Лекция 8

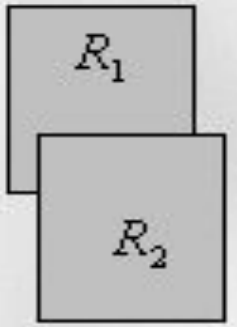
Продолжение Лекции 6:

1. Варианты объединения запросов
2. Представления (View)

Объединение

$$REZ = R \cup S$$

Пример:



$$R[Q,T] \cup S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cup \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \end{bmatrix}$$

UNION-объединение

UNION объединяет результаты двух запросов по следующим правилам:

1. каждый из объединяемых запросов должен содержать одинаковое число столбцов;
2. тип значений из попарно объединяемых столбцов должен быть одинаковым или приводимым: нельзя объединять значения из столбца типа **integer** и столбца типа **varchar**;
3. из результирующего набора автоматически исключаются совпадающие строки

```
SQL> select * from tbl1;

        F1          F2 F3
---------- ---------- -----
         1          10 abc
         2          10 aaa
         3          20 bbb

SQL> select * from tbl2;

        F1          F2 F3
---------- ---------- -----
         1          10 cde
         2          40 ddd
         3          40 eee
         3          40 eee
```

```
SQL> select * from tbl1
  2  union
  3  select * from tbl2;

        F1          F2 F3
---------- ---------- -----
         1          10 abc
         1          10 cde
         2          10 aaa
         2          40 ddd
         3          20 bbb
         3          40 eee

6 строк выбрано.
```

UNION-объединение

1. Если в строку вставляется какая-либо константа, добавляемая в запросе, то ее значение также влияет на идентичность строк
2. Выполнение **UNION**-объединения, использующего выражения :

```
SQL> select f1,f3,'tbl1'from tbl1
  2  union
  3  select f1,f3,'t1' from tbl1;

        F1 F3    'TBL
---------- ----- ----
         1 abc   t1
         1 abc   tbl1
         2 aaa   t1
         2 aaa   tbl1
         3 bbb   t1
         3 bbb   tbl1

6 строк выбрано.
```

- Стандарт не накладывает никаких ограничений на упорядочивание строк в результирующем наборе.
- Так, некоторые СУБД сначала выводят результат первого запроса, а затем - результат второго запроса. СУБД Oracle автоматически сортирует записи по первому указанному столбцу даже в том случае, если для него не создан индекс.
- Для того чтобы явно указать требуемый порядок сортировки, следует использовать фразу **ORDER BY**. При этом можно использовать как имя столбца, так и его номер

```
SQL> select f1,f3,'tbl1'from tbl1
  2  union
  3  select f1,f3,'t1' from tbl1
  4  order by 3;

        F1 F3    'TBL
---------- ----- ----
         1 abc   tbl1
         2 aaa   tbl1
         3 bbb   tbl1
         1 abc   t1
         3 bbb   t1
         2 aaa   t1

6 строк выбрано.
```

UNION ALL

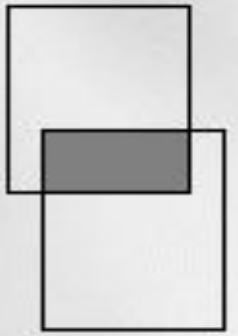
- выполняет объединение двух подзапросов аналогично фразе **UNION** со следующими исключениями:
- совпадающие строки не удаляются из формируемого результирующего набора;
- объединяемые запросы выводятся в результирующем наборе последовательно без упорядочивания.
- При объединении более двух запросов для изменения порядка выполнения операции объединения можно использовать скобки

```
SQL> (select * from tbl1
  2  union
  3  select* from tbl2)
  4  union all
  5  select * from tbl1;

        F1          F2 F3
---------- ---------- -----
         1          10 abc
         1          10 cde
         2          10 aaa
         2          40 ddd
         3          20 bbb
         3          40 eee
         3          50 eee
         1          10 abc
         2          10 aaa
         3          20 bbb

10 строк выбрано.
```

Пересечение

$$REZ = R \cap S$$

Пример:



$$R[Q,T] \cap S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cap \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 5 & a \\ 1 & b \end{bmatrix}$$

INTERSECT-объединение

- Фраза **INTERSECT** позволяет выбрать только те строки, которые присутствуют в каждом объединяемом результирующем наборе.
- Пример объединения запросов как пересекающихся множеств:

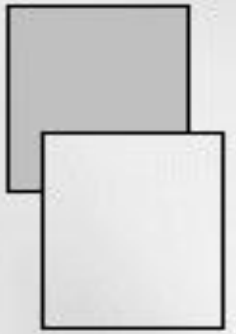
```
SQL> select * from tbl1;

        F1          F2 F3
---------- ---------- -----
         1          10 abc
         2          10 aaa
         3          20 bbb

SQL> select * from tbl2;

        F1          F2 F3
---------- ---------- -----
         1          10 cde
         2          40 ddd
         3          40 eee
         3          40 eee
         3          50 eee
         1          10 abc

6 строк выбрано.
```

```
SQL> select * from tbl1
  2  intersect
  3  select * from tbl2;

        F1          F2 F3
---------- ---------- -----
         1          10 abc

SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
```

Вычитание

$$REZ = R \setminus S$$

Пример:



$$R[Q,T] - S = \begin{bmatrix} 5 & a \\ 3 & a \\ 9 & a \\ 1 & b \\ 2 & b \\ 4 & b \end{bmatrix} \cap \begin{bmatrix} 5 & a \\ 10 & b \\ 15 & c \\ 2 & d \\ 6 & a \\ 1 & b \end{bmatrix} = \begin{bmatrix} 3 & a \\ 9 & a \\ 2 & b \\ 4 & b \end{bmatrix}$$

EXCEPT-объединение

- Фраза **EXCEPT** позволяет выбрать только те строки, которые присутствуют в первом объединяемом результирующем наборе, но отсутствуют во втором результирующем наборе.
- Фразы **INTERSECT** и **EXCEPT** должны поддерживаться только при полном уровне соответствия стандарту SQL-92. Так, некоторые СУБД вместо фразы **EXCEPT** поддерживают опцию **MINUS**

```
SQL> select * from tbl1;

        F1          F2 F3
---------- ---------- -----
         1          10 abc
         2          10 aaa
         3          20 bbb

SQL> select * from tbl2;

        F1          F2 F3
---------- ---------- -----
         1          10 cde
         2          40 ddd
         3          40 eee
         3          40 eee
         3          50 eee
         1          10 abc

6 строк выбрано.
```

```
SQL>  select * from tbl2
  2   minus
  3   select * from tbl1;

        F1          F2 F3
---------- ---------- -----
         1          10 cde
         2          40 ddd
         3          40 eee
         3          50 eee

SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
SQL>
```

- Если не используется ключевое слово **ALL** (по умолчанию подразумевается **DISTINCT**), то при выполнении операции автоматически устраняются дубликаты строк.
- Если указано **ALL**, то количество дублированных строк подчиняется следующим правилам (n1 - число дубликатов строк первого запроса, n2 - число дубликатов строк второго запроса):

INTERSECT ALL: min(n1, n2) и

EXCEPT ALL: n1 - n2, если n1>n2.

Фраза **CORRESPONDING BY** позволяет использовать в объединяемых запросах различное число столбцов: в результирующий набор будут включены только столбцы, указанные в списке.

- Этот список также определяет порядок включения столбцов в результирующий набор.

Создание представлений

Основные объекты БД

-

1. таблицы;
2. индексы;
3. представления;
4. триггеры;
5. хранимые процедуры и функции;
6. курсоры;
7. системный словарь, содержащий метаданные.

СУБД MySQL

Практически полная поддержка стандарта SQL.

- MySQL 5.0 содержит следующие нововведения:

1. хранимые процедуры и функции;
2. обработчики ошибок;
3. курсоры;
4. триггеры;
5. представления;
6. информационная схема (так называемый системный словарь, содержащий метаданные).
7. Тип таблиц Maria (начиная с версии 5.2.x — Aria) — расширенная версия хранилища MyISAM, с добавлением средств сохранения целостности данных после краха.

- MariaDB — ответвление СУБД MySQL, разрабатываемое сообществом. Толчком к созданию стала необходимость обеспечения свободного статуса СУБД (под лицензией GPL), в противовес неопределенной политике лицензирования MySQL компанией Oracle.[3]

Что такое представление?

- Типы таблиц, с которыми вы имели дело до сих пор, назывались — *базовыми таблицами*. Это — таблицы, которые содержат данные.
- Однако имеется другой вид таблиц — представления. *Представления* — это таблицы, чье содержание выбирается или получается из других таблиц.
- Они работают в запросах и операторах DML точно также как и основные таблицы, но не содержат никаких собственных данных.
- Представления подобны окнам, через которые вы просматриваете информацию (как она есть, или в другой форме, как вы потом увидите), которая фактически хранится в базовой таблице.
- *Представление* — это фактически запрос, который выполняется всякий раз, когда представление становится темой команды. Вывод запроса при этом в каждый момент становится содержанием представления.

Представления (VIEW)

- Представляют собой временные, производные (виртуальные) таблицы и являются объектами базы данных, информация в которых не хранится постоянно, как в базовых таблицах, а формируется динамически при обращении к ним.
- Обычные таблицы относятся к базовым, т.е. содержащим данные и постоянно находящимся на устройстве хранения информации. Представление не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц.
- Применение представлений позволяет разработчику базы данных обеспечить каждому пользователю или группе пользователей наиболее подходящие способы работы с данными, что решает проблему простоты их использования и безопасности.
- Содержимое представлений выбирается из других таблиц с помощью выполнения запроса, причем при изменении значений в таблицах данные в представлении автоматически меняются.
- Представление - это фактически тот же запрос, который выполняется всякий раз при участии в какой-либо команде.
- Результат выполнения этого запроса в каждый момент времени становится содержанием представления.
- У пользователя создается впечатление, что он работает с настоящей, реально существующей таблицей.

Представления

- У СУБД есть две возможности реализации представлений.
- Если его определение простое, то система формирует каждую запись представления по мере необходимости, постепенно считывая исходные данные из базовых таблиц.
- В случае сложного определения СУБД приходится сначала выполнить такую операцию, как материализация представления, (сохранить информацию, из которой состоит представление, во временной таблице). Затем система приступает к выполнению пользовательской команды и формированию ее результатов, после чего временная таблица удаляется.
- Представление - это предопределенный запрос, хранящийся в базе данных, который выглядит подобно обычной таблице и не требует для своего хранения дисковой памяти.
- Для хранения представления используется только оперативная память.
- В отличие от других объектов базы данных представление не занимает дисковой памяти за исключением памяти, необходимой для хранения определения самого представления.

Представление может содержать:

- подмножество записей из таблицы БД, отвечающее определённым условиям (например, при наличии одной таблицы «Люди» можно создать два представления «Мужчины» и «Женщины», в каждом из которых будут записи только о людях соответствующего пола);
- подмножество столбцов таблицы БД, требуемое программой (например, из реальной таблицы «Сотрудники» представление может содержать по каждому сотруднику только ФИО и табельный номер);
- результат обработки данных таблицы определёнными операциями (например, представление может содержать все данные реальной таблицы, но с приведением строк в верхний регистр и обрезанными начальными и концевыми пробелами);
- результат объединения (join) нескольких таблиц (например, при наличии таблиц «Люди», «Адреса», «Улицы», «Фирмы и организации» возможно построение представления, которое будет выглядеть как таблица, для каждого человека содержащее его личные данные, адрес места жительства, название организации, где он работает, и адрес этой организации);
- результат слияния нескольких таблиц с одинаковыми именами и типами полей, когда в представлении попадают все записи каждой из сливаемых таблиц (возможно, с исключением дублирования);
- результат группировки записей в таблице
- практически любую комбинацию вышеперечисленных возможностей.

Оператор создания представления (упрощенная форма)

CREATE VIEW < ИМЯ ПРЕДСТАВЛЕНИЯ>

[(< СПИСОК СТОЛБЦОВ>)] AS <SQL - ЗАПРОС >

- Если список имен столбцов в представлении не задан, то каждый столбец представления получает имя соответствующего столбца запроса.

Команда CREATE VIEW

- Она состоит из слов **CREATE VIEW** (*СОЗДАТЬ ПРЕДСТАВЛЕНИЕ*), *имени представления*, которое нужно создать, слова **AS** (*КАК*), и далее *запроса*, как в следующем примере:
- **CREATE VIEW Londonstaff**
  AS SELECT *
  FROM Salespeople
  WHERE city = 'London';

- **Select * FROM Londonstaff;**
-
- **========== SQL Execution Log ==========**

```
SELECT *
FROM  Londonstaff;
=============================================
 snum     sname       city        comm
------   ----------  ----------  -------
 1001     Peel        London      0.1200
 1004     Motika      London      0.1100
=============================================
```

Модифицирование представлений

- **CREATE VIEW Salesown AS SELECT snum, sname, city FROM Salespeople;**
- ========== SQL Execution Log ==========

```
| SELECT *                                                    |
| FROM  Salesown;                                             |
| ============================================================|
|  snum     sname       city                                  |
| ------   ---------  ----------                              |
|  1001     Peel       London                                 |
|  1002     Serres      San Jose                              |
|  1004     Motika      London                                |
|  1007     Rifkin     Barcelona                              |
|  1003     Axelrod     New York                              |
 ============================================================
```

- Представление может теперь изменяться командами модификации DML, но модификация не будет воздействовать на само представление. Команды будут на самом деле перенаправлены к базовой таблице:
- **UPDATE Salesown SET city = 'Palo Alto‘ WHERE snum = 1004;**
- Его действие идентично выполнению той же команды в таблице Продавцов. Однако, если значение комиссионных продавца будет обработано командой UPDATE
- **UPDATE Salesown SET comm = .20 WHERE snum = 1004;**
- она будет отвергнута, так как поле comm отсутствует в представлении Salesown.

Групповые представления

- *Групповые представления* — это представления, которые содержит предложение GROUP BY, или которые основываются на других групповых представлениях.
- **CREATE VIEW Totalforday**
  AS SELECT odate, COUNT(DISTINCT cnum), COUNT(DISTINCT snum), COUNT(onum), AVG(amt), SUM(amt)
- **FROM Orders GROUP BY odate;**
- Предположим, что каждый день вы должны следить за порядком номеров заказчиков, номерами продавцов, принимающих Заказы, номерами Заказов, средним от Заказов, и общей суммой приобретений в Заказах.
- SELECT * FROM Totalforday;

Представления и объединения

- Представления не требуют, чтобы их вывод осуществлялся из одной базовой таблицы.
- Почти любой допустимый запрос SQL может быть использован в представлении, он может выводить информацию из любого числа базовых таблиц, или из других представлений.
- **CREATE VIEW Nameorders**
  AS SELECT onum, amt, a.snum, sname, cname
  FROM Orders a, Customers b, Salespeople c
  WHERE a.cnum = b.cnum AND a.snum = c.snum;
- Можно также объединять представления с другими таблицами, или базовыми таблицами или представлениями, поэтому вы можете увидеть все Заказы Axelrod и значения его комиссионных в каждом Заказе:
- SELECT a.sname, cname, amt comm
  FROM Nameorders a, Salespeople b
  WHERE a.sname = 'Axelrod' AND b.snum = a.snum;

Представления и подзапросы

- Представления могут также использовать и подзапросы, включая соотнесенные подзапросы.
- Предположим, ваша компания предусматривает премию для тех продавцов, которые имеют заказчика с самой высокой суммой **Заказа** для любой указанной даты. Вы можете проследить эту информацию с помощью представления:
- CREATE VIEW Elitesalesforce
  AS SELECT b.odate, a.snum, a.sname,
  FROM Salespeople a, Orders b
  WHERE a.snum = b.snum AND b.amt =
- (SELECT MAX (amt) FROM Orders c WHERE c.odate =
  b.odate);

Чего не могут делать представления

- Имеются большое количество типов представлений, которые являются доступными только для чтения.
- Это означает, что их можно запрашивать, но они не могут подвергаться действиям команд модификации. Мы будем рассматривать эту тему в лаб.работах.
- Имеются также некоторые виды запросов, которые не допустимы в определениях представлений.
- Одиночное представление должно основываться на одиночном запросе; *объединение* (**UNION**) и *объединение всего* (**UNION ALL**) не разрешаются.
- Упорядочение по **ORDER BY** никогда не используется в определении представлений. Вывод запроса формирует содержание представления, которое напоминает базовую таблицу и является по определению неупорядоченным.

Удаление представлений

- Синтаксис удаления представления из базы данных подобен синтаксису удаления базовых таблиц:
- **DROP VIEW <view name>;**
- Помните, вы должны являться владельцем представления, чтобы иметь возможность удалить его.

Оператор CREATE VIEW

```
CREATE VIEW table_name [(field .,:) ]
    AS (SELECT_operator
     [WITH [CASCADED | LOCAL]
     CHECK OPTION ] );
```

- Оператор запроса **SELECT**, использующийся для построения представления, может иметь две формы:
- Расширяемая форма оператора **SELECT** задается как конструкция **SELECT** *(не менять синтаксис представления при изменении оператором **ALTER TABLE** структуры таблицы: добавлении новых столбцов или удалении столбцов),
- Постоянная форма оператора **SELECT** задается как конструкция **SELECT список_столбцов**, жестко фиксируя имена столбцов, входящих в запрос.
- Как будет влиять изменение основных таблиц на представление, можно указать в операторе **ALTER TABLE**:
  - фраза **RESTRICT** определяет ограничение, отменяющее изменение таблицы, если на данный столбец есть ссылки в представлениях (а также в ограничениях и предикатах);
  - фраза **CASCADE** указывает, что все представления, использующие удаляемый столбец, также будут удалены (а также все внешние ключи, имеющие ссылки на удаляемый столбец или ограничения **FOREIGN KEY**).

Оператор ALTER TABLE

- ALTER TABLE table_name
  - { ADD [COLUMN] column_name column_type [(size)] [column_ constraint] }
  - | { ALTER [COLUMN]column_name {SET DEFAULT value } | DROP DEFAULT }
  - | { DROP [COLUMN] column_name RESTRICT|CASCADE }
  - | { ADD table_ constraint }
  - | { DROP CONSTRAINT constraint_name RESTRICT | CASCADE };
- Поддержка оператора ALTER TABLE необходима только для полного уровня соответствия стандарту, однако, большинство коммерческих СУБД реализует этот оператор, но с некоторыми изменениями и расширениями
- **`ALTER TABLE tbl1 DROP COLUMN f2 CASCADE;`**

Изменение данных в представлениях

- Если для представления указывается оператор **DELETE**, **INSERT** или **UPDATE**, то все изменения происходят как над представлением, так и над основными таблицами, используемыми для создания представления.
- Не во все представления можно внести изменения. Так, представления могут быть изменяемыми или постоянными.
- Стандарт позволяет внесение изменений всегда только в одну основную таблицу.
- Однако большинство коммерческих СУБД позволяют вносить изменения и в две связанные между собой таблицы, но с некоторыми оговорками.
- Стандарт SQL-92 определяет, что представление является ***изменяемым***, если выполнены следующие условия:
- запрос, используемый для создания представления, извлекает данные только из одной таблицы;
- если в запросе, используемом для создания таблицы, в качестве таблицы выступает представление, то оно также должно быть изменяемым;
- не разрешается никаких объединений таблиц, даже самой с собой;
- запрос, используемый для создания представления, не должен содержать вычислимых столбцов, агрегирующих функций и фраз **DISTINCT**, **GROUP BY** и **HAVING**;
- в запросе, используемом для создания представления, нельзя ссылаться дважды на один и тот же столбец.

Опции [WITH [CASCADED | LOCAL] CHECK OPTION

Для изменяемого представления можно указывать фразу WITH CHECK OPTION, позволяющую предотвращать "потерю строк" в представлениях. Так, если эта фраза указана, то при внесении изменений в таблицу будет проверен предикат, указанный в запросе, использованном для создания таблицы. Если предикат не возвращает значение TRUE, то изменения не будут внесены.

Например, если запрос создан оператором

- CREATE VIEW v_tbl1 AS (SELECT f1,f2, f3 FROM tbl1 WHERE f2>100) WITH CHECK OPTION;

, то вставка строки не будет произведена:

- INSERT INTO v_tbl1 (f1,f2,f3) VALUES (1,50,'abc');

Фраза WITH CHECK OPTION может быть расширена до:

- WITH CASCADED CHECK OPTION - предикаты проверяются во всех вложенных запросах;
- WITH LOCAL CHECK OPTION - предикаты проверяются только в запросе, использованном для создания данного представления;

Опции [WITH [CASCADED | LOCAL] CHECK OPTION

Для представления, созданного операторами

- CREATE VIEW v_1 AS (SELECT f1,f2, f3 FROM tbl1 WHERE f2>100);,
- CREATE VIEW v_2 AS (SELECT f1,f2, f3 FROM v_1 WHERE f2>50) WITH LOCAL CHECK OPTION;,

добавление строки будет выполнено:

- INSERT INTO v_2 (f1,f2,f3) VALUES (1,30,'abc');.

Эта строка будет добавлена в основную таблицу, но не будет видна в представлении, посредством которого она была добавлена.

По умолчанию предполагается, что для WITH CHECK OPTION используется фраза CASCADED.