Лекция 6

SQL запросы

Неопределенное значение NULL

```
WHERE   Заказ_Фирма.Код_фирма  IS  NULL.
```

Понятие неопределенного значения было внесено в концепции баз данных позднее. Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени. Это значение при появлении дополнительной информации в любой момент времени может быть заменено на некоторое конкретное значение. При сравнении неопределенных значений не действуют стандартные правила сравнения: одно неопределенное значение никогда не считается равным другому неопределенному значению,
Для выявления равенства значения некоторого атрибута неопределенному применяют специальные стандартные предикаты;

<имя атрибута> IS NULL
И
<имя атрибута> IS NOT NULL.

Неопределенное значение NULL

'Введение Null-значений вызвало необходимость модификации классической двузначной логики и превращения ее в трехзначную.
Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	B	Not A	A & B	A | B
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

Неопределенное значение NULL

'Введение Null-значений вызвало необходимость модификации классической двузначной логики и превращения ее в трехзначную.
Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	B	Not A	A & B	A | B
1	1	0	1	1
1	0	0	0	1
1	Null	0	?	1
0	1	1	0	1
0	0	1	0	0
0	Null	1	?	Null
Null	1	Null	Null	?
Null	0	Null	0	Null
Null	Null	Null	Null	Null

Неопределенное значение NULL

'Введение Null-значений вызвало необходимость модификации классической двузначной логики и превращения ее в трехзначную.
Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	B	Not A	A & B	A | B
1	1	0	1	1
1	0	0	0	1
1	Null	0	Null	1
0	1	1	0	1
0	0	1	0	0
0	Null	1	0	Null
Null	1	Null	Null	1
Null	0	Null	0	Null
Null	Null	Null	Null	Null

Неопределенное значение NULL

Склад

Код_товара	Получено	Остаток	Добавить поле
1	25	17	
2	25		
3	25	4	
4	25	15	
5	25	17	
6	25	17	
7	25	14	
8	25	15	
9	25	15	
10	25		
*			

Склад | Запрос9 | Запрос10

Код_товара	Наименование_т	Остаток
2	Дырокол	
10	Блокнот	

```
SELECT Склад.Код_товара, Товар.Наименование_т, Склад.Остаток
FROM Склад, Товар
WHERE ((Склад.Остаток) Is Null) AND Склад.Код_товара=Товар.Код_товара);
```

Агрегатные функции и группировка. bonus

```
БД  ≪Банк≫,
F = (N, ФИО, Филиал, ДатаОткрытия, ДатаЗакрытия, Остаток);
Q, = (Филиал, Город);
```

предположим, что мы хотим найти суммарный остаток на счетах в филиалах. Можно сделать раздельный запрос для каждого из них, выбрав SUM(Остаток) из таблицы для каждого филиала.
GROUP BY, позволит поместить их все в одну команду:

```
SELECT Филиал,  SUM(Остаток)
FROM F  GROUP BY Филиал;
```

GROUP BY применяет агрегатные функции независимо для каждой группы, определяемой с помощью значения поля Филиал. Группа состоит из строк с одинаковым значением поля Филиал, функция SUM применяется отдельно для каждой такой группы, то есть суммарный остаток на счетах подсчитывается отдельно Для каждого филиала. Значение поля, к которому применяется GROUP BY, имеет, по определению, только одно значение на группу вывода, Как и результат работы агрегатной функции. Поэтому мы можем совместить в одном запросе агрегат и поле. Вы можете также использовать GROUP BY с несколькими полями.

Агрегатные функции и группировка. bonus

Предположим, что мы хотели бы увидеть только те суммарные значения остатков на счетах, которые превышают $5000. Чтобы увидеть суммарные остатки свыше $5000, необходимо использовать предложение HAVING. Предложение HAVING определяет критерии, используемые, чтобы удалять определенные группы из вывода, точно так же как предложение WHERE делает это для индивидуальных строк.

```
SELECT Филиал, SUM(Остаток)
FROM F
GROUP BY Филиал
HAVING SUM(Остаток) > 5000;
```

Аргументы в предложении HAVING подчиняются тем же самым правилам, что и в предложении SELECT, где используется GROUP BY. Они должны иметь одно значение на группу вывода.

Агрегатные функции и группировка. bonus

Следующая команда будет запрещена:

```
SELECT Филиал, SUM(Остаток)
FROM F
GROUP BY Филиал
HAVING ДатаОткрытия = 27/12/1999:
```

Поле ДатаОткрытия не может быть использовано в предложении HAVING, потому
что оно может иметь больше чем одно значение на группу вывода. Чтобы избежать такой ситуации, предложение HAVING должно ссылаться только на агрегаты и поля, выбранные GROUP BY. Имеется правильный способ сделать вышеупомянутый запрос:

```
SELECT Филиал, SUM(Остаток)
FROM F
WHERE ДатаОткрытия = '27/12/1999'
GROUP BY Филиал
```

Смысл данного запроса следующий: найти сумму остатков по каждому филиалу счетов, открытых 27 декабря 1999 года.

Агрегатные функции и группировка. bonus

HAVING может использовать только аргументы, которые имеют одно значение на группу вывода. Практически, ссылки на агрегатные функции — наиболее общие, но и поля, выбранные с помощью GROUP BY, также допустимы. Например, мы хотим увидеть суммарные остатки на счетах филиалов в Санкт-Петербурге, Пскове и Урюпинске:

```
SELECT Филиал, SUM(Остаток)
FROM F, Q
WHERE F.Филиал = Q.Филиал
GROUP BY Филиал
HAVING Филиал IN ("Санкт-Петербург", "Псков", "Урюпинск");
```

Поэтому в арифметических выражениях предикатов, входящих в условие выборки раздела HAVING, прямо можно использовать только спецификации столбцов указанных в качестве столбцов группирования в разделе GROUP BY. Остальные столбцы можно специфицировать только внутри спецификаций агрегатных
функций COUNT, SUM, AVG, MIN и MAX, вычисляющих в данном случае некоторое агрегатное значение для всей группы строк.

Агрегатные функции и группировка. bonus

Аналогично обстоит дело с подзапросами входящими в предикаты условия выборки раздела HAVING: если в подзапросе используется характеристика текущей группы, то она может задаваться только путем ссылки на столбцы группирования.

Результатом выполнения раздела HAVING является сгруппированная таблица, содержащая только те группы строк, для которых результат вычисления условия поиска есть TRUE, В частности, если раздел HAVING присутствует в табличном выражении, не содержащем GROUP BY, то результатом его выполнения будет либо
пустая таблица, либо результат выполнения предыдущих разделов табличного выражения, рассматриваемый как одна группа без столбцов группирования.

Внешние/внутренние соединения

Операция внутреннего соединения

```
SELECT <имена столбцов> FROM <имена
таблиц>
WHERE <условие соединения>
```

в этом случае в результирующее отношение попадали только сцепленные по заданным условиям кортежи исходных отношений, для которых эти условия были определены и истинны.

В действительности часто необходимо объединять таблицы таким образом, чтобы в результат попали все строки из первой таблицы, а вместо тех строк второй таблицы, для которых не выполнено условие соединения, в результат попадали бы неопределенные значения NULL. Или наоборот, включаются все строки из правой (второй) таблицы, а отсутствующие части строк из первой таблицы дополняются неопределенными значениями. Такие объединения названы **внешними**

Внешние/внутренние соединения

В общем случае синтаксис части FROM выглядит следующим образом: '

```
FROM <список исходных таблиц> ,
< выражение естественного объединения > |
< выражение объединения > |
< выражение перекрестного объединения > |
< выражение запроса на объединение >
```

< выражение естественного объединения > :

<имя_таблицы 1> NATURAL INNER |LEFT | RIGHT JOIN <имя_таблицы_2>

< выражение объединения > :

<имя_таблицы 1> INNER |LEFT | RIGHT JOIN <имя_таблицы_2> ON условие

Внешние/внутренние соединения

В общем случае синтаксис части FROM выглядит следующим образом: '

```
FROM <список исходных таблиц> ,
< выражение естественного объединения > |
< выражение объединения > |
< выражение перекрестного объединения > |
< выражение запроса на объединение >
```

< выражение перекрестного объединения > :

<имя_таблицы 1> CROSS JOIN <имя_таблицы_2>

< выражение запроса на объединение > :

выражение запроса на объединение :
<имя_таблицы_1> UNION JOIN <имя_таблицы_2>

Внешние/внутренние соединения

В этих определениях INNER — означает внутреннее объединение

Если заданы ключевые слова LEFT, RIGHT, то объединение всегда считается внешним.

LEFT — левое объединение, то есть в результат входят все строки таблицы 1, а части результирующих кортежей, для которых не было соответствующих значений в таблице 2, дополняются значениями NULL (неопределенно).

Ключевое слово RIGHT означает правое внешнее объединение, и в отличие от левого объединения в этом случае в результирующее отношение включаются все строки таблицы 2, а недостающие части из таблицы 1 дополняются неопределенными значениями

Внешние/внутренние соединения

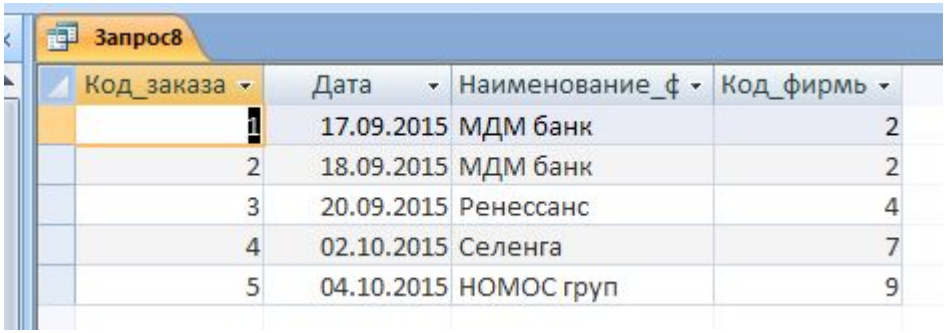
```
SELECT Заказ_фирма.Код_заказа, Заказ_фирма.Дата, Фирма.
Наименование_ф, Заказ_фирма.Код_фирмы
FROM Заказ_фирма RIGHT JOIN Фирма ON Заказ_фирма.Код_фирмы =
Фирма.Код_фирмы
```

Запрос8

Код_заказа	Дата	Наименование_ф	Код_фирмы
		Альфа капитал	
1	17.09.2015	МДМ банк	2
2	18.09.2015	МДМ банк	2
		Группа ВТБ	
3	20.09.2015	Ренессанс	4
		Глобэкс	
		Ритм	
4	02.10.2015	Селенга	7
		Банк Левобережны	
5	04.10.2015	НОМОС груп	9
(№)			

Внешние/внутренние соединения

```
SELECT Заказ_Фирма.Код_заказа, Заказ_Фирма.Дата, Фирма.
Наименование_ф, Заказ_Фирма.Код_фирмы
FROM Заказ_Фирма LEFT JOIN Фирма ON Заказ_Фирма.Код_фирмы =
Фирма.Код_фирмы
```



Запрос8

Код_заказа	Дата	Наименование_ф	Код_фирмы
1	17.09.2015	МДМ банк	2
2	18.09.2015	МДМ банк	2
3	20.09.2015	Ренессанс	4
4	02.10.2015	Селенга	7
5	04.10.2015	НОМОС груп	9

```
SELECT Заказ_Фирма.Код_заказа, Заказ_Фирма.Дата, Фирма.
Наименование_ф, Заказ_Фирма.Код_фирмы
FROM Заказ_Фирма INNER JOIN Фирма ON Заказ_Фирма.Код_фирмы =
Фирма.Код_фирмы
```

```
SELECT Заказ_Фирма.Код_заказа, Заказ_Фирма.Дата, Фирма.
Наименование_ф, Заказ_Фирма.Код_фирмы
FROM Заказ_Фирма , Фирма   WHERE Заказ_Фирма.Код_фирмы =
Фирма.Код_фирмы
```

Примеры:

БД ≪Библиотека≫,

BOOKS (ISBN, TITL, AUTOR, COAUTOR, YEARIZD, PAGES)
READER(NUM_READER, NAME_READER, ADRESS, HOME_PHONE, WORK_PHONE.
BIRTH_DAY)
EXEMPLARE (INV, ISBN, YES_NO, NUM_READER, DATE_IN, DATE_OUT)

ISBN — уникальный шифр книги;
TITL — издание книги;
AUTOR — фамилия автора;
COAUTOR — фамилия соавтора;
YEARIZD - год издания;
PAGES — число страниц

NUM_READER — уникальный номер читательского билета;
NAME_READER — фамилию и инициалы читателя;
ADRESS — адрес читателя;
HOME_PHONE — номер домашнего телефона;
WORK_PHONE — номер рабочего телефона;
BIRTH_DAY — дату рождения читателя.

БД ≪Библиотека≫

BOOKS (ISBN, TITL, AUTOR, COAUTOR, YEARIZD, PAGES)
READER(NUM_READER, NAME_READER, ADRESS, HOME_PHONE, WORK_PHONE. BIRTH_DAY)
EXEMPLARE (INV, ISBN, YES_NO, NUM_READER, DATE_IN, DATE_OUT)

INV — уникальный инвентарный номер экземпляра книги;
ISBN - шифр книги, который определяет, какая это книга, и ссылается на сведения из первой таблицы;
YES_NO - признак наличия или отсутствия в библиотеке данного экземпляра в текущий момент;
NUM_READER — номер читательского билета, если книга выдана читателю, и Null в противном случае;
DATE_IN — если книга у читателя, то это дата, когда она выдана читателю;
DATE_OUT — дата, когда читатель должен вернуть книгу в библиотеку.

Определим перечень книг у каждого читателя; если у читателя нет книг, то номер экземпляра книги равен NULL. Для выполнения этого поиска нам надо использовать левое внешнее объединение, то есть мы берем все строки из таблицы READER и соединяем со строками из таблицы EXEMPLARE, если во второй таблице нет строки с соответствующим номером читательского билета, то в строке результирующего отношения атрибут EXEMPLARE.INV будет иметь неопределенное значение NULL:

```
SELECT   READER.NAME_READER,  EXEMPLARE.INV
FROM  READER
    LEFT JOIN
        EXEMPLARE
 ON
 READER.NUM_READER = EXEMPLARE.NUM_READER
```

Операция *запроса па объединение эквивалентна операции теоретико-множест-*венного объединения в алгебре. При этом требование эквивалентности схем исходных отношений сохраняется. Запрос на объединение выполняется по следующей схеме:

```
SELECT - запрос
UNION
SELECT - запрос
UNION
SELECT - запрос
```

Все запросы, участвующие в операции объединения, не должны содержать выражений, то есть вычисляемых полей.
Например, нужно вывести список читателей, которые держат на руках книгу ≪Идиот≫ или книгу ≪Преступление и наказание≫. Вот как будет выглядеть запрос:

```
SELECT  READER.NAME_READER
    FROM READER, EXEMPLARE, BOOKS
    WHERE EXEMPLARE.NUM_READER= READER.NUM_READER
    AND
    EXEMPLRE.ISBN = BOOKS.ISBN
        AND
    BOOKS.TITLE = "Идиот"
UNION
SELECT  READER.NAME_READER
        FROM READER, EXEMPLARE,BOOKS
        WHERE EXEMPLARE.NUM_READER= READER.NUM_READER
        AND
        EXEMPLARE.ISBN = BOOKS.ISBN
        AND
        BOOKS.TITLE = "Преступление и наказание"
```

По умолчанию при выполнении запроса на объединение дубликаты кортежей всегда исключаются. Поэтому, если найдутся читатели, у Которых находятся на руках обе книги, то они все равно в результирующий список попадут только один раз.

Запрос на объединение может объединять любое число исходных запросов. Так, к предыдущему запросу можно добавить еще читателей, которые держат на руках книгу ≪Замок≫:

```
UNION
SELECT READER.NAME_READER
FROM READER, EXEMPLARE,BOOKS
WHERE EXEMPLARE.NUM_READER = READER.NUM_READER
AND
EXEMPLRE.ISBN = BOOKS.ISBN
AND
BOOKS.TITLE = "Замок"
```

В том случае, когда вам необходимо сохранить все строки из исходных отношений, необходимо использовать ключевое слово ALL в операции объединения. В случае сохранения дубликатов кортежей схема выполнения запроса на объединение будет выглядеть следующим образом:

```
SELECT - запрос
UNION
SELECT - запрос
UNION
SELECT - запрос
```

Однако тот же результат можно получить простым изменением фразы WHERE
первой части исходного запроса, соединив локальные условия логической операцией ИЛИ и исключив дубликаты кортежей.

```
SELECT DISTINCT READER.NAME_READER
FROM READER,  EXEMPLARE.BOOKS

 WHERE  EXEMPLARE.NUM_READER  = READER.NUM_READER
 AND
 EXEMPLRE.ISBN = BOOKS.ISBN    AND
 BOOKS.TITLE = "Идиот"  OR BOOKS.TITLE = "Преступление и
 наказание" OR BOOKS.TITLE = "Замок"
```

Ни один из исходных запросов в операции UNION не должен содержать предложения упорядочения результата ORDER BY, однако результат объединения может быть упорядочен, для этого предложение ORDER BY с указанием списка столбцов упорядочения записывается после текста последнего исходного SELECT-запроса.