Содержание

Нормализация баз данных
1НФ (Первая Нормальная Форма)
- Аномалии обновлений
  - Аномалии вставки (INSERT)
  - Аномалии обновления (UPDATE)
  - Аномалии удаления (DELETE)
- Функциональные зависимости
  - Определение понятия функциональной зависимости

2НФ (Вторая Нормальная Форма)
- Анализ декомпозированных отношений
  - Оставшиеся аномалии вставки (INSERT)
  - Оставшиеся аномалии обновления (UPDATE)
  - Оставшиеся аномалии удаления (DELETE)

3НФ (Третья Нормальная Форма)
Алгоритм нормализации (приведение к 3НФ)
Заключение по нормализации
- Сравнение нормализованных и ненормализованных моделей
- Нормальные формы более высоких порядков
  - НФБК (Нормальная Форма Бойса-Кодда)
  - 4НФ (Четвертая Нормальная Форма)
  - 5НФ (Пятая Нормальная Форма)

Нормализация баз данных.

Методику нормализации таблиц-отношений разработал американский ученый Кодд в 1970 г. Ее суть сводится к приведению таблиц к той или иной нормальной форме. Были выделены три нормальные формы – 1НФ, 2НФ, 3НФ. Позже стали выделять нормальную форму Бойса–Кодда (НФБК), а затем 4НФ и 5НФ. Каждая последующая нормальная форма вводит определенные ограничения на хранимые в базе данные.

Реляционная база данных считается эффективной, если все ее таблицы находятся как минимум в 3НФ.

Проектирование схемы БД должно решать задачи минимизации дублирования данных, упрощения и ускорения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных (удаления, вставки, изменения). Для решения подобных проблем проводится **нормализация отношений**.

Однако в технологии работы с хранилищами данных может использоваться обратный прием - **денормализация отношений** с целью увеличения скорости выполнения запросов к очень большим объемам архивных данных (так как в архивах данные не изменяются и можно отдать предпочтение скорости выборки данных).

В рамках реляционной модели данных Э.Ф. Коддом были разработаны принципы нормализации отношений.

Нормализация - это формальный метод анализа отношений на основе их первичного ключа и существующих связей. Её задача - это замена одной схемы (или совокупности отношений) БД другой схемой, в которой отношения имеют более простую и регулярную структуру.

Моделирование структуры базы данных при помощи алгоритма нормализации имеет серьезные **недостатки**: методика нормализации предполагает первоначальное размещение всех атрибутов проектируемой предметной области в одном отношении, что является очень неестественной операцией:

- Интуитивно разработчик сразу проектирует несколько отношений в соответствии с обнаруженными сущностями. Даже если совершить насилие над собой и создать одно или несколько отношений, включив в них все предполагаемые атрибуты, то совершенно неясен смысл полученного отношения.
- Невозможно сразу определить полный список атрибутов. Пользователи имеют привычку называть разными именами одни и те же вещи или наоборот, называть одними именами разные вещи. Для проведения процедуры нормализации необходимо выделить зависимости атрибутов, что тоже очень нелегко.

В реальном проектировании структуры базы данных применяются другой метод - так называемое **семантическое моделирование**, которое представляет собой моделирование структуры данных, опирающееся на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты диаграмм "сущность-связь (ER)" с построением концептуальной модели базы данных. Обычно проектирование с помощью ER-диаграмм позволяет построить БД, нормализованную до 3НФ или НФБК.

Но, нормализацию удобно применять для проверки ER-модели.

1. 1НФ (Первая Нормальная Форма).

Определение 1НФ.

Первая нормальная форма (1НФ) связана с понятиями простого и сложного атрибутов. Простой атрибут - это атрибут, значения которого атомарные (т.е. неделимы). Сложный атрибут может иметь значение, представляющее собой объединение нескольких значений одного (например, ШхВхГ) или разных доменов (например, Адрес). В первой нормальной форме устраняются повторяющиеся (многозначные) атрибуты (например, в телефон1,телефон2) или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.

Отношение приведено к 1НФ, если все его атрибуты - простые, т.е. значение атрибута не должно быть множеством или повторяющейся группой. Для приведения таблиц к 1НФ необходимо разбить сложные атрибуты на простые, а многозначные атрибуты вынести в отдельные отношения.

Свойства отношения, следующие из его определения:

- В отношении нет одинаковых кортежей.
- Кортежи не упорядочены.
- Атрибуты не упорядочены и различаются по наименованию.
- Все значения атрибутов атомарные.

В ходе логического моделирования на первом шаге данные размещают в одном отношении, имеющем следующие атрибуты:

СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ (Н_СОТР, ФАМ, Н_ОТД, ТЕЛ, Н_ПРО, ПРОЕКТ, Н_ЗАДАН)

где: Н_СОТР - табельный номер сотрудника; ФАМ - фамилия сотрудника; Н_ОТД - номер отдела, в котором числится сотрудник; ТЕЛ - телефон сотрудника; Н_ПРО - номер проекта, над которым работает сотрудник; ПРОЕКТ - наименование проекта, над которым работает сотрудник; Н_ЗАДАН - номер задания, над которым работает сотрудник

Т.к. каждый сотрудник в каждом проекте выполняет ровно одно задание, то в качестве потенциального ключа отношения необходимо взять пару атрибутов {Н_СОТР, Н_ПРО}.

В текущий момент состояние предметной области отражается следующими фактами:

- Сотрудник Иванов, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 1 и во втором проекте "Климат" задание 1.
- Сотрудник Петров, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 2.
- Сотрудник Сидоров, работающий во 2 отделе, выполняет в первом проекте "Космос" задание 3 и во втором проекте "Климат" задание 2.

Это состояние отражается в таблице (выделены ключевые атрибуты):

Таблица 1. Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ	**Н_ПРО**	ПРОЕКТ	Н_ЗАДАН
1	Иванов	1	11-22-33	1	Космос	1
1	Иванов	1	11-22-33	2	Климат	1
2	Петров	1	11-22-33	1	Космос	2
3	Сидоров	2	33-22-11	1	Космос	3
3	Сидоров	2	33-22-11	2	Климат	2

2. Аномалии обновления.

При анализе таблицы отношения СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ, можно видеть, что данные хранятся в ней с большой избыточностью. Во многих строках повторяются фамилии сотрудников, номера телефонов, наименования проектов. Кроме того, в данном отношении хранятся вместе независимые друг от друга данные - и данные о сотрудниках, и об отделах, и о проектах, и о работах по проектам. Пока никаких действий с отношением не производится, это не страшно, Но, как только состояние предметной области изменяется, то, при попытках соответствующим образом изменить состояние базы данных, возникают разные проблемы.

Исторически эти проблемы получили название аномалии обновления. Аномалии можно определить как противоречие между моделью предметной области и физической моделью данных, поддерживаемых средствами конкретной СУБД. Более строгое определения понятия аномалий выходит за пределы данного курса.

Т.к. аномалии проявляют себя при выполнении операций, изменяющих состояние базы данных, то различают следующие виды аномалий:

- Аномалии вставки (INSERT)
- Аномалии обновления (UPDATE)
- Аномалии удаления (DELETE)

В отношении СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ можно привести примеры следующих аномалий:

2.1. Аномалии вставки (INSERT).

В отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ нельзя вставить данные о сотруднике, который пока не участвует ни в одном проекте. Действительно, если, например, во втором отделе появляется новый сотрудник, скажем, Пушников, и он пока не участвует ни в одном проекте, то мы должны вставить в отношение кортеж (4, Пушников, 2, 33-22-11, null, null, null). Это сделать невозможно, т.к. атрибут Н_ПРО (номер проекта) входит в состав потенциального ключа, и, следовательно, не может содержать null-значений.

А также нельзя вставить данные о проекте, над которым пока не работает ни один сотрудник.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

2.2. Аномалии обновления (UPDATE).

Фамилии сотрудников, наименования проектов, номера телефонов повторяются во многих кортежах отношения. Поэтому если сотрудник меняет фамилию, или проект меняет наименование, или меняется номер телефона, то такие изменения необходимо одновременно выполнить во всех местах, где эта фамилия, наименование или номер телефона встречаются, иначе отношение станет некорректным (например, один и тот же проект в разных кортежах будет называться по-разному). Таким образом, обновление базы данных одним действием реализо-

вать невозможно. Для поддержания отношения в целостном состоянии необходимо написать триггер, который при обновлении одной записи исправлял данные и в других местах.

Причина аномалии - избыточность данных, также порожденная тем, что в одном отношении хранится разнородная информация.

Вывод - увеличивается сложность разработки базы данных. База данных, основанная на такой модели, будет работать правильно только при наличии дополнительного программного кода в виде триггеров.

2.3. Аномалии удаления (DELETE).

При удалении некоторых данных может произойти потеря другой информации. Например, если закрыть проект "Космос" и удалить все строки, в которых он встречается, то будут потеряны все данные о сотруднике Петрове. Если удалить сотрудника Сидорова, то будет потеряна информация о том, что в отделе номер 2 находится телефон 33-22-11. Если по проекту временно прекращены работы, то при удалении данных о работах по этому проекту будут удалены и данные о самом проекте (наименование проекта). При этом если был сотрудник, который работал только над этим проектом, то будут потеряны и данные об этом сотруднике.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и о проектах, и о работах по проекту).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

3. Функциональные зависимости.

Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ находится в 1НФ, при этом, как было показано выше, логическая модель данных не адекватна модели предметной области. Таким образом, первой нормальной формы недостаточно для правильного моделирования данных.

Для устранения указанных аномалий (а на самом деле для правильного проектирования модели данных!) применяется метод нормализации отношений. Нормализация основана на понятии функциональной зависимости атрибутов отношения.

3.1. Определение понятия функциональной зависимости.

Определение. Пусть R - отношение. Множество атрибутов Y функционально зависимо от множества атрибутов X (X функционально определяет Y) тогда и только тогда, когда для любого состояния отношения R для любых кортежей r1,r2 $\in$ R из того, что r1X=r2Y следует что r1Y=r2Y (т.е. во всех кортежах, имеющих одинаковые значения атрибутов X, значения атрибутов Y также совпадают в любом состоянии отношения R). Символически функциональная зависимость записывается как X$\rightarrow$Y.

Множество атрибутов X называется детерминантом функциональной зависимости, а множество атрибутов Y называется зависимой частью.

Замечание 1. Если атрибуты X составляют потенциальный ключ отношения R, то любой атрибут отношения R функционально зависит от X.

Пример. В отношении СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ можно привести следующие примеры функциональных зависимостей:

- Зависимость атрибутов от ключа отношения:
  {Н_СОТР,Н_ПРО}→ФАМ; {Н_СОТР,Н_ПРО}→Н_ОТД; {Н_СОТР,Н_ПРО}→ТЕЛ;
  {Н_СОТР,Н_ПРО}→ПРОЕКТ; {Н_СОТР,Н_ПРО}→Н_ЗАДАН
- Зависимость атрибутов, характеризующих сотрудника от таб. номера сотрудника:
  Н_СОТР→ФАМ; Н_СОТР→Н_ОТД; Н_СОТР→ТЕЛ
- Зависимость наименования проекта от номера проекта:
  Н_ПРО→ПРОЕКТ
- Зависимость номера телефона от номера отдела:
  Н_ОТД→ТЕЛ

Замечание 2. Приведенные функциональные зависимости не выведены из внешнего вида отношения, приведенного в таблице 1. Эти зависимости отражают взаимосвязи, обнаруженные между объектами предметной области и являются дополнительными ограничениями, определяемыми предметной областью. Таким образом, функциональная зависимость - семантическое понятие. Она возникает, когда по значениям одних данных в предметной области можно определить значения других данных. Например, зная табельный номер сотрудника, можно определить его фамилию, по номеру отдела можно определить телефона. Функциональная зависимость задает дополнительные ограничения на данные, которые могут храниться в отношениях. Для корректности базы данных (адекватности предметной области) необходимо при выполнении операций модификации базы данных проверять все ограничения, определенные функциональными зависимостями.

3.2. Отличие от математического понятия функциональная зависимости.

Функциональная зависимость атрибутов отношения напоминает понятие функциональной зависимости в математике. Но это не одно и то же.

Если рассматривать математическое понятие функции, то для фиксированного значения x соответствующее значение функции y=f(x) всегда одно и то же. Например, если задана функция y=x^2, то для значения x=2 соответствующее значение y всегда будет равно 4. В противоположность этому в отношениях значение зависимого атрибута может принимать различные значения в различных состояниях базы данных и, самое главное, эти значения могут меняться непредсказуемо.

Например, атрибут ФАМ функционально зависит от атрибута Н_СОТР. Предположим, что сейчас сотрудник с табельным номером 1 имеет фамилию Иванов, т.е. при значении детерминанта равного 1, значение зависимого аргумента равно "Иванов". Но сотрудник может сменить фамилию, например на "Сидоров". Теперь при том же значении детерминанта, равного 1, значение зависимого аргумента равно "Сидоров".

Функциональная зависимость атрибутов утверждает лишь то, что для каждого конкретного состояния базы данных по значению одного атрибута (детерминанта) можно однозначно определить значение другого атрибута (зависимой части). Но конкретные значение зависимой части могут быть различны в различных состояниях базы данных.

4. 2НФ (Вторая Нормальная Форма).

Определение 2НФ. Отношение R находится во второй нормальной форме (2НФ) тогда и только тогда, когда отношение находится в 1НФ и нет неключевых атрибутов, зависящих от части сложного (составного) ключа. (Неключевой атрибут - это атрибут, не входящий в состав никакого потенциального ключа).

Замечание. Если потенциальный ключ отношения является простым, то отношение автоматически находится во 2НФ.

Пример. Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ не находится в 2НФ, т.к. есть атрибуты, зависящие от части сложного (составного) ключа:

- Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника является зависимостью от части сложного ключа: Н_СОТР→ФАМ; Н_СОТР→Н_ОТД; Н_СОТР→ТЕЛ
- Зависимость наименования проекта от номера проекта является зависимостью от части сложного ключа: Н_ПРО→ПРОЕКТ

Для того, чтобы устранить такие зависимости нужно произвести декомпозицию отношения на несколько отношений. При этом те атрибуты, которые зависят от части сложного ключа, выносятся в отдельное отношение.

Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ декомпозируем на три отношения - СОТРУДНИКИ_ОТДЕЛЫ, ПРОЕКТЫ, ЗАДАНИЯ.

Отношение СОТРУДНИКИ_ОТДЕЛЫ (Н_СОТР, ФАМ, Н_ОТД, ТЕЛ):

Таблица 2 Отношение СОТРУДНИКИ_ОТДЕЛЫ

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Функциональные зависимости:

- Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника: Н_СОТР→ФАМ; Н_СОТР→Н_ОТД; Н_СОТР→ТЕЛ
- Зависимость номера телефона от номера отдела: Н_ОТД→ТЕЛ

Отношение ПРОЕКТЫ (Н_ПРО, ПРОЕКТ):

Таблица 3 Отношение ПРОЕКТЫ

Н_ПРО	ПРОЕКТ
1	Космос
2	Климат

Функциональные зависимости: Н_ПРО→ПРОЕКТ

Отношение ЗАДАНИЯ (Н_СОТР, Н_ПРО, Н_ЗАДАН):

Таблица 4 Отношение ЗАДАНИЯ

Н_СОТР	Н_ПРО	Н_ЗАДАН
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

Функциональные зависимости: {Н_СОТР, Н_ПРО}→Н_ЗАДАН

4.1. Анализ декомпозированных отношений.

Отношения, полученные в результате декомпозиции, находятся в 2НФ. Действительно, отношения СОТРУДНИКИ_ОТДЕЛЫ и ПРОЕКТЫ имеют простые ключи, следовательно автоматически находятся в 2НФ, отношение ЗАДАНИЯ имеет сложный ключ, но единственный неключевой атрибут Н_ЗАДАН функционально зависит от всего ключа {Н_СОТР, Н_ПРО}.

4.1.1. Часть аномалий обновления устранена:

- INSERT. Так, данные о сотрудниках и проектах теперь хранятся в различных отношениях, поэтому при появлении сотрудника, не участвующего ни в одном проекте просто добавляется кортеж в отношение СОТРУДНИКИ_ОТДЕЛЫ. Точно также, при появлении проекта, над которым не работает ни один сотрудник, просто вставляется кортеж в отношение ПРОЕКТЫ.
- UPDATE. Фамилии сотрудников и наименования проектов теперь хранятся без избыточности. Если сотрудник сменит фамилию или проект сменит наименование, то такое обновление будет произведено в одном месте.
- DELETE. Если по проекту временно прекращены работы, но требуется, чтобы сам проект сохранился, то для этого проекта удаляются соответствующие кортежи в отношении ЗАДАНИЯ, а данные о самом проекте и данные о сотрудниках, участвовавших в проекте, остаются в отношениях ПРОЕКТЫ и СОТРУДНИКИ_ОТДЕЛЫ.

Тем не менее, часть аномалий разрешить не удалось.

4.1.2. Оставшиеся аномалии вставки (INSERT)

В отношение СОТРУДНИКИ_ОТДЕЛЫ нельзя вставить кортеж (4, Пушников, 1, 33-22-11), т.к. при этом получится, что два сотрудника из 1-го отдела (Иванов и Пушников) имеют разные номера телефонов, а это противоречит модели предметной области. В этой ситуации можно предложить два решения, в зависимости от того, что реально произошло в предметной области. Другой номер телефона может быть введен по двум причинам - по ошибке человека, вводящего данные о новом сотруднике, или потому что номер в отделе действительно изменился.

Тогда можно написать триггер, который при вставке записи о сотруднике проверяет, совпадает ли телефон с уже имеющимся телефоном у другого сотрудника этого же отдела. Если номера отличаются, то система должна задать вопрос, оставить ли старый номер в отделе или заменить его новым. Если нужно оставить старый номер (новый номер введен ошибочно), то кортеж с данными о новом сотруднике будет вставлен, но номер телефона будет у него будет тот, который уже есть в отделе (в данном случае, 11-22-33). Если же номер в отделе действительно изменился, то кортеж будет вставлен с новым номером, и одновременно будут изменены номера телефонов у всех сотрудников этого же отдела. И в том и в другом случае не обойтись без разработки громоздкого триггера.

Причина аномалии - избыточность данных, порожденная тем, что в одном отношении хранится разнородная информация (о сотрудниках и об отделах).

Вывод - увеличивается сложность разработки базы данных. База данных, основанная на такой модели, будет работать правильно только при наличии дополнительного программного кода в виде триггеров.

4.1.3. Оставшиеся аномалии обновления (UPDATE)

Одни и те же номера телефонов повторяются во многих кортежах отношения. Поэтому если в отделе меняется номер телефона, то такие изменения необходимо одновременно выполнить во всех местах, где этот номер телефона встречаются, иначе отношение станет некорректным. Таким образом, обновление базы данных одним действием реализовать невозможно. Необходимо написать триггер, который при обновлении одной записи корректно исправляет номера телефонов в других местах.

Причина аномалии - избыточность данных, также порожденная тем, что в одном отношении хранится разнородная информация.

Вывод - увеличивается сложность разработки базы данных. База данных, основанная на такой модели, будет работать правильно только при наличии дополнительного программного кода в виде триггеров.

4.1.4. Оставшиеся аномалии удаления (DELETE)

При удалении некоторых данных по-прежнему может произойти потеря другой информации. Например, если удалить сотрудника Сидорова, то будет потеряна информация о том, что в отделе номер 2 находится телефон 33-22-11.

Причина аномалии - хранение в одном отношении разнородной информации (и о сотрудниках, и об отделах).

Вывод - логическая модель данных неадекватна модели предметной области. База данных, основанная на такой модели, будет работать неправильно.

Заметим, что при переходе ко второй нормальной форме отношения стали почти адекватными предметной области. Остались также трудности в разработке базы данных, связанные с необходимостью написания триггеров, поддерживающих целостность базы данных. Эти трудности теперь связаны только с одним отношением СОТРУДНИКИ_ОТДЕЛЫ.

5. 3НФ (Третья Нормальная Форма).

Определение 3НФ. Отношение R находится в третьей нормальной форме (3НФ) тогда и только тогда, когда отношение находится в 2НФ и все неключевые атрибуты взаимно независимы. Атрибуты называются взаимно независимыми, если ни один из них не является функционально зависимым от другого.

Отношение СОТРУДНИКИ_ОТДЕЛЫ не в 3НФ, т.к. имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела): Н_ОТД$\rightarrow$ТЕЛ.

Для того, чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения на несколько отношений. При этом те неключевые атрибуты, которые являются зависимыми, выносятся в отдельное отношение.

Отношение СОТРУДНИКИ_ОТДЕЛЫ декомпозируем на два: СОТРУДНИКИ, ОТДЕЛЫ.

Отношение СОТРУДНИКИ (Н_СОТР, ФАМ, Н_ОТД):

Таблица 5. Отношение СОТРУДНИКИ

Н_СОТР	ФАМ	Н_ОТД
1	Иванов	1
2	Петров	1
3	Сидоров	2

Функциональные зависимости: Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника: Н_СОТР→ФАМ; Н_СОТР→Н_ОТД; Н_СОТР→ТЕЛ

Отношение ОТДЕЛЫ (Н_ОТД, ТЕЛ):

Таблица 6. Отношение ОТДЕЛЫ

Н_ОТД	ТЕЛ
1	11-22-33
2	33-22-11

Функциональные зависимости: номер телефона от номера отдела: Н_ОТД→ТЕЛ.

Обратим внимание на то, что атрибут Н_ОТД, не являвшийся ключевым в отношении СОТРУДНИКИ_ОТДЕЛЫ, становится потенциальным ключом в отношении ОТДЕЛЫ. Именно за счет этого устраняется избыточность, связанная с многократным хранением одних и тех же номеров телефонов.

Вывод. Таким образом, все обнаруженные аномалии обновления устранены. Реляционная модель, состоящая из четырех отношений СОТРУДНИКИ, ОТДЕЛЫ, ПРОЕКТЫ, ЗАДАНИЯ, находящихся в третьей нормальной форме, является адекватной описанной модели предметной области, и требует наличия только тех триггеров, которые поддерживают ссылочную целостность. Такие триггеры являются стандартными и не требуют больших усилий в разработке.

6. Алгоритм нормализации (приведение к 3НФ).

Алгоритм нормализации (приведения отношений к 3НФ) описывается следующим образом.

Шаг 1 (Приведение к 1НФ).

На первом шаге задается одно или несколько отношений, отображающих понятия предметной области. По модели предметной области (не по внешнему виду полученных отношений!) выписываются обнаруженные функциональные зависимости. Все отношения автоматически находятся в 1НФ (если они соответствуют определению понятию отношения):

- В отношении нет одинаковых кортежей.
- Кортежи не упорядочены.
- Атрибуты не упорядочены и различаются по наименованию.
- Все значения атрибутов атомарные.

Шаг 2 (Приведение ко 2НФ).

Если в некоторых отношениях обнаружена зависимость атрибутов от части сложного ключа, то проводим декомпозицию этих отношений на несколько отношений следующим образом: те атрибуты, которые зависят от части сложного ключа выносятся в отдельное отношение вместе с этой частью ключа. В исходном отношении остаются все ключевые атрибуты:

Исходное отношение:

R(K1,K2,A1,...,An,B1,...,Bm). Ключ: {K1,K2} - сложный.

Функциональные зависимости:

{K1,K2} → {A1,...,An,B1,...,Bm} - зависимость всех атрибутов от ключа отношения.
{K1}→{A1,…,An} - зависимость некоторых атрибутов от части сложного ключа.

Декомпозированные отношения:

R1(K1,K2,B1,...,Bm) - остаток от исходного отношения. Ключ {K1,K2}.
R2(K1,A1,...,An) - атрибуты и часть сложного ключа, вынесенные из исходного отношения. Ключ K1.

Шаг 3 (Приведение к 3НФ).

Если в некоторых отношениях обнаружена зависимость некоторых неключевых атрибутов от других неключевых атрибутов, то проводим декомпозицию этих отношений следующим образом: те неключевые атрибуты, которые зависят от других неключевых атрибутов выносятся в отдельное отношение. В новом отношении ключом становится детерминант (определителем) функциональной зависимости:

Исходное отношение:

R(K,A1,...,An,B1,...,Bm). Ключ: K.

Функциональные зависимости:

K→{A1,...,An,B1,...,Bm} - зависимость всех атрибутов от ключа отношения.
{A1,...,An}→{B1,...,Bm} - зависимость некоторых неключевых атрибутов от других неключевых атрибутов.

Декомпозированные отношения:

R1(K,A1,...,An) - остаток от исходного отношения. Ключ K.
R2(A1,...,An,B1,...,Bm) - атрибуты, вынесенные из исходного отношения вместе с детерминантом функциональной зависимости. Ключ {A1,...,An}.

7. Заключение по нормализации.

Замечание. На практике, при создании логической модели данных, как правило, не следуют прямо приведенному алгоритму нормализации. Опытные разработчики обычно сразу строят отношения в 3НФ. Кроме того, основным средством разработки логических моделей данных являются различные варианты ER-диаграмм. Особенность этих диаграмм в том, что они сразу позволяют создавать отношения в 3НФ или в НФБК.

Тем не менее, приведенный алгоритм важен по двум причинам.

- Во-первых, этот алгоритм показывает, какие проблемы возникают при разработке слабо нормализованных отношений.
- Во-вторых, как правило, модель предметной области никогда не бывает правильно разработана с первого шага. Эксперты предметной области могут забыть о чем-либо упомянуть, разработчик может неправильно понять эксперта, во время разработки могут измениться правила, принятые в предметной области, и т.д. Все это может привести к появлению новых зависимостей, которые отсутствовали в первоначальной модели предметной области.
- Вывод. Нормализация нужна чтобы убедиться, что отношения остались в 3НФ и логическая модель не ухудшилась, т.е. для проверки реляционной модели полученной из ER-модели.

7.1. Сравнение нормализованных и ненормализованных моделей данных.

Результаты анализа критериев, по которым оценивают влияние логического моделирования данных на качество физических моделей данных и производительность базы данных:

Критерий	Отношения слабо нормализованы (0НФ,1НФ, 2НФ)	Отношения сильно нормализованы (3НФ и выше)
Адекватность базы данных предметной области	ХУЖЕ (-)	ЛУЧШЕ (+)
Легкость разработки и сопровождения базы данных	СЛОЖНЕЕ (-)	ЛЕГЧЕ (+)
Скорость выполнения вставки, обновления, удаления	МЕДЛЕННЕЕ (-)	БЫСТРЕЕ (+)
Скорость выполнения выборки данных	БЫСТРЕЕ (+)	МЕДЛЕННЕЕ (-)

Как видно из таблицы, более сильно нормализованные отношения оказываются лучше спроектированы (три плюса, один минус). Они больше соответствуют предметной области, легче в разработке, для них быстрее выполняются операции модификации базы данных. Правда, это достигается ценой некоторого замедления выполнения операций выборки данных.

У слабо нормализованных отношений единственное преимущество - если к базе данных обращаться только с запросами на выборку данных, то для слабо нормализованных отношений

такие запросы выполняются быстрее. Это связано с тем, что в таких отношениях уже как бы произведено соединение отношений и на это не тратится время при выборке данных. Для часто используемых запросов в высокопроизводительных БД могут проводить процесс частичной **денормализации** отношений.

Таким образом, выбор степени нормализации отношений зависит от характера запросов, с которыми чаще всего обращаются к базе данных.

> Отношения, находящиеся в 1НФ являются "плохими" в том смысле, что они не удовлетворяют выбранным критериям - имеется большое количество аномалий обновления, для поддержания целостности базы данных требуется разработка сложных триггеров.

Отношение R находится во второй нормальной форме (2НФ) тогда и только тогда, когда отношение находится в 1НФ и нет неключевых атрибутов, зависящих от части сложного ключа.

> Отношения в 2НФ "лучше", чем в 1НФ, но еще недостаточно "хороши" - остается часть аномалий обновления, по-прежнему требуются триггеры, поддерживающие целостность базы данных.

Отношение R находится в третьей нормальной форме (3НФ) тогда и только тогда, когда отношение находится в 2НФ и все неключевые атрибуты взаимно независимы.

> Отношения в 3НФ являются самыми "хорошими" с точки зрения выбранных нами критериев - устранены аномалии обновления, требуются только стандартные триггеры для поддержания ссылочной целостности.

7.2. Нормальные формы более высоких порядков.

Переход от ненормализованных отношений к отношениям в 3НФ может быть выполнен при помощи алгоритма нормализации.

Алгоритм нормализации заключается в последовательной декомпозиции отношений для устранения функциональных зависимостей атрибутов от части сложного ключа (приведение к 2НФ) и устранения функциональных зависимостей неключевых атрибутов друг от друга (приведение к 3НФ).

Нормализация схемы БД до третьей нормальной формы (3НФ) в большинстве случаев вполне достаточно, для разработки вполне работоспособные базы данных.

Но, существуют нормальные формы более высоких порядков, устраняющие оставшиеся аномалии, зависимости и избыточности за счёт дальнейшей декомпозиции отношений.

Это нормальная форма Бойса-Кодда (НФБК), четвертая нормальная форма (4НФ) и пятая нормальная форма (5НФ).

Нормальная форма Бойса-Кодда (НФБК).

НФБК стоит особняком от всех НФ. Для того, чтобы отношение находилось в НФБК не нужно, чтобы оно было в 3НФ.

Если в отношении имеется несколько потенциальных ключей, то оно не подходит под формулировку 3НФ, т.к. не все атрибуты зависят только от первичного ключа, они могут зависеть и от потенциальных ключей.

Для таких отношений и была задумана НФБК, которая гласит: отношение находится в НФБК в том случае, если все атрибуты находятся в функциональной зависимости от потенциального ключа.

Например, в отношении определены следующие атрибуты: №п/п, № зачетной книжки, ФИО студента, оценка. В данном отношении определен ПК - №п/п. Атрибуты ФИО и оценка зависят как от ПК, так и от № зачетной книжки. Следовательно данное отношение не находится в 3НФ, оно находится в НФБК, т.к. атрибут № зачетной книжки может быть выбран в качестве ПК, т.е. является потенциальным ключом.

Нормализацию отношений выше НФБК используют очень редко. Чем выше степень нормализации, тем больше таблиц будет получено, а это усложнит работу с БД.