**Лекция 5:**

1. Функциональные зависимости. Нормализация: формы 1НФ, 2НФ, 3НФ. Декомпозиция без потерь.

**Практика по НФ:**

* + привести пример функциональной зависимости
  + всегда ли реляционная база данных находится в 1НФ
  + привести пример аномалий добавления, удаления, обновления для ненормализованных БД
  + обсудить, как нормализация влияет на скорость работы
  + практикум по нормализации схемы БД
  + случаи, когда требуется **денормализация.**

**Лекция 5:**

Теперь давайте поговорим о том, как в теории нужно правильно проектировать БД.

## Нормальные формы

*Упражнение 1:* Вспомнить определение первичного, потенциального и альтернативного ключа. Привести примеры ключей.

Зачем нужна нормализация данных?

Под нормализацией понимается разнесение атрибутов из одной большой таблицы в несколько меньших без потерь информации. Необходимость нормализации обоснована тем, что в таком случае данные сохраняют целостность и корректность, так как хранятся после нормализации только в одном отношении базы, отсутствует избыточность данных (или неполнота в том числе).

В неудачно спланированной базе данных возможны различные проблемы с хранением и обработкой данных: аномалии удаления, обновления, вставки. Конечной целью нормализации является получение проекта БД, в которой исключена избыточность и противоречивость информации. Все аномалии работы с данными приведем далее по мере разбора нормальных форм.

В теории реляционных БД обычно выделяются шесть видов нормальных форм (1нф, 2нф, 3нф, нф Бойса-Кодда, 4нф, 5нф + Еще есть DKNF), но на самом деле их намного больше, указанные формы являются наиболее популярными. Мы рассмотрим первые три. Да и обычно БД выше, чем 3НФ не приводят, так как в 3НФ уже практически все важные аномалии исключены.

## 1НФ.

В первой нормальной форме значения всех атрибутов отношения должны быть атомарными (т.е. значения элементов в домене не являются ни списками, ни множествами).

Таблица находится в 1НФ, если она удовлетворяет условию (свойства отношений):

* + В отношении нет одинаковых кортежей.
  + Кортежи не упорядочены.
  + Атрибуты не упорядочены и различаются по наименованию.
  + Все значения атрибутов **атомарны**.

*Замечание: Пример отношения не в 1нф.*

*Допустим, у нас есть желаемые атрибуты: Фамилия, Имя, Отчество, а в отношении в базе заведен атрибут ФИО. Таким образом, данный атрибут не будет атомарным.*

*Либо база имеет отношение к хранению телефонных номеров, но список телефонных номеров хранится в полях таблицы через запятую и тд.*

Пример данных не в 1НФ (ключ ндок):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Ндок | Поставщик | Товар | Город |
| 15.10.15 | 1 | Иванов | Мышь к = 15, ц = 20; Монитор к = 5, ц = 600 | Долгопрудный |
| 22.10.15 | 2 | Петров | Монитор к = 3, ц = 500; Клавиатура к = 20, ц = 35 | Москва |

Проблемы:

Аномалия выборки (Select)

Пока мы не привели хотя бы к 1НФ – невозможно посчитать запросом физическое количество поставляемого товара.

Аномалия включения (Insert):

Если у нас появится новый поставщик, который на текущий момент еще ничего не поставил, то мы не сможем включить его в нашу таблицу, так как у нас нет сведений о товарах им поставляемых.

Аномалия удаления (Delete):

Если мы удалим заказ № 1 (перенесем в архив), то вся информация про поставщика «Иванова» у нас потеряется.

Аномалия обновления (Update):

Если через некоторое время после работы с поставщиком «Сидоровым», поставщик сменил свое название с «Сидорова» на «Сидорова И.И.», то часть записей, связанных с Сидоровым «Поставщик» будет иметь значение «Сидоров», а часть «Сидоров И.И.».

Если мы захотим переименовать Сидорова во всех строках, то наш запрос будет менять большое количество данных (за всё время закупки у Сидорова).

Если мы хотим обновить наименование товара 1 (Мышь, например), то, как мы это будем делать? (Структура запроса для не 1 НФ?)

Здесь, кроме того, возникает вопрос, а действительно ли мы должны изменить название поставщика/название товара во всех строках?

Как привести такие данные к 1НФ? Для этого нужно разнести данные из списка по отдельным столбцам.

*Упражнение 2:* Пусть есть таблица со следующими данными (жирным шрифтом отмечены ключи), считаем, что каждый поставщик может быть только в одном городе:

**НомерДокумента**

Дата

**Товар**

Поставщик

Город

Колво

Цена

Объем

Масса

Почему эти поля выбраны в качестве ключевых? Изменится ли колво строк в новой таблице? Согласны ли Вы с тем, что эта таблица уже находится в 1 НФ?

Это новая схема данных для исходной таблицы. Здесь артикул, название товара, колво и цена разнесены по разным столбцам, а разные товары в одном документе разнесены в разные строки.

Кроме того, здесь добавлены дополнительные столбцы, которые мы будем использовать в дальнейшей нормализации.

Описание проблем, связанных с такой структурой: Остались аномалии удаления, обновления и вставки. С аномалией выборки мы разобрались.

Дополнение. Также характеристики товара, такие как объем и масса будут повторяться в базе столько раз, сколько данный товар встречался в базе. Аналогичная ситуация с атрибутом «Поставщик» и «Город».

Введем понятие функциональной зависимости. **Функциональная зависимость** обозначает, что по значению одного атрибута всегда можно однозначно определить другой. Например, зная наименование товара можно однозначно определить его массу, объем. Понятно, что в обратную сторону зависимость необязательная, например, монитор и принтер могут весить одинаково, поэтому зависимость масса -> товар – не соответствует действительности, в то время как, конкретный монитор может иметь только одну определенную массу. (Пример функциональной зависимости в мат анализе: на функции y=x\*x; y=4 - >x=+-2). Детерминант зависимости – та часть функциональной зависимости, от которой зависит зависимая часть.

*Упражнение 3:* привести пример функциональной зависимости.

## 2НФ.

Для простоты понимания будем считать, что в таблице у нас один потенциальный ключ, он же первичный ключ отношения.

Отношение находится во 2НФ, тогда и только тогда, когда она находится в 1НФ, и каждый неключевой атрибут зависит полностью от ее первичного ключа. То есть, если для столбцов X и Y первый является ключом, а Z – является подмножеством X, то ситуации, когда Z -> Y, быть не может.

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

*Упражнение 4*: приведение схемы из упр.2 к 2НФ.

Видим, что в нашем случае:

#### НомерДокумента

Дата

Поставщик

Город

#### НомерДокумента

**Товар**

Колво

Цена

## Товар

Объем

Масса

Какие проблемы остаются при такой структуре хранения информации?

Видим, что все так же осталась проблема со сменой названия поставщика. И нельзя завести нового поставщика без заказа. Если удалить заказ N, то информация о поставщике K потеряется. Как провести переименования объектов?

## 3НФ.

3НФ – отношение находится в 2НФ и не содержит транзитивных зависимостей неключевых атрибутов от какого либо ключа.

Если в некоторых отношениях обнаружена зависимость одних неключевых атрибутов от других неключевых атрибутов, то проводим декомпозицию этих отношений: те неключевые атрибуты, которые зависят от других неключевых атрибутов, выносятся в отдельное отношение.

В нашем случае есть зависимость:

НомерДокумента -> Поставщик, Поставщик -> Город;

НомерДокумента -> Город.

Поэтому отношение Поставщик -> Город необходимо вынести в отдельное отношение.

Ключом становится детерминант зависимости «Поставщик».

#### НомерДокумента

Дата

Поставщик

#### НомерДокумента

**Товар**

Колво

Цена

## Товар

Объем

Масса

## Поставщик

Город

Понятно, что декомпозиция должна проводится без потерь информации. (см. Теорема Хеза)

Кроме того, название у товара может меняться в процессе работы, то есть товар оставаться тем же самым, а название меняться (незначительно). Соответственно для учета таких случаев нужно завести идентификатор товара. Кроме того, для городов возможна другая ситуация, когда один и тот же город может писаться по-разному. Например, Иванов и Сидоров оба из Санкт-Петербурга, но для Иванова город указан – «Питер», а для Сидорова «Санкт-Петербург», соответственно, аналитик при дальнейшем анализе базы не сможет без дополнительных усилий сказать, что Петров и Сидоров из одного города. Поэтому предлагается следующая схема:

#### НомерДокумента

Дата

Поставщик\_ID

#### НомерДокумента

**Товар\_ID**

Колво

Цена

## Товар\_ID

Товар

**Товар\_ID**

Объем

Масса

## Поставщик\_ID

Поставщик

## Поставщик\_ID

Город\_ID

**Город\_ID**

Город

*Коммент: здесь есть таблицы, которые связаны друг с другом 1-к-1, например Товар\_ID – Товар и Товар\_ID-Объем-Масса. При выполнении ДЗ можете указывать эти поля в одной таблице, с упоминанием о том, что это не совсем 3НФ, так как есть транзитивная зависимость. То есть покажите, что Вы понимаете, поэтому это не совсем 3НФ.*

*Упражнение 5*: В данной схеме возможен следующий недочет – когда, например, поставщик проживает в нескольких городах. В данной схеме такую ситуацию реализовать нельзя. Почему?

**Задание 1:**

Предложите схему хранения полиномов от нескольких переменных в базе данных.

**Пример для самостоятельной работы у доски:**

База «Персонала», необходимая, например, для службы безопасности компании.

Какую информацию о человеке необходимо хранить:

ФИО (возможно ФИО менялись)

Пол

Дата рождения

Место рождения

Номер, серия паспорта, когда, где и кем выдан

(учесть, что паспортов может быть за жизнь человека несколько, кроме того может быть загран паспорт или свидетельство о рождении)

Образование, номер документа об образовании (аттестат, дипломы и тд)

Адреса проживания (постоянная прописка, регистрация)

Родственники (родители, братья, сестры, мужья, жены, дети…)

Во-первых, необходимо завести Идентификатор человека (хотя бы внутрифирменный), так как номер паспорта не является определяющим атрибутом, хотя бы по причине того, что в 20 лет (45 лет, по утере) паспорт у человека меняется.

Для хранения документов с учетом возможности их изменения в течение жизни человека, для порядкового номера паспорта в жизни человека нужно завести Версия\_ID (порядковый номер документа определенного типа в жизни человека), кроме того, для нормализации учета документов, можно завести поле «тип\_документа» (гражданский паспорт, загранпаспорт, аттестат, диплом).

# Идентификатор человека

**Версия\_ФИО\_ID (здесь ФИО не разделено отдельно на Ф, И и О, также остается для сам.работы)**

ФИО (возможно ФИО менялись)

Пол

Дата рождения

Место рождения

**Тип\_документа\_ID (паспорт, загранпаспорт, свидетельство о рождении)**

**Версия\_документа\_ID**

Номер, серия паспорта, когда, где и кем выдан

**Тип\_образования\_ID (школа, институт, второе высшее, специальное)**

Образование, номер документа об образовании

**Тип\_адреса\_ID (постоянный, регистрация)**

**Версия\_адреса\_ID**

Адреса проживания (постоянная прописка, регистрация)

**Тип\_связи\_ID (жена, муж, ребенок, отец, мать, сестра и тд)**

**Версия\_связи\_ID**

Родственники (список указываемых родственников)

Далее, такое отношение, находящееся уже в 1НФ достаточно просто привести к 2НФ, вынося части зависящие от части ключа, например:

## Идентификатор

**Тип\_документа\_ID (паспорт, загранпаспорт, свидетельство о рождении)**

**Версия\_документа\_ID (порядковый номер документа в жизни человека)**

Номер

серия паспорта

когда выдан

где выдан

кем выдан

Действителен С

Действителен По

Процесс приведения к 3НФ остается для самостоятельной работы.

## Денормализация – действие обратное нормализации. Когда необходимо? Необходимо в том случае, когда использование 1НФ упростит составление и выполнение запросов к БД.

Подумать, в каком случае запросы на выборку будут работать быстрее? Для 1НФ или 3НФ?

Пример: Покупатель изменил адрес, но через месяц попросил перепечатать ему старые заказы с указанием старого адреса. Что в этом случае делать?

Типы связей между таблицами (было на 1-м занятии)

**Один-к-одному** – каждой строке одной таблицы соответствует одна строка (или ни одной) второй таблицы. С другой стороны, каждая строка второй таблицы должна быть связана только с одной строкой первой таблицы.

Пример: таблица «Товар» (поле Товар\_ID – первичный ключ) и таблица «Склад», в которой, например, хранится информация о текущем остатке на складе (поле товар – первичный ключ).

**Один-ко-многим** – каждой строке первой таблицы соответствует ноль, одна или более строк второй таблицы. С другой стороны, каждая строка второй таблицы должна быть связана только с одной строкой первой таблицы.

Пример: таблица «Товары» и таблица «Документы\_данные». Частным случаем связи «один-ко-многим» является связь «один-к-одному».

**Многие-ко-многим** – любой строке первой таблицы может соответствовать ноль, одна или несколько строк второй таблицы. При этом каждая строка второй таблицы может быть связана с одной или более строк первой таблицы. Связь многие-ко-многим реализуется с помощью связывающей таблицы.

Пример: таблица «Документы» и таблица «Товары» по полям товар-ндок через промежуточную таблицу «Документы\_данные», или таблица «Товары» и таблица «Покупатели» через промежуточные таблицы «Документы» и «Документы\_данные» по полям «товар» и «покупатель».

Примерный набор заголовков для базы данных «Продажи»:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Документы** |  | **Документы\_данные** |  | **Товары** |  | **Покупатели** |
| **ндок** |  | **ндок** |  | **Товар\_ID** |  | **Покупатель\_ID** |
| Дата |  | **Товар\_ID** |  | Наименование |  | ФИО |
| Покупатель\_ID |  | Цена |  | Цена |  |  |
|  |  | Колво |  | Остаток |  |  |
|  |  |  |  | Объем |  |  |
|  |  |  |  | Масса |  |  |

Какие недостатки у хранения данных в разных таблицах? Может оказаться так, что из одной таблицы данные удалили (например, удалили строки из «Документы\_данные», которые соответствуют закупке товаров по документу с ндок = 5, но информация о покупке сохранилась в таблице «Документы», если же в таблице «Документы» еще есть поле «Сумма», в котором есть информация о суммарной стоимости покупки, то при подсчете денег в кассе может возникнуть проблема: данные по базе, посчитанные разными способами будет отличаться). Поэтому можно поддерживать ссылочную целостность.

**Ссылочная целостность –** некоторая система правил, используемых в Access/MS SQL для поддержания связей между записями в связанных таблицах, а также обеспечивающих защиту от случайного удаления или изменения связанных данных. Подробнее, когда можно установить целостность данных, можно прочитать в Help для Access/SQL Express (в файле с определениями есть инструкция по созданию схемы данных).

Ссылочная целостность позволяет, например, исключить случаи, когда в документах появляется товар, которого нет в номенклатуре, позволяет при удалении документа удалять и все его данные, при обновлении внешнего ключа каскадно обновлять его во всех таблицах и тд.

**Задание 1:**

В приложении к архиву есть краткая инструкция, как создавать схему данных в Access или MS SQL. Если это Вам интересно (включая поведение таблиц в случае создания разных видов ссылочной целостности).

## Типы соединения таблиц:

**Операции реляционной алгебры:**

1. Объединение, пересечение, разность и декартово произведение
2. Выборка, проекция, соединение, деление

**Выборка:** возвращает отношение, содержащее кортежи из заданного отношения, причем данные удовлетворяют указанным условиям. (SELECT \* WHERE)

**Проекция:** Возвращает отношение, содержащее все кортежи (подкортежи) заданного отношения, которые остались в этом отношении после исключения из него некоторых атрибутов. (SELECT Поле1)

**Произведение:** Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям. (SELECT \* FROM A,B – декартово произведение)

Объединение: **Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим. (UNION)**

**Пересечение:** Возвращает отношение, содержащее все кортежи, которые принадлежат одновременно двум заданным отношениям.

**Разность:** Возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух заданных отношений и не принадлежат второму.

**Соединение:** Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей, принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах (причем эти общие значения в результирующем кортеже появляются один раз, а не дважды). (SELECT FROM INNER JOIN)

**Деление:** Для заданных двух унарных отношений и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

**Реализация в SQL:**

***Замечание:***

В запросах обращаться можно не только к таблицам, но и другим запросам на выборку данных. Обращение происходит точно также.

Что делать, если нужно использовать данные из нескольких таблиц? Таблицы нужно соединять, объединять и тд.

**1. Операции соединения. Типы соединений:**

Соединение таблиц в MS Access производится с помощью операций: JOIN (декартово произведение, CROSS JOIN, «,»), INNER JOIN, LEFT JOIN, RIGHT JOIN (а также FULL JOIN в MS SQL Server).

**INNER JOIN:**

Синтаксис:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A INNER JOIN B ON A.поле = B.поле

С помощью такого запроса из декартового произведения кортежей множества A и B в результат выборки попадут только те, у которых значения в указанных кортежах совпадают.

В принципе, результат работы INNER JOIN аналогичен работе JOIN с WHERE:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A, B

WHERE A.поле = B.поле

Но, во-первых, читаемость запроса, в котором соединено много таблиц, ухудшается, во-вторых, в таком синтаксисе нельзя использовать различные хинты для ускорения запросов (о различных хинтах можно почитать в MSDN по ключевому слову «hints»).

Пример:

Две таблицы:

Документы и Покупатели:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** |  | **ID** | **Покупатель** |
| 1 | 10 |  | 10 | Иванов |
| 2 | 11 |  | 15 | Сидоров |

Результат выполнения запроса

SELECT \*

FROM Документы, Покупатели

Будет

|  |  |  |  |
| --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** | **ID** | **Покупатели** |
| 1 | 10 | 10 | Иванов |
| 2 | 11 | 10 | Иванов |
| 1 | 10 | 15 | Сидоров |
| 2 | 11 | 15 | Сидоров |

SELECT \*

FROM Документы INNER JOIN Покупатели ON

Документы.Покупатель\_ID = Покупатель.ID

|  |  |  |  |
| --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** | **ID** | **Покупатель** |
| 1 | 10 | 10 | Иванов |

Примечание:

1. Соединять можно не только 2 таблицы, но и больше, описывая соединение по нужным полям. Пример для 3-х:

SELECT A.\*, B.\*, C.\*

FROM A

INNER JOIN B ON

A.поле1 = B.поле1 AND

A.поле2 = B.поле2

INNER JOIN C ON

A.поле3 = C.поле3

WHERE C.поле4 = 1234

1. Соединение можно проводить не только по условию равенства, но и по условию <, >, <> и тд. Кроме того, можно проводить различные операции со сравниваемыми значениями:

INNER JOIN B ON A.поле1 = B.поле1 + 1 (если поле1 - числовое )

1. Соединение по нескольким полям также необязательно проводить только с помощью AND, можно использовать, например, OR.

**Что делать, если необходимо соединить таблицу саму с собой?**

Для таких целей используются псевдонимы (или алиасы). Кроме того, псевдонимы можно использовать и в том случае, если название таблицы, из которой Вы берете данные, слишком длинное или сложное для написания. После создания для таблицы псевдонима, в SELECT, WHERE к полям этой таблицы следует обращаться через псевдоним.

SELECT NewNameA.\*, NewNameB.\* (указывать атрибуты для вывода)

FROM A as NewNameA INNER JOIN B as NewNameB ON NewNameA.поле = NewNameB.поле

WHERE NewNameA.поле2 = 2345

|  |  |  |
| --- | --- | --- |
| **Документы** |  | **Документы\_данные** |
| **ндок** |  | **ндок** |
| Дата |  | **Товар\_ID** |
| Покупатель\_ID |  | Колво |
| Сумма |  | Цена |

**Данные в таблицах следующие:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ндок | Дата | Покупатель\_ID | Сумма |  | ндок | Товар\_ID | Колво | Цена |
| 1 | 01.11.15 | 1 | 170 |  | 1 | 1 | 3 | 20 |
| 2 | 02.11.15 | 2 | 235 |  | 1 | 2 | 2 | 55 |
| 3 | 02.11.15 | 2 | 50 |  | 2 | 3 | 3 | 25 |
|  |  |  |  |  | 2 | 4 | 4 | 40 |
|  |  |  |  |  | 3 | 2 | 1 | 50 |

**Задача 3:**

Что выведется в результате выполнения запроса:

SELECT Документы.ндок, SUM(Сумма)

FROM Документы INNER JOIN Документы\_данные as Данные ON

Документы.ндок = Данные.ндок

GROUP BY Документы.ндок

Запустите запрос на этих данных, подумайте, понимаете ли Вы этот результат?

Чтобы стало совсем понятно, почему именно такой ответ, то подумайте, группировка и суммирование работают на результатах виртуальной таблицы, полученной в результате соединения таблиц Документы и Документы\_данные.

Запустите запрос

SELECT \*

FROM Документы INNER JOIN Документы\_данные as Данные ON

Документы.ндок = Данные.ндок

Сколько строк в результате? Сколько раз встречается каждый номер документа и сумма из таблицы Документы?