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

ЯЗЫК SQL. КОРРЕЛИРОВАННЫЕ ВЛОЖЕННЫЕ ПОДЗАПРОСЫ

Методические указания
к лабораторной работе №5
по дисциплине
«Теория баз данных»
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. Коррелированные вложенные подзапросы. Методические указания к лабораторной работе №5 по дисциплине «Теория баз данных», 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. – 6 с.

Цель методических указаний: ознакомится с принципом работы коррелированных подзапросов.

Допущено учебно-методическим центром в качестве методических указаний.

1 ОСНОВНЫЕ ПОЛОЖЕНИЯ

1.1 Коррелированные вложенные подзапросы

Вложенные подзапросы и операция JOIN предоставляют большие возможности по манипулированию данными из нескольких таблиц. Существует еще один способ - использовать коррелированные подзапросы, он более сложен для понимания, но в некоторых случаях позволяет формулировать запросы самым коротким образом. Кроме того, есть некоторые вещи, доступные для соединения и недоступные для вложенных подзапросов, и наоборот.

Например, при использовании подзапроса становятся возможным использовать агрегатные функции в предикате запроса. При использовании соединений в выводе запроса могут участвовать поля из всех таблиц соединения.

Коррелированным называется подзапрос, который использует псевдоним таблицы определенный не во вложенном запросе, а во внешнем. При этом вложенный запрос выполняется много раз - для каждой строки внешней таблицы. Пусть требуется найти всех покупателей, совершивших покупки 10.03.1990. Можно записать следующий коррелированный запрос:

```
SELECT *
FROM Customers outer
WHERE 10.03.1990 IN
 (SELECT odate
   FROM Orders inner
   WHERE outer.cnum = inner.cnum);
```

Логика работы запроса такова: внутренний запрос производит соединение таблицы покупателей и таблицы покупок (outer.cnum = inner.cnum). Для тех строк, у которых условие WHERE выполняется, запрос возвращает дату покупки. Таким образом, для каждого покупателя формируется множество дней, когда он делал покупки. Полученное множество передается во внешний запрос. В случае если 10.03.1990 входит в сформированное множество, внешний запрос возвращает всю строку (*) строку из таблицы Customers.

Приведем процесс выполнения коррелированного подзапроса в алгоритмической форме:

1. Выбрать строку из таблицы во внешнем запросе (строка-кандидат).
2. Сохранить значения полей строки-кандидата в псевдониме.
3. Выполнить подзапрос. Везде, где найдена ссылка на псевдоним из внешнего запроса, подставлять значения из текущей строки-кандидата. Использование значения из строки-кандидата называется внешней ссылкой.
4. Оценить предикат внешнего запроса на основе результатов подзапроса. Если предикат верен - строка выбирается для вывода
5. Перейти на следующую строку во внешнем запросе.

Подобный запрос можно гораздо легче записать другим способом. Например:

```
SELECT DISTINCT *
FROM Customers, Orders
WHERE Customers.cnum = Orders.cnum AND Orders.odate = 10.03.1990;
```

Более показательным является следующий пример. Пусть необходимо вывести имена и номера всех продавцов, которые имеют более одного заказчика

```
SELECT snum, sname
FROM Salespeople main
WHERE 1 <
 (SELECT COUNT (*)
   FROM Customers
   WHERE snum = main.snum );
```

В случае, если перед именем поля не указывается имя псевдонима, то данное поле относится к текущему подзапросу. Таким образом, поле snum во вложенном подзапросе относится к

таблице Customers, а не к Salespeople. В случае, если такого поля в соответствующей таблице нет, оно ищется в подзапросе верхнего уровня. Так, если бы поле snum отсутствовало в таблице Customers, оно относилось бы к Salespeople.

1.2 Соотнесение таблицы со своей копией

Коррелированные запросы можно писать, основываясь только на одной таблице. Данное свойство делает их незаменимыми при написании аналитических запросов (т.е. запросов, производящих сложный анализ данных). Например, пусть нам необходимо найти все покупки со значениями суммы покупки выше среднего значения для каждого покупателя (т.е. надо найти все покупки данного покупателя, найти среднюю сумму покупок, и выдать все покупки со стоимостью выше средней):

```
SELECT *
FROM Orders outer
WHERE amt > =
 (SELECT AVG (amt)
   FROM Orders inner
   WHERE inner.cnum = outer.cnum);
```

Существуют так называемые специальные операторы SQL, они имеют смысл только для подзапросов. Отличительная особенность специальных операторов - они принимают подзапрос как аргумент, точно так же, как это делает IN.

1.3 Оператор EXISTS

EXISTS(X) - булевский оператор. Он получит значение «истинна», если запрос X вернет хоть одну строку. Допустим, нам необходимо узнать список продавцов, у которых есть более одного покупателя

```
SELECT DISTINCT snum
FROM Customers outer
WHERE EXISTS
 (SELECT *
   FROM Customers inner
   WHERE inner.snum = outer.snum AND inner.cnum <> outer.cnum );
```

Для каждой строки-кандидата из внешнего запроса (продавец, проверяемый в настоящее время), внутренний запрос находит строки, у которых совпадают значения поля snum (номер продавца), но не совпадают значение поля cnum (номер покупателя). Если не указать DISTINCT, каждый продавец будет выбран один раз для каждого своего заказчика.

На практике очень часто приходится использовать EXISTS вместе с NOT. Допустим, нам необходимо вывести список продавцов, за которыми числится только один покупатель

```
SELECT DISTINCT snum
FROM Customers outer
WHERE NOT EXISTS
 (SELECT *
   FROM Customers inner
   WHERE inner.snum = outer.snum AND inner.cnum <> outer.cnum );
```

Кроме EXISTS в подзапросах возможно использование еще двух операторов - ANY и ALL.

1.4 Оператор ANY

Оператор ANY становится верным, если значение из верхнего подзапроса совпадает, по крайней мере, с одним значением из вложенного подзапроса. Например, если значение - кандидат равно 1, а вложенный подзапрос вернул {1, 2, 3}, оператор ANY станет верным, (внешний

запрос проверяет на равенство). Например, если нам нужно найти всех продавцов, которые живут в тех же городах, что и покупатели, можно записать следующий запрос:

```
SELECT *
FROM Salespeople
WHERE city = ANY
 (SELECT city
   FROM Customers);
```

Существует синоним оператора ANY – SOME. Так как SQL похож на английский, то одни предложения, верно, звучат с ANY, другие - с SOME. Действие операторов эквивалентно.

ANY используется в сочетании со знаками равенства < (или <=), > (или >=), =:

– > ANY (>= ANY) означает больше (больше или равно), по крайней мере, одного значения или, что равносильно, *больше (больше или равно) минимального значения* из списка. Например, > ANY (1,2,3) означает больше 1;

– < ANY (<= ANY) означает меньше (меньше или равно), по крайней мере, одного значения или, что равносильно, *меньше (меньше или равно) максимального значения* из списка. Например, < ANY (1,2,3) означает меньше 3;

= ANY означает проверку существования, поэтому он эквивалентен условию IN. Однако, квантор != ANY будет не равносилен условию NOT IN. Квантор != ANY означает «не a, или не в, или не с», в то время как условие NOT IN означает «не a, и не в, и не с».

1.5 Оператор ALL

Вторым допустимым оператором является ALL. Действие его противоположно оператору ANY. Оператор ALL становится верным, если все значения из вложенного подзапроса равны значению-кандидату из внешнего запроса. Например, если значение - кандидат равно 1, а вложенный подзапрос вернул {1, 1, 1}, оператор ALL станет верным, (внешний запрос проверяет на равенство).

Например, если необходимо найти всех продавцов, у которых рейтинг выше, чем у любого продавца из Рима, можно записать следующий запрос:

```
SELECT *
FROM Customers
WHERE rating > ALL
 (SELECT rating
   FROM Customers
   WHERE city = Rome );
```

Операторы ANY и ALL можно выразить через EXISTS в коррелированном подзапросе, в явном виде они нужны лишь для упрощения записи запроса. Обратное утверждение не верно - т.е. не все то, что можно выполнить с помощью EXISTS, можно сделать с помощью ANY и ALL.

ALL используется в сочетании со знаками равенства < (или <=), > (или >=), =:

– < ALL (<= ALL) означает, что выражение должно быть меньше (меньше или равно) каждого значения из списка. Это условие выполняется в случае, если выражение *меньше (меньше или равно) минимального значения* из списка;

– > ALL (>= ALL) означает, что выражение должно быть больше (больше или равно) каждого значения из списка. Это условие выполняется в случае, если выражение *больше (больше или равно) максимального значения* из списка. В контексте подзапроса квантор > ALL означает, что текущая строка будет удовлетворять условию, указанному во внешнем запросе, если значение в указанном столбце будет больше всех значений, которые возвращаются подзапросом. Например, > ALL (1, 2, 3) означает больше чем 3;

– = ALL – означает, что выражение должно быть равно каждому значению из списка (конечно, это выполнится только в случае, если в списке одно значение или несколько одинаковых значений). Текущая строка будет удовлетворять условию, указанному во внешнем запросе, если значение в сравниваемом столбце будет равно каждому значению, которое возвращается подзапросом.

- != ALL – эквивалентено условию NOT IN.

ALL чаще используется с неравенствами, нежели с равенствами, так как значение может быть «равным для всех» результатом подзапроса, только если все результаты фактически идентичны.

Замечание 1: Когда говорят, что значение больше (или меньше) чем любое (ANY) из набора значений, это то же самое, что сказать, что оно больше (или меньше) чем любое отдельно взятое из этих значений. И наоборот, сказать, что некоторое значение не равно всему (ALL) набору значений, это то же самое, что сказать, что в наборе нет такого же значения.

Замечание 2: В случае если подчиненный запрос вернул пустое множество, оператор ALL становится верным, а ANY - ложным.

2 ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Записать запрос, соединяющий таблицу со своей копией.
2. Привести пример коррелированного запроса, использующего две разные таблицы.
3. Продемонстрировать следующие возможности SQL
   - работу оператора EXISTS;
   - работу оператора ALL;
   - работу оператора ANY.
4. Написать отчет.

3 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.

4 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Коррелированные вложенные подзапросы?
2. Для чего таблицам назначается псевдоним? Приведите примеры назначения и использования псевдонимов в запросах.
3. Операторы: EXISTS, ALL, ANY – назначение, правила использования?
4. Какой из операторов EXISTS, ALL, ANY являются альтернативой друг другу? Приведите примеры.