### Лекция 8.

1. Повторение пройденного: декартово произведение
2. Построение гистограмм
3. Повторение пройденного: LEFT JOIN и UNION
4. INTERSECT, EXCEPT

**Присылаемое ДЗ:**

Домашние задания находятся в архиве на сайте. Домашнее задание включает в себя аналитическую задачу и пакет запросов. Дедлайн – 10.12.17 23:59.

Решения и вопросы по задаче следует присылать на адрес nixlab@nix.ru с заголовком "Парадигмы: задача 3 (4)", в теле письма указать ФИО, группу.

Контрольная пройдет 07.12.17 в 18:35 в 117 гк

**Лекция 8.**

**Задача (Повторение пройденного)**

У Вас есть таблица с числами от 0 до 9, как сгенерировать таблицу с числами от 0 до 999?

Возможный вариант:

DECLARE @T TABLE (val int)

INSERT INTO @T

VALUES (0), (1), (2), (3), (4), (5), (6), (7), (8), (9)

SELECT T1.val\*100 + T2.val\*10 + T3.val new

FROM @T T1, @T T2, @T T3

ORDER BY new asc

Если Вы включите опцию «Client Statistics» (посмотрите в панели инструментов, недалеко от Execution Plan), то получите некую информацию по скорости работы запроса на 10 запусках:

У меня получилось так (на разных запусках по разному, но в среднем числа похожие):

|  |  |
| --- | --- |
|  | Avg on 10 Trials, ms |
| Client processing time | 9.2 |
| Total execution time | 30.7 |
| Wait time on server replies | 21.5 |

Для сравнения, давайте попробуем использовать SQL не как SQL. И получим эту таблицу в цикле:

DECLARE @new TABLE (val int)

DECLARE @i int

SET @i = 0

SET NOCOUNT ON

WHILE @i < 1000

BEGIN

INSERT INTO @new

VALUES (@i)

SET @i = @i + 1

END

SET NOCOUNT OFF

SELECT \*

FROM @new

Данные из client statistics (у Вас могут быть другие значения, но обратите внимание на то, что в среднем время выполнения дольше)

|  |  |
| --- | --- |
|  | Avg on 10 Trials, ms |
| Client processing time | 30.8 |
| Total execution time | 64.7 |
| Wait time on server replies | 33.9 |

Обратите внимание, что этот запрос будет работать дольше.

Если написать аналогичные запросы для 100 000, то таблица по скорости:

Для декартового произведения

|  |  |
| --- | --- |
|  | Avg on 10 Trials, ms |
| Client processing time | 216.8 |
| Total execution time | 344.2 |
| Wait time on server replies | 127.4 |

То есть менее секунды

Для цикла:

|  |  |
| --- | --- |
|  | Avg on 10 Trials, ms |
| Client processing time | 2116.7 |
| Total execution time | 2147.4 |
| Wait time on server replies | 30.7 |

Уже 2 секунды и более.

**Задача: Построение гистограммы:**

У Вас есть таблица: для каждой частицы скорость, с которой она пролетела мимо датчика. Необходимо при помощи SQL получить данные для построения гистограммы.

|  |  |
| --- | --- |
| **N** | **V** |
| 1 | 10 |
| 2 | 12 |
| 3 | 13 |
| 4 | 12 |
| 5 | 11 |

Что такое гистограмма распределения скоростей: для каждой скорости мы должны вычислить, сколько у нас частиц, которые пролетели с такой скоростью.

В чём минусы следующего запроса?

DECLARE @Data TABLE (N int, V float)

SELECT V, COUNT(\*)

FROM @Data

GROUP BY V

V – float, а значит различных значений скоростей может быть достаточно много. И такая гистограмма будет непоказательна. Нужно разбить скорости по корзинам (обычно одинаковой ширины) и уже для корзин посчитать, сколько частиц попало в каждую корзину.

Например, вот так:

DECLARE @h float

SET @h = 2

SELECT FLOOR(V/@h)\*@h B, FLOOR(V/@h + 1)\*@h E, COUNT(\*)

FROM @Data

GROUP BY FLOOR(V/@h)\*@h, FLOOR(V/@h + 1)\*@h

Здесь регулируется ширина корзины. Подумайте, а как изменить этот запрос, чтобы можно было задавать не шаг для гистограммы, а количество «корзин» для статистики?

**В MSSQL есть вспомогательные операторы Except и Intersect.**

Оба оператора возвращают уникальные значения, получающиеся в результате сравнения двух запросов.

EXCEPT возвращает все уникальные строки из левого запроса (верхнего), которые не найдены в правом.

INTERSECT возвращает уникальные строки, которые возвращают оба запроса.

Основные правила синтаксиса аналогичны условиям для UNION: типы полей должны быть совместимы, колво столбцов, возвращаемых обоими запросами, должно быть одинаково. NULL значения считаются равными (то есть возвращаются по одному разу)

Имена колонок используемых в ORDER BY должны ссылаться на названия колонок, возвращаемых левой частью запроса.

В случае использования подряд нескольких INTERSECT и EXCEPT нужно помнить, что INTERSECT приоритетнее, чем EXCEPT. В остальном они выполняются в порядке следования.

**Примеры:**

**INTERSECT**

1. Повторные значения:

DECLARE @A TABLE (id1 int, id2 int)

INSERT INTO @A

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

DECLARE @B TABLE (id1 int, id2 int)

INSERT INTO @B

VALUES (1, 4), (1, 3), (2, 5)

SELECT id1

FROM @A

INTERSECT

SELECT id1

FROM @B

Ответ: 1 единожды

1. NULL значения

DECLARE @A TABLE (id1 int, id2 int)

INSERT INTO @A

VALUES (1, 2), (1, 3), (NULL, NULL)

DECLARE @B TABLE (id1 int, id2 int)

INSERT INTO @B

VALUES (1, 4), (1, 3), (NULL, 1)

SELECT id1

FROM @A

INTERSECT

SELECT id1

FROM @A

Ответ: 1, NULL

**EXCEPT**

1. Повторные значения

DECLARE @A TABLE (id1 int, id2 int)

INSERT INTO @A

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

DECLARE @B TABLE (id1 int, id2 int)

INSERT INTO @B

VALUES (1, 4), (1, 3), (2, 5)

SELECT id2

FROM @A

EXCEPT

SELECT id2

FROM @B

Ответ: 2 (1 раз)

1. NULL-значения

DECLARE @A TABLE (id1 int, id2 int)

INSERT INTO @A

VALUES (1, 2), (1, 3), (NULL, NULL)

DECLARE @B TABLE (id1 int, id2 int)

INSERT INTO @B

VALUES (1, 4), (1, 3), (NULL, 1)

SELECT id2

FROM @A

EXCEPT

SELECT id2

FROM @B

Ответ: NULL, 2

**Порядок выполнения:**

1. EXCEPT по порядку:

DECLARE @A TABLE (id1 int)

INSERT INTO @A

VALUES (3)

DECLARE @B TABLE (id1 int)

INSERT INTO @B

VALUES (3), (2)

DECLARE @C TABLE (id1 int)

INSERT INTO @C

VALUES (3), (1)

SELECT id1

FROM @A

EXCEPT

SELECT id1

FROM @B

EXCEPT

SELECT id1

FROM @C

-- ответ: 0 строк. Сначала из А вычитается B

-- если было бы наоборот, то вывелось бы 3.

1. Сначала INTERSECT, после EXCEPT

DECLARE @A TABLE (id1 int)

INSERT INTO @A

VALUES (3)

DECLARE @B TABLE (id1 int)

INSERT INTO @B

VALUES (3), (2)

DECLARE @C TABLE (id1 int)

INSERT INTO @C

VALUES (3), (1)

SELECT id1

FROM @B

EXCEPT

SELECT id1

FROM @A

INTERSECT

SELECT id1

FROM @C

Ответ: 2. Иначе было бы пусто.

Если есть еще UNION, то UNION и EXCEPT выполняются в порядке появления в запросе слева направо. При этом первый приоритет у INTERSECT.

**Вопрос**: как получить разницу двух таблиц при помощи EXCEPT?

DECLARE @A TABLE (id1 int, id2 int, id3 int)

INSERT INTO @A

VALUES

(1, 2, 3), (4, 5, 6), (1, 2, 3), (NULL, 1, 2), (NULL, NULL, NULL)

DECLARE @B TABLE (id1 int, id2 int, id3 int)

INSERT INTO @B

VALUES (4, 5, 6), (7, 8, 9), (NULL, 1, 2)

Перед тем как писать запрос, подумайте, а что по Вашему мнению является разницей двух таблиц?

Давайте вспомним, как мы получали разницу двух таблиц при помощи LEFT JOIN?

SELECT A.\*

FROM @A A LEFT JOIN @B B ON

A.id1 = B.id1 AND

A.id2 = B.id2 AND

A.id3 = B.id3

WHERE B.id1 is NULL

|  |  |  |
| --- | --- | --- |
| id1 | id2 | id3 |
| 1 | 2 | 3 |
| 1 | 2 | 3 |
| NULL | 1 | 2 |
| NULL | NULL | NULL |

Соответствует ли это Вашим ожиданиям?

Посмотрите на выдачу у EXCEPT:

SELECT \*

FROM @A

EXCEPT

SELECT \*

FROM @B

|  |  |  |
| --- | --- | --- |
| id1 | id2 | id3 |
| NULL | NULL | NULL |
| 1 | 2 | 3 |

Подумайте, что изменится, если Вы добавите в таблицу B строку с NULL значениями во всех полях?

PS: обратите внимание, что, если у Вас есть NULL значения, то писать запрос на разницу таблиц при помощи LEFT JOIN нужно аккуратно. В результате нужно всегда перед написанием запроса думать, а что Вам нужно, нужны ли дублирующие строки, нужно ли выводить NULL значения и тд.

**Задача:** вывести все товары, которые покупатель с идентификатором = 1 не покупал после 01.11.17 (включая и 01.11.17)

Пример данных:

DECLARE @Товары TABLE (good\_id int primary key)

INSERT INTO @Товары

VALUES (1), (2), (3)

DECLARE @Продажи TABLE (doc\_id int, Data datetime, IDC int, good\_id int, Qty int, primary key (doc\_id, good\_id))

INSERT INTO @Продажи VALUES

(1, '20170901', 1, 2, 1),

(2, '20170901', 1, 3, 1),

(3, '20171101', 1, 3, 1),

(4, '20171101', 2, 1, 1)

Видим, что 1-ого покупателя подходящие товары – это 1 и 2. 1 – он не покупал вообще (взаимоотношения 2-ого покупателя и 1-ого товара нас не интересуют), 2 – покупал до 01.09.17.

Варианты запросов:

LEFT JOIN с условиями

SELECT Товары.good\_id

FROM @Товары Товары LEFT JOIN @Продажи Продажи ON

Товары.good\_id = Продажи.good\_id AND

Продажи.IDC = 1 AND

Продажи.Data >= '20171101'

WHERE Продажи.doc\_id is NULL

Подумайте, что случится, если условие на идентификатор покупателя и дату продажи перенести в WHERE?

EXCEPT:

SELECT good\_id

FROM @Товары

EXCEPT

SELECT good\_id

FROM @Продажи

WHERE IDC = 1 AND Data >= '20171101'

LEFT JOIN и подзапрос:

SELECT Товары.good\_id

FROM @Товары Товары LEFT JOIN

(

SELECT good\_id

FROM @Продажи

WHERE IDC = 1 AND Data >= '20171101'

) ТоварыПокупателя ON

Товары.good\_id = ТоварыПокупателя.good\_id

WHERE ТоварыПокупателя.good\_id is NULL