B.C.A. (Sem – VI)

B.C.A. - 601

Building Application Using PHP

Purushottam Singh

Unit:- 3

Unit – 3

Database Access with MySql:-

Understanding the Database Design Process:-

The Importance of Good Database Design:-

- A good database design is important in ensuring consistent data, elimination of data redundancy, efficient execution of queries and high performance application.
- Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.
- Database design is the structure a database uses to plan, store and manage data.
- Data consistency is achieved when a database is designed to store only useful and required data.
- The outline of the table allows data to be consistent.
- Implementation of primary keys and unique constraints ensures consistency in the stored data.
- Cascading also ensures data uniformity. Implementing cascading of parent and child tables ensures that only those child records with a valid parent record exist.
- A normalized database design eliminates data redundancy, which reduces unnecessarily large volumes of data.
- Duplication can be avoided by creating a table of possible values and using a key to refer to the value.

Types of Table Relationship:-

2015 1st Qst 2016 (6)

- There are four relationships in database.
- **One to One Relationship:** One entity is associated with another entity. For Ex: Each employee is associated with one department.

Employees

EmpID	EmpFirst Name	EmpLast Name	Home Phone	*<< other fields >>*
100	Zachary	Erlich	553-3992	.....
101	Susan	McLain	790-3992	.....
102	Joe	Rosales	551-4993	.....

Compensation

EmpID	Hourly Rate	Commission Rate	*<< other fields >>*
100	25.00	5.0%	.....
101	19.75	3.5%	.....
102	22.50	5.0%	.....

2016 (1)

- **One to Many Relationships:** One entity is associated with many other entities. For Ex: A company is associated with all working employees in one branch/office/country.

Customers

Customer ID	CustFirst Name	CustLast Name	*<< other fields >>*
9001	Papi	Litwin	.....
9002	Alison	Balter	.....
9003	Andy	Baron	.....
9004	Chris	Kunicki	.....
9005	Mary	Chipman	.....

Customer Rentals

Customer ID	Video ID	Checkout Date
9002	80115	09/26/01
9001	64558	09/28/01
9003	10202	09/28/01
9003	11354	09/28/01
9003	78422	10/02/01
9005	30556	09/26/01
9004	20655	10/05/01

- **Many to Many Relationships:** Many entities are associated with many other entities. For Ex: In a company many employees are associated with multiple projects (completed/existing), and at the same time, projects are associated with multiple employees.

Students

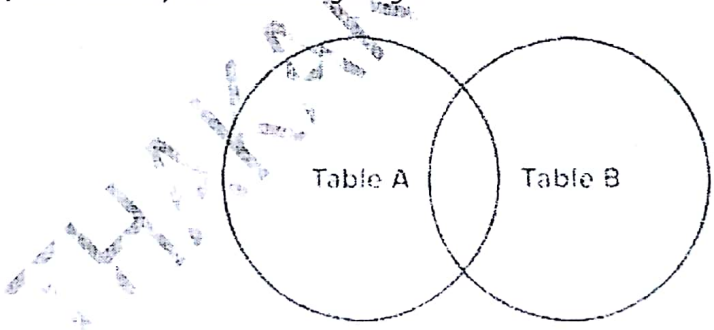
Student ID	StudFirst Name	StudLast Name	StudStreet Address	StudCity	StudState	StudZipcode	<< other fields >>
60001	Zacnary	Erlich	1204 Bryant Road	Seattle	WA	98125	.....
60002	Susan	McLain	101 C Street, Apt. 32	Redmond	WA	98052	.....
60003	Joe	Rosales	201 Cherry Lane SE	Redmond	WA	98073	.....
60004	Diana	Barlet	4141 Lake City Way	Woodinville	WA	98072	.....
60005	Tom	Wickerath	2100 Mineola Avenue	Bellevue	WA	98006	.....

Classes

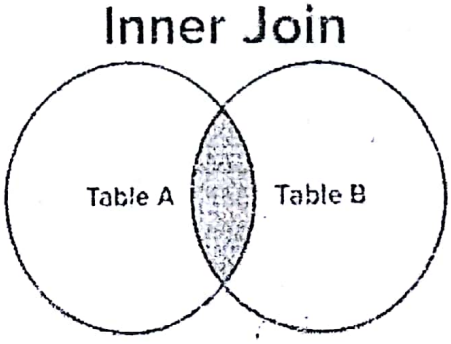
Class ID	Class Name	Class Category	Credits	Instructor ID	Classroom	<< other fields >>
900001	Advanced Calculus	Math	5	220087	2201	.....
900002	Advanced Music Theory	Music	3	220039	7012	.....
900003	American History	History	5	220148	3305	.....
900004	Computers in Business	Computer Science	2	220387	5115	.....
900005	Computers in Society	Computer Science	2	220387	5117	.....
900006	Introduction to Biology	Biology	5	220498	3112	.....
900007	Introduction to Database Design	Computer Science	5	220516	5105	.....
900008	Introduction to Physics	Physics	4	220097	2205	.....
900009	Introduction to Political Science	Political Science	5	220337	3308	.....

SQL Join Types:- 2015(1)

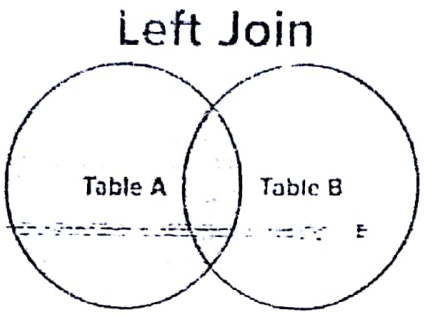
- Let's say we have two sets of data in our relational database: table A and table B, with some sort of relation specified by primary and foreign keys. The result of joining these tables together can be visually represented by the following diagram:



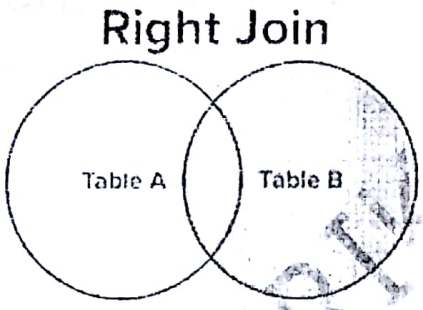
- The extent of the overlap, if any, is determined by how many records in Table A match the records in Table B.
- There are four types of join in table.
- **Inner Join:** - Select all records from Table A and Table B, where the join condition is met.



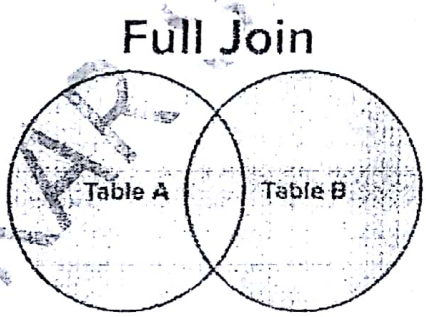
- **Left Join** - Select all records from Table A, along with records from Table B for which the join condition is met (if at all).



- **Right Join:** - Select all records from Table B, along with records from Table A for which the join condition is met (if at all).



- **Full Join:** - Select all records from Table A and Table B, regardless of whether the join condition is met or not.



- **Examples of SQL Join Types:-** Let's use the tables we introduced in the "What is a SQL join?" section to show examples of these joins in action. The relationship between the two tables is specified by the **customer_id key**, which is the **"primary key"** in **customers** table and a **"foreign key"** in the **orders table**:

Table - Customer

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1	George	Washington	gwashington@usa.gov	3200 Mt Vernon Hwy	Mount Vernon	VA	22121
2	John	Adams	jadams@usa.gov	1250 Hancock St	Quincy	MA	02169
3	Thomas	Jefferson	tjefferson@usa.gov	931 Thomas Jefferson Pkwy	Charlottesviile	VA	22902
4	James	Madison	jmadison@usa.gov	11350 Constitution Hwy	Orange	VA	22960
5	James	Monroe	jmonroe@usa.gov	2050 James Monroe Parkway	Charlottesville	VA	22902

Table - Order

1	07/04/1776	$234.56	1
2	03/14/1760	$78.50	3
3	05/23/1784	$124.00	2
4	09/03/1790	$65.50	3

- **Note:** **(1)** Not every customer in our customers table has placed an order.
  (2) There are a few orders for which no customer record exists in our customers table.

- **Query for Inner Join:** - Let's say we wanted to get a list of those customers who placed an order and the details of the order they placed. This would be a perfect fit for an inner join, since an inner join returns records at the intersection of the two tables.

```
select first_name, last_name, order_date, order_amount
from customers c
inner join orders o
on c.customer_id = o.customer_id
```

Output of Inner Joinn

George	Washington	07/4/1776	$234.56
John	Adams	05/23/1784	$124.00
Thomas	Jefferson	03/14/1760	$78.50
Thomas	Jefferson	09/03/1790	$65.50

- **Query for Left Join:** - If we wanted to simply append information about orders to our customers table, regardless of whether a customer placed an order or not, we would use a left join. A left join returns all records from table A and any matching records from table B.

```
select first_name, last_name, order_date, order_amount
from customers c
left join orders o
on c.customer_id = o.customer_id
```

Output of Left Join

George	Washington	07/04/1776	$234.56
John	Adams	05/23/1784	$124.00
Thomas	Jefferson	03/14/1760	$78.50
Thomas	Jefferson	09/03/1790	$65.50
James	Madison	NULL	NULL
James	Monroe	NULL	NULL

- **Query for Right Join:** - Right join is a mirror version of the left join and allows to get a list of all orders, appended with customer information.

```
select first_name, last_name, order_date, order_amount
from customers c
right join orders o
on c.customer_id = o.customer_id
```

Output of Right Join

George	Washington	07/04/1776	$234.56
Thomas	Jefferson	03/14/1760	$78.50
John	Adams	05/23/1784	$124.00
Thomas	Jefferson	09/03/1790	$65.50
NULL	NULL	07/21/1795	$25.50
NULL	NULL	11/27/1787	$14.40

- **Query for Full Join:** - Finally, for a list of all records from both tables, we can use a full join.

```
select first_name, last_name, order_date, order_amount
from customers c
full join orders o
on c.customer_id = o.customer_id
```

Output of Full Join

George	Washington	07/04/1776	$234.56
Thomas	Jefferson	03/14/1760	$78.50
John	Adams	05/23/1784	$124.00
Thomas	Jefferson	09/03/1790	$65.50
NULL	NULL	07/21/1795	$25.50
NULL	NULL	11/27/1787	$14.40
James	Madison	NULL	NULL
James	Monroe	NULL	NULL

Understanding of Normalization: - 12016 6. coloh Types

Normalizing Techniques: -

- More often than not database users have not able to understand what the various normalization techniques are.
- Applying and knowing the principles of normalize and implement them to your daily database design is a simple process and this can come with performance improvements.

What is Normalization: 12016 CO

- Normalization is a process of efficiently organizing your data in your database.
- The goals in doing so: Eliminate all the redundant data and ensure data dependencies.
- Both of these goals are worth achieving.
- Normalizing tables would reduce the amount of space a database may consume.
- And as said earlier each of these is set of rules.
- Each of these rules is called as "Normal Form" or NF in short.
- All the rules are cummulative in nature.
- Meaning if we have three of the rules adhered then we are in the 3NF.

First Normal Form (1NF): -

- Eliminate repeating groups from the same table.
- Aggregate similiar data in separate tables and identify each row with an unique identifier.
- In simple language if we were to say each attribute of the relation would be atomic in nature for 1NF.
- Look at the example below to understand better.

Un-Normalized Data:

Manager	Persons
Vinod	Shaju , Manoj, Ashok, Naveen
Rajiv	Sarawana, Salil, Kannan

1NF

Manager	Persons
Vinod	Shaju
Vinod	Manoj
Vinod	Ashok
Vinod	Naveen
Rajiv	Kannan
Rajiv	Salil
Rajiv	Sarawana

Second Normal Form (2NF): -

- Moving forward lets take a look at the rules that goven 2NF. We get a step even more closer to remove duplicate records.
- Remove data that apply to multiple rows and place them in a separate table
- Relate the above table with foreign keys
- Consider the below example to understand the same.

Un-Normalized Data:

ID	Team	Contact	Meeting #Time
1	SQL	Vinod	2
2	ASP .NET	Rajiv	1.5
3	SQL	Vinod	3
4	SQL	Vinod	1

2NF

Team	Contact
SQL	Vinod
ASP .NET	Rajiv

ID	Team	Meeting #Time
1	SQL	2
2	ASP .NET	1.5
3	SQL	3
4	SQL	1

Third Normal Form (3NF): -

- This is the most preferred normalization technique followed for most of the database.
- Eliminate all fields that donot depend on the Primary key.
- Values in a record that are not part of that record's key do not belong in the table. In general, any time the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table.

Un-Normalized Data:

ID	Contact	City	Country	Code
1	Vinod	Bangalore	India	091
2	Rajiv	Bangalore	India	091

3NF

ID	Contact	Code
1	Vinod	091
2	Rajiv	091

Code	City	Country
091	Bangalore	India

- **Note:** There are 4NF otherwise called as Boyce-Codd normal form (BCNF). I wouldnot deal much into this form as it becomes far beyond practicle limits to have such a requirement. The rule is, we are in BCNF if and only if every determinant is a candidate key.

Learning Basic SQL Commands: -

Table Creation:\-

- The CREATE TABLE statement is used to create a table in MySQL.
- The data type specifies what type of data the column can hold.
- After the data type, you can specify other optional attributes for each column:
  1. NOT NULL - Each row must contain a value for that column, null values are not allowed
  2. DEFAULT value - Set a default value that is added when no other value is passed
  3. UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
  4. AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
  5. PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT
- Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.
- Example: -

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Insert Row: -

- After a database and a table have been created, we can start adding data in them.
- Here are some syntax rules to follow:
  1. The SQL query must be quoted in PHP
  2. String values inside the SQL query must be quoted
  3. Numeric values must not be quoted
  4. The word NULL must not be quoted
- The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)
```

Select Command Using Where Clause: -

- The SELECT statement is used to select data from a database.
- The result is stored in a result table, called the result-set.

```
SELECT column_name,column_name FROM table_name;

SELECT * FROM table_name;
```

- The WHERE clause is used to extract only those records that fulfill a specified criterion.

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

- Example: -

```
SELECT * FROM Customers WHERE Country='Mexico';
SELECT * FROM Customers WHERE CustomerID=1;
```

Update Command: -

- Change the value of the "City" column of a record in the "Customers" table:
- Example: -

```
UPDATE Customers SET City='Hamburg' WHERE CustomerID=1;

UPDATE table_name SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

- UPDATE Multiple Columns

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Frankfurt'
WHERE CustomerID=1;
```

- UPDATE Multiple Records

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

Delete Command: -

- The DELETE statement is used to delete records in a table.
- The DELETE statement is used to delete rows in a table.
- SQL DELETE Syntax

```
DELETE FROM table_name WHERE some_column=some_value;
```

- SQL DELETE Example

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste' AND ContactName='Maria Anders';
```

- Delete All Data

```
DELETE FROM table_name;

DELETE * FROM table_name;
```

Replace Command: -

- The REPLACE function is easy to use and very handy with an UPDATE statment.
- Replace searches for certain characters in a string and replaces them with other characters. So this statement:

```
SELECT Replace('SQLTeam.com Rocks!', 'Rocks', 'Rolls')
```

- REPLACE searches the the first string for any occurance of the the second string and replaces it with the third string.
- You can also do replacements of different sizes. For example,

```
SELECT Replace('SQLTeam.com Rocks!', 'Rocks', 'is cool')
```

String Function: -

- SQL string functions are used primarily for string manipulation. The following table details the important string functions.

Name	Description
ASCII()	Returns numeric value of left-most character
BIN()	Returns a string representation of the argument
BIT_LENGTH()	Returns length of argument in bits
CHAR_LENGTH()	Returns number of characters in argument
CHAR()	Returns the character for each integer passed
CHARACTER_LENGTH()	A synonym for CHAR_LENGTH()
CONCAT()	Returns concatenated string
CONV()	Converts numbers between different number bases

Date and Time Function: -

- As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets complicated.
- Before talking about the complications of querying for dates, we will look at the most important built-in functions for working with dates.
- The following table lists the most important built-in date functions in MySQL.

Function	Description
NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date or date/time expression
EXTRACT()	Returns a single part of a date/time
DATE_ADD()	Adds a specified time interval to a date
DATE_SUB()	Subtracts a specified time interval from a date
DATEDIFF()	Returns the number of days between two dates
DATE_FORMAT()	Displays date/time data in different formats

- The following table lists the most important built-in date functions in SQL Server.

Function	Description
GETDATE()	Returns the current date and time
DATEPART()	Returns a single part of a date/time
DATEADD()	Adds or subtracts a specified time interval from a date
DATEDIFF()	Returns the time between two dates
CONVERT()	Displays date/time data in different formats

Stored Procedure: -

- A subprogram is a program unit/module that performs a particular task.
- These subprograms are combined to form larger programs.
- This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program which is called the calling program.
- PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms
  1. Functions: these subprograms return a single value, mainly used to compute and return a value.
  2. Procedures: these subprograms do not return a value directly, mainly used to perform an action.
- A procedure is created with the CREATE OR REPLACE PROCEDURE statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

- Where,
  1. procedure-name specifies the name of the procedure.
  2. [OR REPLACE] option allows modifying an existing procedure.
  3. The optional parameter list contains name, mode and types of the parameters. IN represents that value will be passed from outside and OUT represents that this parameter will be used to return a value outside of the procedure.
  4. procedure-body contains the executable part.
  5. The AS keyword is used instead of the IS keyword for creating a standalone procedure.

Indexing: - 12015(1)

- The CREATE INDEX statement is used to create indexes in tables.
- Indexes allow the database application to find data fast; without reading the whole table.
- An index can be created in a table to find data more quickly and efficiently.
- The users cannot see the indexes, they are just used to speed up searches/queries.

Syntax: -
```
CREATE INDEX index_name ON table_name (column_name)
```

Example: -
```
CREATE INDEX PIndex ON Persons (LastName, FirstName)
```

Sorting: -

- We have seen SQL SELECT command to fetch data from MySQL table.
- When you select rows, the MySQL server is free to return them in any order, unless you instruct it otherwise by saying how to sort the result.
- But you sort a result set by adding an ORDER BY clause that names the column or columns you want to sort by.

Order By

Syntax: -
```
SELECT field1, field2 table_name1, table_name2 ORDER BY field1, [field2...] [ASC [DESC]]
```

Example: -
```
SELECT * from tutorials_tbl ORDER BY tutorial_author ASC
```

```
$sql = 'SELECT tutorial_id, tutorial_title, tutorial_author, submission_date
        FROM tutorials_tbl ORDER BY  tutorial_author DESC';
```

Using MySQL with PHP: - 16(1)

Connecting to MySQL and selecting the database: -

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.
- MySQL is a database system used on the web.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use.
- MySQL uses standard SQL.
- MySQL compiles on a number of platforms.
- Before we can access data in the MySQL database, we need to be able to connect to the server.

Example (MySQLi Procedural)
```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Retriving Query Results: -

- The SELECT statement is used to select data from one or more tables.
- The following example shows the same as the example above, in the MySQLi procedural way.

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);

// Check connection
if (!$conn)
{
   die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0)
{
   // output data of each row
   while($row = mysqli_fetch_assoc($result))
  {
      echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"].
"<br>";
  }
} else
{
   echo "0 results";
}

mysqli_close($conn);
?>
```

Records Updating: - 12016.(01) 6.

- The UPDATE statement is used to update existing records in a table.

Syntax

```
UPDATE table_name SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";
if (mysqli_query($conn, $sql)) {
   echo "Record updated successfully";
} else {
   echo "Error updating record: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Record Addition: -

- After a database and a table have been created, we can start adding data in them.
- Here are some syntax rules to follow:
  1. The SQL query must be quoted in PHP
  2. String values inside the SQL query must be quoted
  3. Numeric values must not be quoted
  4. The word NULL must not be quoted
- The INSERT INTO statement is used to add new records to a MySQL table.

Syntax

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Deletion Record: -

- The DELETE statement is used to delete records from a table.

Syntax

```
DELETE FROM table_name WHERE some_column = some_value;
```

Example

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";
if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```