Запросы к нескольким таблицам

Соединение таблиц

- Одна из наиболее важных особенностей запросов SQL — это их способность определять связи между многочисленными таблицами и выводить информацию из них в терминах этих связей, всю внутри одной команды.
- Этот вид операции называется — объединением, которое является одним из видов операций в реляционных базах данных. В реляционном подходе это связи которые можно создавать между позициями данных в таблицах.
- Используя соединения, мы непосредственно связываем информацию с любым номером таблицы, и таким образом способны создавать связи между сравнимыми фрагментами данных. При соединении, таблицы представленные списком в предложении FROM запроса, отделяются запятыми.
- Предикат запроса может ссылаться к любому столбцу любой связанной таблицы и, следовательно, может использоваться для связи между ими. Обычно, предикат сравнивает значения в столбцах различных таблиц чтобы определить, удовлетворяет ли WHERE установленному

ИМЕНА ТАБЛИЦ И СТОЛБЦОВ

- Полное имя столбца таблицы фактически состоит из имени таблицы, сопровождаемого точкой и затем именем столбца.
- Имеются несколько примеров имен :

Salespeople.snum

Salespeople.city

Orders.odate

- До этого, вы могли опускать имена таблиц потому что вы запрашивали только одну таблицу одновременно, а SQL достаточно интеллектуален чтобы присвоить соответствующий префикс, имени таблицы.
- Даже когда вы делаете запрос многочисленных таблиц, вы еще можете опускать имена таблиц, если все ее столбцы имеют различные имена. Но это не всегда так бывает.
- Например, мы имеем две типовые таблицы со столбцами называемыми city. Если мы должны связать эти столбцы (кратковременно), мы будем должны указать их с именами Salespeople.city или Customers.city, чтобы SQL мог их различать.

СОЗДАНИЕ СОЕДИНЕНИЯ

- Предположим что вы хотите поставить в соответствии вашему продавцу ваших заказчиков в городе в котором они живут, поэтому вы увидите все комбинации продавцов и заказчиков для этого города.
- Вы будете должны брать каждого продавца и искать в таблице Заказчиков всех заказчиков того же самого города.
- Вы могли бы сделать это, введя следующую команду (вывод показывается в Рисунке 1.)

КОМАНДА

```
SELECT Customers.cname, Salespeople.sname,
Salespeople.city
FROM Salespeople, Customers
WHERE Salespeople.city = Customers.city;
```

РИСУНОК 1. Соединение двух таблиц

```
================  SQL Execution Log ============
| SELECT Customers.cname, Salespeople.sname,     |
| Salespeople.city                               |
| FROM Salespeople, Customers                    |
| WHERE Salespeople.city = Customers.city        |
| ============================================== |
|   cname        cname              city         |
|  -------      --------           ----          |
|  Hoffman      Peel               London        |
|  Hoffman      Peel               London        |
|  Liu          Serres             San Jose      |
|  Cisneros     Serres             San Jose      |
|  Hoffman      Motika             London        |
|  Clemens      Motika             London        |
================================================
```

- Так как это поле city имеется и в таблице Продавцов и таблице Заказчиков, имена таблиц должны использоваться как префиксы.
- Хотя это необходимо только когда два или более полей имеют одно и то же имя, в любом случае это хорошая идея включать имя таблицы в соединение для лучшего понимания и непротиворечивости.
- Несмотря на это, мы будем, в наших примерах далее, использовать имена таблицы только когда необходимо, так что будет ясно, когда они необходимы а когда нет.

- Что SQL в основном делает в соединении — так это исследует каждую комбинацию строк двух или более возможных таблиц, и проверяет эти комбинации по их предикатам.
- В предыдущем примере, требовалась строка продавца Peel из таблицы Продавцов и объединение ее с каждой строкой таблицы Пользователей, по одной в каждый момент времени.

- Если комбинация производит значение которое делает предикат верным, и если поле city из строк таблиц Заказчика равно London, то Peel — это то запрашиваемое значение которое комбинация выберет для вывода.

- То же самое будет затем выполнено для каждого продавца в таблице Продавцов (у некоторых из которых не было никаких заказчиков в этих городах).

СОЕДИНЕНИЕ ТАБЛИЦ ЧЕРЕЗ СЫЛОЧНУЮ ЦЕЛОСТНОСТЬ

- Эта особенность часто используется просто для эксплуатации связей встроенных в базу данных. В предыдущем примере, мы установили связь между двумя таблицами в соединении.
- Это прекрасно. Но эти таблицы, уже были соединены через snum поле.
- Эта связь называется состоянием ссылочной целостности.
- Используя соединение можно извлекать данные в терминах этой связи.

- Например, чтобы показать имена всех заказчиков соответствующих продавцам которые их обслуживают, мы будем использовать такой запрос:

```
SELECT Customers.cname, Salespeople.sname
FROM Customers, Salespeople
WHERE Salespeople.snum = Customers.snum;
```

- Вывод этого запроса показывается в Рисунке 2.

- Это — пример соединения, в котором столбцы используются для определения предиката запроса, и в этом случае, snum столбцы из обеих таблиц, удалены из вывода.
- И это прекрасно. Вывод показывает какие заказчики каким продавцом обслуживаются; значения поля snum которые устанавливают связь — отсутствуют.
- Однако если вы введете их в вывод, то вы должны или удостовериться что вывод понятен сам по себе или обеспечить

РИСУНОК 2. Объединение продавцов с их заказчикам

```
================  SQL Execution Log ============
| SELECT Customers.cname, Salespeople.sname,     |
| FROM Salespeople, Customers                    |
| WHERE Salespeople.snum = Customers.snum        |
| ============================================== |
|    cname         sname                         |
|   -------      --------                        |
|   Hoffman      Peel                            |
|   Giovanni     Axelrod                         |
|   Liu          Serres                          |
|   Grass        Serres                          |
|   Clemens      Peel                            |
|   Cisneros     Rifkin                          |
|   Pereira      Motika                          |
================================================
```

СОЕДИНЕНИЯ ТАБЛИЦ ПО РАВЕНСТВУ ЗНАЧЕНИЙ В СТОЛБЦАХ И ДРУГИЕ ВИДЫ ОБЪЕДИНЕНИЙ

- Соединения которые используют предикаты основанные на равенствах называются — объединениями по равенству.
- Все примеры до настоящего времени, относились именно к этой категории, потому что все условия в предложениях WHERE базировались на математических выражениях использующих знак равно (=). Строки **'city = 'London' и 'Salespeople.snum = Orders.snum'** — примеры таких типов равенств найденных в предикатах.

- **Соединения по равенству** — это вероятно наиболее общий вид объединения, но имеются и другие. Вы можете, фактически, использовать любой из реляционных операторов в соединении. Здесь показан пример другого вида соединения (вывод показывается в Рисунке.3):

```
SELECT sname, cname
FROM Salespeople, Customers
WHERE sname < cname AND rating < 200;
```

РИСУНОК 3. Соединение основанное на неравенстве

```
================  SQL Execution Log ============
| SELECT sname, cname                            |
| FROM Salespeople, Customers                    |
| WHERE sname < cname                            |
| AND rating < 200;                              |
| ============================================== |
|     sname        cname                         |
|    --------    -------                         |
|    Peel         Pereira                        |
|    Motika       Pereira                        |
|    Axelrod      Hoffman                        |
|    Axelrod      Clemens                        |
|    Axelrod      Pereira                        |
================================================
```

- Эта команда не часто бывает полезна. Она воспроизводит все комбинации имени продавца и имени заказчика так, что первый предшествует последнему в алфавитном порядке, а последний имеет оценку меньше чем 200.

- Обычно, вы не создаете сложных связей подобно этой, и, по этой причине, вы вероятно будете строить наиболее общие соединения по равенству, но вы должны хорошо знать и другие возможности.

СОЕДИНЕНИЕ БОЛЕЕ ДВУХ ТАБЛИЦ

- Вы можете также создавать запросы объединяющие более двух таблиц.
- Предположим что мы хотим найти все порядки заказчиков не находящихся в тех городах где находятся их продавцы.
- Для этого необходимо связать все три наши типовые таблицы (вывод показывается в Рисунке. 4)

```
SELECT onum, cname, Orders.cnum,
Orders.snum
FROM Salespeople, Customers,Orders
WHERE Customers.city < > Salespeople.city
    AND Orders.cnum = Customers.cnum
    AND Orders.snum = Salespeople.snum;
```

РИСУНОК 4. Объединение трех таблиц

```
================  SQL Execution Log ==============
| SELECT onum, cname, Orders.cnum, Orders.snum     |
| FROM Salespeople, Customers, Orders              |
| WHERE Customers.city < > Salespeople.city        |
| AND Orders.cnum = Customers.cnum                 |
| AND Orders.snum = Salespeople.snum;              |
| ================================================ |
|    onum     cname          cnum      snum        |
|   ------   -------       -----     -----         |
|    3001    Cisneros        2008      1007        |
|    3002    Pereira         2007      1004        |
|    3006    Cisneros        2008      1007        |
|    3009    Giovanni        2002      1003        |
|    3007    Grass           2004      1002        |
|    3010    Grass           2004      1002        |
================================================
```

- Хотя эта команда выглядит скорее как комплексная, вы можете следовать за логикой, просто проверяя — что заказчики не размещены в тех городах где размещены их продавцы (совпадение двух snum полей), и что перечисленные порядки — выполнены с помощью этих заказчиков (совпадение порядков с полями cnum и snum в таблице Порядков).

РЕЗЮМЕ

- Теперь можно больше не ограничиваетесь просмотром одной таблицы в каждый момент времени.
- Кроме того, вы можете делать сложные сравнения между любыми полями любого числа таблиц и использовать полученные результаты чтобы решать какую информацию вы бы хотели видеть.
- Фактически, эта методика настолько полезна для построения связей, что она часто используется для создания их внутри одиночной таблицы. Это будет правильным: вы сможете объединить таблицу с собой, а это очень удобная вещь.

РАБОТА С SQL

1. Напишите запрос, который бы вывел список номеров порядков, сопровождающихся именем заказчика, который создавал эти порядки.
2. Напишите запрос, который бы выдавал имена продавца и заказчика для каждого порядка после номера порядков.
3. Напишите запрос, который бы выводил всех заказчиков, обслуживаемых продавцом с комиссионными выше 12%. Выведите имя заказчика, имя продавца и ставку комиссионных продавца.
4. Напишите запрос, который вычислил бы сумму комиссионных продавца для каждого порядка заказчика с оценкой выше 100.

СОЕДИНЕНИЕ ТАБЛИЦЫ С СОБОЙ

КАК ДЕЛАТЬ СОЕДИНЕНИЕ ТАБЛИЦЫ С СОБОЙ ?

- Для объединения таблицы с собой, вы можете сделать каждую строку таблицы, одновременно, и комбинацией ее с собой и комбинацией с каждой другой строкой таблицы. Вы затем оцениваете каждую комбинацию в терминах предиката, также как в соединениях мультитаблиц.

- Это позволит вам легко создавать определенные виды связей между различными позициями внутри одиночной таблицы — с помощью обнаружения пар строк со значением поля, например. Вы можете изобразить соединение таблицы с собой, как соединение двух копий одной и той же таблицы.

- Таблица на самом деле не копируется, но SQL выполняет команду так, как если бы это было сделано. Другими словами, это соединение — такое же, как и любое другое соединение между двумя таблицами, за исключением того, что в данном случае обе таблицы идентичны.

ПСЕВДОНИМЫ

- Синтаксис команды для объединения таблицы с собой, тот же что и для соединения многочисленных таблиц, в одном экземпляре. Когда вы объединяете таблицу с собой, все повторяемые имена столбца, заполняются префиксами имени таблицы.
- Чтобы ссылаться к этим столбцам внутри запроса, вы должны иметь два различных имени для этой таблицы. Вы можете сделать это с помощью определения временных имен называемых — переменными диапазона, переменными корреляции или просто — псевдонимами.
- Вы определяете их в предложении FROM запроса.
- Это очень просто: вы набираете имя таблицы, оставляете пробел, и затем набираете

- Имеется пример, который находит все пары заказчиков имеющих один и тот же самый рейтинг (вывод показывается в Рисунке 5.)

```
SELECT first.cname, second.cname, first.rating FROM
Customers first, Customers second WHERE
first.rating = second.rating;
```

(Обратите внимание что на Рисунке 5, как и в некоторых дальнейших примерах, полный запрос не может уместиться в окне вывода, и следовательно будет усекаться.)

РИСУНОК 5. Соединение таблицы с собой

SQL Execution Log

Giovanni	Giovanni	200
Giovanni	Liu	200
Liu	Giovanni	200
Liu	Liu	200
Grass	Grass	300
Grass	Cisneros	300
Clemens	Hoffman	100
Clemens	Clemens	100
Clemens	Pereira	100
Cisneros	Grass	300
Cisneros	Cisneros	300
Pereira	Hoffman	100
Pereira	Clemens	100
Pereira	Pereira	100

- В вышеупомянутой команде, SQL ведет себя так, как если бы он соединял две таблицы называемые 'первая' и 'вторая'. Обе они — фактически, таблицы Заказчика, но псевдонимы разрешают им быть обработанными независимо.
- Псевдонимы первый и второй были установлены в предложении FROM запроса, сразу после имени копии таблицы.
- Обратите внимание что псевдонимы могут использоваться в предложении SELECT, даже если они не определены в предложении FROM. Это — очень хорошо.
- SQL будет сначала допускать любые такие псевдонимы на веру, но будет отклонять команду если они не определены далее в предложении FROM запроса.

- Псевдоним существует только пока команда выполняется. Когда запрос заканчивается, псевдонимы, используемые в нем, больше не имеют никакого значения.

- Теперь, когда имеются две копии таблицы Заказчиков, чтобы работать с ними, SQL может обрабатывать эту операцию точно также, как и любое другое соединение — берет каждую строку из одного псевдонима и сравнивает ее с каждой строкой из другого псевдонима.

УСТРАНЕНИЕ ИЗБЫТОЧНОСТИ

- Обратите внимание, что наш вывод имеет два значение для каждой комбинации, причем второй раз в обратном порядке. Это потому, что каждое значение показано первый раз в каждом псевдониме, и второй раз (симметрично) в предикате.
- Следовательно, значение **A** в псевдониме сначала выбирается в комбинации со значением **B** во втором псевдониме, а затем значение **A** во втором псевдониме выбирается в комбинации со значением B в первом псевдониме.

- В нашем примере, Hoffman выбрался вместе с Clemens, а затем Clemens выбрался вместе с Hoffman. Тот же самый случай с Cisneros и Grass, Liu и Giovanni, и так далее.
- Кроме того каждая строка была сравнена сама с собой, чтобы вывести строки такие как — Liu и Liu.
- Простой способ избежать этого состоит в том, чтобы налагать порядок на два значения, так чтобы один мог быть меньше чем другой или предшествовал ему в алфавитном порядке.
- Это делает предикат ассиметричным, поэтому те же самые значения в обратном порядке не будут выбраны снова.

ПРИМЕР

- SELECT tirst.cname, second.cname, first.rating FROM Customers first, Customers second WHERE first.rating = second.rating AND first.cname < second.cname;

(Вывод этого запроса показывается в Рисунке 6)

РИСУНОК 6. Устранение избыточности вывода в соединении с собой.

```
================  SQL Execution Log ==============
| SELECT first.cname, second.cname, first.rating  |
| FROM Customers first, Customers second          |
| WHERE first.rating = second.rating              |
| AND first.cname < second.cname                  |
| ============================================== |
|   cname       cname      rating                 |
|  -------   ---------   -------                  |
|  Hoffman     Pereira       100                  |
|  Giovanni    Liu           200                  |
|  Clemens     Hoffman       100                  |
|  Pereira     Pereira       100                  |
|  Gisneros    Grass         300                  |
================================================
```

- Hoffman предшествует Periera в алфавитном порядке, поэтому комбинация удовлетворяет обеим условиям предиката и появляется в выводе.
- Когда та же самая комбинация появляется в обратном порядке — когда Periera в псевдониме первой таблицы сравнивается с Hoffman во второй таблице псевдонима — второе условие не встречается.
- Аналогично, Hoffman не выбирается при наличии того же рейтинга что и он сам потому что его имя не предшествует ему самому в алфавитном порядке.
- Если бы вы захотели включить сравнение строк с ними же в запросах подобно этому, вы могли бы просто использовать <= вместо <.

ПРОВЕРКА ОШИБОК

- ПРОВЕРКА ОШИБОК Таким образом мы можем использовать эту особенность SQL для проверки определенных видов ошибок.
- При просмотре таблицы Порядков, вы можете видеть что поля cnum и snum должны иметь постоянную связь.
- Так как каждый заказчик должен быть назначен к одному и только одному продавцу, каждый раз когда определенный номер заказчика появляется в таблице Порядков, он должен совпадать с таким же номером продавца.

Команда определяет любые несогласованности в этой области.

- SELECT first.onum, first.cnum, first.snum, second.onum, second.cnum, second.snum FROM Orders first, Orders second
- WHERE first.cnum = second.cnum
- AND first.snum < > second.snum;

- Хотя это выглядит сложно, логика этой команды достаточно проста. Она будет брать первую строку таблицы Порядков, запоминать ее под первым псевдонимом, и проверять ее в комбинации с каждой строкой таблицы
- Порядков под вторым псевдонимом, одну за другой. Если комбинация строк удовлетворяет предикату, она выбирается для вывода. В этом случае предикат будет рассматривать эту строку, найдет строку где поле cnum=2008 а поле snum=1007, и затем рассмотрит каждую следующую строку с тем же самым значением поля cnum.

- Если он находит что какая-то из них имеет значение отличное от значения поля snum, предикат будет верен, и выведет выбранные поля из текущей комбинации строк.

- Если же значение snum с данным значением cnum в наш таблице совпадает, эта команда не произведет никакого вывода.

БОЛЬШЕ ПСЕВДОНИМОВ

- Хотя соединение таблицы с собой — это первая ситуация, когда понятно, что псевдонимы необходимы, вы не ограничены в их использовании что бы только отличать копию одной таблицы от ее оригинала.
- Вы можете использовать псевдонимы в любое время, когда вы хотите создать альтернативные имена для ваших таблиц в команде.
- Например, если ваши таблицы имеют очень длинные и сложные имена, вы могли бы определить простые односимвольные псевдонимы, типа a и b, и использовать их вместо имен таблицы в предложении SELECT и предикате.
- Они будут также использоваться с соотнесенными подзапросами.

ЕЩЕ БОЛЬШЕ КОМПЛЕКСНЫХ ОБЪЕДИНЕНИЙ

- Вы можете использовать любое число псевдонимов для одной таблицы в запросе, хотя использование более двух в данном предложении SELECT * будет излишеством.
- Предположим что вы еще не назначили ваших заказчиков к вашему продавцу.
- Компания должна назначить каждому продавцу первоначально трех заказчиков, по одному для каждого рейтингового значения.

- Вы лично можете решить, какого заказчика какому продавцу назначить, но следующий запрос вы используете, чтобы увидеть все возможные комбинации заказчиков, которых вы можете назначать.

```
SELECT a.cnum, b.cnum, c.cnum
FROM Customers a, Customers b, Customers c
WHERE a.rating = 100 AND b.rating = 200 AND
c.rating = 300;
```

(Вывод показывается в Рисунке 7)

РИСУНОК 7. : Комбинация пользователей с различными значениями рейтинга

SQL Execution Log

cnum	cnum	cnum
2001	2002	2004
2001	2002	2008
2001	2003	2004
2001	2003	2008
2006	2002	2004
2006	2002	2008
2006	2003	2004
2006	2003	2008
2007	2002	2004
2007	2002	2008
2007	2003	2004
2007	2003	2008

- Как вы можете видеть, этот запрос находит все комбинации заказчиков с тремя значениями оценки, поэтому первый столбец состоит из заказчиков с оценкой 100, второй с 200, и последний с оценкой 300.
- Они повторяются во всех возможных комбинациях.
- Это — сортировка группировки которая не может быть выполнена с GROUP BY или ORDER BY, поскольку они сравнивают значения только в одном столбце вывода.

- Вы должны также понимать, что не всегда обязательно использовать каждый псевдоним или таблицу которые упомянуты в предложении FROM запроса, в предложении SELECT.

- Иногда, предложение или таблица становятся запрашиваемыми исключительно потому что они могут вызываться в предикате запроса.

Например, следующий запрос находит всех заказчиков размещенных в городах где продавец Serres (snum 1002) имеет заказчиков .

```
SELECT b.cnum, b.cname
FROM Customers a, Customers b
WHERE a.snum = 1002
    AND b.city = a.city;
```

(вывод показывается в Рисунке 8)

РИСУНОК 8. Нахождение заказчиков в городах относящихся к Serres

```
================  SQL Execution Log ============
| SELECT b.cnum, b.cname                          |
| FROM Customers a, Customers b                   |
| WHERE a.snum = 1002                             |
| AND b.city = a.city;                            |
| ================================================|
|   cnum      cname                               |
| ------    ---------                             |
|   2003      Liu                                 |
|   2008      Cisneros                            |
|   2004      Grass                               |
================================================
```

- Псевдоним **a** будет делать предикат неверным за исключением случая когда его значение столбца snum = 1002. Таким образом псевдоним опускает все, кроме заказчиков продавца Serres.
- Псевдоним **b** будет верным для всех строк с тем же самым значением города что и текущее значение города для **a**; в ходе запроса, строка псевдонима **b** будет верна один раз когда значение города представлено в **a**.

- Нахождение этих строк псевдонима **b** — единственная цель псевдонима **a**, поэтому мы не выбираем все столбцы подряд.
- Как вы можете видеть, собственные заказчики Serres выбираются при нахождении их в том же самом городе что и он сам, поэтому выбор их из псевдонима a необязателен.
- Короче говоря, псевдоним находит строки заказчиков Serres, Liu и Grass.
- Псевдоним **b** находит всех заказчиков разме- щенных в любом из их городов (San Jose и Berlin соответственно) включая, конечно, самих — Liu и Grass.

- Вы можете также создать соединение которое включает и различные таблицы и псевдонимы одиночной таблицы.

- Следующий запрос объединяет таблицу Пользователей с собой: чтобы найти все пары заказчиков обслуживаемых одним продавцом.

В то же самое время, этот запрос объединяет заказчика с таблицей Продавцов с именем этого продавца.

```
SELECT sname, Salespeople.snum, first.cname,
second.cname
FROM Customers first, Customers second,
Salespeople
WHERE first.snum = second.snum
    AND Salespeople.snum = first.snum
    AND first.cnum < second.cnum;
```

(вывод показан на Рисунке 9)

РИСУНОК 9. Соединение таблицы с собой и с другой таблицей

```
================  SQL Execution Log ==================
| SELECT cname, Salespeople.snum, first.cname          |
| second.cname                                         |
| FROM Customers first, Customers second, Salespeople  |
| WHERE first.snum  = second.snum                      |
| AND Salespeople.snum = first.snum                    |
| AND first.cnum < second.cnum;                        |
| ===================================================|
|  cname      snum        cname       cname            |
|  ------   ------      --------    --------           |
|  Serres     1002        Liu         Grass            |
|  Peel       1001        Hoffman     Clemens          |
 ====================================================
```

РЕЗЮМЕ

- Теперь Вы понимаете возможности объединения и можете использовать их для ограничения связей с таблицей, между различными таблицами, или в обоих случаях.
- Вы могли видеть некоторые возможности объединения при использовании его способностей.
- Вы теперь познакомились с терминами порядковые переменные, корреляционные переменные и предложения (эта терминология будет меняться от изделия к изделию, так что мы предлагаем Вам познакомится со всеми тремя терминами).
- Кроме того Вы поняли, немного, как в действительности работают запросы. Следующим шагом после комбинации многочисленных таблиц или многочисленых копий одной таблицы в запросе, будет комбинация многочисленных запросов, где один запрос будет производить вывод который будет затем управлять работой другого запроса. Это другое мощное средство SQL.

РАБОТА С SQL

1. Напишите запрос, который бы вывел все пары продавцов, живущих в одном и том же городе. Исключите комбинации продавцов с ними же, а также дубликаты строк, выводимых в обратным порядке.
2. Напишите запрос, который вывел бы все пары порядков по данным заказчикам, именам этих заказчиков, и исключал дубликаты из вывода, как в предыдущем во- просе.
3. Напишите запрос, который вывел бы имена (cname) и города (city) всех заказчиков с такой же оценкой (rating) как у Hoffmana. Напишите запрос, использующий поле cnum Hoffmana а не его оценку, так чтобы оно могло быть использовано если его оценка вдруг изменится.