МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»

ЯЗЫК SQL. ЗАПРОСЫ НА ОСНОВЕ НЕСКОЛЬКИХ ТАБЛИЦ

Методические указания
к лабораторной работе №4
по дисциплине
«Управление данными»
для студентов, обучающихся по направлениям
09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика» по учебному плану подготовки бакалавров
дневной и заочной форм обучения

Севастополь
2020

УДК 004.92

Язык SQL. Запросы на основе нескольких таблиц. Методические указания к лабораторной работе №4 по дисциплине «Управление данными», для студентов, обучающихся по направлениям 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика» по учебному плану подготовки бакалавров дневной и заочной форм обучения /Сост. Ю.В. Доронина, А.В. Волкова – Севастополь: Изд-во СГУ, 2020. – 9 с.

Цель методических указаний: изучить способы получения информации из нескольких таблиц, способы выполнения и принцип действия рекурсивных запросов, научится использовать вложенные подзапросы.

Допущено учебно-методическим центром в качестве методических указаний.

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ

Запросы к нескольким таблицам являются наиболее часто используемым типом запросов. Реляционные БД нормализованы, и хранимая информация разбита по большому количеству таблиц**.**

Структура, используемых отношений (Salespeople, Customers, Orders) и их содержимое приведено в лабораторной работе №2 п.2.2 Тестовая база данных.

1.1 Простое соединение двух таблиц

Существует два основных способа соединения таблиц – с помощью оператора JOIN языка SQL, и с помощью условия в разделе WHERE. Например, соединить таблицу **Customers** и таблицу **Salespeople** по полю **city** можно следующим образом:

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

Для выполнения данного запроса сервер произведет ***декартово произведение*** двух таблиц, после чего выполнит операцию селекции нужных строк, затем – проекцию нужных полей. Декартово произведение – это операция, которая требует для своего выполнения максимальное количество памяти и процессорного времени, по сравнению с другими известными операциями. Данный способ неэффективен.

В случае если используется, оператор JOIN, запрос будет выглядеть так:

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

В данном случае сервер БД не будет производить декартово произведение, сервер воспользуется индексами для определения совпадающих значений поля city, выберет строки с совпадающими значениями, а затем произведет проекцию. Данная операция происходит на порядок быстрее. В случае, если по полям учавствующим в соединении, не создан индекс, сервер вынужден производить декартово произведение.

Можно сделать следующие выводы:

- если соединение таблиц происходит по ключам, выгоднее использовать JOIN;
- если запрос выполняется медленно, нужно создать индекс по полю – параметру соединения (CREATE INDEX).

1.2 Соединение более чем двух таблиц

Соединение более чем двух таблиц по форме не отличается от простого соединения. Например, чтобы соединить три таблицы по некоторому условию, можно записать:

```
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;
```

Тот же запрос, переписанный с использованием JOIN, выглядит более сложно:

```
SELECT onum, cname, Orders.cnum, Orders.snum
FROM (Orders JOIN Customers ON Orders.cnum = Customers.cnum)
   JOIN Salespeople ON Orders.snum = Salespeople.snum
WHERE
   Customers.city <> Salespeople.city;
```

1.3 Псевдонимы и рекурсивные объединения

На практике часто встречается ситуация, когда необходимо соединять таблицу саму с собой. Запрос, выполняющий такое соединение, называется рекурсивным.

Таблица «Студент» *(*Students*)*, представленная на рисунке 1, содержит следующие атрибуты:

- CNAME – имя студента;
- RATING – рейтинг студента;
- SPES – специальность.

CNAME	RATING	SPES
Иванов	42	ИС
Калинин	46	А
Петров	42	А
Сидоров	34	ИС
Шахов	46	М

Рисунок 1 – Таблица «Студент»

Например, если необходимо вывести все пары студентов с одинаковым рейтингом, можно записать следующий запрос:

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

Здесь в списке вывода SELECT указываются поля cname и rating таблицы first, и поле cname таблицы second. В разделе FROM указывается, что first и second – просто псевдонимы для таблицы Students. Для выполнения запроса сервер создаст две копии таблицы Students, одну с именем first, другую с именем second, выполнит запрос так, как будто это две разные таблицы, и уничтожит копии. Естествнно, сервер физически не создает копий таблиц, но с псевдонимами в запросе он работает так, будто это две разные таблицы.

Можно заметить, что вывод запроса будет повторять каждую пару дважды – сначала «Иванов – Петров», затем «Петров – Иванов». Кроме того, вывод содержит строки «Иванов – Иванов», «Петров – Петров». Это происходит потому, что сервер берет первую строку из первого псевдонима и сравнивает ее со всеми строками из второго псевдонима. Будут выбраны строки «Иванов – Иванов» и «Иванов – Петров». Затем он переходит к следующей строке и снова сравнивает ее со всеми строками из второго псевдонима, и так далее. Будут выбраны строки «Петров – Иванов» и «Петров – Петров».

Чтобы избежать дубликатов, надо определить еще одно условие, налагающее отношение порядка на сравниваемые строки. Например, сравнивать имена.

```
SELECT
   first.cname, second.cname, first.rating
FROM
   Students first, Students second
WHERE
   first.rating = second.rating AND first.cname < second.cname;
```

1.4 Вложенные подзапросы

Вложенные подзапросы так же служат для получения информации из нескольких таблиц. С их помощью можно выполнять рекурсивные запросы. Очень часто на практике встречаются ситуации, когда запрос выражается очень сложно через соединения, и легко – через вложенный подзапрос, и наоборот. На конкретном сервере БД запрос с использованием JOIN может выполняться очень долго, а с использованием подзапроса – быстро.

Пример запроса с вложенным подзапросом:

```
SELECT *
FROM Orders
WHERE snum =
      (SELECT snum
       FROM Salespeople
       WHERE sname = 'Motika');
```

Так как подзапрос стоит после знака равенства, он должен возвращать только одно значение. В случае если подзапрос вернет более чем одно значение, произойдет ошибка.

Внимание! При записи подзапроса допустима следующая форма:

<имя/константа> <оператор> <подзапрос>,
а не <подзапрос> <оператор> <имя/константа> или
<подзапрос> <оператор> <подзапрос>.

Во вложенных подзапросах можно использовать агрегатные функции:

```
SELECT *
FROM Orders
WHERE amt >
      (SELECT AVG(amt)
       FROM Orders
       WHERE odate = '10/04/1990');
```

Ограничение на вложенный подзапрос то же самое – он должен возвращать единственное значение.

В случае, если подзапрос возвращает несколько записей, вместо операций сравнения нужно использовать IN:

```
SELECT *
FROM Orders
WHERE snum IN
      (SELECT snum
       FROM Salespeople
       WHERE city = 'London');
```

Данный запрос более просто записывается с использованием соединения:

```
SELECT onum, amt, odate, cnum, Orders.snum
FROM Orders, Salespeople
WHERE Orders.snum = Salespeople.snum
      AND Salespeople.city = 'London';
```

Допустимо использовать выражение, основанное на столбце, а не просто сам столбец в предложении SELECT подзапроса:

```
SELECT *
FROM Customers
WHERE cnum =
 (SELECT snum + 1000
  FROM Salespeople
  WHERE sname = 'Serres');
```

Также допустимы подзапросы в выражении HAVING:

```
SELECT rating, COUNT(DISTINCT cnum)
FROM Customers
GROUP BY rating
HAVING rating >(SELECT AVG (rating)
            FROM Customers
            WHERE city = San Jose');
```

2 СОЕДИНЕНИЕ ТАБЛИЦ С ПОМОЩЬЮ ОПЕРАТОРА JOIN

Оператор JOIN используются для запроса данных из двух или нескольких таблиц связанных между собой ключами.

Ключом является столбец (или комбинация столбцов) с уникальным значением для каждой строки. Каждое значение первичного ключа должно быть уникальным в пределах таблицы. Цель состоит в том, чтобы связать данные всех таблиц вместе, не повторяя все данные в каждой таблице.

Существуют следующие типы JOIN, доступные к использованию в языке SQL:

– JOIN (или INNER JOIN) – возвращает строки, когда есть хотя бы одно совпадение в обеих таблицах;

– LEFT JOIN – возвращает строки из левой таблицы, даже если их нет в правой таблице;

– RIGHT JOIN – возвращает строки из правой таблицы, даже если их нет левой таблице;

– FULL JOIN – возвращает строки, когда есть хоть одно совпадение в любой из таблиц;

– CROSS JOIN – возвращает все возможные сочетания каждой строки с каждой.

Примеры использования различных типов оператора JOIN будем рассматривать для таблиц Persons и Orders.

Есть таблица «Persons»:

Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Есть таблица «Orders»:

O_Id	OrderNo	P_Id
1	77895	3
2	44678	3
3	22456	1
4	24562	1
5	34764	15

Столбец «Id» является первичным ключом в таблицы «Persons».

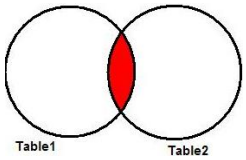
Столбец «O_Id» является первичным ключом в таблицы «Orders» и колонка «P_Id» относится к колонке «Id» в таблице «Persons».

Таким образом, связь между таблицами обеспечивается с помощью колонок «Id» и «P_Id».

2.1 INNER JOIN (естественное соединение)

INNER JOIN возвращает строки, когда есть хотя бы одно совпадение в обеих таблицах. INNER JOIN это тоже что и JOIN. Синтаксис оператора INNER JOIN следующий:

```
SELECT column_name(s)
FROM tab1
[INNER] JOIN tab2
ON tab1.column_name = tab2.column_name
```



Пример использования INNER JOIN: из таблиц Persons и Orders необходимо выбрать все лица, имеющие какие-либо заказы. Для этого используется следующий запрос:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
INNER JOIN Orders ON Persons.Id = Orders.P_Id
ORDER BY Persons.LastName
```

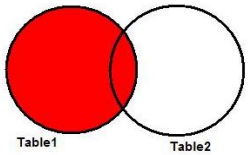
Результат запроса:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678

2.2 LEFT JOIN

LEFT JOIN возвращает строки из левой таблицы (tab1), даже если их нет в правой таблице (tab2). В некоторых базах данных LEFT JOIN имеет имя LEFT OUTER JOIN. Синтаксис оператора LEFT JOIN следующий:

```
SELECT column_name(s)
FROM tab1
LEFT [OUTER] JOIN tab2
ON tab1.column_name = tab2.column_name
```



Пример использования LEFT JOIN: теперь из таблиц Persons и Orders необходимо получить список всех лиц и их заказов. Для этого используется следующий й запрос:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
LEFT JOIN Orders ON Persons.Id = Orders.P_Id
ORDER BY Persons.LastName
```

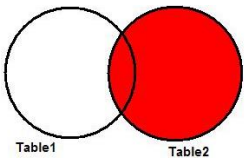
Результат запроса:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	77895
Pettersen	Kari	44678
Svendson	Tove	

2.3 RIGHT JOIN

RIGHT JOIN возвращает строки из правой таблицы (tab2), даже если их нет левой таблице (tab1). В некоторых базах данных RIGHT JOIN имеет имя RIGHT OUTER JOIN. Синтаксис оператора RIGHT JOIN следующий:

```
SELECT column_name(s)
FROM tab1
RIGHT [OUTER] JOIN tab2
ON tab1.column_name = tab2.column_name
```



Пример использования RIGHT JOIN: теперь из таблиц Persons и Orders необходимо получить список всех лиц и их заказов. Для этого используется следующий запрос:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
RIGHT JOIN Orders ON Persons.Id = Orders.P_Id
ORDER BY Persons.LastName
```

Результат запроса:

LastName	FirstName	OrderNo
		34764
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	44678
Pettersen	Kari	77895

2.4 CROSS JOIN (декартово произведение)

CROSS JOIN – это объединение SQL по которым каждая строка одной таблицы объединяется с каждой строкой другой таблицы. Это объединение редко требуется. Синтаксис оператора CROSS JOIN следующий:

```
SELECT column_name(s)
FROM tab1
CROSS JOIN tab2
```

Пример использования CROSS JOIN: из таблиц Persons и Orders необходимо вывести все возможные сочетания каждой строки из таблицы «Persons» с каждой строкой таблицы «Orders». Для этого используется следующий запрос:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
CROSS JOIN Orders
```

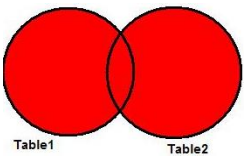
Результат запроса:

LastName	FirstName	OrderNo
Hansen	Ola	77895
Hansen	Ola	44678
Hansen	Ola	22456
Hansen	Ola	24561
Hansen	Ola	34764
Svendson	Tove	77895
Svendson	Tove	44678
Svendson	Tove	22456
Svendson	Tove	24561
Svendson	Tove	34764
Pettersen	Kari	77895
Pettersen	Kari	44678
Pettersen	Kari	22456
Pettersen	Kari	24561
Pettersen	Kari	34764

2.5 FULL JOIN

FULL JOIN возвращает строки, когда есть хоть одно совпадение в любой из таблиц. Синтаксис оператора FULL JOIN следующий:

```
SELECT column_name(s)
FROM tab1
FULL [OUTER] JOIN tab2
ON tab1.column_name = tab2.column_name
```



Пример использования FULL JOIN: из таблиц Persons и Orders необходимо получить список всех людей и заказов. Для этого используется следующий запрос:

```
SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo
FROM Persons
FULL JOIN Orders ON Persons.Id = Orders.P_Id
ORDER BY Persons.LastName
```

Результат запроса:

LastName	FirstName	OrderNo
		34764
Hansen	Ola	22456
Hansen	Ola	24562
Pettersen	Kari	44678
Pettersen	Kari	77895
Svendson	Tove	

3 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Записать запросы, соединяющие две таблицы с помощью JOIN и без него.
2. Записать запросы, соединяющие более чем две таблицы с помощью JOIN и без него.
3. Продемонстрировать следующие возможности SQL:
   - использование псевдонимов на примере рекурсивного запроса.
   - привести пример запроса с подзапросом
   - использование агрегатных функций в подзапросе
   - подзапросы, возвращающие единственное и множественные значения
   - подзапросы, использующие вычисление
   - использование подзапросов в HAVING
4. Написать отчет.

4 СОДЕРЖАНИЕ ОТЧЕТА

1. Отчет состоит из титульного листа, цели работы, описания процесса выполнения работы и вывода.
2. Отчет должен содержать исходные данные, тексты запросов и результаты их выполнения.

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Операции реляционной алгебры (операторной формы записи)?
2. Бинарные операторы над отношениями, операторная форма записи?
3. В чем различие соединения таблиц по условию и с использованием JOIN?
4. Свойства операции соединения?
5. В чем различие вложенных запросов и запросов с соединением?
6. Какие формы записи подзапроса недопустимы?
7. В чем особенность подзапроса, перед которым стоит знак арифметического сравнения?