ЛАБОРАТОРНАЯ РАБОТА №4
ССЫЛОЧНАЯ ЦЕЛОСТНОСТЬ ДАННЫХ.
СОЗДАНИЕ СХЕМЫ БАЗЫ ДАННЫХ.

4.1 Цель работы

Ознакомиться с принципами построения схемы базы данных и обеспечения целостности данных в базе.

4.2 Введение

Свое название реляционные базы данных (РБД) получили, в том числе, потому, что таблицы в БД не существуют независимо друг от друга. Таблицы взаимосвязаны друг с другом, т.е. действие, произведенное в одной таблице, вызовет некоторые действия в другой таблице. Существует три основных класса связей между таблицами: один к одному (1:1), один ко многим (1:M), и многие ко многим (M:M). На практике связи первого типа используются редко. Связи третьего типа не реализуются в РБД напрямую, одну связь многие ко многим приводят к двум связям один ко многим.

4.3 Порядок выполнения работы

1) Проанализировать схему БД своего варианта задания (вариант тот же, что и в лабораторной работе №1), выделить и классифицировать **все существующие связи**, определить необходимые ограничения целостности.
2) Создать еще не созданные таблицы, изменить существующие таким образом, чтобы они могли участвовать в связях (с помощью оператора ALTER TABLE).
3) В процессе создания таблиц установить связи между таблицами.
4) Составить запросы на ввод данных в главную и подчиненную таблицу. Проверить работу ограничений на значения первичного ключа обеих таблиц и внешнего ключа подчиненной таблицы.
5) Составить запросы на обновление и удаление данных для проверки работы ограничений целостности связей между таблицами. Проверить работу ограничений целостности в случаях установки каскадирования и запрета удаления и обновления данных.

Дополнительно:

6) Разработать приложение для визуализации и коррекции данных двух взаимосвязанных таблиц в соответствии с вариантом задания.

4.4 Варианты заданий

Варианты задания представлены в таблице 4.1. В варианте указаны имена взаимосвязанных таблиц **для клиентского приложения**.

Таблица 4.1 – Варианты заданий к лабораторной работе №4

№ вар-та	Имя главной таблицы	Имя подчиненной таблицы	№ струк-туры
1.	Поезд	Рейс	10
2.	Специальность	Студент	8
3.	Персона	Персона_Дело	12
4.	Личность	Ребенок	4
5.	Фирма	Гостиница	17
6.	Автор	Публикация	1
7.	Фирма	Сотрудник	15
8.	Служащий	Трудовая деятельность	7
9.	Преподаватель	Курс	2
10.	Предмет	Экзамен	9
11.	Диагноз	Курс лечения	18
12.	Хирург	Операция	3
13.	Факультет	Преподаватель	11
14.	Команда	Результат соревнований	13
15.	Фирма	Помещение	5
16.	Машина	Доп. комплектующие	19
17.	Профессия	Служащий	6
18.	Машина	Перевозка	16
19.	Партия	Партиец	20
20.	Концертный зал	Концерт	14
21.	Персона	Родственник	12
22.	Курс_препарат	Курс лечения	18
23.	Преподаватель	Курс	8
24.	Гостиница	Номер	17
25.	Факультет	Факультет_Предмет	11
26.	Личность	Родитель	4
27.	Соревнование	Результат соревнований	13
28.	Образование	Служащий_Образование	7
29.	Помещение	Телефон	5
30.	Партиец	Партийные взносы	20

4.5 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:

– концептуально-логическая схема базы данных согласно варианту,

– список связей между объектами (сущностями) БД с описанием каждой связи (тип связи (1:1, 1:M), главная таблица и подчиненная, поля, участвующие в связи (внешний ключ подчиненной таблицы, первичный ключ главной таблицы) и ограничения целостности для этих полей (допустимость Null-значений), идентифицирующая или неидентифицирующая связь),

– тексты всех запросов на создание таблиц с соответствующими первичными и внешними ключами и заданными условиями целостности связей (каскадирование обновления, удаления или запрет), сформулированные на естественном языке и структурированном языке запросов SQL;

– содержание таблиц, участвующих в запросах на обновление и удаление данных до и после соответствующих действий;

– тексты запросов на удаление и обновление данных из главной таблицы и подчиненной таблицы, участвующих в связи 1:M, сформулированные на естественном языке и структурированном языке запросов SQL,

– результаты выполнения запросов на удаление и обновление данных для главной и подчиненной таблицы, участвующей в связи 1:M в случае каскадирования и запрета.

– *дополнительно*: описание программы и экранной формы для визуализации и редактирования данных двух взаимосвязанных таблиц

4) Выводы.

4.6 Контрольные вопросы

1) Что понимается под объектом (сущностью) БД.
2) Охарактеризуйте возможные типы связей между сущностями.
3) Используя концептуально-логическую схему БД для своего варианта, приведите пример замены связи типа M:M двумя связями 1 :M за счет введения в БД дополнительной таблицы.
4) Какие связи между объектами являются идентифицирующими, а какие нет?
5) Как задаются связи между таблицами на языке SQL?
6) Сформулируйте ограничения целостности, которые накладываются на значения поля (совокупности полей), являющегося первичным ключом реляционной таблицы и поля (совокупности полей), являющегося внешним ключом.
7) Какие возможности контроля целостности связей поддерживаются

СУБД? Охарактеризуйте процессы ввода, обновления и удаления данных в главной и подчиненной таблице в каждом случае.

4.7 Рекомендации по выполнению работы

4.7.1 Понятие сущности, связи и целостности базы данных

Процесс проектирования базы данных обычно состоит из трех этапов:

– исследование предметной области;

– анализ данных (сущностей и их атрибутов);

– определение отношений между сущностями и определение первичных и вторичных (внешних) ключей.

В процессе проектирования определяется структура реляционной БД (состав таблиц, их структура и логические связи). Структура таблицы определяется составом столбцов, типом данных и размерами столбцов, ключами таблицы.

К базовым понятиям модели БД «сущность – связь» относятся: сущности, связи между ними и их атрибуты (свойства).

Сущность – любой конкретный или абстрактный объект в рассматриваемой предметной области. Сущности – это базовые типы информации, которые хранятся в БД (в реляционной БД каждой сущности назначается таблица). К сущностям могут относиться: студенты, клиенты, подразделения и т.д. Экземпляр сущности и тип сущности – это разные понятия. Понятие тип сущности относится к набору однородных личностей, предметов или событий, выступающих как целое (например, студент, клиент и т.д.). Экземпляр сущности относится, например, к конкретной личности в наборе. Типом сущности может быть студент, а экземпляром – Петров, Сидоров и т. д.

Связь - взаимосвязь между сущностями в предметной области. Связи представляют собой соединения между частями БД (в реляционной БД – это соединение между записями таблиц).

Сущности - это данные, которые классифицируются по типу, а связи показывают, как эти типы данных соотносятся один с другим. Если описать некоторую предметную область в терминах сущность - связь, то получим модель сущность - связь для этой БД.

Связь устанавливается между двумя общими полями (столбцами) двух таблиц. Существуют связи с отношением «один-к-одному», «один-ко-многим» и «многие-ко-многим».

Отношения, которые могут существовать между записями двух таблиц:

1) один - к - одному, каждой записи из одной таблицы соответствует одна запись в другой таблице;

2) один - ко - многим, каждой записи из одной таблицы соответствует несколько записей другой таблице;

3) многие - к - одному, множеству записей из одной таблице соответствует одна запись в другой таблице;

4) многие - ко - многим, множеству записей из одной таблицы соответствует несколько записей в другой таблице.

Тип отношения в создаваемой связи зависит от способа определения

связываемых полей:

1) отношение «один-ко-многим» создается в том случае, когда только одно из полей является полем первичного ключа или уникального индекса.

2) отношение «один-к-одному» создается в том случае, когда оба связываемых поля являются ключевыми или имеют уникальные индексы.

3) отношение «многие-ко-многим» фактически представляется двумя отношениями «один-ко-многим» и третьей (связующей) таблицей, первичный ключ которой состоит из полей первичного ключа двух других таблиц.

Ключ – это столбец (может быть несколько столбцов), позволяющий установить связь с записями в другой таблице. Существуют ключи двух типов: первичные и вторичные (внешние).

Первичный ключ – это одно или несколько полей (столбцов), комбинация значений которых однозначно определяет каждую запись в таблице. Первичный ключ не допускает значений Null (отсутствия значений) и всегда должен иметь уникальный индекс. Первичный ключ используется для связывания таблицы с внешними ключами в других таблицах.

Внешний (вторичный) ключ - это одно или несколько полей (столбцов) в таблице, содержащих ссылку на поле или поля первичного ключа в другой таблице. Значения в поле внешнего ключа могут повторяться. Внешний ключ определяет способ объединения таблиц.

Из двух логически связанных таблиц одну называют таблицей первичного ключа или главной таблицей, а другую таблицей вторичного (внешнего) ключа или подчиненной таблицей. СУБД позволяют сопоставить родственные записи из обеих таблиц и совместно вывести их в форме, отчете или запросе.

Существует три типа первичных ключей: ключевые поля счетчика (счетчик), простой ключ и составной ключ.

Поле счетчика (Тип данных «Счетчик»). Тип данных поля в базе данных, в котором для каждой добавляемой в таблицу записи в поле автоматически заносится уникальное числовое значение.

Если поле содержит уникальные значения, такие как коды или инвентарные номера, то это поле можно определить как первичный ключ. В качестве ключа можно определить любое поле, содержащее данные, если это поле не содержит повторяющиеся значения или значения Null.

В случаях, когда невозможно гарантировать уникальность значений каждого поля, существует возможность создать ключ, состоящий из нескольких полей. Чаще всего такая ситуация возникает для таблицы, используемой для связывания двух таблиц (связь типа многие - ко -многим).

Необходимо еще раз отметить, что в поле первичного ключа должны быть только уникальные значения в каждой строке таблицы, т.е. совпадение не допускается, а в поле вторичного или внешнего ключа совпадение значений в строках таблицы допускается.

Если возникают затруднения с выбором подходящего типа первичного ключа, то в качестве ключа целесообразно ввести поле типа «счетчик».

Связь вида 1:М в реляционных СУБД используется наиболее часто, поэтому ее мы рассмотрим наиболее подробно.

В предыдущих лабораторных работах нами была разработана таблица

«Список абонентов». Очевидно, что для этой таблицы первичным ключом является поле счетчика ID, так как его значение однозначно определяет значения полей FIO и bdate. Предоставим СУБД MySQL информацию о том, что это ключевое поле. Это можно сделать с помощью SQL-скрипта вида:

```
ALTER TABLE `phonelib` DROP PRIMARY KEY, ADD PRIMARY KEY(`ID`)
```

или с помощью визуального интерфейса в среде phpMyAdmin (рисунок 4.1).

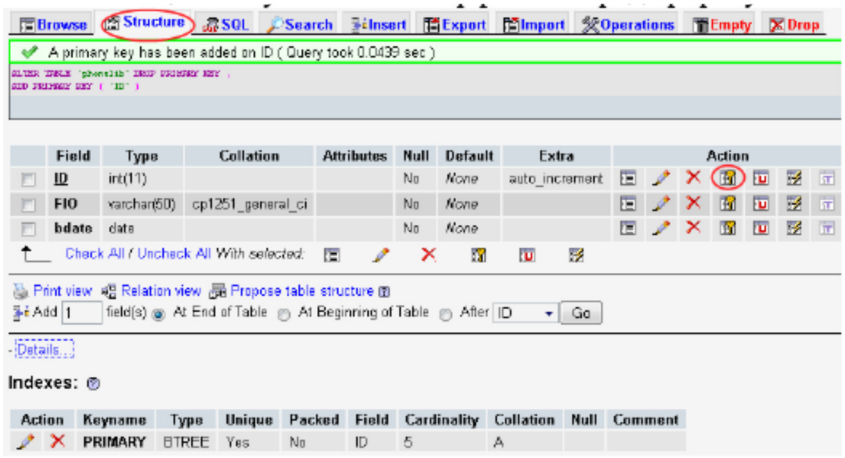

Рисунок 4.1 – Создание первичного ключа в среде phpMyAdmin

Теперь выполним анализ предметной области. Мы уже определили, что абонент характеризуется фамилией, именем, отчеством и датой рождения. Каждый абонент может иметь несколько телефонных номеров: домашний, мобильный, рабочий и т.п. Отношение сущность-связь для базы данных показано на рисунке 4.2.

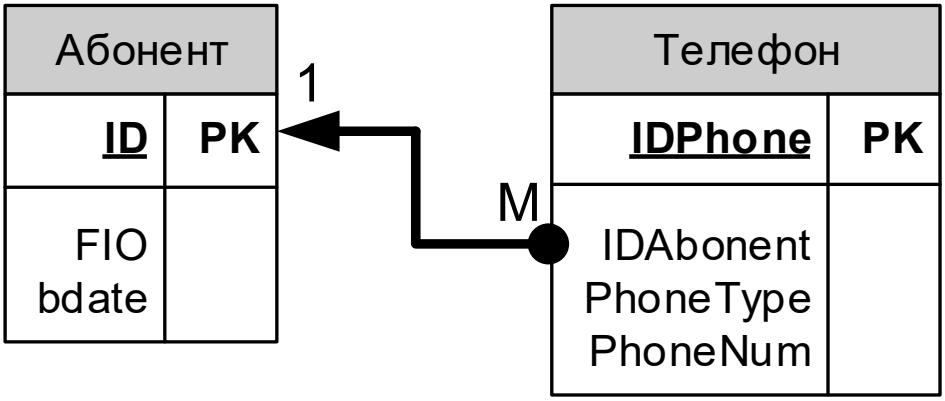

Рисунок 4.2 - Отношение сущность-связь

Очевидно, что поля «ID» и «IDAbonent» в обеих таблицах имеет одинаковый смысл – номер, присвоенный абоненту. В случае если таблицы не имеют связи друг с другом, ничто не мешает нам добавить в таблицу «Телефон» строку, например, (142, «Домашний», «0506521318»), несмотря на то, что в таблице «Абонент» нет абонента с номером 142. Это нарушение целостности данных, так как такая строка может быть занесена, но она не соответствует действительности. Чтобы подобная ситуация стала невозможной, необходимо установить связь между таблицами по полю «ID». В этом случае сервер БД

автоматически проследит, чтобы значение поля «IDAbonent» из вставляемой строки существовало в поле «ID» таблицы «Телефон».

Подобной проверки достаточно, чтобы условие целостности данных выполнялось при добавлении данных в таблицы. Но его недостаточно при манипулировании данными. Рассмотрим ситуацию, когда мы удаляем из таблицы «Абонент» некоторую запись, например абонента с номером 1. В случае если таблицы не связанны, удаление абонента повлечет за собой изменение только одной таблицы. В таблице «Телефон» останутся сведения о телефонных номерах абонента с номером 1. Такая ситуация – также нарушение целостности данных, так как абонента в БД нет, а его телефонные номера присутствуют. В случае если таблицы связанны, удаление абонента должно повлечь за собой удаление всех его номеров (тогда говорят, что удаление каскадируется).

Такая же ситуация с модификацией данных в родительской таблице. Если абонент с номером 1 изменил номер на 5, то в связанных таблицах изменение родительской таблицы повлечет за собой автоматическое изменение в полях связи дочерних таблиц.

Сформируем ограничения, обеспечивающие целостность базы данных «Телефонный справочник». Следует отметить, что понятие ссылочной целостности появилось в MySQL сравнительно недавно (начиная с версии 5.0) и большая часть «движков» ее не поддерживает. Поэтому мы и используем «движок» InnoDB при создании таблиц. Интерфейс phpMyAdmin также не имеет возможности визуально устанавливать внешние ключи и определять правила каскадного удаления и обновления базы данных. Поэтому создание дочерней таблицы мы будем выполнять с помощью SQL-скрипта

```
CREATE TABLE `MyPhoneNums` (
  `IDPhone` int(11) NOT NULL AUTO_INCREMENT,
  `IDAbonent` int(11) DEFAULT NULL,
  `PhoneType` varchar(20) DEFAULT NULL,
  `PhoneNum` varchar(12) DEFAULT NULL,
  PRIMARY KEY (`IDPhone`),
  KEY `abonent_key` (`IDAbonent`),
  CONSTRAINT `abonent_key` FOREIGN KEY (`IDAbonent`) REFERENCES `phonelib`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB AUTO_INCREMENT=2
DEFAULT CHARSET=cp1251;
```

В данном скрипте фраза PRIMARY KEY (‘IDPhone’) означает, что поле счетчика IDPhone является первичным ключом, KEY ‘abonent_key’ (‘IDAbonent’) CONSTRAINT ‘abonent_key’ FOREIGN KEY (‘IDAbonent’) REFERENCES ‘phonelib’ (‘ID’) ON DELETE CASCADE ON UPDATE CASCADE определяет внешний ключ и устанавливает правила каскадного обновления и удаления записей в родительской таблице phonelib.

Разработаем WEB-приложение для вывода и коррекции телефонного справочника. Скрипт phlib.php практически совпадает с текстом, приведенным в лабораторной работе №2. Для коррекции списка телефонов абонента, добавим в столбец «FIO» ссылки на корректирующий скрипт phonelist.php с указанием ID абонента. Таким образом, цикл вывода данных будет иметь вид:

```
while($row=mysql_fetch_array($result))
{ echo "<tr>";
  echo "<td>".$row['ID']."</td>";
  echo "<td>";
  echo "<A HRef=phonelist.php?id=".$row['ID'];
  echo "&FIO=".$row['FIO'].">".$row['FIO']."</a>";
  echo "</td> <td>".$row['bdate']."</td>";// выводим данные
  echo "</tr>";
};
```

Скрипт phonelist.php аналогичен скрипту phlib.php, за исключением того, что часть информации передается через метод Get (ID и FIO), а часть - через метод Post (PhoneType и PhoneNum).

В приведенном ниже примере реализована только операция «Добавить»:

```
…
/* Обработка данных, полученных методом post */
if ($_POST['add'])
{
   $sql = mysql_query("INSERT  into MyPhoneNums(IDAbonent,PhoneType,PhoneNum)
          values (".$_GET['id']."', '".$_POST['my_PhoneType']."',
'".$_POST['my_PhoneNum']."');");
}

/* составить запрос для вставки информации  в таблицу */
$query = "SELECT * FROM $userstable where IDAbonent=".$_GET['id'];
/* Выполнить запрос. Если произойдет ошибка - вывести ее. */
$result=mysql_query($query) or die(mysql_error());
echo "<TABLE BORDER=1>";
echo "<TR>";
echo "<TH>Тип</TH><TH>Номер</TH>";
echo "</TR>";
while($row=mysql_fetch_array($result))
{ echo "<tr>";
  echo "<td>"
  echo $row['PhoneType']."</td><td>".$row['PhoneNum']."</td>";
  echo "</tr>";
};
echo "</TABLE>";
```

Результат работы приложения представлен на рисунке 4.3.

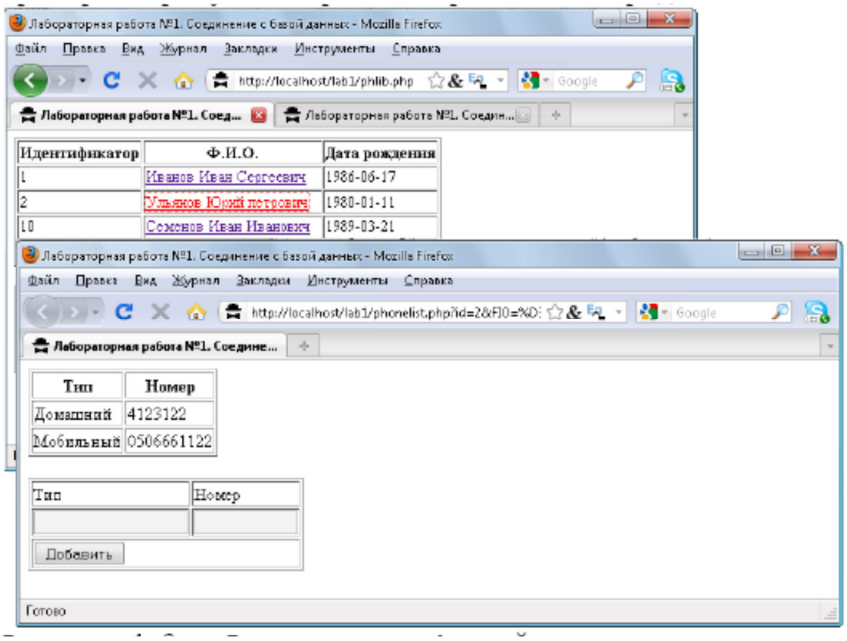

Рисунок 4.3 – Редактор телефонной книги