PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

Faculty of Engineering & Technology

Department of Information and Communication Engineering

Lab Report

Course Title: Database Management Systems Sessional

Course Code: ICE-3106

SUBMITTED BY:	SUBMITTED TO:
Name: Md. Omar Faruk Roll:190605 Session:2018-2019 3rd Year 1st Semester	Md. Tofail Ahmed Assistant Professor Department of Information and Communication Engineering Pabna University of Science and Technology ..... Signature

L NO	NAME OF THE PROGRAM	
1	Study and Implementation of DML Commands of SQL with Suitable Example: i) Insert ii) Delete iii)Update	
2	Study and Implementation of DDL Commands of SQL with Suitable Example i)Create ii)Alter iii)Drop	
3	Study and Implementation of DML Commands of i)Select Clause ii)From Clause iii)Where Clause	
4	Study and Implementation of DML Commands of i)Group By & Having Clause ii)Order By Clause iii)Create View, Indexing & Procedure Clause	
5	Study and Implementation of SQL Commands of Join Operations with Example i)Cartesian Product ii) Nutural Join iii)Left Outer Join iv)Right Outer Join v)Full Outer Join	
6	Study and Implementation of Aggregate Function with Example i)Count Function ii)Max Function iii)Min Function iv)Avg Function	
7	Study and Implementation of Triggering System on Database Table Using SQL Commands with Example.	
8	Study and Implementation of SQL Commands to Connect MySQL Database with Java or PHP.	

Experiment NO:- 1

Experiment Name: study and implement of DML Commands of SQL with suitable example

- Insert
- Delete
- Update

Objectives:

1. To Understand and use of Data manipulation language to write query for database
2. To study how to insert, delete and update data in the database

Theory: A Datamanipulation language is a language that enables users to access a manipulate as organized by appropriate data model.

(1) The Retrival of information stored in the database

(2) Insertion of new information into the database.

(3) Deletion of information from database

(4) Modification of information stored in the database

(a) Insertion: To insert data into a relation we either specify a tuple to be inserted or write a query whose result is a set of tuples to be inserted

The simplest "insert" statement is a request to insert one tuple. Suppose we want to insert the fact that there is a course CS-437 in the computer science department with the title "Database systems" and four credit hours in a "course table". The statement will be

```
insert into course
        values ('CS-437', 'Database systems',
                'Com Sci', 4);
```

(11) Deletion:

A delete request is expressed in much the same way as a query. We can delete only whole tuples we can not delete value only particulars attributes.

SQL Expression for delete operation:

delete from r
where p

where p represent predicate and represent relation. Delete all tuples in the instructor relation pretaining to instractor in the finance department

Query →

delete from instractor
where dept-name = "finance".

(c) Update:

In cretian situations, we may wish to change a value in a tuple without changing all values in the tuple. for this purpose, the update statement can be used.

for example, suppose that annual salary increase are being made, and salaries for all instractors are to be incread by * 5 present.

we write,

```
update instructor
set salary = salary * 1.05;
```

Source code:

```
use master
create database university_180610
---TO use the university_180610 database

use university_180610
--create a table named department
( dept_name varchar(20),
  building varchar(15),
  budget numeric(12,2);
  primary key (dept_name);

insert into department values ('Biology','watson',90000)
insert into department values ('Comp Sci','taylor',100000)
insert into department values ('Elec.Eng','Taylor',85000)
insert into department values('Finance','Painter',120000)

insert int department values('History','pointer',50000)
insert into department values('Music','pakward',
                              80000)
```

```
--To show the department table with attribute
  and values.
--i insert int.it
select * from department
--To delete one tuple
delete from department where dept_name=
 'Biology'
--To update department
updat department set budget = budget *
 1.05 where budget < 85800
```

Output :

	Dept_name	building	budget
1	Biology	Watson	75000.00
2	Chemistry	Painter	100000.00
3	CSE	Taylor	90000.00
4	EEE	Taylor	80000.00
5	ICE	Watson	95000.00
6	Physics	Painter	85,000.00

	dept_name	building	budget
1	Chemistry	Painter	100000.00
2	CSE	Taylor	90000.00
3	EEE	Taylor	950000.00
4	ICE	Watson	80000.00
5	Physics	Painter	851000.00
6			

Experiment No: 02

Experiment name: Study and implementation of DDL commands of SQL with the suitable example.

- Create
- Alter
- Drop

Objectives:

① To study how to create, alter and drop table in database

② To understand and use of Data Definition language to write query for database

Theory:

We define SQL relation by using the create table command.

Data Definition Language :- we specify a database schema by a set of definitions expressed by a special language called data definition language

Create:

The following command create a relation department in the database.

```
create table department
    (  dept_name  varchar(20),
       building   varchar (15),
       budget     numeric (12,2),
       primary key (dept_name));
```

The general form of create table command is

```
create table r {
    A1 ... D1
    A2     D2
    ...  ...
    An     Dn
      (integrity constant1)
      (integrity constant));
```

where r is the name of the relation each A_i is the name of an attribute in the schema of relation r and D_i is the domain of Attribute A_i, that is D_i specifies the type of attribute A_i:

Drop

The drop command delete all information about the dropped relation from the database

The command is

drop table r;

Alter:

we use the alter table command to add attributes to an existing relation. All tuples in the relation are assigned null as the value for the new attribute the form of the alter table command is,

alter table r and A D;

where r is the name of an existing

A is the name of the to be added

and D is the type of the added attribute

We can' drop attributes from a relation by the command

altertable r drop A,

Source code:

```
use master
create database university
use University
--create a table name instanactor
create table instractor (
    ID varchar(5),
    name varchar(20) not null,
    dept_name varchar(21),
    salary numeric(8,2),
    primary key (ID));
```

Insert into instractors values ('1001', 'Srinivasal', 'com;sci', 65000)

Insert into instractions values ('12121', 'CWU', Finance', 90000)

insert into instractions values ('15151', 'Mozond', 'Music', 40000)

insert into instractions values ('22222', 'Einestain', 'physics', 95000)

insert into instractor values ('32343', 'EI Sid', 'History' 6000)

insert into instractor values ('33456', 'Gold',

Output:

	ID	name	dept-name	Salary
1	10101	Onar	physics	৳ 8000.00
2	10121	Fawk	Music	8500.00
3	10333	Maria	Biology	9000.00
4	12453	Sagor	ICT	75000.00
5	14544	Hossain	Chemistry	100000.00
6	15353	Amin	CSE	95000.00

	ID	name	dept-name	Salary	Course-ID
1	10101	Onar	physics	8000.00	NULL
2	10121	Fawk	music	8500.00	NULL
3	10333	Maria	Biology	9000.00	NULL
4	12453	Sagor	ICT	75000.00	NULL
5	14544	Hossain	Chemistry	100000.00	NULL
6	15353	Amin	CSE	95000.00	NULL

Select * from instructors

alter table instructors add course_no char(20);

drop table instructors.

Experiment no:- 03

Name of Experiment :- Study and implementation of DML commands of

- Select clause
- from clasuse
- where clause

Objectives:-

① To understand and use of the SQL queries

② To study how to implement select, from, where, clause in database

Theory: The basic structure of an SQL Quiries consist of three clauses:

- Select
- from
- where.

A query takes as its input the relations listen in the from classe, operates on them as specified in the where and select clauses and then produces a relation as the result.

The role of each clause in as follows —

① The select clause is used to list the attributes desired in the result of query

② The from clause is a list of the relation to be accessed in the evaluation of the query.

③ The where clause as a predicate involving attributes of the relation in the from clause.

A typical SQL query has the form:

(select $A_1, A_2 \ldots A_n$
from $r_1, r_2 \ldots r_n$
where p;

Queries on a single relation

Let us consider an example

"Find the name of all the instructions".

Instructor names are found in the Instructor relation, so we put that relation in the from clause. the instructor's name appears in the name attribute, so we put that in the select clause

```
select name
from instructor;
```

The result is a relation consisting of single attribute with the heading name

Queries on multiple relations:

Suppose we want to answer the following query:

"Retrive the names of all instructors along with their department names and department building name"

The query can be written in SQL as

```
select name, instructor, dept_name building
    from instructor, department
    where instructor.dept_name = department.
                                 dept_name
```

The select clause may contain arithmetic expression involving the operations +, -, *, and / operating on constant on attribute of tuples. For example

```
select ID, Salary * 1.1
from instructor.
```

Source code:

```
use university
select dept_name from instructor;
select name
from instructor
where dept_name = "physics";
```

Output:

	dept_name
1	Physics
2	Music
3	Biology
4	Chemistry
5	ICE
6	CSE

	Name
1	Omar

Experiment NO: 04

Name of the experiment :- Study and implement of DML command of

- Group By & Having clause
- Order By clause
- Create view, Indexing and procedure clause.

Objectives:

1. To understand and use the SQL queries
2. To study how to implement group by, having by, order by clause SQL code
3. To create view, indexing and procedure clause in database

Theory:

There are circumstances where we would like to apply aggregate function in this case we use group by and having clause.

the aggregate functions are:—

- Average : avg
- Maximum : max
- Minimum : min
- Total : sum
- Count : count

Group By clause: In some cases we apply aggregate function not only to a single set of tuples, but also to a group of sets of tuples; we specify this the attribute or attributes give in the group by clauses are placed in one group.

Example:-

Find the average salary in each department

Query: select dept-name, avg (salary) as avg_salary.

from instructor
group by dept_name;

Having clause:

It is useful to state a conditions that applies to groups rather than to tuples for example we want to see only those departments where the average salary of the instructors is more than 4200, this condition does not apply to a single tuple, rather it applies to each group constructed by the group by clause. To express such as query us ing clause.

We express this query:

```
select dept-name, avg (salary) as avg-
salary from instructor
group by dept-name
having avg (salary) 4200
```

Order by clause:

SQL offers the users to some control over the order in which tuples in a relation are displayed the 'order By' Blause the tuples in the result of a query to appear in sorted order

For example

```
select *
    from instructor
    order by salary dise, name ase;
```

By default, the order by clause lists in asending order, To specify the sort order, we may specify, "dese" for dsending order, "ase" for asending order.

Create view:

we define a view in SQL by using the create view command. to define, we must given the view name and must state the query that computes the view.

the form of the create view command is

create view v as <query expression>;

where <query expression> is only large query expression. The view name is represented by v.

Indexing: An index on an attribute of a relation is a data structure that allow the database system to find those tuples in the relation that a specified values for that attribute efficiency without scanning through all tuples of the realations

we create an index with the create index command which takes the form

create index (<index-name>

Source code:

```
use university
select dept_name, avg (salary) as avg_salary
```

```
from instructor
group by dept_name;

Select dept_name, avg (Salary) as avg_salary
from instructor
group by dept_name
having avg (salary) > 4200;
select * from instructor order by salary
                desc, names.

create view faculty as
select ID, name, dept_name
from instructor
  create index dept_index as instructor(
                dept_name);

CREATE PROCEDURE instruct_proc
As
BEGIN
Select name as dbtions_name from instructor
where ID = '15151'
END EXEC instruct_proc
Select * from instructor.
```

Outputs

	dept-name	Avg_Salary
1	Biology	90000.0000
2	Chemistry	910000.0000
3	CSE	95000.0000
4	ICE	75000.0000
5	MUSIC	85000.0000
6	physics	68000.0000

	dept-name	avg-Salary
1	Biology	90000.0000
2	Chemistry	100000.0000
3	CSE	95000.0000
4	MUSIC	85000.0000

	ID	name	dept_name	Salary	course_id
0	14544	Kamal	Chemisty	10000.00	NULL
2	15353	Samad	CSE	95000.00	NULL
3	10333	Salam	Biology	90000.00	NULL
4	10121	Rakib	MUSIC	85000.00	NULL
5	12453	Rafid	ICE	75000.00	NULL
6	10101	Anwar	physics	68000.0	NULL

	authors_name
1	Rakib

Experiment no-5

Experiment name: Study and implement of SQL commands of join operation with example.

- Cartession product
- Natural join
- Left outer join
- Right outer join
- Full outer join

Objectives:

① To understand and use of the cartesion product in SQL queries in database

② To study the commands of join operations and their implemetation on database

Theory:

Join operation: Join operations that allow the programmer to write some queries in a more natural way and to express some queries that deficult to do with only the cartiesion product. The goal of creating a join condition is that it helps to combine the data from two or more

DBMS tables. It is denoted by $\bowtie$.

The natural join :- The natural join operation operates on two relations and produces a relation as the result, unlike the cartesian product of two relation which concatenates each tuple of the first relation with every tuple of the second, natural join considers only those pairs of tuples with the same value on those attributes that appears in the schemas of both relations

for example,

we write the query for all students in the university who have taken some course, find their names the course ID of all courses they took as,

```
select name, coursed-id
from student natural
join takes;
```

Syntax:

Select $A_1, A_2 \dots A_n$
from r_1 natural join r_2 natural join
natural join r_n
where P;

* Outer join:

The outer-join operation works in a manner similar to the join operations but it preserves those tuples that would be lost in a join by creating tuples in the result containing null values. There are three forms of outer join:

① The left-outer join:

The left outer join preserves tuples only in the relation named before the left outer join operation

Example: Select *
from r_1 natural left

(ii) The Right-outer join -

The Right-outer join preserves tuples only in the relation named after the right outer join operation

Exppl :

Select $A_1, A_2 \ldots A_n$

from r_2 natural right outer join r_1

(iii) The full outer join - the full outer join preserves tuple in both relations It is the union of a left outer join and the corresponding right outer join

Select *

from r_1 natural full outer join r_2

Cartesian product :- the cartesian product operation allow us to combine information from any two relation

Source code:-

```
use university
  Select * from instructor
  Slect * from departments
--- cartesion product
Select building, departments, dept_name, Salary
--- join operation
Select Salary, buiding
from departments join instractors on
departments, dept_name = instructor.
dept-name

--- left outer join
  select * from departments left outer
                -join instructor on
  departmens.dept_name = instructors.dept_
  name
-- Right outer join
  select * from instractor right
  outer join departments on.
```

department.dept_name = instructor.dept_name

-- Full outer join

select * from instructor full outer join department on

department.dept_name = instructors.dept_name.

Output:

Cartesian product

	building	dept-name	Salary
1	Painter	physics	68000.00
2	watson	Biology	90000.00
3	Watson	ICE	75000.00
4	painter	Chemistry	100000.00
5	Paylon	CSE	95000.00

join:

	building	Salary
1	painter	68000.00
2	watson	90000.00
3	watson	750000.00
4	Painter	100000.00
5	Painter	9500.00

Right outer join:

	dept-name	building	budget	ID	name	deptname
1	Biology	watson	70000.0	10333	Srlam	Biology
2	Chemistry	painter	100000.00	14544	kamal	Chemistry
3	CSE	onur	90000.0	15355	Garney	CSE
4	EEE	Farruk	950000.0	NULL	NULL	NULL
5	ICE	watson	871063	12453	Rafid	ICE
6	Physics	painter	8500.0	10101	Anwar	Physics

Full onder join:

	ID	name	dept-name	Salary	dept-name	Bilding
1	10101	Anwar	physics	68000.0	Physics	Bilogy
2	10121	Rakib	music	8500.0	NULL	Chemisty
3	10333	Srlam	Bilogy	9000.0	Biology	CSE
4	12453	Rakib	ICE	750000	ICE	IEE
5	14544	kamal	Chemisty	9500.0	Chemistry	Tatyor
6	NULL	NULL	NULL	NULL	CSE	Onor

Experiment on: 06

Name of the experiment: Study and Implementation of Aggregate function with exam.

- Count function
- Max function
- Min function
- Avg function

Objectives:

1. To understand and use of the aggregate function on database
2. To study how to implement count() max(), min(), Avg(), function in SQL on database

Theory:

Aggregate functions are functions that take a collection of values as input and return a single value. SQL often five standard built-in

functions.

- Average : avg
- Minimum : min
- Maximum : max
- Total : sum
- Count : Count

The input to sum and avg must be a collection of numbers, but the other operations can operate on collections of non-numeric data types, such as string as well

Avg(): This avg() function returns the average value of some collection of number in a data table

Example:

```
select avg(salary) as average salary
from instructor
where dept_name = "physics";
```

Min(): This min() functions returns the minimum value from some collection of numbers under on attribute in a relation.

Example:

```
Select min(salary) as minimum-salary
from instructor
where dept-name = "physics";
```

This query returns the minimum salary of a physics department.

max(): max() function returns the maximum value from some collection of numeric values under as attribute in a relation

Example:

```
Select max(salary) as maximum=
                      salary
from instructions
where dept-name = "physics";
```

Count() : count() functions returns number of tuples in a relation. usually.

The notation for this function in SQL
Count(*)

Example:

To find the number of tuples in the instructor relation we can write:

```
: Select count(*)
  from instructor;
```

There are some cases where must eliminate duplicates before computing an aggregate function:

for example

```
select count (distinct ID)
from teaches
where semester = 'Spring' and
                 year = 2018;
```

This return the total number of instructor who teach a course in the spring 2018 semester.

Source code:

```
use university
select count(ID) as count-ID
from instructor;
select max(salary) as max-salary from instruc
  for
select min(salary) as min-salary from instructors
select avg(salary) as avg-salary
from instractor;
```

Output:

	Course_id
1	6

	max_salary
1	100000.0

	min_salary
1	85500.0000

Experiment no:- 07

Name of Experiment :- Study and implementation of Triggering system on Database table using SQL commands with example

Objectives:

① To understand the triggering system on Database table

② To understand how-trigger can be used for automatically updating a table when an insert / update / delete statement takes place in another table.

Theory:

Trigger is a special type of procedure which is attached to a table and is only executed when an insert, update, or delete occurs on that table. One day to specify the modification actions that find the trigger when it is create

unlike stored procedure, trigger can not be explicity excuted. Once we enter a trigger into the database, the database system takes on the responsibility of executing it whenever the specified event occurs and the corresponding condition is satisfied

System

CRETE TRIGGER trigger-name on

For INSERT | DELETE | UPDATE

AS

BEGIN

TRIGGER BODY

END

where create trigger creates or replace an existing trigger with the trigger name.

Source code:

```
create database storetable
use storetable
create table customer(
(cust_id char(5) primary key
check (cust_id like
[[cs][0-9][0-9][0-9]
[0-9])),
cuse_fname char(12) NOT NULL
cost_lname varchar(12)
cust_address);
insert customer values ('C001', 'Omar',
'Faruk', 'pabna').
create table items
( item_id char(3) primary key check
(item_id like ([P][0-9]
[0-9][0-9][0-9]));
item_name char(12),
item_category char(10),
item_price float(12) check (item_price>20)
```

```
item_qon int check (item_qon>=0)
item_lst_sold datetime default getdate()
);

select * from items
insert items values('P0001','Jumnan',
                    'laptop',125.52k)

insert items value('P0003','Realem','10-2-2021
                    'phone',80.5)

insert item values('P00004','HP','laptop',
                   '5-14-20','103.5','2-18-22')

create table transact
( tran_id char(5) check (tran_id like
'C'[L1][0-9][0-9][0-9][0-9][0-9][0-9])

item_id char(5) foreign key references
                    items(item_id),
cust_id char(5) foreign key 'reffernces
                    customer(cust_id),
```

```
trans_type char(1),
tran_quantity int check (trans_quantity>0),
tran_date datetime default getdate()
)
select * from transcwet

-- Trigger
create trigger test on items FOR INSERT
AS
BEGIN
Declare @item_id char(5), @@amount
char(12), @tran_type char(1),
Select @item_id = item_id, @amount =
        trans_quantity
                                    )
If (@tran_type = '5')
    Update Items set item_qoh = item_qoh-@
                amount where item=id =
                @item_id
```

```
ELSE
    update Items set item_qoh = item_qoh +
        @amount where item_id = @item_id

END
Insert transact value ('10000004', '00001',
                       '0001', '5', '08',)
select * from transact
select * from Items.
```

Experiment No:-08

Name of the Experiment: Study and implementation of SQL commands to connect MySQL database with java or php

Objectives:

① To understand and use the SQL queries on database

② To study the SQL commands to connect mysql database with java or php

Theory:

To connect java application with the my sql data base, we need to allow following step.

In this example we are using MySQL as the database so we need to know following information, for the my SQL database.

① Driver class: The driver class for the mysql database is com.mysql.Jdbc.Driver

② Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/sonoo where jbdc is the API. mysql is the database, localhost is the server name on which mysql is running. we may also use IP address 3306 is the part number and sonoo is the database name. we may use only database, in such case we need to replace the sonoo with out database name

③ Username: The default username for the mysql database is root

④ Password: It is the password given by the user at the time of installing the mysql database. In this case, we are

as the password

Source code:

```
import java.sql.*;
public class TestSQLJava
{
connection con;
TestSQLJava()
  {
    Create connection();
  }
  void create connection()
    {
      try {
          con = DriverManager.get connection
          ("Jdbc:odbc:Test Data"), }
catch (Exception exp) { System.out.print'
      (No connection with DB" + exp); }
  }
Void show data()
  {
    try {
```

```
{ Statement st = con.createStatement();
  ResultSet rs = st.executeQuery
    ("select * from Authors");
    systemout.println(rs.getString(1)+
    "\t" + rs.getString(2));
  }
catch (Exception exp) { System.out.print
("Problem with query result" + exc:}
}
void updateData(String au_id, String state)
{ try
  { PreparedStatement pst=
    con.prepareStatement ("Update
    authors set state=? where au-id=?)
pst.setString(1, State);
pst.setString(2, au_id);
pst.executeUpdate();
}
```