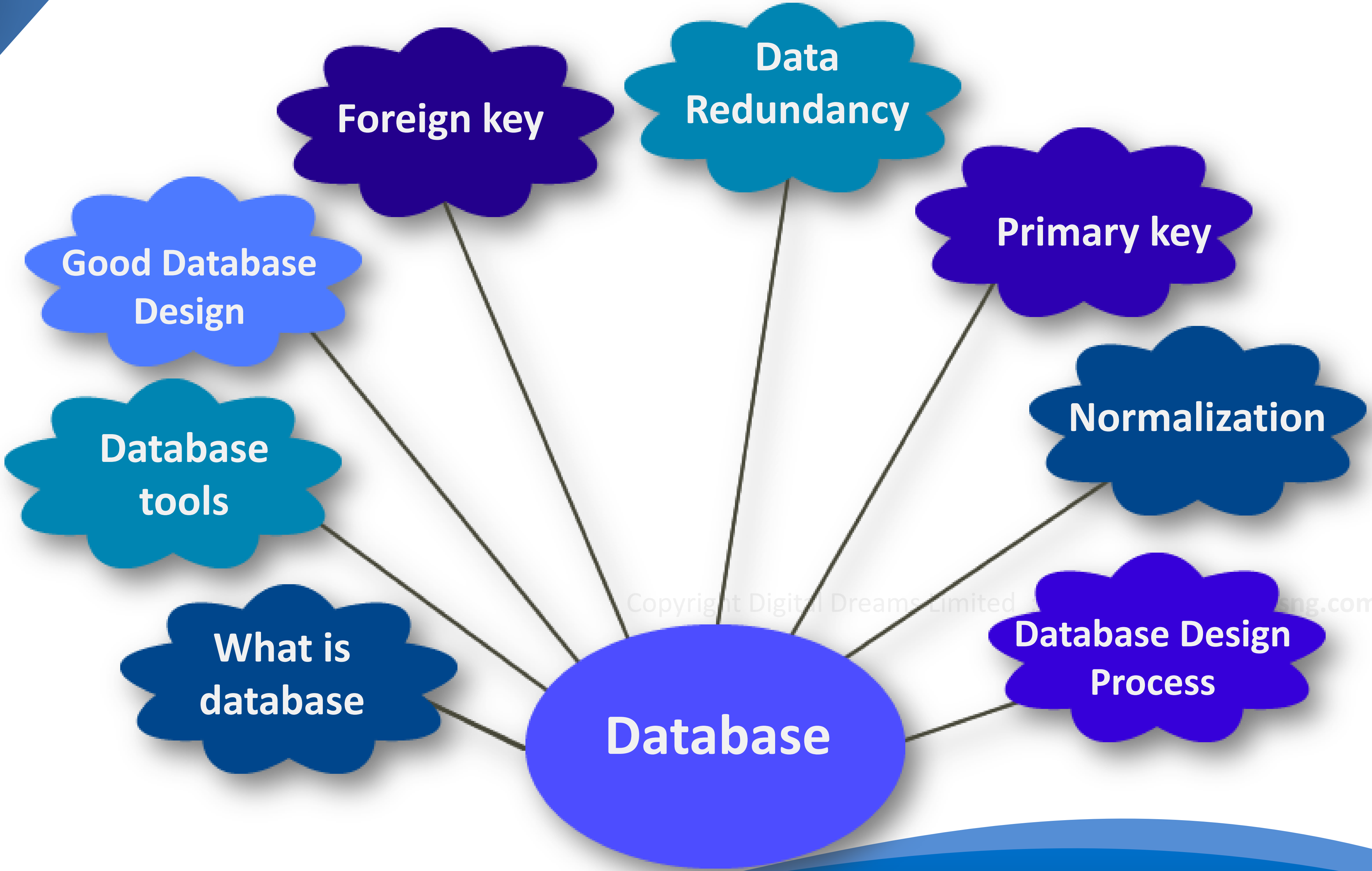
Database

Presenter:
Ossai Chinedu


Foreign key
Data Redundancy
Good Database Design
Primary key
Normalization
Database tools
What is database
Database Design Process
Database

What is Database?

Database is any collection of data, or information, that is specially organized for rapid search and retrieval.

Or Database is a collection of information that is organized so that it can easily be accessed, managed, and updated.

Database Tools

- MySQL
- Ms Access
- Oracle
- Dbase etc.

Good Database Design

Good database design is crucial for a high performance application. If a database doesn't have optimized relationships—normalization—it won't be able to perform as efficiently as possible.

The benefits of a well-planned and designed database are numerous, and it stands to reason that the more work you do up front, the less you'll have to do later because the results are costly. So, before you even start coding your application, spend a lot of time designing your database.

Proper database design is the only way your application will be efficient, flexible, and easy to manage and maintain. An important aspect of database design is to use relationships between tables instead of throwing all your data into one long flat file.

Types of relationships include

- One-to-one relationships
- One-to-many relationships
- Many-to-many relationships

Database Keys

keys are used to tie tables together. These relationships come in several forms:

What is Primary key in a table
A primary key, also called a primary keyword, is a key in a relational database that is unique for each record. It is designated to uniquely identify all table records in database.

What is Foreign key in a table
Foreign key is a column in one table which is **primary key** on another table.
Foreign key and Primary key is used to define relationship between two tables in relational database.

For **example** in **Employee** and **Department** relationship, we have two tables **Department(**dept_id, dept_name**)** and **Employee (**emp_id, emp_name, dept_id**).**
dept_id is primary key in Department table and foreign key in Employee table.

Data Redundancy

Data Redundancy is a condition created within a database or storage technology in which the same piece of data is held in two separate places.

This can also mean two different fields within a single database.

A good database should not encourage data redundancy.

Database Design Process

The greatest problem in application design is a lack of forethought. As it applies to database-driven applications, the design process must include a thorough evaluation of your database—what it should hold, how data relates to each other, and most importantly, is it scalable?

The general steps in the design process are:

- Define the objective.
- Design the data structures (tables, fields).
- Discern relationships.
- Define and implement business rules.
- Create the application.

Database Design Process

Creating the application is the last step—not the first! Many developers take an idea for an application, build it, then go back and try to make a set of database tables fit into it. This approach is completely backwards, inefficient, and will cost a lot of time and money.

Before starting any application design process, sit down and talk it out. If you can't describe your application—including the objectives, audience, and target market—then you're not ready to build it, let alone model the database.

Once you can describe the actions and nuances of your application to other people and have it make sense to them, you can start thinking about the tables you want to create. Start with big flat tables because, once you write them down, your newfound normalization skills will take over and you will be able to find your redundancies and visualize your relationships.

Normalization

Normalization is the art of organizing your database in such a way that your tables are related where appropriate and flexible for future growth.

Using relationships to properly organize your data is called **normalization.** There are many levels of normalization, but the primary levels are the **first**, **second**, and **third normal forms.** Each level has a rule or two that must be followed.
Following all of the rules will help ensure that your database is well organized and flexible.
The sets of rules used in normalization are called **normal forms.** If your database design follows the first set of rules, it's considered in the **first normal form**.

Normalization

If the first three sets of rules of normalization are followed, your database is said to be in the **third normal form.**

you'll learn about each rule in the first, second, and third normal forms and hopefully will follow them as you create your own applications. You'll be using an example set of tables for a **students** and **courses** database and taking it to the third normal form.

Flat Table

Before launching into the first normal form, you have to start with something that needs to be fixed. In the case of a database, it's the flat table. A flat table is like a spreadsheet—many, many columns. There are no relationships between multiple tables; all the data you could possibly want is right there in that flat table. This scenario is inefficient and consumes more physical space on your hard drive than a normalized database.

In your **students** and **courses** database, assume you have the following fields in your flat table:

- **StudentName—**The name of the student.
- **CourseID1—**The ID of the first course taken by the student.
- **CourseDescription1—**The description of the first course taken by the student.
- **CourseIntructor1—**The instructor of the first course taken by the student.
- **CourseID2—**The ID of the second course taken by the student.

Flat Table

- **CourseDescription2—**The description of the second course taken by the student.
- **CourseIntructor2—**The instructor of the second course taken by the student.

➢ Repeat CourseID, CourseDescription, and CourseInstructor columns many more times to account for all the classes a student can take during their academic career.

➢ With what you've learned so far, you should be able to identify the first problem area: CourseID, CourseDescription, and CourseInstructor columns are repeated groups.

➢ Eliminating redundancy is the first step in normalization, so next you'll take this flat table to first normal form. If your table remained in its flat format, you could have a lot of unclaimed space, and a lot of space being used unnecessarily—not an efficient table design!

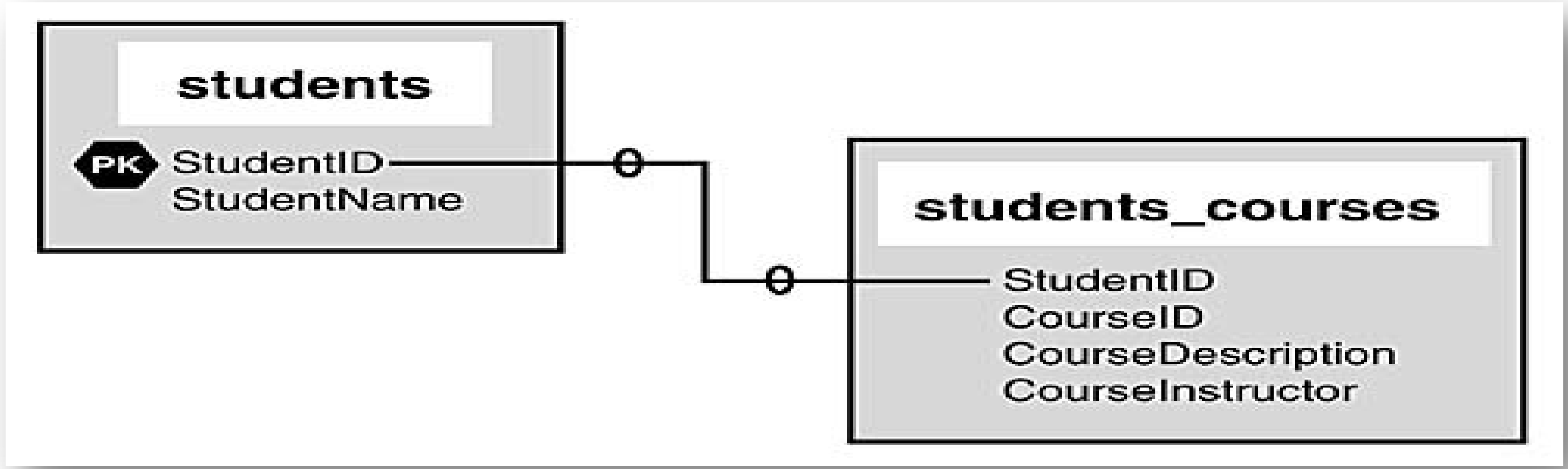
First Normal Form

The rules for the first normal form include

- Eliminate repeating information.
- Create separate tables for related data.

If you think about the flat table design, with many repeated sets of fields for the student and courses database, you can identify two distinct topics: **students** and **courses**. Taking your student and courses database to the first normal form would mean that you create two tables: one for **students** and one for **courses,** as shown.

First Normal Form



Breaking the flat table into two tables.

Your two tables now represent **a one-to-many relationship** of one student to many courses. Students can take as many courses as they wish and are not limited to the number of CourseID/CourseDescription/CourseInstructor groupings that existed in the flat table.

Second Normal Form

Second Normal Form

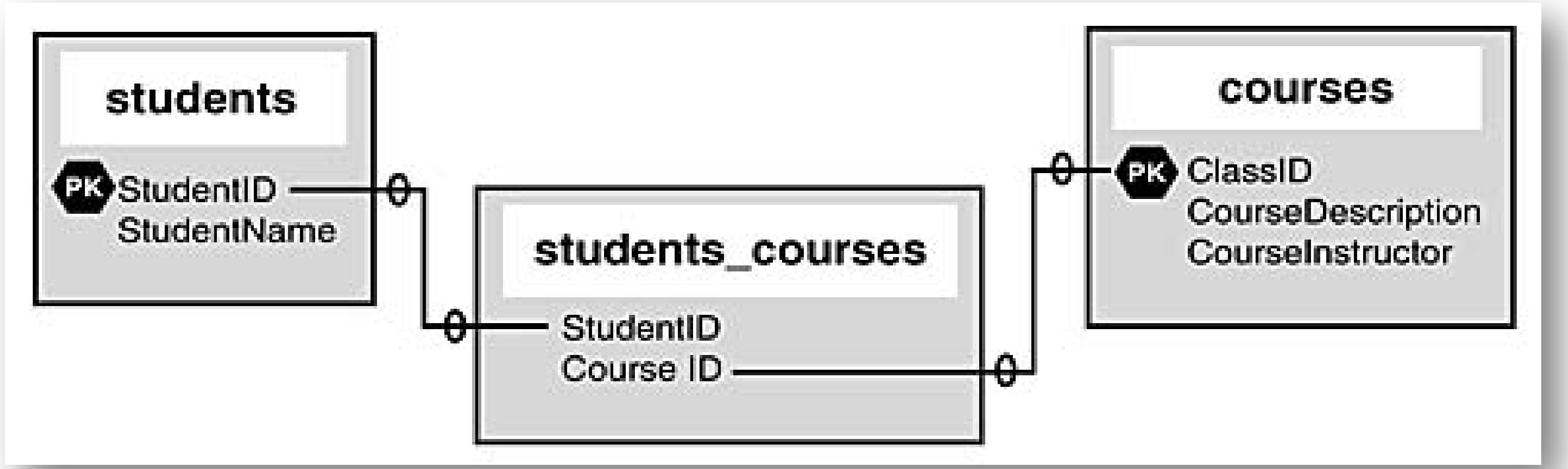
The rule for the second normal form is

- No non-key attributes depend on a portion of the primary key.

In plain English, this means that if fields in your table are not entirely related to a primary key, you have more work to do. In the students and courses example, it means breaking out the courses into their own table, and modifying the students_courses table.

CourseID, CourseDesc, and CourseInstructor can become a table called courses with a primary key of CourseID. The students_courses table should then just contain two fields: StudentID and CourseID. You can see this new design in as shown.

Second Normal Form



Taking your tables to second normal form.
This structure should look familiar to you as a many-to-many relationship using an intermediary mapping table.
The third normal form is the last form we'll look at, and you'll find it's just as simple to understand as the first two.

Third Normal Form

The rule for the third normal form is

- No attributes depend on other non-key attributes.

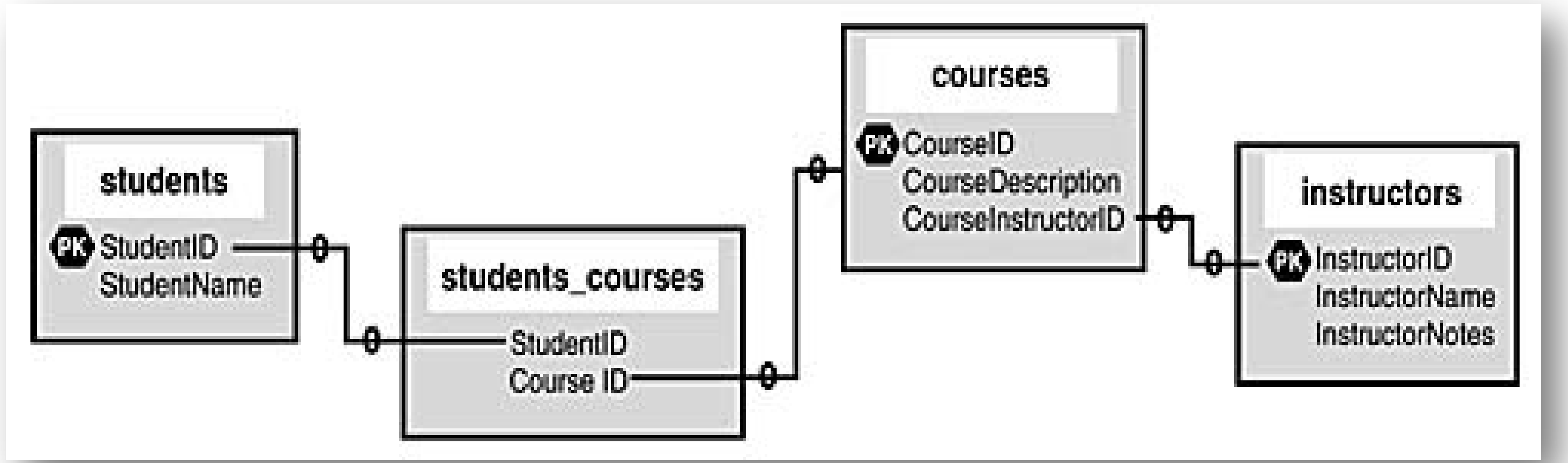
This rule simply means that you need to look at your tables and see if more fields exist that can be broken down further and that aren't dependent on a key.

Think about removing repeated data and you'll find your answer—**instructors.**

Inevitably, an **instructor** will teach more than one class. However, **CourseInstructor** is not a key of any sort.

So, if you break out this information and create a separate table purely for the sake of efficiency and maintenance (as shown), that's the third normal form.

Third Normal Form



Taking your tables to third normal form.
Third normal form is usually adequate for removing redundancy and allowing for flexibility and growth.

Creating Database

To see the default database that exist in the saver

```
mysql> SHOW DATABASES;
```

Use the SHOW statement to find out what databases currently exist on the server:

Creating a database

```
mysql> CREATE DATABASE student_db;
```

Selecting Database

Creating a database does not select it for use; you must type **Use** followed by database **name**.
Your database needs to be created only once, but you must select it for use each time you begin a mysql session.

```
mysql> USE student_db
```

Note:Selecting database does not require a semicolon.

Creating Tables

CREATE TABLE statement to specify the layout of your table:

```
mysql> CREATE TABLE bioinfo (id int(5) auto_increment primary
key, name VARCHAR(20), course VARCHAR(20),
    -> dept VARCHAR(20), sex CHAR(1));
```

After creating a database,we may deside to see the tables

```
mysql> SHOW TABLES;
```

Describing Tables

To verify that your table was created the way you expected, use a DESCRIBE statement:

Note:**bioinfo** is the table name.

mysql> **DESCRIBE bioinfo;**

Loading/ Inserting Data into Tables

```
mysql> INSERT INTO bioinfo
    -> VALUES (‘john',‘database',‘computer science',‘m');
```

Retrieving Data from Tables

```
mysql> SELECT * FROM TABLENAME;

Mysql> SELECT COLUMNNAME FROM TABLENAME;
```

Updating table and Deleting table

Editing/updating table

mysql> **UPDATE bioinfo SET sex = ‘f' WHERE name = ‘john';**

Deleting a table

mysql> **DELETE FROM bioinfo;**

Updating table and Deleting table

To load data to the database through Notepad

LOAD DATA LOCAL INFILE “C:/Documents and Settings/user/Desktop/pet.txt”INTO TABLE pet;

To delete a column called death in pet table

ALTER TABLE pet DROP COLUMN death;

To add a column called death in a pet table

ALTER TABLE pet ADD death DATE;

To add a column to a specific position in a pet table

ALTER TABLE pet ADD COLUMN phone VARCHAR(225) AFTER SEX;

To change a specific column name

ALTER TABLE pet CHANGE COLUMN death live VARCHAR(225);

To duplicate existing table structure

CREATE TABLE pee LIKE pet;

Primary Key

```
CREATE TABLE product(
 id int NOT NULL,
 item CHAR(30) ,
Cost_price  CHAR(30) ,
Selling _price CHAR(30) ,
    PRIMARY KEY (id)
 );
// the first insert will  work while the subsequent ones will not work
mysql> INSERT INTO product
   -> VALUES (“omo”, “3000”,’”3500”);

mysql> INSERT INTO product
   -> VALUES ('‘ 1“, “omo”, “3000”,’”3500”);
```

Primary Key

```
CREATE TABLE product(
 id int NOT NULL AUTO_INCREMENNT,
 item CHAR(30) ,
Cost_price  CHAR(30) ,
Selling _price CHAR(30) ,
    PRIMARY KEY (id)
 );

mysql> INSERT INTO product
    -> VALUES ("“ “, “klin”, “3000”,“”3500”);
```

Creating Connection Page.

```
$db = mysqli_connect(“localhost”,”root”,”password”,”dbname”) or die(“Could
not connect to database”);

If($db->connect_errno>0)
{
        die($db->connect_error);
}
```

Creating Table Page

```
<?php
require_once "connect.php";
$createtable= "create table if not exists studentinfom(
id int(11) primary key auto_increment,
firstname varchar(225),
middlename varchar(225),
surname varchar(225),
date_of_birth varchar(255) not null,
feedback text,
phone varchar(55);
```

Creating Table Page Cont.

```
picture_ref varchar(55))";
$sql = $db->query($createtable) or die($db->error);
if(isset($sql)){echo "studentinfom table creation successful.<br />";}else{echo
"studentinfom table was not created .<br />";}
?>
```

```
//datetime varchar(25) NOT NULL,
//datetime varchar(25) NOT NULL default '',
//date datetime,
//picture varchar(55),
//feedback text,
//phone int not null,
//phone varchar(225)
```

Insert into database page A

```
<?php
require_once "connect.php";

//mysqli_real_escape_string(to ensure that no error will show if the person type ',/ etc

$firstname = mysqli_real_escape_string($_POST['firstname']);
        $middlename =mysqli_real_escape_string($_POST['middlename']);
        $surname =mysqli_real_escape_string($_POST['surname']);
```

Insert into database page B

```
$day = $_POST["day"];
$month = $_POST["month"];
$year = $_POST["year"];
$date_of_birth =$day."/".$month."/".$year;
$date=mysqli_real_escape_string($_POST['date']);
$feedback=mysqli_real_escape_string($_POST['feedback']);
        $phone=mysqli_real_escape_string($_POST['phone']);
```

Insert into database page C

```
$picture_ref = basename($_FILES["picture_ref"]["name"]);
{
    $target_path = "images/";
    $target_path = $target_path . basename(
$_FILES["picture_ref"]["name"]);
    if(move_uploaded_file($_FILES['picture_ref']['tmp_name']
, $target_path) )

                }
```

Insert into database page D

```
mysql_select_db("beni",$con);
$sql="INSERT INTO studentinfom(firstname,
middlename,surname,date_of_birth,feedback,
phone,picture_ref)
VALUES
('$firstname','$middlename','$surname','$date_of_birth',
'$feedback','$phone','$picture_ref')";
if (!$db->query($sql))
  {
  die($db->error);
  } else {
echo "Your Form has been successfully Submitted----Thanks!--------";
}
?>
```

Insert into database code ALL

```
<?php
require_once "connect.php";
if (!$con)
  {
  die('Could not connect: ' . mysqli_error());
  }
  //mysqli_real_escape_string(to ensure that no error will show if the person type
',/ and etc
  $firstname = mysqli_real_escape_string($_POST['firstname']);
            $middlename =mysqli_real_escape_string($_POST['middlename']);
            $surname =mysqli_real_escape_string($_POST['surname']);
$day = $_POST["day"];
$month = $_POST["month"];
$year = $_POST["year"];
```

Insert into database code ALL

```
$date_of_birth =$day."/".$month."/".$year;
$date=mysqli_real_escape_string($_POST['date']);
        $feedback =mysqli_real_escape_string($_POST['feedback']);
        $phone=mysqli_real_escape_string($_POST['phone']);
$picture_ref = basename($_FILES["picture_ref"]["name"]);

{
$target_path = "images/";
$target_path = $target_path . basename( $_FILES["picture_ref"]["name"]);
        if(move_uploaded_file($_FILES['picture_ref']['tmp_name'], $target_path) )
                {}  }
```

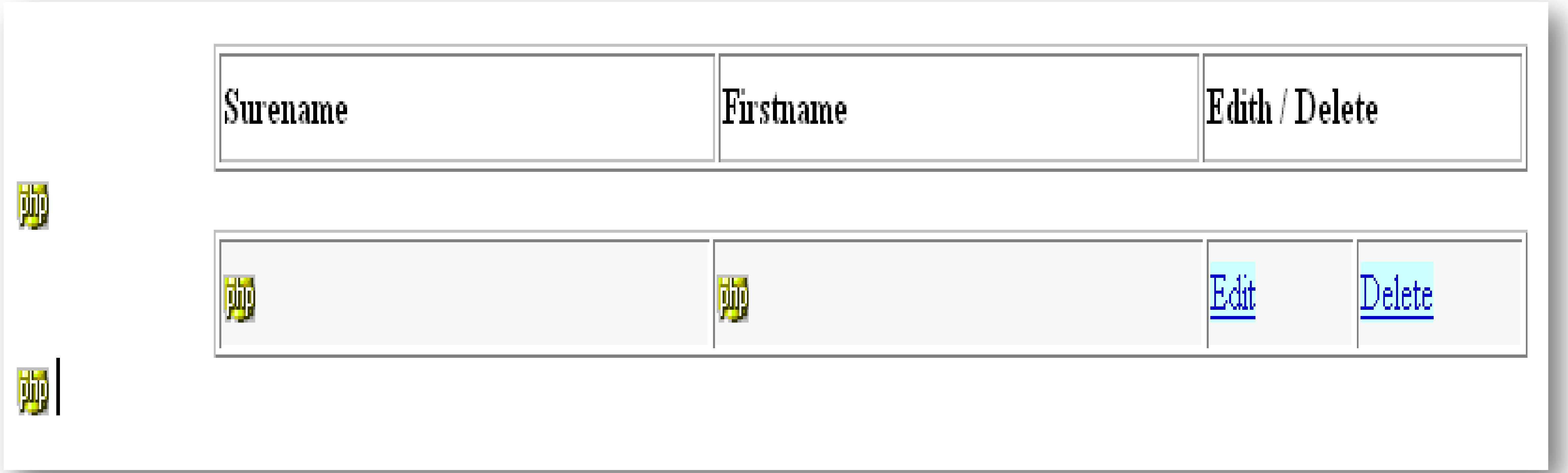
Insert into database code ALL

```
$sql="INSERT INTO studentinfom(
firstname,
middlename,
surname,
date_of_birth,
feedback,
phone,
picture_ref)
```

Insert into database code ALL

```
VALUES
('$firstname','$middlename','$surname','$date_of_birth',
'$feedback','$phone','$picture_ref')";
if (!$db->query($sql))
  {
  die('Error: ' . mysqli_error());
  } else {
echo "Your Form has been successfully Submitted----Thanks!--------";
}
?>
```

Editing page1 Interface



Surename	Firstname	Edith / Delete	
		Edit	Delete

Editing page1 code1

```
require_once "connect.php";
$result = $db->query("SELECT id,firstname,surname FROM studentinfom", $con) or die($db-
>error);
while($row =$result->fetch_assoc())
  {
          $id[]=$row['id'];                         ~~// or  $id=$row['id'];~~
          $firstname[]=$row['firstname'];           ~~// or   $firstname=$row['firstname'];~~
          $surname[]=$row['surname'];              ~~// or   $surname=$row['surname'];~~
  }
           $result=$db->query("select count(*) as count from studentinfom",$con) or
die($db->error);
          list($totalsize)=$result->num_rows;
?>
```

Editing page1 code2

```
<?php  if(isset($_GET['id']))
$id2=htmlspecialchars($_GET['id']);
if(isset($id2)){
		$sql="SELECT id,firstname,surname FROM studentinfom where id='$id2'";
}else{
		$sql="SELECT id,firstname,surname FROM  studentinfom";
}
$result = $db->query($sql) or die($db->error);
while($row = $result->fetch_assoc())
  {
		$firstname[]=$row['firstname'];
		$surname[]=$row['surname'];
				}

?>
```

Editing page1 code2

```
<?php  if(isset($_GET['id']))
$id2=htmlspecialchars($_GET['id']);
if(isset($id2)){$sql="SELECT id,firstname,surname FROM studentinfom where
id='$id2'";}else{$sql="SELECT id,firstname,surname FROM  studentinfom";}
$result = $db->query($sql) or die($db->error);
while($row = $result->fetch_assoc())
  {
             $firstname[]=$row['firstname'];
             $surname[]=$row['surname'];
                           }
?>
```

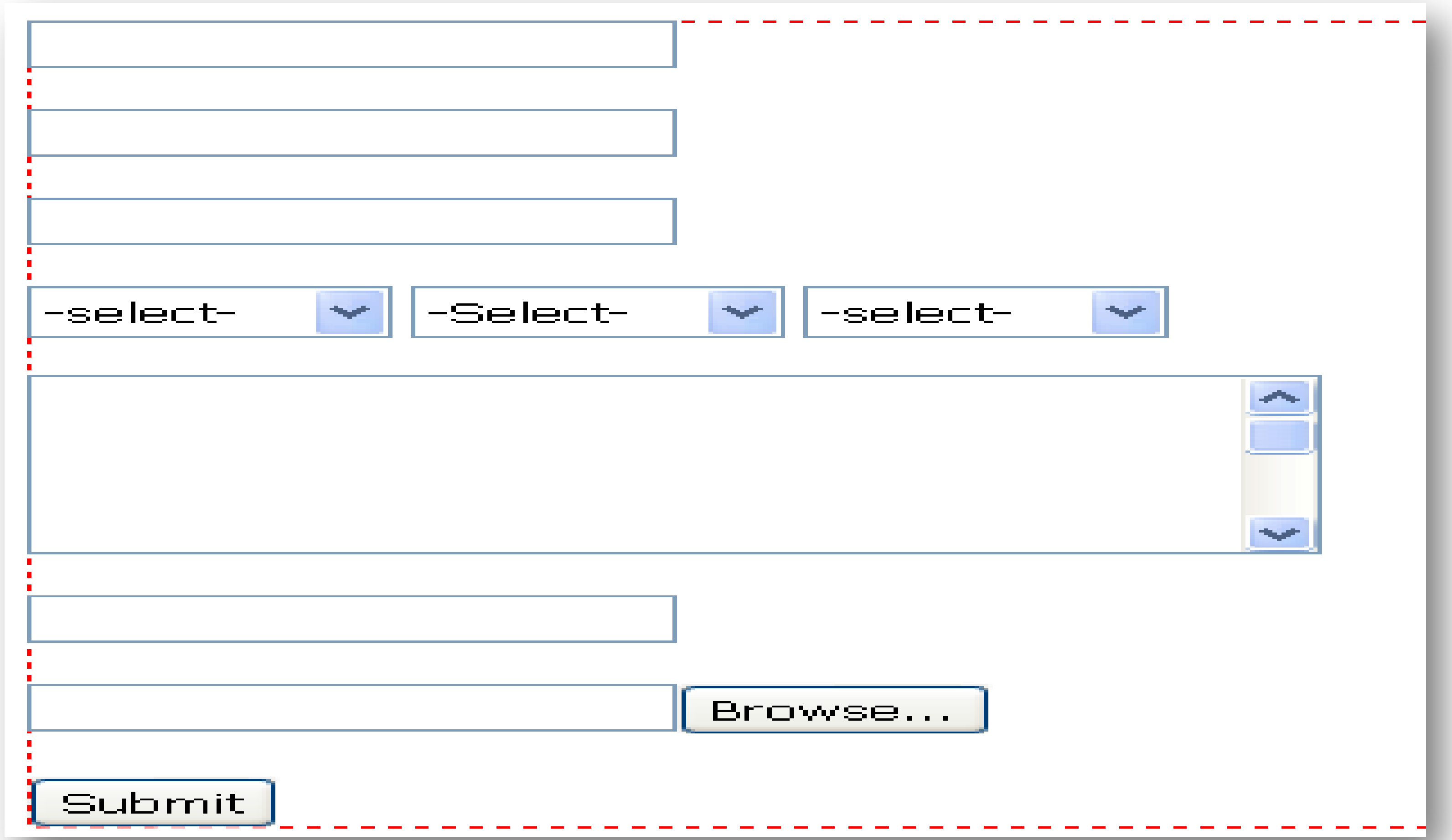
Editing page1 code3

```
<table width="700" border="1" align="center">
  <tr>
    <td  width="260" height="34" bgcolor="#FFFFFF">Id</td>
    <td width="252" height="34" bgcolor="#FFFFFF">Firstname</td>
    <td  width="166" bgcolor="#FFFFFF"><span class="style4">Edith /
Delete</span></td> </tr>
</table>
 <?php $length = count($id);
              for($i=0; $i<$length; $i+=1){$cj=fmod($i,1); ?>
    <table width="700" border="1" align="center"><tr>
  <td width="210" height="34" bgcolor="#F7F7F7">
  <?php echo $surname[$i];//echo $id[$i];?> </td>
```

Editing page1 code3

```
<td width="210" height="34" bgcolor="#F7F7F7">
  <?php
     echo $firstname[$i]
 ?>
</td>
<td width="60" bgcolor="#F7F7F7"><a href="edit_form_test.php?id=
     <?php
          echo $id[$i];
     ?>">Edit</a></td>
   <td width="67" bgcolor="#F7F7F7"> <a href="deleting.php?id=<?php echo
$id[$i];?>">Delete</a></td>
  </tr></table>
  <?php }?>
 ?>
```

Editing form page2



Editing form page2 code1

```
require_once "connect.php";
if(isset($_GET["id"]))
{
        $id=$_GET["id"];
        $sql="Select * from studentinfom where studentinfom.id=$id";
        $result=$db->query($sql) or die($db->error);
$count=0;
while($row = $result->fetch_assoc())
```

Editing form page2 code1

```
{
        $id=$row['id'];
        $firstname=$row['firstname'];
        $middlename=$row['middlename'];
        $surname=$row['surname'];
        $date_of_birth=$row['date_of_birth'];
        $feedback=$row['feedback'];
                $phone=$row['phone'];
        $picture_ref=$row['picture_ref'];
                $count++;
 } }
        ?>
```

Editing form page2 code2

```
<form action="update_edit_test.php" method="POST"
enctype="multipart/form-data" name="form1">
  <label for="textfield"></label>
  <p> <input type="text" name="firstname" id="textfield" value="<?php echo
$firstname;?>">
</p> <p>
    <label for="textfield2"></label>
    <input type="text" name="middlename" id="textfield2" value="<?php
echo $middlename;?>">
</p>
```

Editing form page2 code2

```
<p>
    <label for="textfield3"></label>
    <input type="text" name="surname" id="textfield3" value="<?php echo
    $surname;?>">
</p>
<p>
  <select name="day" id="">
        <option>-select-</option>
        <option>1</option>
        <option>2</option>
        <option>3</option>
  </select>
```

Editing form page2 code3

```
<select name="month" id="">
    <option>-Select-</option>
    <option>jan</option>
    <option>feb</option>
    <option>march</option>
  </select>
  <select name="year" id="">
    <option>-select-</option>
    <option>2015</option>
    <option>2016</option>
    <option>2017</option>
  </select>
<p>
   <label for="date"></label>
   <input type="text" name="date" id="textfield5" />
</p>
```

Editing form page2 code4

```
<<p>
    <label for="textarea"></label>
    <textarea name="feedback" id="textarea"  value="<?php //echo
$feedback;?>"cols="45" rows="5"><?php echo $feedback;?> </textarea>
  </p><p>
    <label for="textfield4"></label>
    <input type="text" name="phone" id="textfield4"  value="<?php echo
$phone;?>"/>
  </p>  <p>
    <label for="fileField"></label>
    <input type="file" name="picture_ref" id="fileField" value="<?php echo
$picture_ref;?>"/>
  </p>  <p> <label for=""></label>
    <input type="hidden" name="bid" id="bid" value="<?php echo $id;?>" />
    <input type="submit" name="submit" id="submit" value="submit" />
      </p> <br /></form>
```

main editing page3

```
 if(isset($_POST["submit"])){
                         $bid=$_POST["bid"];
                         $firstname = mysqli_real_escape_string($_POST['firstname']);
             $middlename = mysqli_real_escape_string($_POST['middlename']);
             $surname = mysqli_real_escape_string($_POST['surname']);
$day = $_POST["day"];
$month = $_POST["month"];
$year = $_POST["year"];
             $date_of_birth =$day."-".$month."-".$year;
             $phone =  mysqli_real_escape_string($_POST['phone']);
             $feedback = mysqli_real_escape_string($_POST['feedback']);
             $picture_ref = basename($_FILES["picture_ref"]["name"]); {
             $target_path = "images/";
             $target_path = $target_path . basename( $_FILES["picture_ref"]["name"]);
             if(move_uploaded_file($_FILES['picture_ref']['tmp_name'], $target_path) )
                         {}                         }}
```

main editing page3

```
  $sql = "UPDATE studentinfom SET
firstname='$firstname',
middlename='$middlename',
surname='$surname',
date_of_birth='$date_of_birth',
feedback='$feedback',
picture_ref='$picture_ref',
phone='$phone'WHERE id ='$bid'";
                  $result=$db->query($sql) or die($db->error);
if ($result) {
  echo "Your Post has been successfully Editted----Thanks!--------";
die();}
 else{
               die($db->error);}

?>
```

Deleting page

//Save this page with “deleting.php”

```
<?php


$id_del=$_GET["id"];
$result=$db->query("DELETE FROM studentinfom WHERE id='$id_del'");
if ($result)  {
 echo "The selected user has been successfully Deleted----Thanks!--------";
 die(); }
?>
```

Questions
And
Answers

Assignment

Design a database for a shop, keeping record of customers and sales per day.