Формализация информации и Big Data

«02.03.03 - Математическое обеспечение и администрирование информационных систем направленность разработка и администрирование информационных систем»

http//vikchas.ru

Тема 1. Формализация информации

Лекция 6 «Базы данных»

Часовских Виктор Петрович
д-р техн. наук, профессор кафедры ШИиКМ

ФГБОУ ВО «Уральский государственный экономический университет»

Екатеринбург 2023

Нормализация отношений

В реляционных базах данных **схема** содержит как структурную, так и семантическую информацию.

Схема отношения - это именованное множество пар {имя атрибута, имя домена}. Степень или "арность" схемы отношения - мощность этого множества, т.е. количество атрибутов.

Схема базы данных - это набор именованных схем отношений.

Структурная информация связана с объявлением отношений.

Семантическая информация выражается множеством функциональных зависимостей между атрибутами отношений.

Некоторые функциональные зависимости могут приводить к возникновению трудностей (аномалий) при модификации баз данных. Процедура устранения нежелательных функциональных зависимостей (а, следовательно, и связанных с ними аномалий) составляет суть процесса нормализации.

Нормализация - это пошаговый обратимый процесс замены данной схемы (или совокупности отношений) базы данных другой схемой, в которой отношения имеют более простую и регулярную структуру.

В теории нормальных форм определены различные нормальные формы, которые ограничивают типы допустимых функциональных зависимостей отношения. Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет заданному набору ограничений.

Коддом первоначально выделено три нормальных формы - первая (1НФ), вторая (2НФ) и третья (3НФ) и описан механизм, позволяющий любое отношение преобразовать к 3НФ. Далее, Коддом предложена усиленная третья нормальная форма, или нормальная форма Бойса-Кодда (БКНФ).

Четвертая нормальная форма (4НФ) определена Фейджином.

Всего определено 6 форм.

Уровень нормализации отношения зависит от его семантики и не может быть однозначно определен из данных, содержащихся в текущий момент в базе данных.

Семантика базы данных должна быть задана с помощью функциональных зависимостей.

Понятие функциональной зависимости является фундаментальным в теории реляционных баз данных.

В отношении R атрибут Y функционально зависит от атрибута X тогда и только тогда, когда в любой момент времени каждому значению X соответствует в точности одно значение

$$R.X \rightarrow R.Y$$

Пример

Товар	Поставщик	Адрес	Количество	Цена
Товар1	Поставщик1	Адрес1	500	1000
Товар2	Поставщик2	Адрес2	2000	600
Товар3	Поставщик3	Адрес3	400	5000
Товар4	Поставщик4	Адрес4	150	2000

В примере атрибут ***Адрес*** функционально зависит от атрибута ***Поставщик***.

Поставщика → Адрес

Полная функциональная зависимость. Функциональная зависимость

$$R.X \rightarrow R.Y$$

называется полной, если атрибут Y зависит от всей группы атрибутов X и не зависит функционально от любой ее части (подмножества).

Транзитивная функциональная зависимость.

Функциональная зависимость R.X → R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости: R.X → R.Z и R.Z → R.Y

Первая нормальная форма (1НФ). Отношение R называется нормализованным или приведенным к первой нормальной форме, если значения всех его атрибутов простые (атомарные), т.е. значение атрибута не должно быть множеством или повторяющейся группой

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

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

Таблица сотрудников в ненормализованном виде.

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 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

Т.е. 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.

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.

Вторая нормальная форма (2NF) базы данных

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, т.е. ключ идентифицирует каждую строку таблицы. По ключу мы можем обратиться к конкретной строке данных в таблице.

Если ключ **составной**, т.е. состоит из нескольких столбцов, то все остальные неключевые столбцы должны зависеть от всего ключа, т.е. 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, т.е. 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.

Создадим таблицу *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

Таблица удовлетворяет условиям первой нормальной формы, т.е. в ней нет дублирующих строк и все значения атомарны.

Для нормализации этой таблицы до второй нормальной формы нужно внедрить **первичный ключ.**

Добавим для каждого сотрудника табельный номер

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

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

Первичный ключ составной - приведение таблицы ко второй нормальной форме

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.

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

Таблица проектов организации в первой нормальной форме

Как видим, она в первой нормальной форме, значит, мы можем пытаться приводить ее ко второй нормальной форме.

Чтобы привести таблицу ко второй нормальной форме, необходимо определить для нее первичный ключ.

Посмотрев на эту таблицу, 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, т.е. каждое сочетание значений этих столбцов является уникальным.

Таким образом, мы определили первичный ключ и он у нас составной, т.е. состоящий их двух столбцов.

Создадим составной первичный ключ.

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, т.е. 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.

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.

Создадим следующие таблицы:

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

Проекты

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

Участники

Идентификатор проекта	Идентификатор участника
1	1
1	2
1	3
2	2
2	3

Связь проектов и участников этих проектов

Создали 3 таблицы:

1.Проекты, в нее добавили искусственный первичный ключ.

2.Участники, в нее также добавили искусственный первичный ключ.

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

Мы привели таблицы базы данных ко второй нормальной форме.

Третья нормальная форма (3NF) базы данных

Если таблицы базы данных находятся во **второй** нормальной форме, то можно 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.

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

Неключевые 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.

Таблица должна содержать правильные неключевые столбцы.

Таблица сотрудников во второй нормальной форме

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

Проверяем неключевые столбцы, каждый из них должен зависеть только от первичного ключа, и никаким образом к другим неключевым столбцам он не должен относиться.

Видим, что столбец *«**Описание подразделения**»* не зависит напрямую от первичного ключа. Это следует из вопроса *«Каким образом описание подразделения связано с сотрудником?»*. Ответ: *«Атрибут описание подразделения содержит детальные сведения того подразделения, в котором работает сотрудник»*.

Является очевидным, 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.

Необходимо произвести **декомпозицию.**

Необходимо исходную таблицу разбить на две: в первой хранить сотрудников, а во второй подразделения. А для реализации связи в таблице сотрудников создать ссылку на таблицу подразделений, т.е. добавить внешний ключ.

Таблица подразделений 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

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

В таблицах отсутствует транзитивная зависимость, и они находятся в третьей нормальной форме.

Нормальная форма Бойса-Кодда (BCNF)

Между 3 и 4 нормальной формой есть еще и промежуточная нормальная форма, она называется – Нормальная форма Бойса-Кодда (BCNF). Иногда ее еще называют *«**Усиленная третья нормальная форма**»*.

Для перехода к **BCNF,** необходимо, чтобы таблицы были в **3NF.**

Требования нормальной формы Бойса-Кодда следующие:

- Таблица должна находиться в третьей нормальной форме. Здесь все как обычно, т.е. как и у всех остальных нормальных форм, первое требование 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.

Требования **BCNF** 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 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.

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 проектов и кураторов

Таблица 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.

Но таблица 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, т.е. ***«Направление»***, зависит от неключевого атрибута, т.е. ***«Куратора»***.

Чтобы привести данную таблицу к нормальной форме Бойса-Кодда, необходимо, как всегда сделать декомпозицию данного отношения, т.е. разбить эту таблицу на несколько таблиц.

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 кураторов

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

Таблица связи кураторов и проектов

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

Четвертая нормальная форма (4NF) базы данных

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

В четвертой нормальной форме (4NF) в таблицах отсутствуют нетривиальные многозначные зависимости.

В таблицах многозначная зависимость выглядит следующим образом.

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 примере многозначная зависимость обозначается вот так:

A —> B

A —> C

ПРИМЕР

Рассмотрим наш университет: есть дисциплины (курсы), которые изучают студенты, преподаватели, которые читают эти курсы, и аудитории, в которых преподаватели проводят занятия по курсам.

Курсы

Идентификатор курса	Название курса
1	SQL
2	Python
3	Visual Studio 2019

Преподаватели

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

Аудитории

Идентификатор аудитории	Номер аудитории
1	152
2	203
3	305
4	407
5	555

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.

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 университете 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 необходимо соединить эти три сущности в одной таблице. В итоге получается следующая таблица (*для наглядности здесь представлены текстовые значения, а не идентификаторы*).

Курс	Преподаватель	Аудитория
SQL	Иванов И.И.	152
SQL	Иванов И.И.	203
SQL	Сергеев С.С.	305
SQL	Сергеев С.С.	407
Python	John Smith	555
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

Т.е. для каждого курса в этой таблице может быть несколько преподавателей, а также несколько аудиторий.

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?

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 о аудиториях 152 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.

Должны будем 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**.

Необходимо вынести каждую многозначную зависимость в отдельную таблицу, т.е. разнести независимые друг от друга атрибуты, в нашем случае *«Преподаватель»* и *«Аудитория»*, по разным таблицам.

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

Связь курсов и преподавателей

Курс	Аудитория
SQL	152
SQL	203
SQL	305
SQL	407
Python	555
Python	305

Связь курсов и аудиторий.

Пятая нормальная форма (5NF) базы данных

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

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.

Требование пятой нормальной формы (5NF) заключается в том, чтобы в таблице каждая **нетривиальная зависимость соединения** определялась потенциальным ключом этой таблицы.

Однако существуют таблицы, которые не получится декомпозировать на две таблицы без потери данных, т.е. 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.

Таблица будет находиться в **пятой** нормальной форме, если при соединении (**JOIN**) 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, т.е. данные будут отличаться, например, какие-то строки были потеряны, или созданы новые, то в этом случае возникает так называемая **зависимость соединения**, т.е. 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.

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

Рассмотрим таблицу, которая хранит информацию о связи сотрудников с проектами и направлениями работы сотрудников в этих 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

Связь сотрудников с проектами и направлениями работы в проектах

Таблица находиться в 4-ой форме

Понятно, что часть данных каждого из столбцов зависит от части данных другого столбца, т.е. 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.

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

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

Связь сотрудников и направлений

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

Связь проектов и направлений

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.

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.

Необходимо хорошо разбираться в предметной области, чтобы определить зависимости соединения, ведь это действительно очень сложно. Иными словами, если удастся определить эти зависимости соединения, то только в этом случае можно привести таблицу к пятой нормальной форме.

Доменно-ключевая нормальная форма (DKNF) базы данных

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.

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.

Шестая нормальная форма (6NF) базы данных

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.

А 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»***, т.е. 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.

На практике нет никакой необходимости приводить базу данных до какой-то определенной нормальной формы.

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. Необходимо подумать о том, какие именно операции (действия) будут выполняться над данными.

Так все ошибки нормализации станут очевидными и можно увидеть, какие аномалии могут возникнуть в тех или иных случаях, и принимать решения о нормализации, иными словами, необходимо руководствоваться здравым смыслом.