**Лекция 4:**

1. Интересные и полезные задачи.
2. Запросы на группировку: WHERE и HAVING.
3. Что делать, если таблиц больше одной? Декартово произведение таблиц.
4. 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

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,''))

**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.

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.

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).

Подумайте, есть ли различия в результатах следующих запросов?

|  |  |
| --- | --- |
| SELECT Покупатель  FROM Документы  GROUP BY Покупатель | SELECT distinct Покупатель  FROM Документы |

**Комментарий:**

Использующим MySQL:

Такой синтаксис в MySQL:

SELECT Покупатель, MAX(Сумма)

FROM Документы

не выдаст ошибку синтаксиса, но покупатель, которого выведет этот запрос – будет произвольным, совсем не обязательно тем, которому соответствует максимальная 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

**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: %

Возможный вариант решения:

DECLARE @T TABLE (val int)

DECLARE @i int

SET @i = 0

WHILE @i < 1000

BEGIN

SET @i = @i + 1

INSERT INTO @T

VALUES (@i)

END

SELECT T1.val, SUM(CASE WHEN T1.val%T2.val = 0 THEN 1 ELSE 0 END)

FROM @T T1, @T T2

WHERE T1.val > T2.val

GROUP BY T1.val

HAVING SUM(CASE WHEN T1.val%T2.val = 0 THEN 1 ELSE 0 END) = 1

ORDER BY T1.val

**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