ЛАБОРАТОРНАЯ РАБОТА №2.
СРЕДСТВА МАНИПУЛИРОВАНИЯ ДАННЫМИ ЯЗЫКА SQL.

Цель работы:

Изучение операторовINSERT, UPDATE, DELETE и простейших форм оператора SELECT.Изучение методов разработки клиентских приложений для ввода и коррекции данных.

Введение

К базовым средствам манипулирования данными языка SQL относятся «поисковые» варианты операторов UPDATE и DELETE. Эти варианты называются поисковыми, потому что при задании соответствующей операции задается логическое условие, налагаемое на строки адресуемой оператором таблицы, которые должны быть подвергнуты модификации или удалению. Кроме того, в такую категорию языковых средств входит оператор INSERT, позволяющий добавлять строки в существующие таблицы.

Порядок выполнения работы

1) Открыть созданную в л/р№1 базу данных, содержащую одну таблицу в соответствии с вариантом задания.
2) Создать в таблице не менее 10 записей произвольного содержания. В 4-5 записях содержимое поля должно соответствовать условию, указанному в варианте задания, используя интерфейсные свойства оболочки.
3) Вызвать диалог SQL-редактора программы phpMyAdmin. Ввести 4-5 запросов для добавления записи в таблицу:
   - запрос на добавление одного кортежа в таблицу;
   - запрос на добавление одновременно нескольких кортежей в таблицу.
4) Изменить содержимое полей последней введенной записи с помощью SQL-запроса.
5) Изменить содержимое любого поля всех записей, удовлетворяющих условию, с помощью SQL-запроса.
6) Удалить все записи, удовлетворяющие условию, с помощью SQL-запроса.

Дополнительно:

7) Разработать клиентское приложение на языке скриптов PHP для ввода, отображения и коррекции данных таблицы.

Варианты заданий

Варианты задания выбираются в соответствии с порядковым номером фамилии студента в журнале преподавателя. Номер варианта, название таблицы и условие, которому должны соответствовать обновляемые и удаляемые данные указаны в таблице 2.1.

Таблица 2.1 – Варианты заданий к лабораторной работе №2

№ вар-та	Имя таблицы	Условие	№ струк-туры
1.	Поезд	Станция прибытия=4	10
2.	Студент	Стипендия> 100	8
3.	Персона	Количество детей = 2	12
4.	Личность	Пол=’мужской’	4
5.	Гостиница	Разряд = 3	17
6.	Публикация	Тип=’статья’	1
7.	Сотрудник	Оклад > 600	15
8.	Трудовая деятельность	Тип события =’увольнение’	7
9.	Курс	N преподавателя=1	2
10.	Экзамен	N предмета=3	9
11.	Курс лечения	Стоимость < 300	18
12.	Операция	Название Операции = 'Аппендицит’	3
13.	Преподаватель	Должность = 'Ассистент’	11
14.	Результат соревнований	Количество забитых мячей = 2	13
15.	Помещение	Площадь < 20	5
16.	Комплектующие	Стоимость > 200	19
17.	Служащий	Код профессии = 3	6
18.	Перевозка	Место назначения = ‘Москва’	16
19.	Газета	Количество сотрудников > 20	20
20.	Концертный зал	Количество мест = 200	14
21.	Персона	Семейное положение = 'Женат’	12
22.	Курс лечения	Метод лечения = ‘Хирургия’	18
23.	Студент	Номер группы = 41	8
24.	Гостиница	Разряд > 1	17
25.	Преподаватель	Научное звание = 'Доцент’	11
26.	Личность	Пол=’женский’	4
27.	Результат соревнований	Количество пропущенных мячей > 1	13
28.	Трудовая деятельность	Дата = 21.01.2012	7
29.	Помещение	Площадь > 15	5
30.	Газета	Тираж < 10000	20

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) Какие типы данных MySQL Вы знаете?
2) Типы ключей и цели их использования в таблицах БД.
3) Назначение и типы индексов в таблицах БД.
4) Схемы индексирования таблиц.
5) Основные виды связей таблиц в БД и их характеристика.
6) Контроль целостности связей.
7) Какие операции манипулирования с данными Вы знаете?
8) Опишите синтаксис оператора INSERT.
9) Опишите синтаксис оператора UPDATE.
10) Опишите синтаксис оператора DELETE.
11) Каким образом можно определить условия отбора записей для изменения, удаления?
12) Что обозначает символ % в теле строки-образца?

Рекомендации по выполнению работы

Базовые средства манипулирования данными

Оператор INSERT добавляет одну или более новых строк данных к существующей таблице или виду. INSERT - одна из привилегий базы данных, которая контролируется инструкциями GRANT и REVOKE.

Значения вставляются в столбцы строки по порядку их следования, если не задан факультативный список столбцов. Если список столбцов задан на множестве всех доступных столбцов, во все неперечисленные столбцы автоматически вставляется или значение по умолчанию, или NULL.

Если факультативный список столбцов упущен, предложение VALUES должно содержать значения для всех столбцов таблицы.

Чтобы вставить одну строку данных, должно присутствовать предложение VALUES и содержать определенный список значений.

Чтобы вставить несколько строк данных, определите <select_expr>,

которое возвращает уже существующие данные из другой таблицы. Выбранные строки должны соответствовать списку столбцов.

Предупреждение: Допустимо выбирать из той же таблицы, в которую строки вставляются, но эта практика не рекомендуется, так как подобные действия могут привести к бесконечным вставкам строк (зацикливанию).

Синтаксис:

```
INSERT INTO <object> [(col [, col ...])]
{VALUES (<val> [, <val> ...]) | <select_expr>};
<object> = tablename | viewname
<val> = {<constant> | <expr>| <function> | NULL | USER} [COLLATE collation]
```

Обратите внимание: Предложение COLLATE не может быть использовано для BLOB значений.

<constant> = num | "string" | charsetname "string"

<expr> = Допустимое выражение SQL, которое возвращает в одиночное значение столбца.

<function> = {CAST (<val> AS <datatype>)

| UPPER (<val>) | GEN_ID (generator, <val>) }

<select_expr> = SELECT возвращающий ноль или более строк, где число столбцов в каждой строке такое же, как число элементов, которые должны быть вставлены.

INTO <object> Имя существующей таблицы или вида, в которую вставляются данные.

col Имя существующего столбца в таблице или виде, в который вставляются значения.

VALUES (<val> [, <val> ...] Список значению дляя вставки в таблицу или вид. Значения должны быть в том же порядке, как целевые столбцы.

<select_expr> Запрос, который возвращает значения, для вставки в целевые столбцы.

Пример:

Следующая инструкция добавляет строку в таблицу, присваивает значения двум столбцам:

```
INSERT INTO EMPLOYEE_PROJECT (EMP_NO, PROJ_ID) VALUES (52, "DI");
```

Оператор UPDATE изменяет одну или более существующих строк в таблице или виде. Факультативное предложение WHERE может быть использовано, чтобы ограничить UPDATE к некоторому подмножеству строк таблицы. Модификации не могут модифицировать секторы массива. Предупреждение: Если предложение WHERE упущено, UPDATE изменяет все строки в таблице. Обратите внимание: Когда модифицируются BLOB столбцы, UPDATE заменяет весь BLOB целиком новым значением.

Синтаксис:

```
UPDATE {table | view}SET col = <val> [, col = <val> ...] [WHERE
```

```
<search_condition>;<val> = {col [<array_dim>] | <constant> | expr> |
<function>  | NULL | USER  }<array_dim> = [x:y [, x:y ...]]
```

Обратите внимание: внешние скобки должны присутствовать в ссылке на массив.

<constant> = num | "string" | charsetname "string"

<expr> = Допустимое выражение SQL, которое возвращает одиночное значение.

<function> = {CAST (<val> AS <datatype>) | UPPER (<val>) | GEN_ID (generator, <val>) }

table | view - Имя существующей таблицы или вида для модификции.

SET col = <val> Определяет столбцы для изменения и значения, которые требуется присвоить этим столбцам.

WHERE <search_cond> Модифицировать только найденное. Определяет условия, которым строка должна удовлетворять, чтобы изменится.

Примеры:

Следующая инструкция изменяет столбцы для всех строк таблицы:

```
UPDATE CITIES SET POPULATION = POPULATION * 1.03;
```

Следующая инструкция использует предложение WHERE, чтобы ограничить модификацию столбцов подмножеством строк:

```
UPDATE COUNTRY SET CURRENCY = "USDollar" WHERE COUNTRY = "USA";
```

Оператор DELETE определяет одну или более строк, чтобы удалить из таблицы или редактируемого (updatable) вида. DELETE - одна из привилегий, которые контролируются инструкциями GRANT и REVOKE.

Для определения строк, которые следует удалить, может быть использовано факультативное предложение WHERE.

Предостережение: Если не используется предложение WHERE, удалены все строки из таблицы.

Синтаксис:

```
DELETE FROM table [WHERE <search_condition>];
```

Table - Имя таблицы из которой удаляются строки.

<search_condition> Условия поиска, которые определяют строки для удаления. Если это предложение не используется, DELETE воздействует на все строки в определенной таблице или виде.

Примеры:

Следующая инструкция удаляет все строки из таблицы:

```
DELETE FROM EMPLOYEE_PROJECT;
```

Следующая инструкция удаляет строку для служащего #141:

```
 DELETE FROM SALARY_HISTORY WHERE EMP_NO = 141;
```

Манипулирование данными таблицы с помощью phpMyAdmin.

Среда phpMyAdmin предоставляет возможности интерактивного манипулирования данными. Рассмотрим этот процесс на примере редактирования данных таблицы, разработанной в предыдущей лабораторной работе.

Для этого запускаем среду и выбираем из перечня нашу базу данных (рисунок 2.1).

Рисунок 2.1 – Выбор базы данных для редактирования

Далее выбираем для редактирования таблицу phonelib и нужную запись для редактирования (рисунок 2.2).

Рисунок 2.2 – Выбор таблицы и записи для редактирования

Выполняем редактирование записи. Обратите внимание на текст SQL-скрипта для обновления строки таблицы (рисунок 2.3).

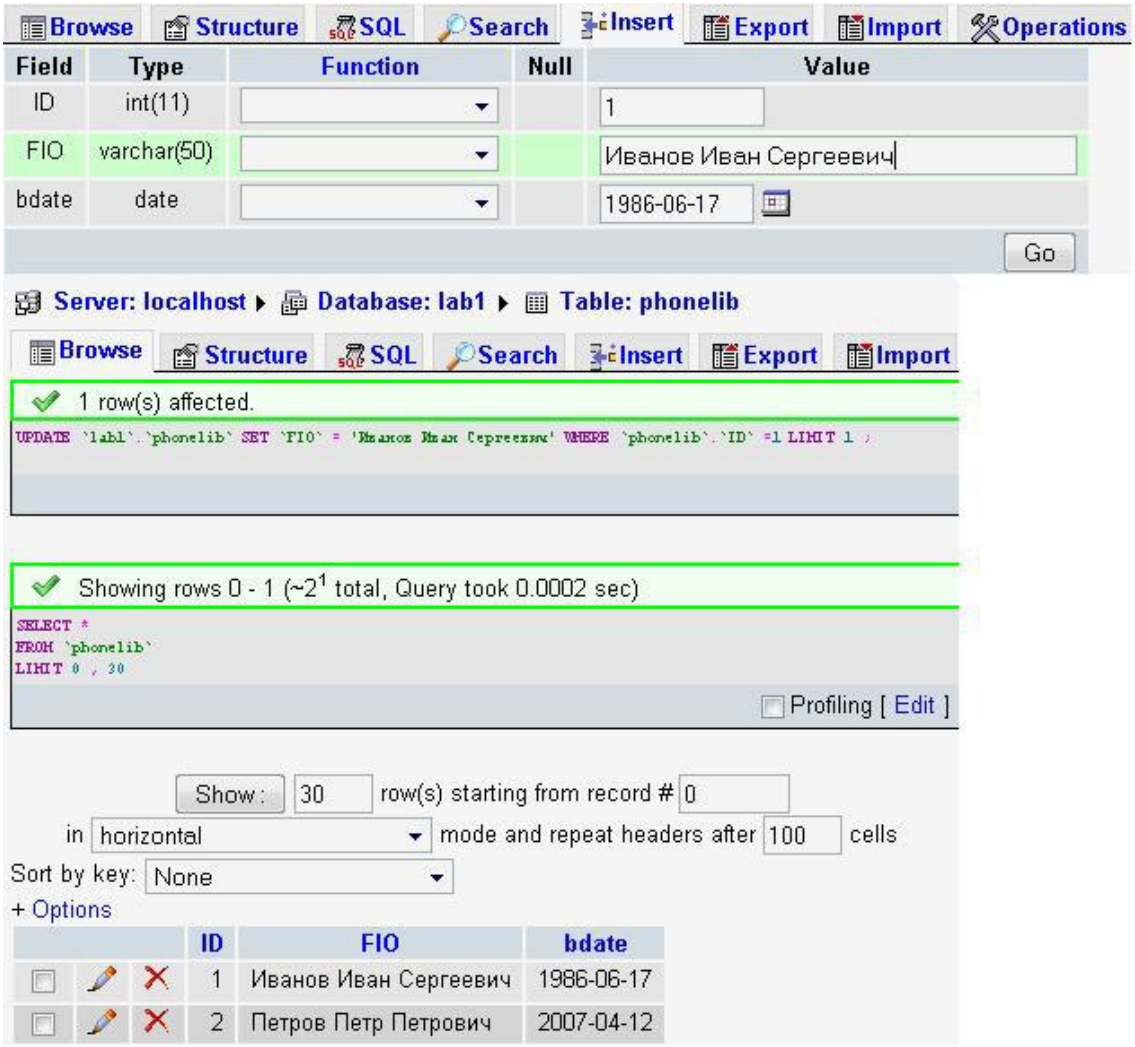

Рисунок 2.3 – Редактирование записи в среде phpMySQLAdmin

Клиентское приложение для манипулирования данными

Модифицируем клиентскоеWEB-приложение, разработанное в лабораторной работе №1. Для управления таблицей добавим после PHP-скрипта в файле phlib.php форму, в которой будет три текстовых поля (inputtype=”text”) для ввода номера, Ф.И.О. и даты рождения, а также три кнопки (inputtype=”submit”) – «Добавить», «Редактировать» и «Удалить». Отметим, что при нажатии кнопок форма будет отправлять данные методом post скрипту, расположенному в том же самом файле (action="phplib.php").

```
<form action="phplib.php" method="post">
<table border="1" align="left">
<tr><td>W</td><td>Ф.И.О.</td><td>Датарождения</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>
</table>
</form>
```

Форма ввода имеет вид, показанный на рисунке 2.4.

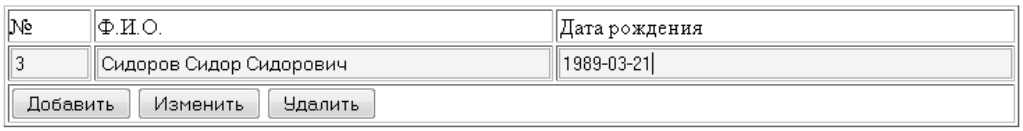

Рисунок 2.4 – Форма модификации данных таблицы

Разработаем обработчик события по нажатию кнопок «Добавить», «Изменить» и «Удалить». Будем считать, что поля редактирования заполнены должным образом, и нам необходимо перенести из них информацию в базу данных.

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

Располагаем описанный выше скрипт после оператора выбора базы данных (mysql_select_db). Примерный результат работы приложения представлен на рисунке 2.5.

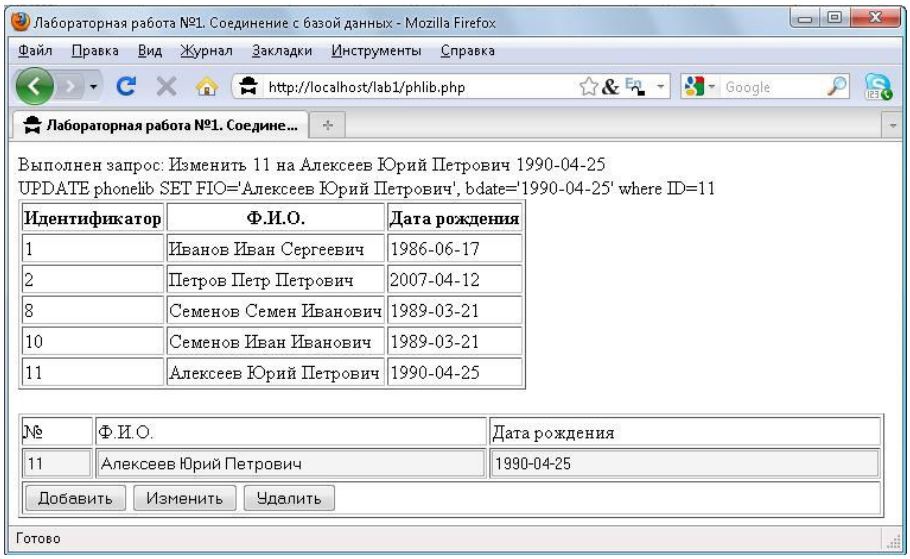

Рисунок 2.5 – Результат работы приложения