МГТУ ИМ. Н.Э. БАУМАНА

Базы Данных

Семинар 4

Операции над множествами

Добавим немного интерактива и узнаем, какие операции над множествами они знают.

Рассмотрим первую операцию - объединение (**UNION**):

```
Синтаксис оператора UNION:

SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions]
UNION [ALL]
SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions];
```

Предложение **UNION** объединяет вывод двух или более SQL запросов в единый набор строк и столбцов. Например чтобы получить всех продавцов и заказчиков размещенных в Лондоне и вывести их как единое целое вы могли бы ввести:

```
SELECT snum, sname
FROM Salespeople
WHERE city = 'London'

UNION

SELECT cnum, cname
FROM Customers
WHERE city = 'London';
```

Операция объединения может быть выполнена только при выполнении следующих условий:

- количество выходных столбцов каждого из запросов должно быть одинаковым;
- выходные столбцы каждого из запросов должны быть совместимы между собой (в порядке их следования) по типам данных;
- в результирующем наборе используются имена столбцов, заданные в первом запросе;
- предложение **ORDER BY** применяется к результату соединения, поэтому оно может быть указано только в конце всего составного запроса.

Пример с множествами:
Чему равно объединение множеств {1, 2} {3}? - {1, 2, 3}
Чему равно объединение множеств {1, 2, 3} {3}? - {1, 2, 3}

UNION работает по той же схеме: Если вы объединяете два запроса и в каждом из которых есть одинаковые данные, другими словами полностью идентичные, оператор union объединит их в одну строку для того чтобы не было дублей.

UNION ALL – это оператор SQL для объединения результирующего набора данных нескольких запросов, а вот данный оператор, выведет уже абсолютно все строки, даже дубли.
Далее рассмотрим оператор пересечения:

Оператор **INTERSECT** используется для возврата результатов 2-х или более запросов SELECT. Тем не менее, он возвращает строки, выбранные для всех запросов или наборов данных. Если запись существует в одном запросе, а в другом нет, то она будет исключена из результирующего набора **INTERSECT**.

Запрос INTERSECT будет возвращать записи в серой затененной области. Эти записи, которые существуют в обоих SELECT1 и SELECT2.
Каждый оператор SELECT в INTERSECT должен иметь одинаковое количество полей в наборах результатов с одинаковыми типами данных.

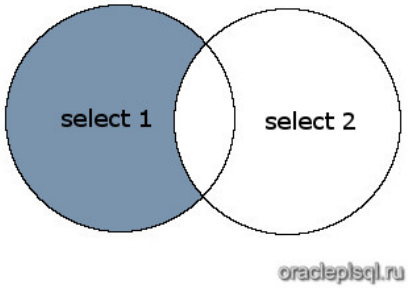
```
Синтаксис оператора INTERSECT:

SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions]
INTERSECT
SELECT expression1, expression2, … expression_n
FROM tables
[WHERE conditions];
```

expression1, expression2, … expression_n - Столбцы или расчеты, которые вы хотите получить.
Tables - Таблицы из которых вы хотите получить записи.
WHERE conditions - Необязательный. Условия, которые должны быть выполнены для выбранных записей.

В обоих SELECT запросах должно быть одинаковое количество expression и иметь схожие типы данных.

Оператор вычитания - MINUS (Oracle)/
EXCEPT(MS SQL)



Oracle оператор **MINUS** используется для возврата всех строк первого запроса SELECT, не возвращаемых вторым SELECT. Каждый запрос SELECT будет определять набор данных. Оператор MINUS возвращает все записи из первого набора данных, а затем из результатов удалит все записи из второго набора данных.

Запрос **MINUS** вернет записи в серой затененной области. Эти записи, которые существуют в SELECT1 и SELECT2. Каждый SELECT в запросе MINUS должны иметь одинаковое количество полей в результирующих наборах с одинаковыми типами данных.

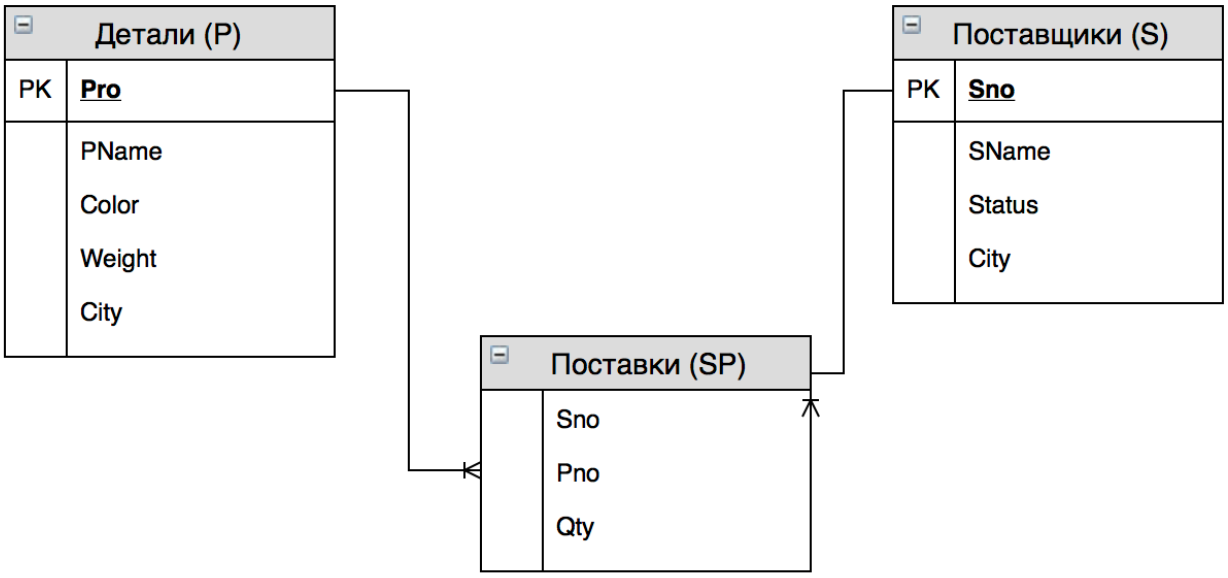
```
Синтаксис оператора MINUS:

SELECT expression1, expression2, … expression_n
  FROM tables
[WHERE conditions]
 MINUS
SELECT expression1, expression2, … expression_n
  FROM tables
[WHERE conditions];
```

expression1, expression2, … expression_n - Столбцы или расчеты, которые вы хотите получить.
Tables - Таблицы из которых вы хотите получить записи.
WHERE conditions - Необязательный. Условия, которые должны быть выполнены для выбранных записей.

В обоих SELECT запросах должно быть одинаковое количество expression и иметь схожие типы данных.

Практическая часть
ER-модель, рассматриваемая на лекции:



1. Получить имена всех поставщиков, поставляющих деталь под номером 2

```
SELECT S.SName
FROM S inner join SP on S.Sno = SP.Sno
WHERE SP.Pno = 2
```

2. Получить имена поставщиков, поставляющих хотя бы одну деталь красного цвета

```
SELECT S.SName
FROM S inner join SP on S.Sno = SP.Sno
       inner join P on SP.Pno = P.Pno
WHERE SP.Color = 'Красный'
```

3. Получить пары имен всех поставщиков <SName1, SNmae2> из одного города

```
SELECT S.SName
FROM S AS FirstS inner join S AS SecondS
     on (FirstS.City = SecondS.City) AND (FirstS.Sno < SecondS.Sno)
```

Не говорить сразу ответ, пусть подумают, напишут, что будет если не будет ограничения на Sno. Спросить какой вид джойна СУБД выберет для соединения этих таблиц в данном запросе (Nested Loops Join), почему?
При составлении Хэш-таблицы ключ соединения хэшируется с помощью специальной функции, которая не гарантирует, что при a < b, f(a) < f(b), поэтому данный вид соединения не подходит.

4. Найти имена всех поставщиков, не поставляющих деталь под номером 2:
Основная идея: Найти тех кто поставляет 2ую деталь (1-ый запрос) и вычесть из из всех поставщиков

```
SELECT S.SName from S
EXCEPT
SELECT S.SName
FROM S inner join SP on (S.Sno = SP.Sno) AND (SP.Pno = 2)
```

Обратить внимание, что в условие соединения добавилось условие из WHERE. Так тоже можно.

5. Найти всех поставщиков, поставляющих все детали:

Распишем поэтапно (После каждого запроса писать результат):
5.1. Сначала надо найти количество деталей:

```
SELECT count(distinct Pno) as cnt
FROM P
```

В результате - 1 ячейка с названием cnt и количеством различных деталей

5.2. Для каждого поставщика узнать, сколько различных деталей он поставляет

```
SELECT Sno, count(distinct Pno) as cntS
FROM SP
GROUP BY Sno
```

5.3 Собираем все вместе:

```
SELECT S.Sname
FROM (SELECT Sno, count(distinct Pno) as cntS
      FROM SP
      GROUP BY Sno) tmp
     inner join S on tmp.Sno = S.Sno
WHERE cntS IN (SELECT count(distinct Pno)
               FROM P )
```

Если будет мало:

6. Найти поставщика, поставляющего больше всего различных деталей (некрасивый способ, красивый будет после следующего семинара)
6.1. Для каждого поставщика узнать, сколько различных деталей он поставляет

```
SELECT Sno, count(distinct Pno) as cntS
FROM SP
GROUP BY Sno
```

6.2. Найти среди данной выборки MAX

```
SELECT MAX(cntS)
FROM( SELECT Sno, count(distinct Pno) as cntS
      FROM SP
      GROUP BY Sno)tmp
```

6.3. Найти тех, у кого количество деталей соответствует максимуму

```
SELECT S.SName
FROM (SELECT Sno, count(distinct Pno) as cntS
      FROM SP
      GROUP BY Sno) groupS
    inner join S on groupS.Sno = S.Sno
WHERE groupS.cntS = (SELECT MAX(cntS)
                     FROM (SELECT Sno, count(distinct Pno) as cntS
                           FROM SP
                           GROUP BY Sno) tmp)
```

Некрасиво, потому что два раза одна и та же группировка, от этого можно избавиться с помощью Обобщенного Табличного Выражения (ОТВ), которое будет рассмотренное на следующем семинаре.