**Лекция 4:**

1. Интересные и полезные задачи.
2. Что делать, если таблиц больше одной? Декартово произведение таблиц.
3. 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 4:**

**Как можно делать «красивый комментарий»:**

Соединять строки можно и спецфункциями, а можно и сложением (главное, не забыть привести к правильному типу):

SELECT 'Значение идентификатора ' + CAST(id as nvarchar(10)) + ' равно ' + CAST(val as nvarchar(10))

FROM @T

Вопрос: как хранить полином от одной переменной в БД?

Если полином приведенный, то можно предложить такой способ.

|  |
| --- |
| **Polinoms** |
| **id** |
| **power** |
| coeff |

Давайте при помощи «красивого» вывода попробуем вывести полином (решение от студентки в 2015-2016 году):

DECLARE @Polinom TABLE (id int, power int, coeff float)

INSERT INTO @Polinom

VALUES (1, 2, 1), (1, 1, 2), (1, 0, 1)

|  |  |  |
| --- | --- | --- |
| id | power | coeff |
| 1 | 2 | 1 |
| 1 | 1 | 2 |
| 1 | 0 | 1 |

DECLARE @n INT, @txt nvarchar(100)

SET @n = 0

SET @txt = ''

SELECT @n = @n + power

FROM @Polinom

SELECT @n

SELECT @txt = @txt + CAST(coeff as nvarchar(5)) + '\*x^' + CAST(power as nvarchar(5)) + '+'

FROM @Polinom

WHERE id = 1

SELECT REVERSE(STUFF(REVERSE(@txt),1,1,''))

(см MSDN: ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de\_6tsql/html/8e1a9387-2c5d-4e51-a1fd-a2a95f026d6f.htm

SELECT @*local\_variable* is typically used to return a single value into the variable. However, when *expression* is the name of a column, it can return multiple values. If the SELECT statement returns more than one value, the variable is assigned the last value that is returned.)

**Задача 1:**

Вы знаете, что подсчета суммы чисел в столбце достаточно использовать функцию SUM. А как посчитать произведение чисел в столбце?

Для упрощения будем считать, что все числа положительные. Самостоятельно подумайте про ситуацию, когда числа произвольные (отрицательные, нули). Попробуйте использовать функцию CASE.

SELECT EXP(SUM(LOG(val)))

FROM Numbers

**Декартово произведение:**

Если Вы указываете таблицы через запятую, значит, что каждая строка одной таблицы соединиться с каждой строкой второй таблицы – декартово произведение строк из одной таблицы на строки в другой таблицы. Если Вы на самом деле хотели соединить иначе, то нужно явно прописать по какому условию Вы соединяете (то есть по какому условию из декартового произведения нужно оставить (отфильтровать) только нужные Вам строки):

DECLARE @T TABLE (id int primary key, val int)

INSERT INTO @T

VALUES (1, 2), (2, 4), (3, 6)

SELECT \*

FROM @T T1, @T T2

SELECT \*

FROM @T T1, @T T2

WHERE T1.id = T2.id

Обратите внимание, что здесь мы соединили таблицу саму с собой. Но можно делать декартово произведение между собой и разных таблиц. Самое главное – не забывать о том, что такое декартово произведение.

**Задача 2:**

Написать запрос, который выводит 3 числа, которые в сумме дают заданное.

DECLARE @param int

SET @param = 100

SELECT T1.val, T2.val, T3.val

FROM T as T1, T as T2, T as T3

WHERE T1.val != T2.val AND T1.val != T3.val AND T2.val != T3.val AND T1.val + T2.val + T3.val = @param

Как изменить запрос, чтобы тройки чисел были уникальными (чтобы не встречались тройки вида {97, 1, 2}, {97, 2, 1}, {1, 2, 97} и тд)?

(Задача с собеседования) Подумайте, а как из таблицы с натуральными числами (от 1 до 1000) вывести только простые числа?

Для получения остатка от деления можно использовать оператор: %

**Полезные конструкции:**

Если нужно посчитать сумму только по числам, которые удовлетворяют условию, лучший способ – это воспользоваться CASE.

DECLARE @T TABLE (id int, val float)

INSERT INTO @T

VALUES (1, -1), (2, 0), (3, 1), (4, -4), (5, 6)

SELECT SUM(1), COUNT(\*), MIN(val), -MAX(-val), MAX(val), -MIN(-val),

SUM(CASE WHEN val > 0 THEN 1 ELSE 0 END),

SUM(CASE WHEN val > 0 THEN power(val, 2) ELSE 0 END)

FROM @T

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.

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). Мы рассмотрим первые три.

## 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?