ЛАБОРАТОРНАЯ РАБОТА №5
РАЗРАБОТКА СЛОЖНЫХ ЗАПРОСОВ К БАЗЕ ДАННЫХ. ЗАПРОСЫ НА ОСНОВЕ НЕСКОЛЬКИХ ТАБЛИЦ. КОРРЕЛИРОВАННЫЕ ВЛОЖЕННЫЕ ПОДЗАПРОСЫ.

5.1 Цель работы

Выполнение лабораторной работы преследует следующие цели:

– изучить способы получения информации из нескольких таблиц;

– записать запросы, демонстрирующие выборки из нескольких таблиц с использованием оператора JOIN и без него;

– изучить способы выполнения и принцип действия рекурсивных запросов;

– научится использовать вложенные подзапросы;

– ознакомиться с возможностями построения вложенных коррелированных подзапросов с применением кванторов;

– ознакомиться с возможностью формирования отчетов в клиентском приложении.

5.2 Введение

Запросы к нескольким таблицам являются наиболее часто используемым типом запросов. Существует два основных способа объединения таблиц – с помощью оператора JOIN языка SQL, и с помощью условия в разделе WHERE. Анализировать информацию из нескольких таблиц в запросах можно путем использования запросов с подзапросами.

5.3 Порядок выполнения работы

1) Для заданной вариантом базы данных разработать и выполнить с помощью SQL-редактора PHPMyAdmin запросы, соединяющие две таблицы с помощью JOIN и без него.

2) Разработать и выполнить с помощью SQL-редактора PHPMyAdmin запросы, соединяющие более чем две таблицы с помощью JOIN и без него.

3) Продемонстрировать с помощью SQL-редактора PHPMyAdmin 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) Ознакомится с принципом и продемонстрировать работу коррелированных подзапросов:

- привести пример соединения таблицы со своей копией;
- привести пример коррелированного запроса, использующего две разные таблицы;
- привести пример запроса с оператором EXIST;
- привести пример запроса с оператором ALL;
- привести пример запроса с оператором ANY.

5) Разработать и выполнить с помощью SQL-редактора PHPMyAdmin обязательные запросы, заданные вариантом (таблица 5.1).

Дополнительно:

6) Модифицировать приложение лабораторной работы №4. Добавить окно для вывода результатов сложных запросов, предусмотренных вариантом задания.

5.4 Варианты заданий

Варианты структуры базы данных совпадают с вариантами к лабораторной работе №4.

Таблица 5.1 – Варианты заданий к лабораторной работе №5

№ вар.	Запросы к базе данных
1.	1. Вывести всех пассажиров поезда № 28 «Севастополь-Киев», выехавших 29.01.2007 2. Вывести все поезда, которые не следуют через станцию «Джанкой» 3. Вывести время отправления первой (по времени) электрички «Севастополь-Симферополь».
2.	1. Вывести Ф.И.О. студентов факультета АВТ 2. Вывести Ф.И.О. студентов, которые не изучают программирование. 3. Вывести Ф.И.О. студентов, получающих максимальную стипендию
3.	1. Вывести всех осужденных, проходящих по делу №26. 2. Вывести всех осужденных, которые не имеют псевдонима. 3. Вывести Ф.И.О. осужденного на максимальный срок
4.	1. Вывести всех служащих, имеющих высшее образование. 2. Вывести Ф.И.О. сотрудников, которые не имеют детей. 3. Вывести Ф.И.О. самого молодого мужчины.
5.	1. Вывести Ф.И.О. клиентов гостиницы «Крым». 2. Вывести гостинцы, в которых нет номеров «Люкс». 3. Вывести все гостиницы максимального разряда.

№ вар.	Запросы к базе данных
6.	1. Вывести Ф.И.О. всех авторов, писавших на тему «Локальные вычислительные сети». 2. Вывести любого автора, который печатался в журнале «Микропроцессорные системы». 3. Вывести Ф.И.О. автора самой последней (по дате) публикации.
7.	1. Вывести всех сотрудников строительных фирм. 2. Вывести всех сотрудников, у которых нет детей 3. Вывести сотрудников с максимальным окладом
8.	1. Вывести всех служащих, говорящих на английском языке 2. Вывести Ф.И.О. служащих, которые не были в отпуске в 2007 г. 3. Вывести Ф.И.О. служащего с максимальным окладом
9.	1. Вывести всех студентов, слушающих курс «СУБД». 2. Вывести Ф.И.О. преподавателей, не имеющих своего курса. 3. Вывести Ф.И.О. преподавателя, который провел последнее (по дате) занятие.
10.	1. Вывести название ВУЗов, осуществляющих подготовку по специальности «компьютерные сети». 2. Вывести Ф.И.О. студентов, которые не сдали ни одного экзамена. 3. Вывести название специальности с минимальным сроком обучения.
11.	1. Вывести все препараты для лечения гриппа. 2. Вывести список диагнозов, для которых существует нетрадиционный метод лечения. 3. Вывести диагноз с минимальной стоимостью лечения.
12.	1. Вывести всех пациентов хирурга Иванова. 2. Вывести всех хирургов, которые ведут хотя бы одного пациента. 3. Вывести Ф.И.О. самого старшего пациента.
13.	1. Вывести все предметы, которые изучают на факультете АВТ 2. Вывести всех преподавателей, которые не преподают на факультете АВТ. 3. Вывести название предметов с максимальным количеством часов.
14.	1. Вывести всех спортсменов, игравших во Франции. 2. Вывести всех спортсменов, которые не играют в Динамо (Киев). 3. Вывести список лучших бомбардиров (забивших максимальное количество мячей).
15.	1. Вывести все телефоны фирмы «Оптима Крым» 2. Вывести названия фирмы, которая не имеет контрагентов 3. Вывести название фирмы, занимающей помещение с максимальной площадью

№ вар.	Запросы к базе данных
16.	1. Вывести инженеров, обслуживающих машины типа Pentium IV 2. Вывести название адрес ВЦ, в которых нет машин типа Macintosh. 3. Вывести список инженеров с минимальным окладом
17.	1. Вывести Ф.И.О. служащих, работающих над проектом «Победа» 2. Вывести названия работ, в которых задействованы программисты 3. Вывести минимальную тарифную ставку
18.	1. Вывести всех водителей, возивших грузы более 10 тонн. 2. Вывести Ф.И.О. любого водителя, который водит КАМАЗ. 3. Вывести название груза максимальной массы.
19.	1. Вывести сумму партвзносов, полученных партией «Зеленых» в 2006 году. 2. Вывести все партии, которые имеют свою газету. 3. Вывести название газеты с максимальным тиражем.
20.	1. Вывести всех артистов, выступивших в «Доме офицеров» 31.12.2006. 2. Вывести всех артистов, которые не выступали в «Альбервил-холле». 3. Вывести название зала с максимальным количеством мест.
21.	1. Вывести Ф.И.О. всех осужденных, дата осуждения которых не меньше 01.01.2000г. 2. Вывести даты рождения всех осужденных, которые не имеют псевдонима. 3. Вывести № и Ф.И.О. осужденного на минимальный срок
22.	1. Вывести побочные эффекты препаратов для лечения пневмонии. 2. Вывести наименования и стоимость курсов лечения, для которых используется медицинское оборудование. 3. Вывести метод лечения с максимальной стоимостью лечения.
23.	1. Вывести номера зачетных книжек студентов факультета ИТиУТС, специальности ИТиКС. 2. Вывести номера групп студентов, которые изучают дискретную математику. 3. Вывести Ф.И.О. преподавателей, имеющих максимальное количество часов по курсам.
24.	1. Вывести номера паспортов клиентов гостиницы «Мрия», проживающих в ней более трех дней. 2. Вывести номера и названия гостиниц, в которых нет трехкомнатных номеров. 3. Вывести все фирмы, имеющие гостиницы максимального разряда.

№ вар.	Запросы к базе данных
25.	1. Вывести даты всех экзаменов, студентов факультета ИТиКС 2. Вывести Ф.И.О. студентов, которые сдают экзамен программирование преподавателю по фамилии Иванов. 3. Вывести Ф.И.О. преподавателей, читающих максимальное количество предметов у студентов факультета ИТиУТС.
26.	1. Вывести даты рождения всех служащих, проживающих в Севастополе. 2. Вывести Ф.И.О. сотрудников, которые имеют 2 и более детей. 3. Вывести дату рождения самого пожилого мужчины.
27.	1. Вывести названия команд, участвовавших в соревнованиях 2000г. 2. Вывести Ф.И.О. и даты рождения спортсменов команды, не пропустившей мячей. 3. Вывести список аутсайдеров (забивших минимальное количество мячей) на соревнованиях 2000г.
28.	1. Вывести Ф.И.О. всех служащих, окончивших СевГУ 2. Вывести Ф.И.О. и даты рождения служащих, которые поступили на работу до 2007 г. включительно. 3. Вывести Ф.И.О. служащего с минимальным размером премии.
29.	1. Вывести все телефоны фирмы с банковским счетом 1211001… 2. Вывести адреса фирм, имеющих складские помещения 3. Вывести контактное лицо фирмы, занимающей торговое помещение минимальной площади.
30.	1. Вывести Ф.И.О. руководителей партий, имеющих свою газету. 2. Вывести профессии партийцев из партии «ЛДПР», которые не заплатили партийные взносы за два месяца. 3. Вывести название помещения минимальной площади, которое занимает партия «Зеленых».

5.5 Содержание отчета

Отчет должен содержать следующие разделы:

1) Титульный лист определенного образца.
2) Тема, цель работы, вариант задания (полностью представленная структура данных).
3) Ход работы:

– исходные данные (концептуально-логическая схема БД, операторы создания всех таблиц БД на языке SQL из л/р №4);

– тексты всех запросов на выборку, обозначенных в разделе 5.3 в соответствии заданием, сформулированные на естественном языке и структурированном языке запросов SQL;

– результаты выполнения запросов (таблицы до и после выполнения запросов);

– *дополнительно*: описание клиентского приложения).

Все запросы должны иметь логический смысл. Без формулировки запросов на естественном языке отчет приниматься не будет.

4) Выводы.

5.6 Контрольные вопросы

1) В чем различие соединения таблиц по условию и с использованием JOIN?
2) В чем различие вложенных запросов и запросов с соединением?
3) Какие формы записи подзапроса недопустимы?
4) В чем особенность подзапроса, перед которым стоит знак арифметического сравнения?
5) Что такое коррелированный подзапрос?
6) Опишите алгоритм выполнения запроса с коррелированным подзапросом.
7) Назначение операторов EXIST, ALL, ANY.

5.7 Рекомендации по выполнению работы

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

5.7.1 Простое объединение двух таблиц

Существует два основных способа объединения таблиц – с помощью оператора JOIN языка SQL, и с помощью условия в разделе WHERE. Например, объединить таблицу Customers и таблицу Salespeople из базы данных, рассмотренной в лабораторной работе №3, 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 мы используем 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 объединение таблиц происходит по ключам, выгоднее использовать JOIN;

– если запрос выполняется медленно, нужно создать индекс по полю – параметру объединения (CREATE INDEX)

5.7.2 Объединение более чем двух таблиц

Объединение более чем двух таблиц по форме не отличается от простого объединения. Например, чтобы объединить 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 AND
```

5.7.3 Псевдонимы и рекурсивные объединения

На практике часто встречается ситуация, когда необходимо объединять таблицу саму с собой. Запрос, выполняющий такое объединение, называется рекурсивным. Например, если мы хотим вывести все пары студентов с одинаковым рейтингом, можно записать следующий запрос:

```
SELECT one.cname, two.cname, one.rating
FROM Students one, Students two
```

WHERE one.rating = two.rating;

Здесь в списке вывода SELECT указываются поля cname и rating таблицы one, и поле cname таблицы two. В разделе FROM указывается, что one и two – просто псевдонимы для таблицы Students. Для выполнения запроса сервер создаст две копии таблицы Students, одну с именем one, другую с именем two, выполнит запрос так, как будто это две разные таблицы, и уничтожит копии. Естественно, 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 наложить еще одно условие, налагающие отношение порядка на сравниваемые строки. Например, сравнивать имена.

```
SELECT one.cname, two.cname, one.rating
FROM Students one, Students two
WHERE one.rating = two.rating AND
      one.cname < two.cname;
```

5.7.4 Вложенные подзапросы

Вложенные подзапросы так же служат для получения информации из нескольких таблиц. С их помощью можно выполнять рекурсивные запросы. Для чего существует два способа сделать одно и то же действие? Дело в том, что 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.

Обратите внимание, что 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');
```

5.7.5 Построение коррелированных вложенных подзапросов

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 объединений в выводе запроса могут участвовать поля из всех таблиц объединения.

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.

Примечание. – Если внешний запрос и внутренний подзапрос обращаются к разным таблицам, то использование псевдонимов не обязательно.

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.

Конечно, 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 нам 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.

5.7.6 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.

5.7.7 Оператор (квантор) 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 значениея поля cnum (номер покупателя). Если не указать DISTINCT, каждый продавец будет выбран один раз для каждого своего заказчика.

На практике очень часто приходится использовать EXIST 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.

5.7.8 Оператор ANY

Условие с 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}, условие с оператором 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.

5.7.9 Оператор ALL

Действие оператора ALL 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 нам 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 EXIST в коррелированном подзапросе, в явном виде они нужны лишь для упрощения записи запроса. Обратное утверждение неверно, т.е. не все то, что можно выполнить с помощью EXIST, можно сделать с помощью ANY и ALL.

Примечание 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.

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