Third Year of Computer Engineering (2019 Course)

Database Management Systems Laboratory (310246)

Lab Manual

Prepared By

Mr. R. P. Sabale

(2025-26)

Sr. No.	Name of Practical	Page No.	Date	Remark
	Group A - SQL & PL/SQL			
1	**ER Modeling :** Study of Open Source Databases: MYSQL and selecting one real time application. Draw ER Diagram And Create Tables.			
2	**SQL Queries:** a) Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View,Index, Sequence, Synonym b)Write at least 10 SQL queries on the suitable database application using SQL DML statements			
3	**SQL Queries - all types of Join, Sub-Query and View:** Write at least 10 SQL queries for suitable database application using SQL DML statements .			
4	Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.			
5	Write a PL/SQL block to create a procedure and function.			
6	Write a PL/SQL block to implement all types Cursors: (All types: Implicit, Explicit) for selected real time application .			
7	Write a PL/SQL program to useDatabase Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).			
8	Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)			
Group B :Large Scale Databases:NOSQL : MongoDB				
9	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)			
10	Implement aggregation and indexing with suitable example using MongoDB			
11	Implement Map reduces operation with suitable example using MongoDB.			
12	Write a program to implement MongoDB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)			
13	Group C Mini Project : Student should develop application in group of 2-3 students and submit the Project Report			

INDEX

Practical No. 1

Title : Select one real time application and Draw ER Diagram And Create Tables.

Objectives : To develop basic, intermediate and advanced Database programming skills

Theory : A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

So now days, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored in to different tables and relations are established using primary keys or other keys known as foreign keys.

A Relational DataBase Management System(RDBMS) is a software that:

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables

RDBMS Terminology:

Before we proceed to explain database system, let's revise few definitions related to database.

Database: A database is a collection of tables, with related data.

Table: A table is a matrix with data. A table in a database looks like a simple spreadsheet. **Column**: One column (data element) contains data of one and the same kind, for example the Column postcode.

Row: A row (tuple, entry or record) is a group of related data, for example the data of one subscription.

Redundancy: Storing data twice, redundantly to make the system faster.

Primary Key: A primary key is unique. A key value cannot occur twice in one table. With a key, you can find at most one row.

Foreign Key: A foreign key is the linking pin between two tables.

Compound Key: A compound key(composite key) is a key that consists of multiple columns, Because one column is not sufficiently unique.

Index: An index in a database resembles an index at the back of a book.

Referential Integrity: Referential Integrity makes sure that a foreign key value always points to An existing row.

Entity relationship diagram (ERD):

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database.ER diagrams are used to sketch out the design of a database.

Common Entity Relationship Diagram Symbols:

There are five main components of an ERD:

- **Entities**, which are represented by rectangles.

- An entity is an object or concept about which you want to store information.
- A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.

- **Actions**, which are represented by diamond shapes, show how two entities share information in the database.In some cases, entities can be self-linked. For example, employees can supervise other employees.
-

- **Attributes**, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute.

- A multivalued attribute can have more than one value.

- For example, an employee entity can have multiple skill values.
- A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.

- **Connecting lines**, solid lines that connect attributes to show the relationships of entities in the diagram.
- **Cardinality** specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional..

Lab Work: Take a real-time application like an "Online Shopping System" and design a database using the Entity-Relationship (ER) model.

ER Model:

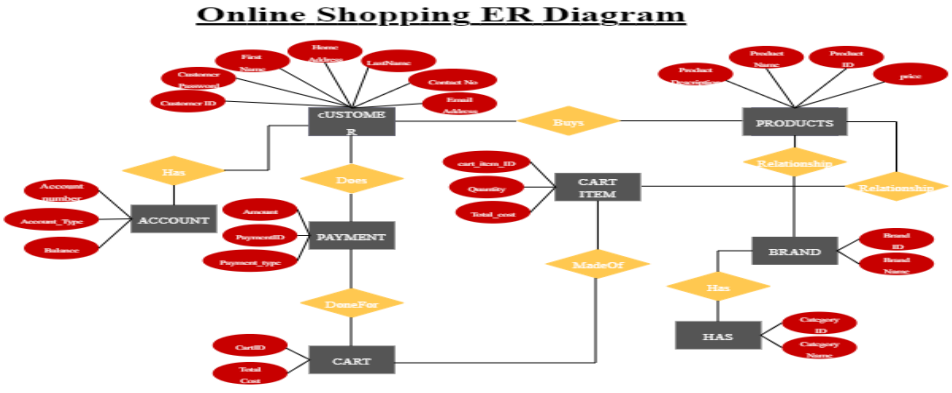
Entities:

1. Customer
   - Customer ID (Primary Key)
   - Name
   - Email
   - Phone Number
   - Address
2. Product
   - Product ID (Primary Key)
   - Product Name
   - Description
   - Price
   - Quantity
3. Order
   - Order ID (Primary Key)
   - Customer ID (Foreign Key)
   - Order Date
   - Total Amount
4. Order Item
   - Order Item ID (Primary Key)
   - Order ID (Foreign Key)
   - Product ID (Foreign Key)
   - Quantity
   - Price
5. Payment
   - Payment ID (Primary Key)
   - Order ID (Foreign Key)
   - Payment Method
   - Payment Date

Relationships:

1. Customer-Order: One customer can place many orders (One-to-Many).
2. Order-Order Item: One order can have many order items (One-to-Many).
3. Product-Order Item: One product can be part of many order items (One-to-Many).
4. Order-Payment: One order can have one payment (One-to-One).

ER Diagram: Draw diagram with the entities as rectangles, attributes as lists inside the rectangles, and relationships



.

Converting ER Model to Tables:

Tables:

1. Customer
   - Customer ID (Primary Key)
   - Name
   - Email
   - Phone Number
   - Address
2. Product
   - Product ID (Primary Key)
   - Product Name
   - Description
   - Price
   - Quantity
3. Order
   - Order ID (Primary Key)
   - Customer ID (Foreign Key referencing Customer table)
   - Order Date
   - Total Amount
4. Order Item
   - Order Item ID (Primary Key)
   - Order ID (Foreign Key referencing Order table)
   - Product ID (Foreign Key referencing Product table)
   - Quantity
   - Price
5. Payment
   - Payment ID (Primary Key)
   - Order ID (Foreign Key referencing Order table)
   - Payment Method
   - Payment Date

This design captures the key entities and relationships in an online shopping system, and the tables can be used to store and manage data for the application.

Conclusion: we have selected a real-time application "Online Shopping System" and Designed a database using the Entity-Relationship (ER) model.

. .

Practical No. 2 -A

Title: Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View,Index, Sequence, Synonym on selected real time application

Objectives: To study SQL DDL statements

LAB WORK:

SQL queries:Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table,View,Index,Sequence,Synonym

mysql>create database dbms;

mysql> use dbms;

1. **Create table:**

mysql> create table stud (RollNo int, Name varchar(20), Class varchar(20));

//Show Structure of Table//

mysql> desc stud;

//Alter table to add Branch & MobileNo//

mysql> alter table stud add Branch varchar(20);

mysql> alter table stud add MobileNo int;

mysql> desc stud;

//Insert values into table//

mysql> insert into stud values (1,'ABC','TE','Comp',123);

mysql> insert into stud values (2,'PQR','TE','Comp',456);

mysql> insert into stud values (3,'XYZ','TE','Comp',789);

//Display Table//

mysql> select *from stud;

//Drop particular Column//

mysql> alter table stud drop MobileNo;

mysql> select*from stud;

//Add particular Column//

mysql> alter table stud add MobileNo int;

mysql> select*from stud;

//Update in table//

mysql> update stud set Class='SE' where RollNo='3';

mysql> select*from stud;

mysql> select RollNo,Name,Class from stud;

//Truncate-Delete all rows//

mysql> truncate table stud;

mysql> select *from stud;

Empty set (0.03 sec)

mysql> desc stud;

//Drop table//

mysql> drop table stud;

mysql> desc stud;

ERROR 1146 (42S02): Table 'dbms.stud' doesn't exist

2. **View**

mysql> use dbms;
Database changed

//Create Table//
mysql> create table stud(RollNo int, Name varchar(20), Class varchar(20));

mysql> insert into stud values (1,'ABC','comp');

mysql> insert into stud values (2,'qaz','comp');

mysql> insert into stud values (3,'wsx','comp');

mysql> select*from stud;

//Create view from table//
mysql> create view std as select RollNo,Name from stud;

mysql> select*from std;

//Modify View by create or replace statement//
mysql> create or replace view std as select Name from stud where Name='wsx';

mysql> select*from std;

mysql> drop view std;

mysql> select*from std;

ERROR 1146 (42S02): Table 'dbms.std' doesn't exist

3. INDEX

mysql> use dbms;
Database changed

//create Table for Index//
mysql> create table stud(RollNo int, Name char(20),Class char(20), Branch char(20));

mysql> insert into stud values (1,'qaz','TE','COMP');

mysql> insert into stud values (2,'wsx','TE','COMP');

mysql> insert into stud values (3,'edc','TE','COMP');

mysql> insert into stud values (4,'rfv','TE','COMP');

mysql> select*from stud;

//Create INDEX//
mysql> create index ind_1 on stud(Class);

mysql> create index ind_2 on stud(Name);
//Display INDEX//
mysql> show indexes from stud;

Conclusion: **Thus we have studied how to use SQL DDL Statements.**

Practical No. 2-B

Title:- Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.

Objectives:- To study SQL DML statements

THEORY: Data Manipulation Language (DML):data manipulation language (DML) is a family of syntax elements similar to a computer programming language used for selecting, inserting, deleting and updating data in
database. Performing read-only queries of data is sometimes also considered a component of DML.
Data manipulation language comprises the SQL data change statements, [2] which modify stored data but not the schema or database objects.
Data manipulation languages have their functional capability organized by the initial word in a statement, which is almost always a verb. In the case of SQL, these verbs are:

- SELECT ... FROM ... WHERE ...
- INSERT INTO ... VALUES ...
- UPDATE ... SET ... WHERE ...
- DELETE FROM ... WHERE ...

LABWORK:
SQLQueries: Design at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.
Example: Create table for student Marksheet:

SrNo	SeatNo	Name	Class	Branch	DC	EME	MCA	DNMS	CNS
1	T190273001	Shubhangi	TE	E&TC	41	42	48	40	47
2	T190273002	Shital	TE	E&TC	40	40	48	45	48
3	T190273003	Shruti	TE	E&TC	90	80	89	83	74
4	T190273004	Nikita	TE	E&TC	81	81	85	92	65
5	T190273005	Ashwini	TE	E&TC	47	40	44	50	60
6	T190273006	Manjukesh	TE	E&TC	40	16	32	21	36

SQL Queries used:

1. See available database:
   - **mysql> show databases;**
2. Choose Database from available:
   - **mysql> use trupti;**

3. Create table with name Student_Marksheet according to given example:
   - **mysql> create table Student_Marksheet(SrNo int, SeatNo varchar(20), Name varchar(20), Class varchar(10), Branch varchar(10), DC int, EME int, MCA int, DBMS int, CNS int);**
4. See Structure of table with **desc** command:
   - **mysql> desc Student_Marksheet;**
5. Enter alteast 6 Records in Student_Marksheet table:
   - **mysql> insert into Student_marksheet values (1,'T190273001','Shubhangi','TE','E&TC',41,42,48,40,47);**
   - **mysql> insert into Student_marksheet values (2,'T190273002','Shital','TE','E&TC',40,40,48,45,48);**
   - **mysql> insert into Student_marksheet values (3,'T190273003','Shruti','TE','E&TC',90,80,89,83,74);**
   - **mysql> insert into Student_marksheet values (4,'T190273004','Nikita','TE','E&TC',81,81,85,92,65);**
   - **mysql> insert into Student_marksheet values (5,'T190273005','Ashwini','TE','E&TC',47,40,44,50,40);**
   - **mysql> insert into Student_marksheet values (6,'T190273006','Manjukesh','TE','E&TC',40,16,32,21,36);**
6. Display all records in Student_Marksheet by using **SELECT*FROM** command:
   - **mysql> select*from Student_Marksheet;**
7. Now add another two column as Total_Marks and Average _Marks into Student_Marksheet by using **ALTER TABLE** command:
   - **mysql> alter table Student_Marksheet add Total_Marks int;**
   - **mysql> alter table Student_Marksheet add Avrage_Marks int;**
8. Display all records in Student_Marksheet by using **SELECT*FROM** command:
   - **mysql> select*from Student_Marksheet;**
9. Now set value of Total_Marks Column using **UPDATE AND SET OPERATOR** command
   - **mysql> update Student_Marksheet set Total_marks=DC+EME+MCA+DBMS+CNS;**
   - **mysql> update Student_Marksheet set Avrage_marks=Total_Marks/5;**
10. Display all records in Student_Marksheet by using **SELECT*FROM** command:
    - **mysql> select*from Student_Marksheet;**
11. Display who are get maximum marks in DBMS subject using **SELECT FROM AND MAX** Command:

- **mysql> select Name,DBMS from Student_Marksheet where DBMS=(select max(DBMS) from Student_Marksheet);**

12. Display who are get minimum marks in DBMS subject using **SELECT FROM AND MIN** Command:
    - **mysql> select Name,DBMS from Student_Marksheet where DBMS=(select min(DBMS) from Student_Marksheet);**
13. Display who are get maximum Total_Marks using **SELECT FROM AND MAX** Command:
    - **mysql> select Name,Total_Marks from Student_Marksheet where Total_Marks=(select max(Total_marks) from Student_Marksheet);**
14. Display Students names who are gets marks in DC subjects in between 40 and 80 using **SELECT FROM WHERE CAUSE** Command:
    - **mysql> select Name,DC from Student_Marksheet where DC between 40 and 80;**
15. Add a column of Percentage in Student_Marksheet table to find % of each students by using **ALTER TABLE** Command:
    - **mysql> alter table Student_Marksheet add Percentage int;**
16. Calculate Percentage of each student using **UPDATE and SET OPERATOR** Command:
    - **mysql> update Student_Marksheet set Percentage=((Total_Marks)/500)*100;**
17. Display all records in Student_Marksheet by using **SELECT*FROM** command:
    - **mysql> select*from Student_Marksheet;**

FINAL OUTPUT :

SrNo	SeatNo	Name	Class	Branch	DC	EME	MCA	DNMS	CNS	Percentage
1	T190273001	Shubhangi	TE	E&TC	41	42	48	40	47	44%
2	T190273002	Shital	TE	E&TC	40	40	48	45	48	44%
3	T190273003	Shruti	TE	E&TC	90	80	89	83	74	83%
4	T190273004	Nikita	TE	E&TC	81	81	85	92	65	81%
5	T190273005	Ashwini	TE	E&TC	47	40	44	50	60	44%
6	T190273006	Manjukesh	TE	E&TC	40	16	32	21	36	29%

Conclusion: 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator designed and executed..

Practical No: 3

Title:- Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

Objectives:- To study all types of Join, Sub-Query and View SQL statements.

LAB WORK:

Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

Example: 1. Create table for Student:

SeatNo	Name	Class	Branch
T190273001	Shubhangi	TE	E&TC
T190273002	Shital	TE	E&TC
T190273003	Shruti	TE	E&TC
T190273004	Nikita	TE	E&TC
T190273005	Ashwini	TE	E&TC
T190273006	Manjukesh	TE	E&TC

2. Create Table for Marksheet:

SeatNo	DC	EME	MCA	DNMS	CNS
T190273001	41	42	48	40	47
T190273002	40	40	48	45	48
T190273003	90	80	89	83	74
T190273004	81	81	85	92	65
T190273005	47	40	44	50	60
T190273006	40	16	32	21	36

SQL Queries used:

18. See available database:
    - **mysql> show databases;**
19. Choose Database from available:
    - **mysql> use trupti;**
20. Create table with name Student according to given example:
    - **mysql> create table Student(SeatNo varchar(20), Name varchar(20), Class varchar(10), Branch varchar(10));**
21. See Structure of table with **desc** command:
    - **mysql> desc Student;**
22. Enter alteast 6 Records in Student table:
    - **mysql> insert into Student values ('T190273001','Shubhangi','TE','E&TC');**

- **mysql> insert into Student_marksheet values ('T190273002','Shital','TE','E&TC');**
- **mysql> insert into Student_marksheet values ('T190273003','Shruti','TE','E&TC');**
- **mysql> insert into Student_marksheet values ('T190273004','Nikita','TE','E&TC');**
- **mysql> insert into Student_marksheet values ('T190273005','Ashwini','TE','E&TC');**
- **mysql> insert into Student_marksheet values ('T190273006','Manjukesh','TE','E&TC');**

23. Create table with name Marksheet according to given example:

- **mysql> create table Marksheet(SeatNo varchar(20), DC int, EME int,MCA int, DBMS int, CNS int);**

24. See Structure of table with **desc** command:

- **mysql> desc Marksheet;**

25. Enter alteast 6 Records in Marksheet table:

- **mysql> insert into Marksheet values ('T190273001',41,42,48,40,47);**
- **mysql> insert into Marksheet values ('T190273002',40,40,48,45,48);**
- **mysql> insert into Marksheet values ('T190273003' ,90,80,89,83,74);**
- **mysql> insert into Marksheet values ('T190273004', 81,81,85,92,65);**
- **mysql> insert into Marksheet values ('T190273005', 47,40,44,50,40);**
- **mysql> insert into Marksheet values ('T190273006',40,16,32,21,36);**

26. Display all records in Student and Marksheet by using **SELECT*FROM** command:

- **mysql> select*from Student;**
- **mysql> select*from Marksheet;**

27. As both Table having SeatNo Column same value then we can perform join function on table Student and Marksheet using **INNER JOIN with CLOUMN NAME FROM TABLE1 & TABLE2** command:

- **mysql> select student.Name,marksheet.DC,marksheet.EME,marksheet.DBMS,marksheet.MCA,marksheet.CNS from student inner join marksheet on student.SeatNo=marksheet.SeatNo;**

28. As both Table having SeatNo Column same value then we can perform join function on table Student and Marksheet using **LEFT JOIN with CLOUMN NAME FROM TABLE1 & TABLE2** command:

- **mysql> select student.SeatNo, student.Name,marksheet.DC from student left join marksheet on student.SeatNo=marksheet.SeatNo;**

29. As both Table having SeatNo Column same value then we can perform join function on table Student and Marksheet using **RIGHT JOIN with CLOUMN NAME FROM TABLE1 & TABLE2** command:

- **mysql> select student.SeatNo, student.Name,marksheet.EME,marksheet.CNS from student right join marksheet on student.SeatNo=marksheet.SeatNo;**

30. As both Table having SeatNo Column same value then we can perform join function on table Student and Marksheet using **CROSS JOIN with CLOUMN NAME FROM TABLE1 & TABLE2** command:

- **mysql> select student.SeatNo, student.Name,marksheet.DC,marksheet.EME from student cross join marksheet;**

31. Pass SUB-QURRIES on table Student or Marksheet

SELECT
FROM
WHERE..... Sub-Quarries
HAVING.....Sub-Quarries

- **Find Maximum marks in EME subject:**
  - **mysql> select SeatNo,EME from marksheet where EME=(select max(EME) from marksheet);**
- **Find all Subject Average Values:**
  - **mysql> select avg(DC),avg(EME),avg(DBMS),avg(MCA),avg(CNS) from marksheet;**
- **Find total Count of Student in Table:**
  - **mysql> select count(SeatNo) from marksheet;**
- **find SUM of all subject in table:**
  - **mysql> select sum(DC),sum(EME),sum(DBMS),sum(MCA),sum(CNS) from marksheet;**
- **Find MIN, MAX & Average marks for DC & CNS subject:**
  - **mysql> select min(DC),max(DC),avg(DC),min(CNS),max(CNS),avg(CNS) from marksheet;**
- **Find Students who got marks greater than 60 in DC Subject:**
  - **mysql> select SeatNo, DC from marksheet having DC>60;**
- **Find Students who got marks in rage of 50 to 90 in MCA Subject:**
  - **mysql> select SeatNo,MCA from marksheet having MCA between 50 and 90;**

32. Create View from marksheet table with **CLOUMN LIST FROM TABLE** command:

- **Create View DC from Marksheet Table:**
  - **mysql> create view DC as select SeatNo, DC from marksheet;**
- **update view by create or replace command**
  - **mysql> create or replace view DC as select SeatNo, DC from marksheet where SeatNo='T190273004';**

- **display all rows in DC by SELECT *FROM:**
  - **mysql> select * from DC;**
- **Delete view by DROP:**
  - **mysql> drop view DC;**

Conclusion: Thus we have studied to use & implement various join operation with nested queries.

Practical No: 4

Title : Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.PL/SQLcode block to calculate the area of a circle for a value of radius varying from 5 to 9.

LAB WORK:

create table areas (r number(2), area number (14,2));

```
declare
r number(5);
area number(14,2);
pi constant number (4,2):=3.14;
begin
r:=5;
while r<=9
loop
```

```
area:=pi*power(r,2);
insert into areas values(r,area );
r:=r+1;
end loop;
end;
/
select * from areas;
```

Conclusion: PL/SQLcode block to calculate the area of a circle for a value of radius varying from 5 to 9. Is executed.

Practical No: 5

Title :- PL/SQL Stored Procedure and Stored Function. : Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored arebetween989 and 900 categoryisfirstclass,ifmarks899and 825categoryisHigherSecond Class

LAB WORK:

```
setautoprint on;
setserveroutput on;
set verify off;
create table marks(rollno number(10), name varchar2(20), sub1 number, sub2 number, sub3 number, sub4 number, sub5 number);
insert into marks values(1,'Karishma',200,250,260,270,150);
insert into marks values(2,'Mayuri',210,220,230,240,250);
insert into marks values(3,'Asim',250,260,270,280,290);
insert into marks values(4,'Vaishnavi',180,190,200,220,250);
```

```
select * from marks;
create table stud_marks(rollno number(10),name varchar2(20), total_marks number(10));
create table Result(rollno number(10), name varchar2(20), class varchar2(30));
create procedure PROC_GRADE
is
sum number(20);
i number(10);
n number(10);
rollnol number(10);
namel varchar2(30);
totall number(10);
classl varchar2(30);
s1 number;
s2 number;
s3 number;
s4 number;
s5 number;
begin
select count(*) into n from marks;
i := 0;
loop
i:=i+1;
selectrollno into rollnol from marks where rollno=i;
select name into namel from marks where rollno=i;
select sub1 into s1 from marks where rollno=i;
select sub2 into s2 from marks where rollno=i;
select sub3 into s3 from marks where rollno=i;
select sub4 into s4 from marks where rollno=i;
select sub5 into s5 from marks where rollno=i;
totall:=a1+s2+s3+s4+s5;
iftotall<=1500 ANDtotall>=990 then
classl:='DISTINCTION';
end if;
iftotall<=989 AND totall>=900 then
classl:='FIRST CLASS';
end if;
iftotall<899 ANDtotall>=825 then
classl:='HIGER SECOND CLASS';
end if;
iftotall<825 then
classl:='PASS CLASS';
end if;
insert into stud_marks values(rollnol,namel,totall);
insert into result values(rollnol,namel,classl);
if i=n then
exit;
end if;
end loop;
end;
/
execute PROC_GRADE;
select * from stud_marks;
select * from result;
```

CONCLUSION: Stored procedure is created.

Practical No: 6

Title :- **Cursors: Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table Cust_New with the data available in the table Cust_Old. If the data in the first table already exist in the second table then that data should be skipped.**

Objective **:-** Learning the concept of cursor in PL/SQL

Theory **:- CURSOR:-**

For the processing of any SQL statement, database needs to allocate memory. This memory is called context area. The context area is a part of PGA (Process global area) and is allocated on the oracle server.

A cursor is associated with this work area used by ORACLE, for multi row queries. A cursor is a handle or pointer to the context area .The cursor allows to process contents in the context area row by row.

There are two types of cursors.

1) Implicit cursor:-Implicit cursors are defined by ORACLE implicitly. ORACLE

defines implicit cursor for every DML statements.

2) Explicit cursor:-These are user-defined cursors which are defined in the declaration section of the PL/SQL block. There are four steps in which the explicit cursor is processed.
    1) Declaring a cursor
    2) Opening a cursor
    3) Fetching rows from an opened cursor
    4) Closing cursor

General syntax for

CURSOR:-

```
DECLARE
    Cursor cursor_name IS select_statement
or query; BEGIN
    Open cursor_name;
Fetch cursor_name into
    list_of_variables; Close
    cursor_name;
END;
```

Where

1) Cursor_name:-is the name of the cursor.
2) Select_statement:-is the query that defines the set of rows to be processed by the cursor.

3) Open cursor_name:-open the cursor that has been previously declared. When cursor is opened following things happen

i) The active set pointer is set to the first row.

ii) The value of the binding variables are examined.

4) Fetch statement is used to retrieve a row from the selected rows, one at a time, into PL/SQL variables.

5) Close cursor_name:-When all of cursor rows have been retrieved, the cursor should be closed.

Explicit cursor attributes:-

Following are the cursor attributes

1. **%FOUND**: - This is Boolean attribute. It returns TRUE if the previous fetch returns a row and false if it doesn‟t.
2. **%NOTFOUND**:-If fetch returns a row it is often used returns FALSE and TRUE if it doesn‟t. This as the exit condition for the fetch loop;

3. **%ISOPEN**:-This attribute is used to determine whether or not the associated cursor is open. If so it returns TRUE otherwise FALSE.
4. **%ROWCOUNT**:-This numeric attribute returns a number of rows fetched by the cursor.

Cursor Fetch Loops

1) Simple

Loop Syntax:-

```
LOOP
    Fetch cursorname into list of variables;
EXIT WHEN
    cursorname%NOTFOUND
    Sequence_of_statements;
END LOOP;
```

2) WHILE

Loop Syntax:-

3) **Cursor FOR Loop Syntax:**

FETCH cursorname INTO list of variables; WHILE cursorname%FOUND

LOOP

Sequence_of_statements;

FETCH cursorname INTO list of variables; END LOOP;

FOR variable_name IN cursorname LOOP

-- an implicit fetch is done here.

-- cursorname%NOTFOUND is also implicitly checked.

-- process the fetch records.

Sequence_of_statements;

END LOOP;

There are two important things to note about :-

i) Variable_name is not declared in the DECLARE section. This variable is implicitly declared by the PL/SQL compiler.

ii) Type of this variable is cursorname%ROWTYPE.

Implicit Cursors

PL/SQL issues an implicit cursor whenever you execute a SQL statement directly in your code, as long as that code does not employ an explicit cursor. It is called an "implicit" cursor because you, the developer, do not explicitly declare a cursor for the SQL statement.

If you use an implicit cursor, Oracle performs the open, fetches, and close for you automatically; these actions are outside of your programmatic control. You can, however, obtain information about the most recently executed SQL statement by examining the values in the implicit SQL cursor attributes. PL/SQL employs an implicit cursor for each UPDATE, DELETE, or INSERT statement you execute in a program. You cannot, in other words, execute these statements within an explicit cursor, even if you want to. You have a choice between using an implicit or explicit cursor only when you execute a single-row SELECT statement (a SELECT that returns only one row).

In the following UPDATE statement, which gives everyone in the company a 10% raise, PL/SQL creates an implicit cursor to identify the set of rows in the table which would be affected by the update:

```
UPDATE employee
   SET salary = salary * 1.1;
```

The following single-row query calculates and returns the total salary for a department. Once again, PL/SQL creates an implicit cursor for this statement:

```
SELECT SUM (salary) INTO
  department_total FROM employee
 WHERE department_number = 10;
```

If you have a SELECT statement that returns more than one row, you must use an explicit cursor for that query and then process the rows returned one at a time. PL/SQL does not yet support any kind of array interface between a database table and a composite PL/SQL datatype such as a PL/SQL table.

LAB WORK:

Write a PL/SQL block of code using parameterized cursor that will merge the data available in newly created table Comp Dep with the data available in the Student. If the data in the first table already exists in the second table then that data should be skipped.

```
setautoprint on;
setserveroutput on;
set verify off;
create table CompDep(Roll number(10),Name varchar(20));
create table Student(Roll  number(10),Name varchar(20));
insert into Student values(1,'a');
insert into Student values(2,'b');
insert into Student values(3,'c');
insert into Student values(4,'d');
insert into CompDep values(2,'b');
insert into CompDep values(5,'e');
insert into CompDep values(6,'f');
declare
cursor cu1 is
selectRoll,Name from Student;
cursor cu2 is
select Roll from CompDep;
rno int;
nm varchar(20);
rno2 int;
begin
open cu1;
open cu2;
```

```
loop  fetch cu1 into rno,nm;
fetch cu2 into rno2;
exit when cu1%found = false;
if rno2 <>rno then insert into CompDep values(rno,nm);
end if;
end loop;
close cu1;
close cu2;
end;
/
Select *from CompDep;
```

Conclusion: cursors implemented.

Practical No: 7

Title :- Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

Objective :-Learning the concept of use of trigger

Theory :- DATABASE TRIGGERS:-

A database trigger is a PL/SQL program unit, which gets fired automatically whenever the data event such as DML or DDL system event. Triggers are associated with a specific table and are fired automatically whenever the table gets manipulated in a predefined way. The act of executing a trigger is called as firing a trigger.

Triggers are similar to procedures in that they are named PL/SQL blocks with declarative, executable and exception handling sections. But the difference is a procedure is executed explicitly from another block via a procedure call but a trigger is executed implicitly whenever the triggering event happens. A procedure can pass arguments but trigger doesn"t accept arguments

A database trigger has following components:-

1.A triggering **Event**

2.A triggering **Constraint**

3.A triggering

Action Trigger categories

Triggers are categorized in various ways.

1)Trigger type 2)Triggering time 3)Triggering event

Trigger types

There are two types of triggers

1. **Statement Trigger**:-A statement trigger is a trigger in which the trigger action is executed once for the manipulation operation that fires the trigger.

2. **Row Trigger**:-A row trigger is a trigger in which the trigger action is performed repeatedly for each row of the table that is affected by the manipulation operation that fires the trigger.Triggering time

Triggers can specify the time of trigger action.

1) Before the triggering event

The trigger action is performed before the operation that fires the trigger is executed. This trigger is used when execution of operation depends on trigger action.

2)After the triggering event

The trigger action is performed after the operation that fires the trigger is executed.
This trigger is used when triggering action depends on the execution of operation.

Triggering Events

Triggering events are the DML operations. These operations are **insert, update and delete** When these operations are performed on a table, the trigger which is associated with the operation is fired. Triggering events divide triggers into three types.

1) DELETE TRIGGER
2) UPDATE TRIGGER
3) INSERT TRIGGER

General syntax for creation of Trigger

Create [or replace] TRIGGER <trigger_name>
<BEFORE | AFTER>
DELETE | [OR] INSERT | [OR] UPDATE[OF <column1>[,<column2>…..]
ON <table_name>
[for each row[when <condition>]
Begin
……… ……
……..
End;

Where

Trigger_name:-tri gger name is the name of the trigger.
Table_name :-is thye table name for which the trigger is defined.
Trigger-condition:-The trigger condition in the when clause,if present is evaluated first.The body of the trigger is executed only when this condition evaluates to true.

Dropping trigger

Suppose you want to drop trigger then the syntax is

Syntax:-Drop trigger trigger_name;

Enabling and Disabling Triggers:

The Trigger can be disabled without dropping them. When the trigger is disabled, it is still exists in data dictionary but never fired, To disable trigger, use alter command.

Syntax:-

Alter TRIGGER trigger_name

DISABLE/ENABLE; For all triggers on a particular

table

Syntax:-

Alter TRIGGER trigger_name (DISABLE/ENABLE) all triggers;

Lab WORK:-Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. Theoldvalueofupdatedordeletedrecordsshouldbeaddedin Library_Audittable.

//Create USER for trigger:

Login to SQL plus with

User ID: sys as sysdba

Password: svit

//Write following Cammands to create user

```
create user C##Computer  identified by svit;

GRANT ALL PRIVILEGES TO C##Computer;

GRANT CONNECT TO C##Computer;

GRANT CREATE SESSION TO C##Computer;

GRANT CREATE TABLE TO  C##COMPUTER;

GRANT UNLIMITED TABLESPACE TO C##Computer;

GRANT CREATE VIEW, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TRIGGER to C##Computer;

GRANT ALTER ANY TABLE to C##Computer;

GRANT ALTER ANY PROCEDURE to C##Computer;

GRANT ALTER ANY TRIGGER to C##Computer;

GRANT DELETE ANY TABLE to C##Computer;

GRANT DROP ANY PROCEDURE to C##Computer;

GRANT DROP ANY TRIGGER to C##Computer;

GRANT DROP ANY VIEW to C##Computer;
```

//Exit from SQL open new window of SQL and Login with following details:

User ID: C##Computer

Password: svit;

```
create table library(bno number(5),bname varchar2(20),author varchar2(20), allowed_days
number(5));
insert into library values(1,'QWE','poi',10);
insert into library values(2,'ASD','lkj',15);
insert into library values(3,'zxc','mnb',10);
select * from library;
create table library_audit(bno number(5),old_all_days number(5),new_all_days
number(5));
create or replace trigger Sample before update or delete on library for each row
begin
insert into library_audit values(:new.bno,:old.allowed_days,:new.allowed_days);
end;
/
select * from library_audit;
update library set allowed_days = 15 where bno=1;
select * from library_audit;
```

Conclusion: Triggers generated.

PRACTICAL No : 8

Title: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

Lab work:

Implementation of MySQL/Oracle database connectivity for online shopping with database navigation operations like add, delete, and edit using Python:

Database Schema

Let's assume we have the following tables in our online shopping database:

- products: stores product information
- customers: stores customer information
- orders: stores order information

```
CREATE TABLE products (
    product_id INT AUTO_INCREMENT,
    product_name VARCHAR(255),
    price DECIMAL(10, 2),
    quantity INT,
    PRIMARY KEY (product_id)
);

CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT,
    name VARCHAR(255),
    email VARCHAR(255),
    password VARCHAR(255),
    PRIMARY KEY (customer_id)
);

CREATE TABLE orders (
    order_id INT AUTO_INCREMENT,
    customer_id INT,
    product_id INT,
    order_date DATE,
    PRIMARY KEY (order_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Python Implementation

We'll create a Python class OnlineShoppingDB to handle database operations:

```
import mysql.connector
import cx_Oracle

class OnlineShoppingDB:
    def __init__(self, host, user, password, database):
        self.host = host
        self.user = user
        self.password = password
        self.database = database

    def connect(self):
        try:
            # Connecting to MySQL database
            connection = mysql.connector.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                database=self.database
            )
            return connection
        except mysql.connector.Error as error:
            raise error

    def add_product(self, connection, product_name, price, quantity):
        try:
            cursor = connection.cursor()
            sql = "INSERT INTO products (product_name, price, quantity) VALUES (%s, %s, %s)"
            cursor.execute(sql, (product_name, price, quantity))
            connection.commit()
            print(f"Product '{product_name}' added successfully.")
        except mysql.connector.Error as error:
            raise error

    def delete_product(self, connection, product_id):
        try:
            cursor = connection.cursor()
            sql = "DELETE FROM products WHERE product_id = %s"
            cursor.execute(sql, (product_id,))
            connection.commit()
            print(f"Product with ID {product_id} deleted successfully.")
        except mysql.connector.Error as error:
            raise error

    def edit_product(self, connection, product_id, product_name, price, quantity):
        try:
            cursor = connection.cursor()
            sql = "UPDATE products SET product_name = %s, price = %s, quantity = %s WHERE product_id = %s"
            cursor.execute(sql, (product_name, price, quantity, product_id))
            connection.commit()
```

```
            print(f"Product with ID {product_id} updated successfully.")
        except mysql.connector.Error as error:
            raise error

    def place_order(self, connection, customer_id, product_id):
        try:
            cursor = connection.cursor()
            sql = "INSERT INTO orders (customer_id, product_id, order_date) VALUES (%s, %s, CURDATE())"
            cursor.execute(sql, (customer_id, product_id))
            connection.commit()
            print(f"Order placed successfully for customer ID {customer_id} and product ID {product_id}.")
        except mysql.connector.Error as error:
            raise error

# Example usage
db = OnlineShoppingDB("localhost", "username", "password", "online_shopping_db")
connection = db.connect()

# Add a product
db.add_product(connection, "Apple iPhone", 999.99, 10)

# Delete a product
db.delete_product(connection, 1)

# Edit a product
db.edit_product(connection, 1, "Apple iPhone 13", 1099.99, 5)

# Place an order
db.place_order(connection, 1, 1)
```

Sample Output:

```
Product 'Apple iPhone' added successfully.
Product with ID 1 deleted successfully.
Product with ID 1 updated successfully.
Order placed successfully for customer ID 1 and product ID 1.
```

For Oracle database connectivity, you can modify the connect method to use cx_Oracle:

```
def connect(self):
    try:
        connection = cx_Oracle.connect(
            user=self.user,
            password=self.password,
            dsn=f"{self.host}:1521/{self.database}"
        )
        return connection
    except cx_Oracle.DatabaseError as error:
        raise error
```

conclusion:
This implementation provides basic database operations for online shopping, including adding, deleting, and editing products, as well as placing orders .

Practical No. 9

Aim : Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

Objectives : Learn the concept of MONGO DB

Theory : **MongoDB** is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB .

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

CRUD is the basic operation of Mongodb ,it stands CREATE , READ , UPDATE, DELETE.

MongoDB ─ 1. Create Collection

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Basic syntax of createCollection() command is as follows:

db.createCollection(name, options)

In the command, name is name of collection to be created. Options are a document and are used to specify configuration of collection.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use:

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexID	Boolean	(Optional) If true, automatically create index on _id field. Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

While inserting the document, MongoDB first checks size field of capped collection, then it

checks max field.

Examples

Basic syntax of createCollection() method without options is as follows:

>use test

switched to db test

>db.createCollection("mycollection")

{ "ok" : 1 }

>

You can check the created collection by using the command show collections.

>show collections

mycollection

system.indexes

2. READ-The find() Method

To query data from MongoDB collection, you need to use MongoDB's find()method.

Syntax

The basic syntax of find() method is as follows:

>db.COLLECTION_NAME.find()

find()method will display all the documents in a non-structured way.

The pretty() Method

To display the results in a formatted way, you can use pretty() method.

Syntax

>db.mycol.find().pretty()

Example

>db.mycol.find().pretty()

{

"_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

>

Apart from find() method, there is findOne() method, that returns only one document.

3. UPDATE

MongoDB's update() and save() methods are used to update document into a collection.

The update() method updates the values in the existing document while the save() method replaces the existing document with the document passed in save() method.

MongoDB Update() Method

The update() method updates the values in the existing document.

The basic syntax of update() method is as follows:

>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)

Example

Consider the mycol collection has the following data.

{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}

{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will set the new title 'New MongoDB Tutorial' of the documents whose

title is 'MongoDB Overview'.

>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB

Tutorial'}})

>db.mycol.find()

{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}

{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

>

By default, MongoDB will update only a single document. To update multiple

documents, you need to set a parameter 'multi' to true.

>db.mycol.update({'title':'MongoDB Overview'},

{$set:{'title':'New MongoDB Tutorial'}},{multi:true})

MongoDB Save() Method

The save() method replaces the existing document with the new document passed in the save() method.

The basic syntax of MongoDB save() method is −

>db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})

Example

Following example will replace the document with the _id '5983548781331adf45ec7'.

```
>db.mycol.save(
{
"_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point
New Topic",
"by":"Tutorials Point"
} )
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"Tutorials Point New
Topic", "by":"Tutorials Point"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

4. DELETE-The remove() Method

MongoDB's remove() method is used to remove a document from the collection.

remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

- deletion criteria: (Optional) deletion criteria according to documents will be removed.
- justOne: (Optional) if set to true or 1, then remove only one document.

Basic syntax of remove() method is as follows:

>**db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)**

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
```

{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

Following example will remove all the documents whose title is 'MongoDB Overview'.

>db.mycol.remove({'title':'MongoDB Overview'})

>db.mycol.find()

{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}

{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

LOGICAL OPERATORS:

AND in MongoDB

Syntax

In the find() method, if you pass multiple keys by separating them by ',' then

MongoDB treats it as AND condition. Following is the basic syntax of AND −

>db.mycol.find({key1:value1,

key2:value2}).pretty() Example

Following example will show all the tutorials written by 'tutorials point' and whose title is

'MongoDB Overview'.

>db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty()

{

"_id":

ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql

database", "by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database",

"NoSQL"], "likes": "100"

}>

For the above given example, equivalent where clause will be ' where by='tutorials point' AND title = 'MongoDB Overview' '. You can pass any number of key, value pairs in find clause.

OR in MongoDB

Syntax : To query documents based on the OR condition, you need to use $or keyword.

Following is the basic syntax of OR −

```
>db.mycol.find( { $or: [ {key1: value1}, {key2:value2} ] } ).pretty()
```

Example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'.

```
>db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty()
```

```
{ "_id":
ObjectId(7df78ad8902
c), "title": "MongoDB
Overview",
"description": "MongoDB is no sql
database", "by": "tutorials point",
"url":
"http://www.tutorialspoint.com",
"tags": ["mongodb", "database",
"NoSQL"], "likes": "100" }
```

Using AND and OR Together Example

The following example will show the documents that have likes greater than 100 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is 'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

```
>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},{"title": "MongoDB Overview"}]}).pretty()
```

```
{
"_id":
ObjectId(7df78ad89
02c), "title":
"MongoDB
Overview",
"description": "MongoDB is no sql
database", "by": "tutorials point",
"url":
"http://www.tutorialspoint.com",
"tags": ["mongodb", "database",
"NoSQL"], "likes": "100" }
```

LAB WORK:

Mongo DB (Installation,BasicCRUDoperations, Execution)

Procedure to open Mongo dB Command line

Open C drive on PC then go to Program files □MonodB□server□version of software□bin folder □Double click on mongod.exe □wait for connection □Double click on mongo.exe□once connection successful then write following Quarries

```
showdbs;
use local;
db.createCollection('Student');
db.Student.insert({'Rno':'1','Name':'Piyush','Class':'TECOMP'});
db.Student.insert({'Rno':'2','Name':'Abhi','Class':'TECOMP'});
db.Student.insert({'Rno':'3','Name':'Ashley','Class':'TECOMP'});
db.Student.insert({'Rno':'4','Name':'Hitesh','Class':'TECOMP'});
db.Student.insert({'Rno':'5','Name':'Pratik','Class':'TECOMP'});
db.Student.insert({'Rno':'6','Name':'Priti','Class':'TECOMP'});
db.Student.find();
db.Student.find().pretty();
db.Student.update({'Name':'Hitesh'},{$set:{'Name':'Henry'}});
db.Student.find().pretty();
db.Student.remove({'Name':'Pratik'},1);
db.Student.remove({'Name':'Priti'},1);
db.Student.find().pretty();
db.Student.drop();
db.Student.find().pretty();
```

Conclusion: Thus we have studied MongoDB Queries using CRUD operations.

Practical No. 10

Aim : Implement aggregation and indexing with suitable example using MongoDB.

Objectives : Learn the concept of MongoDB

Theory : MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you great understanding on MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(*) and with group by is an equivalent of mongodb aggregation.
The aggregate() Method For the aggregation in MongoDB, you should use aggregate() method. Basic syntax of aggregate() method is as follows:
>db.COLLECTION_NAME.**aggregate**(AGGREGATE_OPERATION)

Example

In the collection you have the following data:

{
_id: ObjectId(7df78ad8902c) title: 'MongoDB Overview',
description: 'MongoDB is no
sql database', by_user:
'tutorials point',
url:
'http://www.tutorialspoint.c
om', tags: ['mongodb',
'database', 'NoSQL'], likes:
100
},

{

_id: ObjectId(7df78ad8902d)

title: 'NoSQL Overview',

description: 'No sql database
is very fast', by_user:
'tutorials point',
url:
'http://www.tutorialspoint.c
om', tags: ['mongodb',
'database', 'NoSQL'], likes:
10
},
{

_id:
ObjectId(7df78ad
8902e) title:
'Neo4j
Overview',
description: 'Neo4j is no
sql database', by_user:
'Neo4j',
url: 'http://www.neo4j.com',

tags: ['neo4j', 'database', 'NoSQL'], likes:

750
},

Now from the above collection, if you want to display a list stating how many tutorials are written by each user, then you will use the following aggregate() method:

> db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum :1}}}])

```
{
"result" : [

{

"_id" : "tutorials point", "num_tutorial" : 2
},
{
"_id" : "Neo4j","num_tutorial" : 1
}],
"ok" : 1

}>
```

Sql equivalent query for the above use case will be select by_user, count(*) from mycol group by by_user.

Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement This scan is highly inefficient and require MongoDB to process a large volume of data. Indexes are special data structures, that store a small portion of the data set in an easy -to-traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in the index.

The ensureIndex() Method

To create an index you need to use ensureIndex() method of MongoDB. The basic syntax of ensureIndex() method is as follows().

```
>db.COLLECTION_NAME.ensureIndex({KEY:1})
```

Here key is the name of the file on which you want to create index and 1 is for ascending order. To create index in descending order you need to use -1.

Example

```
>db.mycol.ensureIndex({"title":1})
```

In **ensureIndex()** method you can pass multiple fields, to create index on multiple fields.

```
>db.mycol.ensureIndex({"title":1,"description":-1})
```

ensureIndex() method also accepts list of options (which are optional).

LAB WORK:

Title: Implement aggregation and indexing with suitable example using MongoDB

```
Show dbs;
Use local;
//CREATECOLLECTIONWEBSITE
db.createCollection('website');

//INSERTVALUESINWEBSITE
db.website.insert({'roll':'1','name':'harsh','amount':1000,'url':'www.yahoo.com'});
db.website.insert({'roll':'2','name':'jitesh','amount':2000,'url':'www.yahoo.com'});
db.website.insert({'roll':'3','name':'rina','amount':3000,'url':'www.google.com'});
db.website.insert({'roll':'4','name':'ash','amount':4000,'url':'www.gmail.com'});
db.website.insert({'roll':'5','name':'ash','amount':1000,'url':'www.pvg.com'});

//SUMAGGREGATE

db.website.aggregate({$group:{_id:"$name","total":{$sum:"$amount"}}});

//AVGAGGREGATE

db.website.aggregate({$group:{_id:"$name","total":{$avg:"$amount"}}});

//MINAGGREGATION

db.website.aggregate({$group:{_id:"$name","total":{$min:"$amount"}}});

//MAXAGGREGATION

db.website.aggregate({$group:{_id:"$name","total":{$max:"$amount"}}});

//FIRSTAGGREGATION

db.website.aggregate({$group:{_id:"$name","total":{$first:"$amount"}}});

//LASTAGGREGATION
db.website.aggregate({$group:{_id:"$name","total":{$last:"$amount"}}});

//PUSHAGGREGATION

db.website.aggregate({$group:{_id:"$name","total":{$push:"$amount"}}});

//COUNTAGGREGATION

db.website.aggregate({$group:{_id:"$name","total": {$sum:1}}});

//ADDTOSETAGGREGATE

db.website.aggregate({$group:{_id:"$name","total"{$addToSet:"$amount"}}});
```

//INDEXING

```
db.createCollection('website1');
db.website1.insert({'r':1,'name':'harsh'});
db.website1.find().pretty()
db.website1.createIndex({'name':1})
```

//CREATEINDEXING

```
db.website1.createIndex({'name':-1})

db.website1.getIndexes()

db.website1.createIndex({'name':-1})
```

//DROPINDEX

```
db.website.dropIndex({'name':-1})
```

//GETINDEXING

```
db.website1.getIndexes()
db.website1.find().pretty()
db.website1.createIndex({'name':1})
db.website1.getIndexes()
db.website1.dropIndex({'name':1})
db.website1.getIndexes()
db.website1.createIndex({'name':1,'r':-1})
db.website1.getIndexes()
```

Conclusion: - Thus we have studied use and implementation of aggregation function &indexing function.

Practical No. 11

Aim : **Implement Map reduces operation with suitable example using MongoDB**

Objectives : Learn the concept of NOSQL MongoDB

LAB WORK:

MongoDB - Map reduces operations: Implement Map reduces operation with suitable example using MongoDB.

```
Show dbs;
Use local;
db.createCollection('Journal');
db.Journal.insert({'book_id':1,'book_name':'JavacdOOP','amt':500,'status':'Available'});
db.Journal.insert({'book_id':1,'book_name':'JavaOOP','amt':400,'status':'Not
Available'});
db.Journal.insert({'book_id':1,'book_name':'Java','amt':300,'status':'NotAvailable'});
db.Journal.insert({'book_id':2,'book_name':'Java','amt':300,'status':'Available'});
db.Journal.insert({'book_id':2,'book_name':'OPP','amt':200,'status':'Available'});
db.Journal.insert({'book_id':2,'book_name':'C+','amt':200,'status':'Available'});
db.Journal.insert({'book_id':3,'book_name':'C+','amt':150,'status':'Available'});
db.Journal.insert({'book_id':3,'book_name':'C++','amt':200,'status':'Not Available'});
db.Journal.insert({'book_id':4,'book_name':'OPPC++','amt':300,'status':'Not Available'});
db.Journal.insert({'book_id':5,'book_name':'OPP++','amt':400,'status':'Available'});
db.Journal.insert({'book_id':5,'book_name':'C++','amt':400,'status':'Available'});
db.Journal.insert({'book_id':5,'book_name':'C++Java','amt':400,'status':'Not Available'});
varmapfunction=function(){emit(this.book_id,this.amt)};
varreducefunction=function(key,value){returnArray.sum(value);};
db.Journal.mapReduce(mapfunction,reducefunction,{'out':'new'});
db.Journal.mapReduce(mapfunction,reducefunction,{'out':'new'}).find().pretty();
db.new.find().pretty();
```

Conclusion: Thus we have studied Map reduce function.

Practical No. 12

Title: Write a program to implement MogoDB database connectivity with PHP

Theory: To use MongoDB with PHP, we need to install additional MongoDB PHP driver.

Make a Connection and Select a Database

To make a connection, you need to specify the database name, if the database doesn't exist then MongoDB creates it automatically.

Following is the code to connect to the database –

```
<?php
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";
// select a database
$db = $m->mydb;
echo "Database mydb selected";
?>
```

When the program is executed, it will produce the following result −

Connection to database successfully Database mydb selected

Create a Collection:

Following is the code snippet to create a collection –

```
<?php
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";

// select a database
$db = $m->mydb;
echo "Database mydb selected";
$collection = $db->createCollection("mycol"); echo
"Collection created succsessfully";
?>
```

When the program is executed, it will produce the following result –

Connection to database successfully
Database mydb selected
Collection created succsessfully

Insert a Document

To insert a document into MongoDB, insert() method is used.

Following is the code snippet to insert a document –

```
<?php
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";

// select a database
$db = $m->mydb;
echo "Database mydb selected";
$collection = $db->mycol;
echo "Collection selected succsessfully";
```

```
    $document = array( "title" => "MongoDB",

      "description" => "database", "likes" =>
      100,
      "url" => "http://www.mongo.com/mongodb/", "by",
      "JIT"
    );

$collection->insert($document);
echo "Document inserted successfully";
 ?>
```

When the program is executed, it will produce the following result –
Connection to database successfully Database mydb selected
Collection selected succsessfully Document inserted successfully

Find All Documents

To select all documents from the collection, find() method is used.

Following is the code to select all documents –

```
 <?php
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";

// select a database
$db = $m->mydb;
echo "Database mydb selected";
$collection = $db->mycol;
echo "Collection selected succsessfully";


$cursor = $collection->find();
// iterate cursor to display title of documents


    foreach ($cursor as $document) { echo
      $document["title"] . "\n";
    }
 ?>
```

When the program is executed, it will produce the following result −
Connection to database successfully
Database mydb selected
Collection selected succsessfully { "title": "MongoDB"
}

Update a Document

To update a document, you need to use the update() method.

In the following example, we will update the title of inserted document to MongoDB. Following is the code snippet to update a document –

```
<?php
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";


// select a database
$db = $m->mydb;
echo "Database mydb selected";
$collection = $db->mycol;
echo "Collection selected succsessfully";


// now update the document
$collection->update(array("title"=>"MongoDB"), array('$set'=>array("title"=>"MongoDBNew")));
echo "Document updated successfully";


// now display the updated document
$cursor = $collection->find();


// iterate cursor to display title of documents echo
"Updated document";

foreach ($cursor as $document) { echo
$document["title"] . "\n";
}
?>
```

When the program is executed, it will produce the following result –
Connection to database successfully
Database mydb selected
Collection selected succsessfully
Document updated successfully
Updated document {
"title": "MongoDBNew"
}

Delete a Document: **To delete a document, you need to use remove() method.**

In the following example, we will remove the documents that has the title MongoDB Tutorial. Following is the code to delete a document –

```
<?php
```

```
// connect to mongodb
$m = new MongoClient();
echo "Connection to database successfully";

// select a database
$db = $m->mydb;
echo "Database mydb selected";
$collection = $db->mycol;
echo "Collection selected succsessfully";
// now remove the document
$collection->remove(array("title"=>"MongoDBNew"),false); echo
"Documents deleted successfully";
// now display the available documents
$cursor = $collection->find();

// iterate cursor to display title of documents echo
"Updated document";
foreach ($cursor as $document) { echo
$document["title"] . "\n";
}
?>
```

When the program is executed, it will produce the following result –

Connection to database successfully Database mydb selected
Collection selected succsessfully Documents deleted successfully

In the above example, the second parameter is boolean type and used for justOne field of remove() method.

Conclusion: Thus we learnt and successfully implemented MongoDB database connectivity with mongoDB.

Write a program to implement MongoDB database connectivity with any front end language to

implement Database navigation operations (add, delete, edit etc.)

```
db=client["student"]
collection=db["stud"]
```

```
def create_data():
    r=int(input("Enter roll no:
"))
    n=input("Enter name: ")
    m=input("Enter Marks: ")
    data={"Roll
No":r,"Name":n,"Marks":m}

insert_doc=collection.insert_
one(data)
    print("Record Inserted")

def show():
```

```
show_data=collection.find()
    for data in show_data:
        print(data)

def update():
    r=int(input("Enter roll no to
Update: "))
    m=input("Enter Marks: ")
    update_multiple =
collection.update_many({"Ro
ll No": r}, {"$set":{ "Marks":
m}})
    print(update_multiple)
```

```
    condition={"Roll No":r}
    val={"$set":{"Marks":m}}
    update_single =
collection.update_one(conditi
on,val)
    print(update_single)

def delete():
    r=int(input("Enter roll no to
Delete: "))
    delt={"Roll No": r}
    collection.delete_one(delt)
    print("Record Deleted Roll
No:",r)
```

while True:

Practical No:13 MINI PROJECT

Title: Student should develop application in group of 2-3 students and submit the Project Report which will consist of documentation related to different phases of Software Development Life Cycle:

- Title of the Project, Abstract, Introduction
- Software Requirement Specification
- Conceptual Design using ER features, Relational Model in appropriate Normalize form
- Graphical User Interface, Source Code
- Testing document
- Conclusion

- Develop application considering: Front End : Java/Perl/PHP/Python/Ruby/.net/any other language
  Backend : MongoDB/MySQL/Oracle

.