможно получить ответ на вопрос о том, когда может быть выполнено такое приведение.

## 3.9 Шестая нормальная форма (6NF)

Шестая нормальная форма (6NF) была введена при работе с хронологическими базами данных.

**Хронологическая база данных** – это база, которая может хранить не только текущие данные, но и исторические данные, т.е. данные, относящиеся к прошлым периодам времени. Однако такая база может хранить и данные, относящиеся к будущим периодам времени.

В процессе проектирования хронологических баз данных возникают некоторые особые проблемы, решить которые можно с помощью: горизонтальной декомпозиции и вертикальной декомпозиции.

В данном случае нас интересует вертикальная декомпозиция, процесс которой очень сильно напоминает нашу классическую нормализацию, которую мы рассматривали до пятой нормальной формы включительно.

Иными словами, декомпозиция таблиц, которую мы использовали для приведения этих таблиц к той или иной нормальной форме, по факту и является вертикальной декомпозицией.

В процессе изучения хронологических баз данных исследователи выдвигали доводы в пользу максимально возможной вертикальной декомпозиции таблиц, а не просто их декомпозиции до какой-то определённой степени, которую требует классическая теория нормализации. Общая идея состояла в том, что таблицы должны быть приведены к неприводимым компонентам, под этим подразумеваются такие компоненты, для которых дальнейшая декомпозиция без потерь становится невозможной.

Теперь стоит напомнить, что пятая нормальная форма основана на так называемых зависимостях соединения. Поскольку вертикальная декомпозиция, которая используется в хронологических базах данных, представляет собой классическое разделение таблиц на проекции, была сформулирована новая

нормальная форма, основанная на обобщенном понятии зависимости соединения, поэтому новую форму назвали «Шестая нормальная форма».

Требование шестой нормальной формы заключается в том, что таблица должна удовлетворять всем нетривиальным зависимостям соединения.

Из этого определения следует, что таблица находится в 6NF, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Стоит отметить, что таблица, которая находится в 6NF, также находится и в 5NF, и во всех предыдущих.

Шестая нормальная форма вводит такое понятие как «Декомпозиция до конца», то есть максимально возможная декомпозиция таблиц.

Однако, если в хронологических базах данных такая нормализация может быть полезна, так как она позволяет бороться с избыточностью, то в нехронологических базах данных нормализация таблиц до шестой нормальной формы приведёт к значительному снижению производительности. Кроме этого такая нормализация сделает работу с базой данных очень сложной за счет многократного увеличения количества таблиц.

Поэтому шестую нормальную форму в реальном мире не используют, более того, трудно даже представить себе ситуацию, при которой возникала бы необходимость нормализовать базу данных до шестой нормальной формы. Практического применения шестой нормальной формы, наверное, просто нет.

Процесс проектирования правильной базы данных – это не процесс приведения ее к самой высокой нормальной форме, это компромисс между отсутствием аномалий и приемлемой производительностью.

Поэтому в процессе нормализации базы данных необходимо руководствоваться в первую очередь требованиями к разрабатываемой системе и требованиями предметной области. Вы должны подумать о том, какие именно операции (действия) будут выполняться над данными. Так все ошибки нормализации станут очевидными и Вы сможете увидеть, какие аномалии могут возникнуть в тех или иных случаях, и принимать решения о нормализации, иными словами, Вы должны руководствоваться здравым смыслом.

# Список использованных источников

1 Карпова, Т.С. Базы данных: модели, разработка, реализация [Электронный ресурс]: учебное пособие / Т.С. Карпова. - 2-е изд., исправ. - Москва: Национальный Открытый Университет «ИНТУИТ», 2018. - 241 с. – (ЭБС Университетская библиотека ONLINE). - Режим доступа: http://biblioclub.ru/index.php?page=book&id=429003.

2 Лазицкас, Е.А. Базы данных и системы управления базами данных [Электронный ресурс]: учебное пособие / Е.А. Лазицкас, И.Н. Загумённикова, П.Г. Гилевский. - Минск: РИПО, 2019. - 267 с. – (ЭБС Университетская библиотека ONLINE). – Режим доступа: http://biblioclub.ru/index.php?page=book&id=463305.

3 Программирование для начинающих [Электронный ресурс]: заметки IT-специалиста. – 2021. – Режим доступа: https://info-comp.ru/database-normalization.

4 Васильев, А.А. Базы данных Access [Электронный ресурс]: самоучитель / А.А. Васильев, И.Н.Телина. – Санкт-Петербург: Питер, 2018. - Режим доступа: http://www.litres.ru/aleksandr-vasilev/.

5 Кузнецов, С. Введение в реляционные базы данных [Электронный ресурс] / С. Кузнецов. - Москва: Национальный Открытый Университет «ИНТУИТ», 2018. - 248 с. – (ЭБС Университетская библиотека ONLINE). - Режим доступа: http://biblioclub.ru/index.php?page=book&id=429088.

6 Сирант, О.В. Работа с базами данных [Электронный ресурс] / О.В. Сирант, Т.А. Коваленко. - 2-е изд., испр. - Москва: Национальный Открытый Университет «ИНТУИТ», 2019. - 150 с. – (ЭБС Университетская библиотека ONLINE). – Режим доступа: http://biblioclub.ru/index.php?page=book&id=428978.