**Лекция 4:**

1. Запросы на группировку
2. Агрегатные функции
3. Отличие между WHERE и HAVING
4. Нормализация: формы 1НФ, 2НФ, 3НФ

**Запросы с Группировкой полей:**

Что такое запросы с группировкой? В случае, если нам необходимо посчитать некоторую агрегатную величину по каким-либо полям, используются запросы с группировкой. В этом случае мы задаем группу данных и для каждой группы вычисляем аналитическую функцию.

Возможные агрегатные функции: SUM, AVG, MIN, MAX, *FIRST, LAST*, STDEV, COUNT и тд. FIRST и LAST есть только в MS Access (и в последних версиях MS SQL).

Пример запроса:

SELECT Покупатель\_ID, SUM (Сумма)

FROM Документы

GROUP BY Покупатель\_ID

Такой запрос сгруппирует поля из документов по покупателю и просуммирует значения покупок из документов. То есть в результате такой запрос посчитает по каждому покупателю сумму его закупок из таблицы Документы. (Группировка – берем поля, указанные в выражении GROUP BY, для одинаковых комбинаций вычисляем одну величину).

Пример:

Были данные

|  |  |  |  |
| --- | --- | --- | --- |
| **Ндок** | **…** | **Покупатель** | **Сумма** |
| 1 |  | Петров | 1 |
| 2 |  | Иванов | 10 |
| 3 |  | Сидоров | 14 |
| 4 |  | Петров | 20 |
| 5 |  | Сидоров | 21 |

Результат запроса:

Петров 21

Иванов 10

Сидоров 35

Запрос можно переписать как

SELECT SUM (Сумма)

FROM Документы

GROUP BY Покупатель\_ID

В результате выполнения не выведется фамилия покупателя, но сумма будет посчитана по каждому персонально.

Если не задано ключевое слово GROUP BY, то это значит, что вся таблица является одной большой группой для данных.

Что выведется, если написать

SELECT SUM (Сумма)

FROM Документы

?

Подумайте, что будет в результате (поле Дата есть в таблице Продажи):

SELECT Покупатель, Дата, SUM(Сумма)

FROM Документы

GROUP BY Покупатель, Дата

В MS SQL есть следующая возможность:

SELECT COUNT(Покупатель), COUNT(distinct Покупатель), COUNT(\*)

FROM Документы

Что выведет такой запрос?

Так же в MS SQL distinct можно использовать и внутри других агрегатных функций, например SUM(distinct Сумма).

Упражнение: Как без использования подзапросов вывести покупателей, которые купили суммарно на максимальное количество денег?

SELECT TOP 1 WITH TIES Покупатель\_ID, SUM(Сумма) Оборот

FROM Документы

GROUP BY Покупатель\_ID

ORDER BY 2 desc

Обратите внимание на то, что сортировать можно по номеру поля в SELECT.

**WHERE и HAVING**

Если в результирующем запросе требуется провести отбор по групповому значению, то необходимо использовать команду HAVING. Раздел HAVING может содержать только те аргументы, которые имеют единственное значение для группы.

Можно ли переносить условия из WHERE в HAVING?

Варианты:

SELECT Покупатель, SUM (Сумма\_RUR)

FROM Документы

GROUP BY Покупатель

HAVING Покупатель = ‘Иванов’

и

SELECT Покупатель, SUM (Сумма\_RUR)

FROM Документы

WHERE Покупатель = ‘Иванов’

GROUP BY Покупатель

равносильны.

А вот запросы:

SELECT Покупатель, SUM (Сумма\_RUR)

FROM Документы

WHERE Сумма\_RUR > 100

GROUP BY Покупатель

и

SELECT Покупатель, SUM (Сумма\_RUR)

FROM Документы

GROUP BY Покупатель

HAVING SUM (Сумма\_RUR) > 100

выдадут абсолютно разные результаты.

*Упражнение 1:* Почему???

*Упражнение 2:* Что выдаст запрос:

SELECT Покупатель, SUM (Сумма\_RUR)

FROM Документы

GROUP BY Покупатель

HAVING MIN (Сумма\_RUR) > 100

Что будет результатом следующего запроса:

DECLARE @T TABLE (id int, val float)

INSERT INTO @T VALUES (1, 10), (2, 15)

SELECT MIN(id)

FROM @T

HAVING SUM(val) > 10

Есть небольшая тонкость в использовании HAVING:

Следующий запрос в MS Access отработает, а на SQL Server нет.

SELECT SUM(Сумма)

FROM Документы

HAVING Покупатель\_ID = 2

А вот следующий запрос не будет работать ни там, ни там.

SELECT Покупатель\_ID

FROM Документы

HAVING Покупатель\_ID = 2

Такой запрос будет работать в Access.

SELECT SUM(Сумма)

FROM Документы

WHERE Сумма > 1

HAVING Покупатель\_ID = 2

Но все же настоятельно предлагается писать запросы в соответствии со спецификацией: HAVING может применяться только к колонкам, которые есть в GROUP BY или к агрегатной функции.

**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

**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

## Нормальные формы

*Упражнение 3:* 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 (по постановке ТЗ очень важно знать по отдельности Ф, И, О): 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. Если ТЗ говорит, что эти номера используются для поиска людей по номеру телефона, то логично, что указание через запятую – плохой вариант хранения данных.*

Пример данных не в 1НФ (ключ ндок, считаем, что поставщик только в одном городе):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата | Ндок | Поставщик | Товар | Город |
| 15.10.21 | 1 | Иванов | Мышь к = 15, ц = 20; Монитор к = 5, ц = 600 | Долгопрудный |
| 22.10.21 | 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 все 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 4:* 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 5:* 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 6*: 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 7*: 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.

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

## 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?

Обычно базы данных относительно хорошо нормализованы, соответственно, запросы пишутся не к одной таблице, а соединению многих таблиц, что может работать не всегда эффективно. В этом случае, если у Вас есть требования к готовым отчетам, Вы можете из нескольких таблиц сделать куб, содержащих основную полезную для отчетов информацию.

OLAP – куб (OnLine Analytical Processing) – интерактивный анализ данных – многомерный массив данных, как правило разреженный и долговременно хранимый, используемый в OLAP.

OLAP – технология обработки данных, заключающаяся в подготовке суммарной (агрегированной) информации на основе больших массивов данных, структурированных по многомерному принципу.

В результате OLAP куб как правило содержит ответы на основные вопросы, возникающие у пользователей. И написание запроса сводится либо к простому SELECT, либо к минимальной группировке (но при этом как правило все запросы к одной таблице).

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 т.д.