DATABASE MANAGEMENT SYSTEM LAB CSE-403L

SEMESTER:6TH

LAB REPORT # 5

Submitted By: *Zainab Khalid*
Registration No:*19PWCSE1743*
Section: A
Submitted to: *Mam Naina Said*

DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR

LAB # 05

DATABASE DEVELOPMENT USING MYSQL

TASK 5.1

What is difference between SQL and MySQL? Why is MySQL used? What are its features?

Difference Between SQL and MySQL:

SQL	MySQL
SQL is developed by Microsoft Corporation.	MySQL was developed by MySQL AB but is currently acquired and owned by Oracle Corporation.
SQL is a query programming language that manages RDBMS.	MySQL is a relational database management system that uses SQL.
SQL is primarily used to query and operate database systems.	MySQL allows you to handle, store, modify and delete data and store data in an organized way.
SQL does not support any connector.	MySQL comes with an in-built tool known as MySQL Workbench that facilitates creating, designing, and building databases.
SQL follows a simple standard format without many or regular updates.	MySQL has numerous variants and gets frequent updates.
SQL supports only a single storage engine.	MySQL offers support for multiple storage engines along with plug-in storage, making it more flexible.
SQL does not allow other processors or even its own binaries to manipulate data during execution.	MySQL is less secure than SQL, as it allows third-party processors to manipulate data files during execution.

TASK 5.2

What is database engine? What purpose does it serve? How many types of engines are supported by MySQL? Which database engine is most commonly used and why?

Database Engine:

A database engine (or storage engine) is the underlying software component that a database management system (DBMS) uses to create, read, update and delete (CRUD) data from a database.

The Database Engine component of SQL Server is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data consuming applications in your enterprise.

Types of Storage Engines:

There are two types of storage engines in MySQL: transactional and non-transactional. MySQL supported storage engines:

- InnoDB
- MyISAM
- Memory
- CSV
- Merge
- Archive
- Federated
- Blackhole

Widely Used Storage Engine:

InnoDB is the most widely used storage engine with transaction support. It is an ACID compliant storage engine. It supports row-level locking, crash recovery and multi-version concurrency control. It is the only engine which provides foreign key referential integrity constraint. Oracle recommends using InnoDB for tables except for specialized use cases.

TASK 5.3 :

Consider the Relational Schema given in Figure 6.2 and its tables given in Figure 6.3. Write SQL commands to create all the tables. Take the appropriate attribute type and length from the data provided. (Note: Use the following hierarchy for table creation: 1) Type, Tournament and Team, 2) Member, and 3) Entry).

Type table:

```
MariaDB [(none)]> use lab5task;
Database changed
MariaDB [lab5task]> create table Type(
    -> Type varchar(30) not null,
    -> Fee int not null,
    -> primary key(Type));
Query OK, 0 rows affected (0.336 sec)
```

```
MariaDB [lab5task]> describe Type;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| Type  | varchar(30) | NO   | PRI | NULL    |       |
| Fee   | int(11)     | NO   |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

Tournament table:

```
MariaDB [lab5task]> create table TournamentTable(
    -> TourID int not null auto_increment,
    -> TourName varchar(30),
    -> TourType varchar(30),
    -> primary key(TourID));
Query OK, 0 rows affected (0.425 sec)
```

```
MariaDB [lab5task]> describe TournamentTable;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| TourID   | int(11)     | NO   | PRI | NULL    | auto_increment |
| TourName | varchar(30) | YES  |     | NULL    |                |
| TourType | varchar(30) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
3 rows in set (0.014 sec)
```

Team Table

```
MariaDB [lab5task]> create table Team(
    -> TeamName varchar(30) not null,
    -> PracticeNight varchar(30) not null,
    -> Manager int not null,
    -> primary key(TeamName));
Query OK, 0 rows affected (0.461 sec)
```

```
MariaDB [lab5task]> describe Team;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| TeamName      | varchar(30) | NO   | PRI | NULL    |       |
| PracticeNight | varchar(30) | NO   |     | NULL    |       |
| Manager       | int(11)     | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.015 sec)
```

Member Table:

```
MariaDB [lab5task]> create table Member(
    -> MemberID int not null auto_increment PRIMARY KEY,
    -> LastName varchar(30),
    -> FirstName varchar(30) not null,
    -> Handicap int,
    -> Gender varchar(1) not null,
    -> Team varchar(30),
    -> MType varchar(30) not null,
    -> Coach int,
    -> Phone varchar(15) not null,
    -> JoinDate DATE not null,
    -> foreign key(Team) references Team(TeamName),
    -> foreign key(MType) references Type(Type));
Query OK, 0 rows affected (0.301 sec)

MariaDB [lab5task]> describe Member;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| MemberID  | int(11)     | NO   | PRI | NULL    | auto_increment |
| LastName  | varchar(30) | YES  |     | NULL    |                |
| FirstName | varchar(30) | NO   |     | NULL    |                |
| Handicap  | int(11)     | YES  |     | NULL    |                |
| Gender    | varchar(1)  | NO   |     | NULL    |                |
| Team      | varchar(30) | YES  | MUL | NULL    |                |
| MType     | varchar(30) | NO   | MUL | NULL    |                |
| Coach     | int(11)     | YES  |     | NULL    |                |
| Phone     | varchar(15) | NO   |     | NULL    |                |
| JoinDate  | date        | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
10 rows in set (0.016 sec)
```

Entry Table:

```
MariaDB [lab5task]> create table Entry(
    -> Member int not null,
    -> TourID int not null,
    -> Year int not null,
    -> primary key(Member,TourID,Year),
    -> foreign key(Member) references Member(MemberID),
    -> foreign key(TourID) references TournamentTable(TourID));
Query OK, 0 rows affected (0.542 sec)

MariaDB [lab5task]> describe Entry;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| Member | int(11) | NO   | PRI | NULL    |       |
| TourID | int(11) | NO   | PRI | NULL    |       |
| Year   | int(11) | NO   | PRI | NULL    |       |
+--------+---------+------+-----+---------+-------+
3 rows in set (0.011 sec)
```

TASK 5.5

Using insert command, populate all the records in member, type, entry, team, and tournament tables according to Figure 6.3a and Figure 6.3b.

Member table

```
MariaDB [lab5task]> select * from member;
+----------+----------+-----------+----------+--------+-------+--------+-------+---------+------------+
| MemberID | LastName | FirstName | Handicap | Gender | Team  | MType  | Coach | Phone   | JoinDate   |
+----------+----------+-----------+----------+--------+-------+--------+-------+---------+------------+
|      118 | McKenzie | Melissa   |       30 | F      | NULL  | Junior |   153 | 963270  | 2009-05-10 |
|      138 | Stone    | Michael   |       30 | M      | NULL  | Senior |  NULL | 983223  | 2013-05-13 |
|      153 | Nolan    | Brenda    |       11 | F      | TeamB | Senior |  NULL | 442649  | 2010-07-25 |
|      176 | Branch   | Helen     |     NULL | F      | NULL  | Social |  NULL | 589419  | 2015-11-18 |
|      178 | Beck     | Sarah     |     NULL | F      | NULL  | Social |  NULL | 226596  | 2004-01-06 |
|      228 | Burton   | Sandra    |       26 | F      | NULL  | Junior |   153 | 244493  | 2015-06-21 |
|      235 | Cooper   | William   |       14 | M      | TeamB | Senior |   153 | 722954  | 2012-02-12 |
|      239 | Spence   | Thomas    |       10 | M      | NULL  | Senior |  NULL | 697720  | 2010-06-04 |
|      258 | Olison   | Barbara   |       16 | F      | NULL  | Senior |  NULL | 370186  | 2015-07-11 |
|      286 | Polland  | Robert    |       19 | M      | TeamB | Junior |   235 | 5617681 | 2015-07-26 |
|      290 | Buxton   | Thomas    |       26 | M      | NULL  | Senior |   235 | 268936  | 2012-07-10 |
|      323 | Wilcox   | Daniel    |        3 | M      | TeamA | Senior |  NULL | 665993  | 2013-04-30 |
|      331 | Schmidt  | Thomas    |       25 | M      | NULL  | Senior |   153 | 867492  | 2013-03-20 |
|      332 | Bridges  | Deborah   |       12 | F      | NULL  | Senior |   235 | 279087  | 2011-03-05 |
|      339 | Young    | Betty     |       21 | F      | TeamB | Senior |  NULL | 507813  | 2013-03-30 |
|      414 | Gilmore  | Jan       |        5 | F      | TeamA | Junior |   153 | 459558  | 2011-05-12 |
|      415 | Taylor   | William   |        7 | M      | TeamA | Senior |   235 | 137353  | 2011-11-09 |
|      461 | Reed     | Robert    |        3 | M      | TeamA | Senior |   235 | 994664  | 2009-07-18 |
|      469 | Willis   | Carolyn   |       29 | F      | NULL  | Junior |  NULL | 688378  | 2014-12-27 |
|      487 | Kent     | Susan     |     NULL | F      | NULL  | Social |  NULL | 707217  | 2019-09-19 |
+----------+----------+-----------+----------+--------+-------+--------+-------+---------+------------+
20 rows in set (0.001 sec)
```

Team table

```
MariaDB [lab5task]> insert into team values
    -> ('TeamA','Tuesday',239),
    -> ('TeamB','Monday',153);
Query OK, 2 rows affected (0.149 sec)
Records: 2  Duplicates: 0  Warnings: 0

MariaDB [lab5task]> select * from team;
+----------+---------------+---------+
| TeamName | PracticeNight | Manager |
+----------+---------------+---------+
| TeamA    | Tuesday       |     239 |
| TeamB    | Monday        |     153 |
+----------+---------------+---------+
2 rows in set (0.000 sec)
```

Tournament table

```
MariaDB [lab5task]> insert into tournamenttable values
    -> (24,'Leeston','Social'),
    -> (25,'Kajapoi','Social'),
    -> (36,'WestCoast','Social'),
    -> (38,'Canterburry','Open'),
    -> (40,'Otago','Open');
Query OK, 5 rows affected (0.153 sec)
```

TourID	TourName	TourType
24	Leeston	Social
25	Kajapoi	Social
36	WestCoast	Social
38	Canterburry	Open
40	Otago	Open

Type table

```
MariaDB [lab5task]> insert into type values
    -> ('Associate',60),
    -> ('Junior',150),
    -> ('Senior',300),
    -> ('Social',50);
Query OK, 4 rows affected (0.324 sec)
```

Type	Fee
Associate	60
Junior	150
Senior	300
Social	50

Entry table

```
MariaDB [lab5task]> select * from entry;
```

MemberID	TourID	Year
228	24	2014
228	25	2014
228	36	2014
235	38	2012
235	38	2014
235	40	2013
235	40	2014
239	25	2014
239	40	2012
258	24	2013
258	38	2013
286	24	2012
286	24	2013
286	24	2014
415	25	2014
415	36	2013
415	36	2014
415	38	2012
415	40	2012

TASK 5.6:

Write the query for the following:

a) List the first name, last name, and phone numbers of all the members.

```
select FirstName, LastName, Phone from members;
```

b) List complete information of all the male members.

```
select * from members where Gender = 'M';
```

c) List complete information of all the members who joined after 01-01-2013.

```
select * from members where JoinDate= '01-Jun-13';
```

d) List name of all the members who belonged to Team A.

```
select * from members where Team = 'TeamA';
```

e) List complete information of all the senior members.

```
select * from members where MType= 'Senior';
```

f) List complete information of all the members in order of LastName.

```
select * from members order by LastName asc;
```

g) Retrieve the number of records in Member table.

```
select count(*) from members
```

h) Provide the first name and last name of the two coaches.

```
select FirstName, LastName from members limit 2
```

j) Delete the record from Entry table where Member=415 and TourID=40.

```
DELETE FROM member WHERE Member=415 and TourID=40;
```

k) Update the Fee of Associate in Type table from 60 to 80

```
UPDATE Type
SET Fee= 80;
```

TASK 5.7

MySQL supports various built-in functions belonging to various categories such as numeric functions, string functions, and date & time functions. Write MySQL commands for following numeric functions: ceiling, cos, degrees, log10, mod, radians, round, sqrt, and truncate. Next write MySQL commands for following string functions: concat, upper, lower, repeat, reverse,

regexp, replace, length, ltrim, and rtrim. Finally write MySQL commands for following date & time functions: curdate, week, date_from, quarter, now, sysdate, and date_format.

Numeric Functions:

CEILING:

The CEILING() function returns the smallest integer value that is bigger than or equal to a number.

Syntax:

CEILING(*number*)

Example:
Return the smallest integer value that is greater than or equal to 25:

```
SELECT CEILING(25);
```

COS:

The COS() function returns the cosine of a number.

Syntax

COS(*number*)

Example

Return the cosine of a number:

```
SELECT COS(2);
```

DEGREES:

The DEGREES() function converts a value in radians to degrees.

Syntax

DEGREES(*number*)

Example

Convert the radian value into degrees:

```
SELECT DEGREES(1.5);
```

LOG10:

The LOG10() function returns the natural logarithm of a number to base-10.

Syntax

LOG10(*number*)

Example

Return the base-10 logarithm of 4.5:

```
SELECT LOG10(4.5);
```

MOD:

The MOD() function returns the remainder of a number divided by another number.

Syntax

- MOD(*x*, *y*)
- *x* MOD *y*
- *x* % *y*

Example

Return the remainder of 18/4:

```
SELECT 18 MOD 4;
```

RADIANS:

The RADIANS() function converts a degree value into radians.

Syntax

RADIANS(*number*)

Example

Convert a degree value into radians:

```
SELECT RADIANS(-45);
```

ROUND:

The ROUND() function rounds a number to a specified number of decimal places.

Syntax

ROUND(*number*, *decimals*)

Example

Round the number to 2 decimal places:

```
SELECT ROUND(135.375, 2);
```

SQRT:

The SQRT() function returns the square root of a number.

Syntax

```
SQRT(number)
```

Example

Return the square root of a number:

```
SELECT SQRT(64);
```

TRUNCATE:

The TRUNCATE() function truncates a number to the specified number of decimal places.

Syntax

```
TRUNCATE(number, decimals)
```

Example

Return a number truncated to 2 decimal places:

```
SELECT TRUNCATE(135.375, 2);
```

Date & Time Functions:

CURDATE:

The CURDATE () function returns the current date.The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).This function equals the CURRENT_DATE() function.

Syntax

```
CURDATE()
```

Example

Return the current date:

```
SELECT CURDATE();
```

WEEK:

The WEEK() function returns the week number for a given date (a number from 0 to 53).

Syntax

```
WEEK(date, firstdayofweek)
```

Example

Return the week number for a date:

```
SELECT WEEK("2017-06-15");
```

QUARTER:

The QUARTER() function returns the quarter of the year for a given date value (a number from 1 to 4).

- January-March returns 1
- April-June returns 2
- July-Sep returns 3
- Oct-Dec returns 4

Syntax

```
QUARTER(date)
```

Example

Return the quarter of the year for the date:

```
SELECT QUARTER("2017-06-15");
```

NOW:

The NOW() function returns the current date and time.

Note: The date and time is returned as "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS.uuuuuu (numeric).

Syntax

NOW()

Example

Return current date and time:

```
SELECT NOW();
```

SYSDATE:

The SYSDATE() function returns the current date and time.

Note: The date and time is returned as "YYYY-MM-DD HH:MM:SS" (string) or as YYYYMMDDHHMMSS (numeric).

Syntax

SYSDATE()

Example

Return the current date and time:

```
SELECT SYSDATE();
```

DATE_FORMAT:

The DATE_FORMAT() function formats a date as specified.

Syntax

DATE_FORMAT(*date, format*)

Example

Format a date:

```
SELECT DATE_FORMAT("2017-06-15", "%Y");
```

TASK 5.8

MySQL uses various operators such as Comparison (<, >, <=, >=, ==, and !=), Boolean (AND, OR, and NOT), and Special Operators (Between, Like, IN, Is Null, and Distinct). Explore these.

Comparison Operators:

= = (Equal to):

```
SELECT * FROM Products
WHERE Price = 18;
```

> (Greater than):

```
SELECT * FROM Products
WHERE Price > 30;
```

< (Less than):

```
SELECT * FROM Products
WHERE Price < 30;
```

>= (Greater than or equal to):

```
SELECT * FROM Products
WHERE Price >= 30;
```

<= (Less than or equal to):

```
SELECT * FROM Products
WHERE Price <= 30;
```

<> (Not equal to):

```
SELECT * FROM Products
WHERE Price <> 18;
```

The SQL AND, OR and NOT Operators

The AND and OR operators are used to filter records based on more than one condition:

NOT:

Displays a record if the condition(s) is NOT TRUE

```
SELECT * FROM Customers
WHERE City NOT LIKE 's%';
```

OR:

TRUE if any of the conditions separated by OR is TRUE

```
SELECT * FROM Customers
WHERE City = "London" OR Country = "UK";
```

AND:

TRUE if all the conditions separated by AND is TRUE

```
SELECT * FROM Customers
WHERE City = "London" AND Country = "UK";
```

Special Operators:

BETWEEN:

TRUE if the operand is within the range of comparisons.

```
SELECT * FROM Products
WHERE Price BETWEEN 50 AND 60;
```

IN:

TRUE if the operand is equal to one of a list of expressions

```
SELECT * FROM Customers
WHERE City IN ('Paris','London');
```

LIKE:

TRUE if the operand matches a pattern.

```
SELECT * FROM Customers
WHERE City LIKE 's%';
```

ISNULL:

ISNULL() function lets you return an alternative value when an expression is NULL:

```
SELECT ProductName, UnitPrice * (UnitsInStock + ISNULL(UnitsOnOrder, 0))
FROM Products;
```

DISTINCT:

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```