ЛАБОРАТОРНАЯ РАБОТА №3.
ЭЛЕМЕНТАРНЫЕ ВЫБОРКИ ПОСРЕДСТВОМ ОПЕРАТОРА SELECT. АГРЕГАТНЫЕ ФУНУКЦИИ SQL

Цель работы

Целью лабораторной работы является изучение работы оператора SELECT языка SQL, а также изучение возможностей обработки данных с помощью агрегатных функций языка SQL.

Введение

Большинство операций с базой данных заключается в выборке из нее необходимой пользователю информации. Для этого служит оператор SELECT, позволяющий формировать набор данных, удовлетворяющий условиям поиска и группировки данных. В том числе, оператор SELECT используется для подведения итогов, получения статистической информации и т.д.

Порядок выполнения работы:

1) Изучить рекомендации по выполнению работы и определить вариант задания.
2) Создать SQL-запрос, который отражает все строки таблицы, указанной в варианте задания.
3) Создать SQL-запрос, задающий порядок столбцов, отличный от исходного.
4) С помощью SQL-запроса продемонстрировать действие модификатора DISTINCT.
5) Ограничить вывод данных с помощью SQL-запроса, используя WHERE с простым условием.
6) Ограничить вывод данных с помощью SQL-запроса, используя WHERE и составное условие.
7) С помощью SQL-запроса продемонстрировать действие специальных функций IN, BETWEEN, LIKE и IS NULL в условии.
8) С помощью SQL-запроса продемонстрировать работу специальных функций с условием NOT.
9) Ознакомиться с принципами работы агрегатных функций COUNT, SUM, AVG, MAX, MIN. Составить SQL-запрос с одной из агрегатных функций.
10) С помощью SQL-запроса продемонстрировать использование COUNT(*).
11) С помощью SQL-запроса продемонстрировать выполнение простых вычислений в запросе.

12) Использовать простое вычисление, как параметр агрегатной функции, в SQL-запросе.

13) Ознакомиться с использованием предложения GROUP BY, продемонстрировать его работу с помощью SQL-запроса.

14) Ознакомиться с использованием предложения HAVING, продемонстрировать его работу с помощью SQL-запроса.

15) С помощью SQL-запросов выполнить следующие задания по варианту (табл.3.1):

- получить результат агрегатной функции по полю группировки, (функция и поле группировки указаны в варианте задания);
- осуществить фильтрацию информации по заданному полю;
- выполнить сортировку информации по возрастанию или убыванию значений в поле, выбранном пользователем из списка всех полей таблицы (направление сортировки задано вариантом задания).

Все запросы формулировать на естественном языке и структурированном языке запросов SQL. Проверить работу вышеупомянутых запросов с помощью SQL-редактора программы phpMyAdmin.

Дополнительно:

16) Модифицировать приложение, полученное в результате выполнения лабораторной работы №2 с целью добавления возможностей получения информации, удовлетворяющей различным условиям выборки, согласно варианти и п.15

Варианты заданий

Запросы на выборку формулируются к базе данных, определенной вариантом задания к первой и второй лабораторным работам. В таблице 3.1 приводится имя таблицы БД, к которой необходимо осуществлять запросы, а также данные для выполнения п.15: поле, по которому должна осуществляться фильтрация, поле группировки и агрегатная функция для реализации запроса с группировкой, направление сортировки строк.

Студент может **добавить в таблицу дополнительные неключевые поля**, если сочтет это полезным для смысловой наглядности разрабатываемых запросов.

Таблица 3.1 – Варианты заданий к лабораторной работе №3

№ варта	№ структуры	Имя таблицы	Поле фильтра	Поле группировки	Агрегатная функция	Направление сортировки
1	**2**	**3**	**4**	**5**	**6**	**7**
1	10	Поезд	Маршрут	Станция отправления	Мин. время прибытия	По убыванию

№ варта	№ структуры	Имя таблицы	Поле фильтра	Поле группировки	Агрегатная функция	Направление сортировки
1	**2**	**3**	**4**	**5**	**6**	**7**
2	8	Студент	Ф.И.О.	Номер группы	Макс. стипендия в каждой группе	По возрастанию
3	12	Персона	Ф.И.О.	Пол	Среднее кол-во детей	По убыванию
4	4	Личность	ФИО	Пол	Самый старший каждого пола	По возрастанию
5	17	Гостиница	Адрес	Разряд	Количество гостиниц	По убыванию
6	1	Публикация	Заглавие	Тип	Последняя дата для данного типа	По возрастанию
7	15	Сотрудник	Ф.И.О.	Пол	Количество сотрудников	По убыванию
8	7	Труд. деятельность	Номер служ.	Тип события	Кол-во событий каждого типа	По возрастанию
9	2	Курс	Название курса	Номер преподавателя	Количество курсов у преп.	По убыванию
10	9	Экзамен	Наименов. ВУЗа	Код специальности	Средняя оценка	По возрастанию
11	18	Курс лечения	Метод лечения	Номер диагноза	Средняя стоимость	По убыванию
12	3	Операция	Название операции	Номер патента	Количество операций	По возрастанию
13	11	Преподаватель	Ф.И.О.	Должность	Кол-во преподав. в должности	По убыванию
14	13	Результат соревнований	Название команды	Номер соревнования	Максим. число забитых мячей	По возрастанию
15	5	Помещение	Адрес	Номер фирмы	Максимальная площадь	По убыванию
16	19	Комплект	Наименован.	Номер машины	Макс. Стоимость	По возрастанию

№ вар-та	№ струк-туры	Имя таблицы	Поле фильтра	Поле группи-ровки	Агрегатная функция	Направление сортировки
1	**2**	**3**	**4**	**5**	**6**	**7**
17	6	Служа-щий	Ф.И.О.	Код про-фессии	Число служ. по профес-сиям	По убыва-нию
18	16	Пере-возка	Место назначе-ния	Дата	Средняя цена	По возраста-нию
19	20	Газета	Наимено-ван.	Тираж	Мин. Кол-во сотрудников	По убыва-нию
20	14	Концерт-ный зал	Адрес	Количе-ство мест	Число залов	По возраста-нию
21	12	Персона	Ф.И.О.	Пол	Макс. кол-во детей	По убыва-нию
22	18	Курс ле-чения	Метод лечения	Номер диагноза	Мин. стои-мость	По возраста-нию
23	8	Студент	Ф.И.О.	Номер группы	Ср. стипендия в каждой группе	По убыва-нию
24	17	Гости-ница	Адрес	Разряд	Количество гостиниц	По возраста-нию
25	11	Препода-ватель	Ф.И.О.	Долж-ность	Кол-во препо-дав. в долж-ности	По убыва-нию
26	4	Личность	ФИО	Пол	Самый млад-ший каждого пола	По возраста-нию
27	13	Результат соревно-ваний	Название команды	Номер соревно-вания	Ср. число за-битых мячей	По убыва-нию
28	7	Труд. дея-тельность	Номер служ.	Тип со-бытия	Кол-во собы-тий каждого типа	По возраста-нию
29	5	Помеще-ние	Адрес	Номер фирмы	Мин. пло-щадь	По убыва-нию
30	20	Газета	Наимено-ван.	Тираж	Макс. кол-во сотрудников	По возраста-нию

Содержание отчета

1. Титульный лист определенного образца.
2. Тема, цель работы, вариант задания (полностью представленная структура данных).
3. Ход работы:
   - полное описание действий студента по открытию базы данных и вводу данных, с использованием интерфейсных свойств оболочки (экранные формы с фактическими значениями (по варианту задания), вводимые в диалоговые окна);
   - обязательное представление и подробное описание всех SQL-запросов (описание включает в себя таблицу до выполнения запроса, сам SQL-запрос, таблица после выполнения запроса);
   - *дополнительно*: текст программы.
4. Выводы.

Контрольные вопросы

1) Опишите основные конструкции языка SQL, использованные в лабораторной работе SELECT, DISTINCT, WHERE, IN, BETWEEN, LIKE.
2) Каким образом можно добавить поле в существующую таблицу, внести значения в существующие записи для этого поля по умолчанию?
3) Как ввести в запрос на выборку вычисляемого поля?
4) Для чего используются агрегатные функции. Что является аргументом агрегатной функции. Перечислите агрегатные функции языка SQL.
5) С какой целью используются предложения GROUP BY и HAVING?

Рекомендации по выполнению работы

Агрегатные функции SQL

Агрегатные функции – это функции, которые работают не с одним значением, взятым из строки таблицы, а с группой значений. Например, рассмотрим таблицу 3.2, в которой хранится информация о продажах товаров:

Таблица 3.2 – Продажи товаров

Дата продажи	№ товара	Группа товара	Цена покупки
11.12.89	1	1	10.00
11.12.89	2	1	20
11.12.89	3	2	15
12.12.89	1	1	100
12.12.89	2	1	50

Группировать поля таблицы можно многими способами. Например, можно сгруппировать продажи по дням, тогда сумма поля «Цена покупки» даст общую выручку по дням. Группируя по номеру товара, и суммируя по тому же полю получим сумму продаж товара с таким-то номером за все дни.

Группируя таблицу по группам товара, получим сумму покупок всех товаров данной группы за все дни. В нашем примере мы вычисляем сумму от некоторой группы, сумма в данном случае – агрегатная функция. Группировать можно не только по одному полю, но и по нескольким.

В случае, если запрос использует агрегатные функции, то в списке полей, которые выводит запрос, могут быть только поля – аргументы агрегатных функций. Все остальные поля обязательно должны быть перечислены в предложении GROUP BY (см. ниже).

Общий алгоритм, по которому работают агрегатные функции следующий: сначала таблица упорядочивается по тем полям, по которым осуществляется группировка; затем от каждой сформированной группы вычисляется требуемое значение, которое и заносится в результат.

Существуют следующие агрегатные функции:

- COUNT считает количество строк, которые вернул запрос;
- SUM суммирует значения данного поля;
- AVG находит среднее значение поля;
- MAX находит максимальное значение поля;
- MIN находит минимальное значение поля.

Рассмотрим работу агрегатных функций на примере базы данных, состоящий из трех таблиц – «Торговый агент», «Покупатель», «Сделка». Таблица 3.3 «Торговый агент» (Salespeople) содержит следующие столбцы:

- SNUM – номер агента;
- SNAME – имя агента;
- CITY – город, где работает агента;
- COMM – комиссионные торгового агента.

Таблица 3.3 – Торговый агент (Salespeople)

SNUM	SNAME	CITY	COMM
1001	Peel	London	0.12
1002	Serres	San Jose	0.13
1004	Motika	London	0.11
1007	Rifkin	Barcelona	0.15
1003	Axelrod	New York	0.10

Таблица 3.4 «Покупатель» (заказчик) состоит из следующих столбцов:

- CNUM – номер покупателя;
- CNAME – имя покупателя;
- CITY – город проживания покупателя;
- RATING – некоторый рейтинг, присвоенный покупателю;
- SNUM – номер торгового агента, за которым закреплен покупатель.

Таблица 3.4 – Покупатель (Customers)

CNUM	CNAME	CITY	RATING	SNUM
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003

2003	Liu	SanJose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	SanJose	300	1007
2007	Pereira	Rome	100	1004

Таблица 3.5 «Сделка» (заказ) содержит следующие столбцы:

- ONUM – номер заказа;
- AMT – сумма заказа;
- ODATE – дата заказа;
- CNUM – номер покупателя;
- SNUM – номер торгового агента.

Таблица 3.5 – Сделка (Orders)

ONUM	AMT	ODATE	CNUM	SNUM
3001	18.69	10/03/1990	2008	1007
3003	767.19	10/03/1990	2001	1001
3002	1900.10	10/03/1990	2007	1004
3005	5160.45	10/03/1990	2003	1002
3006	1098.16	10/03/1990	2008	1007
3009	1713.23	10/04/1990	2002	1003
3007	75.75	10/04/1990	2004	1002
3008	4723.00	10/05/1990	2006	1001
3010	1309.95	10/06/1990	2004	1002
3011	9891.88	10/06/1990	2006	1001

Ниже приведены тексты запросов к базе данных.

Чтобы найти сумму всех покупок в таблице «Сделки», мы можем ввести следующий запрос: **SELECT SUM** ((*amt*)) **FROM** *Orders*;

Результат его выполнения – число 26658.4

Запрос «**SELECT AVG** (*amt*) **FROM** *Orders*;» вернет среднее значение сумм сделок -2665.84.

Функция **COUNT** отличается от других агрегатных функций тем, что она не выполняет математических действий над значением столбца. Она считает число значений в данном столбце, или число строк в таблице. Если необходимо подсчитать количество различных значений некоторого поля в таблице, функцию COUNT надо использовать с DISTINCT. Например, чтобы подсчитать количество продавцов в настоящее время описанных в таблице сделок, мы можем использовать следующий запрос:

SELECT COUNT (**DISTINCT** *snum*) **FROM** *Orders*;

Результат – 5.

Иногда возникает необходимость решить обратную задачу – подсчитать количество значений поля вместе с повторениями. Для этого существует описатель **ALL** (он подразумевается по умолчанию). Например:

SELECT COUNT (**ALL** *rating*) **FROM** *Customers*;

Подсчитает количество повторений поля rating с повторениями.
Чтобы подсчитать общее число строк в таблице, используйте функцию COUNT со звездочкой вместо имени поля, как, например, в следующем примере.

SELECT COUNT (*) **FROM** *Customers*;

Результат – 7.
Внимание! Только **COUNT** (*) может подсчитывать значения NULL. Все остальные функции игнорируют неопределенные значения.

Простые вычисления.

Столбцы, значение которых может быть получено с помощью простых арифметических действий через другие столбцы в БД, как правило, не хранятся. SQL предоставляет простой способ производить подобные вычисления. Также можно помещать в некоторый столбец константу. Например, чтобы представить комиссионные торгового агента в процентном отношении, а не в виде десятичного числа можно записать такой запрос:

SELECT *snum, sname, city, comm**100 **FROM** *Salespeople* **FROM** *Salespeople*;

Можно дополнить наш запрос знаком процента, выводимым после поля *comm**100:
SELECT *snum, sname, city*, *comm**100, ' % ' **FROM** *Salespeople*;

Знак процента в данном случае – константное выражение. При выполнении вычислений в запросе допустимы арифметические действия – сложение, вычитание, умножение, деление.

Вычисления в агрегатных функциях.

Допустим, мы хотим вывести 10% от стоимости самой дорогой сделки. Тогда можно записать такой запрос:

SELECT MAX (amt*0.1) **FROM** *Orders*;

Таким образом можно выполнять вычисления с использованием других полей и арифметических действий. Вложение агрегатов не допускается.

Использование GROUP BY.

Как уже было упомянуто ранее, таблицу можно группировать самыми разными способами. В случае, если мы не группируем таблицу вообще, в списке вывода запроса присутствуют только агрегатные функции. Таблица в этом случае рассматривается целиком. В случае, если мы группируем таблицу по какому-то полю, в списке вывода могут присутствовать те поля, относительно которых таблица группируется и агрегатные функции. При этом все поля, не охваченные агрегатными функциями, должны быть перечислены в предложении GROUP BY.

Предположим, что мы хотим найти сделку с максимальной стоимостью для каждого торгового агента. Можно сделать персональный запрос для каждого из них, выбрав MAX(amt) из таблицы сделок. Можно записать следующий запрос:

SELECT *snum*, **MAX** (*amt*) **FROM** *Orders* **GROUP BY** *snum*;

Результат выполнения запроса представлен в таблице 3.6:

Таблица 3.6

snum	
1001	767.19
1002	1713.23
1003	75.75
1014	1309.95
1007	1098.16

В данном случае таблица группируется относительно номера торгового агента, поэтому поле *snum* присутствует в списке вывода запроса и в предложении GROUP BY.

Использование HAVING.

HAVING подобен WHERE – он задает условия отбора групп так же, как это делает WHERE для строк. Например, если мы хотим узнать, какие торговые агенты имеют заработок от одной сделки более чем 3000, и в какой день, можно использовать следующий запрос:

SELECT *snum, odate*, **MAX** ((*amt*))
FROM *Orders*
GROUP BY *snum, odate*
HAVING MAX ((*amt*)) > 3000.00;
Результат работы запроса представлен в таблице 3.7.

Таблица 3.7

Snum	odate	------
1001	10/05/1990	4723.00
1001	10/06/1990	9891.88
1002	10/03/1990	5160.45

Внимание! В предложении HAVING нельзя проверять имена полей на какое-либо условие – для этого существует WHERE. То, что вы проверяете в HAVING должно иметь только одно значение для группы.

Использование условий выборки и агрегатных функций SQL в клиентском приложении

В клиентское приложение, разработанное в предыдущей лабораторной работе добавлены возможности выполнения некоторых запросов на выборку данных из таблицы phlib, которая содержит данные об абонентах, а именно:

- выбрать (подсчитать) число абонентов, родившихся в каждую дату, присутствующую в БД (запрос с группировкой);

- выбрать абонетов, личные номера которых входят в заданный диапазон (границы диапазона задаются пользователем в соответствующих текстовых полях);

- выбрать (фильтровать) абонентов, ФИО которых начинается с подстроки, указанной в соответствующем текстовом поле;

- выбрать (сортировать) всех абонентов в порядке возрастания значений в поле таблицы, имя которого совпадает со значением выбранного элемента в списке выбора.

Для этого в соответствующую форму в файле phlib.php введены дополнительные кнопки, текстовые поля и список выбора. Вид окна показан на рисунке 3.1.

Обработчик нажатия каждой добавленной кнопки должен изменять запрос на выборку данных. Для задания параметров запросов используются соответствующие текстовые поля и список выбора.

Текст файла phlib.php с внесенными изменениями (выделены жирным шрифтом) приведен в приложении.

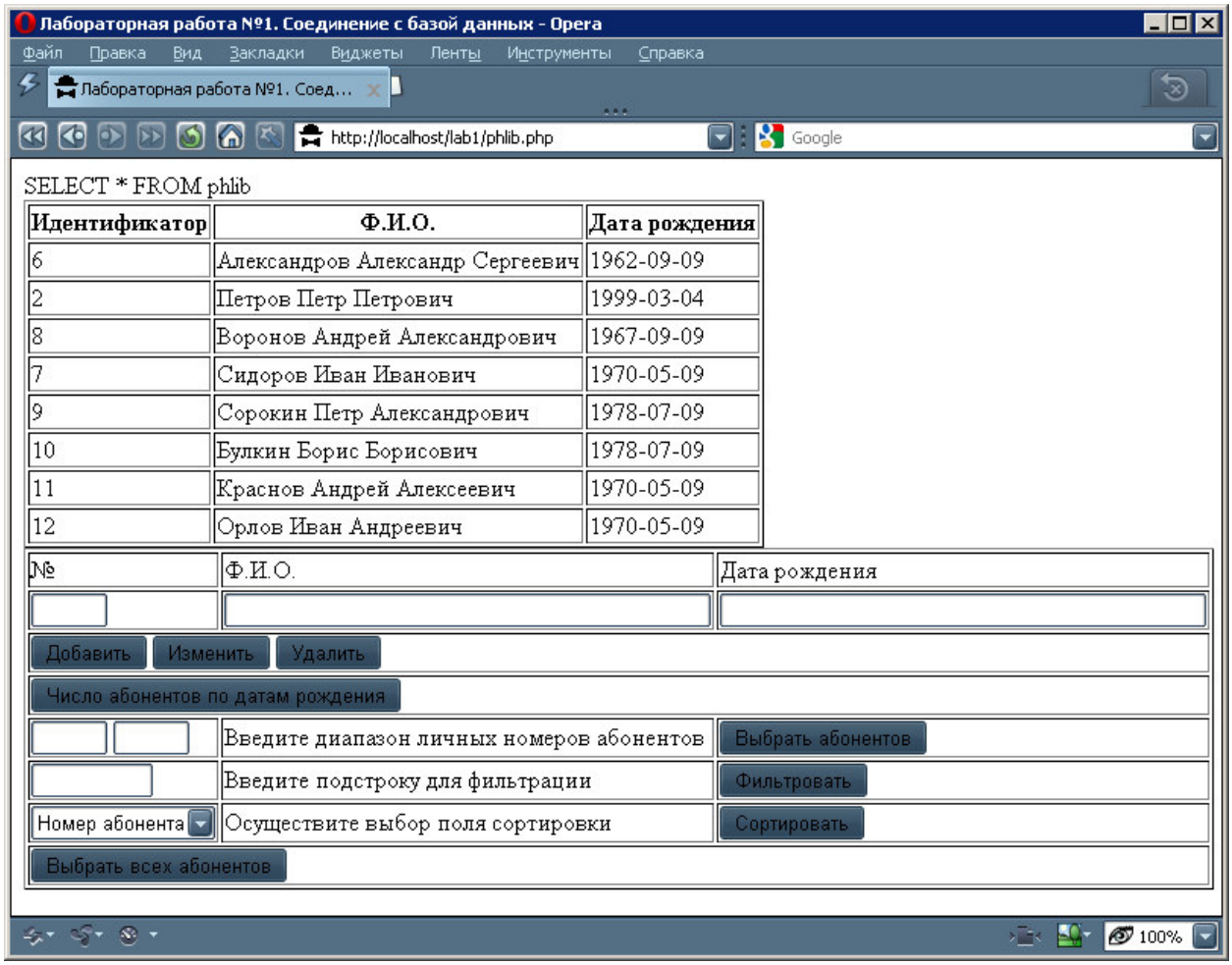

Рисунок 3.1 – Окно, открывающееся при запуске файла phlib.php

ПРИЛОЖЕНИЕ

Текст файла phlib.php

```
<HTML>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<meta name="Author" content="Сергеев Георгий Георгиевич">
<META NAME="ROBOTS" CONTENT="ALL">
<META NAME="Keywords" CONTENT="лабораторная работа, MySQL, соединение с базой данных">
<META NAME="Description" CONTENT="Лабораторная работа №1. Соединение с базой данных">
</head>
<title>Лабораторная работа №1. Соединение с базой данных</title>
<BODY>
 <?
/* Переменные для соединения с базой данных */
$hostname = "localhost";
$username = "root";
$password = "";
$dbName = "lab1";
/* Таблица MySQL, в которой хранятся данные */
$userstable = "phlib";
/* создать соединение */
mysql_connect($hostname,$username,$password) OR DIE("Не могу создать соединение ");
/* выбрать базу данных. Если произойдет ошибка - вывести ее */
mysql_select_db($dbName) or die(mysql_error());

/* Обработка данных, полученных методом post */
if ($_POST['add'])
{
echo "Выполнен запрос: Добавить ".$_POST['my_name']." ".$_POST['my_data']."<BR>";
$sql = mysql_query("INSERT  into phlib(FIO,bdate)
values ('".$_POST['my_name']."', '".$_POST['my_data']."');");
}
if ($_POST['delete'])
{
  echo "Выполнен запрос: Удалить ".$_POST['my_id']."<BR>";
  $sql = mysql_query("DELETE  from phlib where ID=".$_POST['my_id']);
}
if ($_POST['update'])
{
echo "Выполнен запрос: Изменить ".$_POST['my_id']. " на ".$_POST['my_name']."
".$_POST['my_data']."<BR>";
$sql = "UPDATE phlib  SET FIO='".$_POST['my_name']."', bdate='".$_POST['my_data']."'
where ID=".$_POST['my_id'];
  echo $sql."<BR>";
  mysql_query($sql) OR DIE ("Не могу выполнить запрос");
}
```

```
/* Составить запрос для выборки информации */
if ($_POST['count_1'])
   {$query = "SELECT bdate, count(*) as count_col FROM $userstable group by bdate
order by bdate";
    echo $query."<BR>";
   /* Выполнить запрос. */
    $result=mysql_query($query) or die(mysql_error());
    echo "<TABLE BORDER=1>";
    echo"<TR><TH>Дата</TH><TH>Число родившихся</TH></TR>";
   /* Выбрать очередную запись из таблицы. */
    while($row=mysql_fetch_array($result))// берем
    //результаты из каждой строки
    {
    /* Вывести ее в виде HTML*/
     echo "<tr><td>".$row['bdate']."</td> <td>".$row['count_col']."</td> </tr>";
     };
     echo "</TABLE>";}
   else
{
if ($_POST['between_1'])
{$query = "SELECT * FROM $userstable where ID between ".$_POST['min_1']." and
".$_POST['max_1'];
 echo $query."<BR>";}
else if ($_POST['filter'])
       {$query = "SELECT * FROM $userstable where FIO like
'".$_POST['my_filter']."%'";
        echo $query."<BR>";}
    else if ($_POST['case_1'])
           {$query = "SELECT * FROM $userstable order by ".$_POST['case_1'];
            echo $query."<BR>";}

         else {$query = "SELECT * FROM $userstable";
              echo $query."<BR>";}
/* Выполнить запрос. */
$result=mysql_query($query) or die(mysql_error());
echo "<TABLE BORDER=1>";
echo"<TR><TH>Идентификатор</TH><TH>Ф.И.О.</TH><TH>Дата рожде-
ния</TH></TR>";
/* Выбрать очередную запись из таблицы. */
while($row=mysql_fetch_array($result))// берем
//результаты из каждой строки
{
/* Вывести ее в виде HTML*/
echo "<tr><td>".$row['ID']."</td> <td>".$row['FIO']."</td>
<td>".$row['bdate']."</td></tr>";
};
echo "</TABLE>";
}
/* Закрыть соединение */
mysql_close();
?>
<form action="phlib.php" method="post">
```

```
<table border="1" align="left">
<tr> <td>№</td> <td>Ф.И.О.</td> <td>Дата рождения</td> </tr>
<tr>
   <td valign="top"><input name="my_id" type="text"
      size="5" /></td>
   <td valign="top"><input name="my_name" type="text"
      size="50" /></td>
   <td valign="top"><input name="my_data" type="text"
      size="50" /></td>
</tr>
<tr><td colspan=3>
      <input name="add" type="submit" value="Добавить" />
      <input name="update" type="submit" value="Изменить" />
      <input name="delete" type="submit" value="Удалить" />
</td> </tr>

<tr>
<td colspan=3>
      <input name="count_1" type="submit" value="Число абонентов по датам рожде-
ния" />
</td> </tr>
<tr>
<td>
      <input name="min_1" type="text"  size="5" />
      <input name="max_1" type="text"  size="5" />
</td>
<td> Введите диапазон личных номеров абонентов </td>
<td>
      <input name="between_1" type="submit" value="Выбрать абонентов" />
</td>
</tr>

<tr>
   <td valign="top"><input name="my_filter" type="text"
      size="10" /> </td>
   <td> Введите подстроку для фильтрации </td>
   <td valign="top">  <input name="filter" type="submit" value="Фильтровать" />
</td> </tr>
<tr>
<td>
<!тег поместит на форму элемент "список выбора"!>
<select name=case_1>
<option value="ID">Номер абонента
<option value="FIO">ФИО
<option value="bdate">Дата рождения
</select>
</td>
<td> Осуществите выбор поля сортировки </td>
<td valign="top">  <input name="sort" type="submit" value="Сортировать" />
</td>
</tr>
<tr>
```

<td colspan=3 valign="center"> <input name="refr" type="submit" value="Выбрать всех абонентов" />
</tr>
</td>
</table>
</form>

</BODY>
</HTML>