STES's

SINHGAD COLLEGE OF ENGINEERING

Vadgaon(Bk), Pune

Department of Computer Engineering

LABORATORY MANUAL

2022-23

DATABASE MANAGEMENT SYSTEM

TE-COMPUTER ENGINEERING

SEMESTER-I

Subject Code: **310241**

Roll No : **Batch :** Seat No:

TEACHING SCHEME
Lectures: 3Hrs/Week
Practical: 2 Hrs./Week

EXAMINATION SCHEME
Practical: 25 Marks
Term Work: 25 Marks

-: **Name of Faculty**:-
Prof. M.D. Sale
Prof. H.E.Chaudhari
Prof. A. V. Dirgule

Sinhgad Institutes

Sinhgad Technical Educational Society's

SINHGAD COLLEGE OF ENGINEERING

PUNE

CERTIFICATE

This is to certify that

*Mr./ Miss*_____,

*Of Class*_____ *Roll No.*___ *Has completed all the practical work in the subject*_____ *satisfactorily in the Department of*_____ *as prescribed by Savitribai Phule Pune University, in the academic year 2022 – 2023*

Staff In-charge Head of the Department Principal

INDEX

Sr. No.	Name of Assignment	Page No.	Date	Sign
	Group A - Database Programming Languages – SQL, PL/SQL			
1	**ER Modeling and Normalization:** Decide a case study related to real time application in group of 2-3 students and formulate a problem statement for application to be developed. Propose a Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize Relational data model.			
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.** Design at least 10 SQL queries for suitable database application using SQL DML statements: **Insert, Select, Update, Delete with operators, functions and set operator**.			
3	**SQL Queries – all types of Join, Sub-Query and View:** Write at least10 SQL queries for suitable database application using SQL DML statements.			
4	**Unnamed PL/SQL code block:** Use of Control structure and Exception handling is mandatory. Write a PL/SQLblock of code for the following requirements:- Schema: 1. **Borrower**(Roll,Name,DateofIssue, NameofBook, Status 2. **Fine** (Roll, Date, Amt) Accept Roll & N ame of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.			
5.	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.			

6	**Named PL/SQL Block: 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 are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block for using procedure created with above requirement. **Stud_Marks**(name, total_marks) **Result**(Roll,Name, Class)			
7	**Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)** 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.			
8	**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.			
9	**Database Connectivity:** 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 NOSQL Databases			
1	**Study of Open Source NOSQL Database**: MongoDB (Installation, Basic CRUD operations, Execution)			
2	**MongoDB Queries:** Design and Develop MongoDB Queries using **CRUD operations**. (Use CRUD operations, SAVE method, logical operators)			
3	**MongoDB – Aggregation and Indexing:** Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB.			

4	**MongoDB – Map-reduces operations:** Implement Map reduces operation with suitable example using MongoDB.			
5	**Database Connectivity:** Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)			
	Group C: Mini Project			
1	Using the database concepts covered in Group A and Group B, develop an application with following details: 1. Follow the same problem statement decided in Assignment -1 of Group A. 2. Follow the Software Development Life cycle and other concepts learnt in Software Engineering Course throughout the implementation. 3. Develop application considering: • Front End: Java/Perl/PHP/Python/Ruby/.net/any other language • Backend : MongoDB/ MySQL/Oracle 4. Test and validate application using Manual/Automation testing. 5. 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.			

PART-I DATABASE MANAGEMENT SYSTEM

GROUP A

Database Programming Languages – SQL, PL/SQL

Assignment No.	**1**
Title	**ER Modeling and Normalization:** Decide a case study related to real time application in group of 2-3 students and formulate a problem statement for application to be developed. Propose a Conceptual Design using ER features using tools like ERD plus, ER Win etc. (Identifying entities, relationships between entities, attributes, keys, cardinalities, generalization, specialization etc.) Convert the ER diagram into relational tables and normalize Relational data model.
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	Database Management System Laboratory
Signature	

Assignment No. 1

Title : ER Modeling and Normalization:

Ans:

Conceptual Design includes

Ideas −→ High-level design −→ Relational database schema −→ Relational DBMS

• Entity-Relationship model is used in the conceptual design of a database (conceptual level, conceptual schema) • Design is independent of all physical considerations (DBMS, OS, . . .).

Questions that are addressed during conceptual design: – What are the entities and relationships of interest (miniworld)? – What information about entities and relationships among entities needs to be stored in the database? – What are the constraints (or business rules) that (must) hold for the entities and relationships? • A database schema in the ER model can be represented pictorially (Entity-Relationship diagram)

Entity Types, Entity Sets, Attributes and Keys

• Entity: real-world object or thing with an independent existence and which is distinguishable f rom other objects. Examples are a person, car, customer, product, gene, book etc. • Attributes: an entity is represented by a set of attributes (its descriptive properties), e.g., name, age, salary, price etc. Attribute values that describe each entity become a major part of the data eventually stored in a database. • With each attribute a domain is associated, i.e., a set of permitted values for an attribute. Possible domains are integer, string, date, etc.

• Entity Type: Collection of entities that all have the same attributes, e.g., persons, cars, customers etc. • Entity Set: Collection of entities of a particular entity type at any point in time; entity set is typically referred to using the same name as entity type.

Key attributes of an Entity Type • Entities of an entity type need to be distinguishable. •
A superkey of an entity type is a set of one or more attributes whose values uniquely determine each entity in an entity set. • A candidate key of an entity type is a minimal (in terms of number of attributes) superkey. • For an entity type, several candidate keys may exist. During conceptual design, one of the candidate keys is selected to be the primary key of the entity type.

Relationships, Relationship Types, and Relationship Sets
• Relationship (instance): association among two or more entities, e.g., "customer 'Smith' orders product 'PC42' "

• Relationship Type: collection of similar relationships An n-ary relationship type R links n entity types E1, .En. Each relationship in a relationship set R of a relationship type involves entities e1 ∈ E1, . . . , en ∈ En R ⊆ {(e1, . . . , en) | e1 ∈ E1, . . . , en ∈ En} where (e1, . . . , en) is a relationship.

• Degree of a relationship: refers to the number of entity types that participate in the relationship type (binary, ternary, . . .).

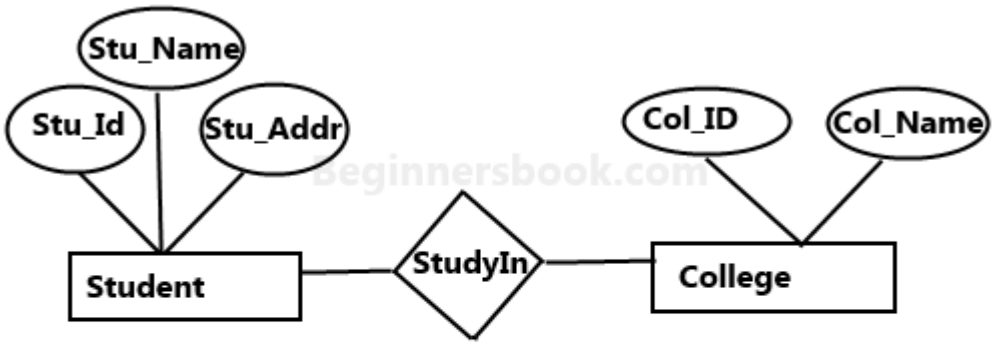
• Roles: The same entity type can participate more than once in a relationship type.

EMPLOYEES reports_to supervisor subordinate Role labels clarify semantics of a relationship, i.e., the way in which an entity participates in a relationship. recursive relationship.

• Relationship Attributes:
A relationship type can have attributes describing properties of a relationship. "customer 'Smith' ordered product 'PC42' on January 11, 2005, for $2345". These are attributes that cannot be associated with participating entities only, i.e., they make only sense in the context of a relationship.
• Note that a relationship does not have key attributes! The identification of a particular relationship in a relationship set occurs through the keys of participating entities.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities nd hese entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept

A simple ER Diagram:



Sample E-R Diagram

In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students

however a student cannot study in multiple colleges at the same time. Student entity has attributes

such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID &

Col_Name.

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms

in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much

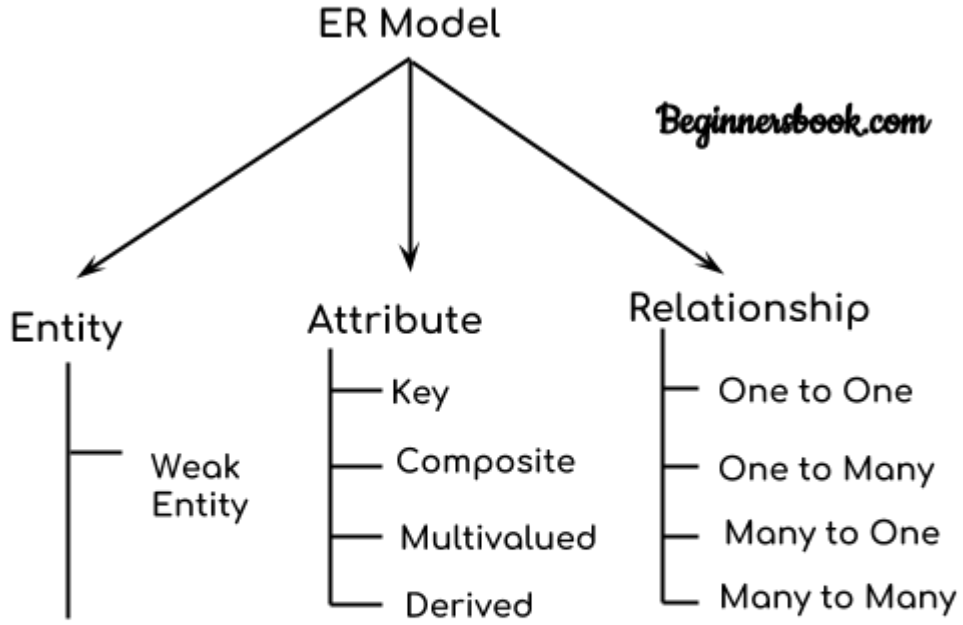
about these terms now, just go through them once.

Rectangle: Represents Entity sets.
Ellipses: Attributes
Diamonds: Relationship Set
Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set

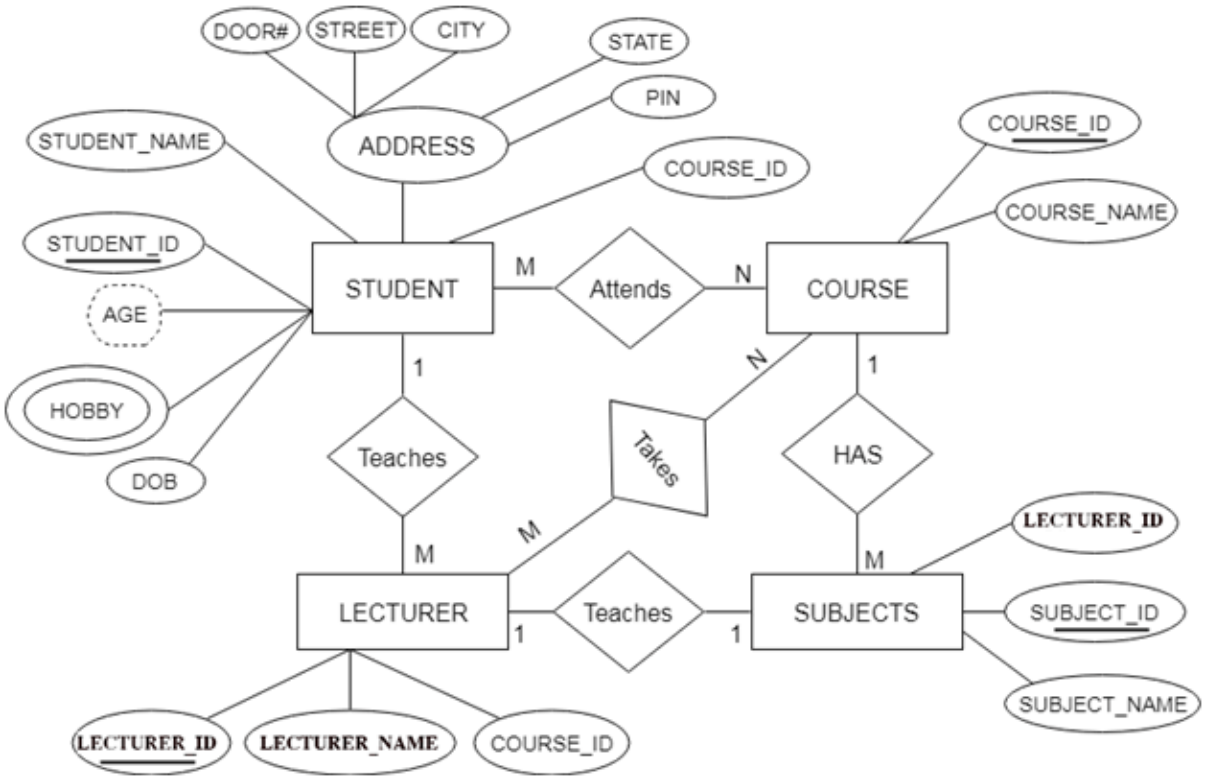
Components of a ER Diagram

Conversion of ER diagram to TableThe database can be represented using the notations, and these notations can be reduced to a collection of tables.In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below:



Components of ER Diagram



There are some points for converting the ER diagram to the table:

- **Entity type becomes a table.**

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

- **All single-valued attribute becomes a column for the table.**

In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table.

Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

- **A key attribute of the entity type represented by the primary key.**

In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID are the key attribute of the entity.

- **The multivalued attribute is represented by a separate table.**

In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values

in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

- **Composite attribute represented by components.**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

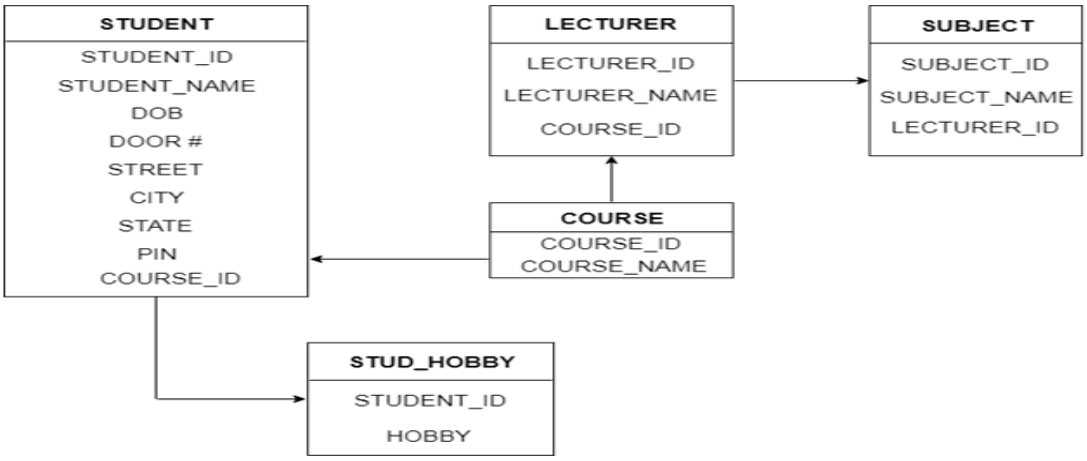
- **Derived attributes are not considered in the table.**

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating

the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below

Figure: Table structure



Normalization:- Normalization is a process that "improves" a database design by generating relations that are of higher normal forms.
The objective of normalization:
"to create relations where every dependency is on the key, the whole key, and nothing but the key".
Normalization Rule
Normalization rules are divided into the following normal forms:

1. 1NF is considered the weakest,
2. 2NF is stronger than 1NF,
3. 3NF is stronger than 2NF, and
4. BCNF is considered the strongest
5. Fourth Normal Form

any relation that is in BCNF, is in 3NF;
any relation in 3NF is in 2NF; and
any relation in 2NF is in 1NF.

First Normal Form (1NF)
If tables in a database are not even in the 1st Normal Form, it is considered as **bad database design**.
For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

Second Normal Form (2NF) :
For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency

Student_id	name	reg_no	branch	address

subject_id	subject_name

- student_id is the primary key and will be unique for every row, hence we can use student_id to fetch any row of data from this table
- This is Dependency and we also call it Functional Dependency.

Third Normal Form (3NF) :

A table is said to be in the Third Normal Form when,

- **It is in the Second Normal form.**

And, it doesn't have Transitive Dependency.

score_id	student_id	subject_id	marks	exam_name	total_marks

- student_id + subject_id.----pk

- exam_name is dependent on both student_id and subject_id.
- total_marks depends on exam_name as with exam type the total score changes
- This is Transitive Dependency. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.Boyce and Codd Normal Form (BCNF)

Boyce-Codd Normal Form or BCNF is an extension to the third normal form, and is also known as 3.5 Normal Form.

For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

1. It should be in the Third Normal Form.
2. And, for any dependency A → B, A should be a super key**.**

Conclusion : Thus we have studied how to modify E-R Model & Normalization.

FAQ :

1. What is ER Model?:
2. Explain in details how to convert ER Diagram to Table Form.
3. What is Normalization:?
4. What is 1NF?
5. What is 2NF?
6. What is 3NF?

Lab Exercise

1 : Suppose you are given the following requirements for a simple database for the National Hockey League (NHL): • the NHL has many teams, • each team has a name, a city, a coach, a captain, and a set of players, • each player belongs to only one team, • each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records, • a team captain is also a player, • a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2). Construct a clean and concise ER diagram for the NHL database

2. A university registrar's office maintains data about the following entities: 1. courses, including number, title, credits, syllabus, and prerequisites; 2. course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; 3. students, including student-id, name, and program; 4. instructors, including identi-cation number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's of-ce.Document all assumptions that you make about the mapping constraints.

3(a) Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

3(b) Construct appropriate tables for the above ER Diagram ?

4(a) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.

4(b) Construct appropriate tables for the above ER Diagram :

5. Consider a database used to record the marks that students get in different exams of different course offerings.

a) Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.

b) Construct an alternative E-R diagram that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering

Assignment No.	**2 (a)**
Title	**SQL Queries:** a. Design and Develop SQLDDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.
Roll No.	
Class	T.E. (C.E.)
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 2 (a)

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

Objectives: To study SQL DDL statements

Theory: SQL – Structured Query Language

Data Definition in SQLCreating Tables

Syntax:-

```
Create table<table
name>
(colume_name 1
datatype size(),
colume_name 2
datatype size(),
....
colume_name n datatype size());
```

e.g. Create table student with the following fields(name,roll,class,branch)

```
Create table student
(name
char(20),
Roll
number(
5), Class
char(10),
Branch
char(15));
```

A table from a table

- **Syntax :**

  CREATE TABLE<TableName>(<ColumnName>,
  <Columnname>)ASSELECT <ColumnName>,
  <Columnname> FROM <TableName>;

  - If the source table contains the records, then new table is also created with the same records present in the source table.

- If you want only structure without records then select statement must havecondition.
- **Syntax:**

  CREATE TABLE <TableName> (<ColumnName>, <Columnname>) ASSELECT <ColumnName>, <Columnname> FROM <TableName> WHERE 1=2; (Or)

  CREATE TABLE <TableName> (<ColumnName>, <Columnname>) ASSELECT <ColumnName>, <Columnname> FROM <TableName> WHERE ColumnName =NULL;

Constraints

The definition of a table may include the specification of integrity constraints. Basicallytwo types of constraints are provided: column constraints are associated with a single column whereas table constraints are typically associated with more than one column. Aconstraint can be named. It is advisable to name a constraint in order to get more meaningful information when this constraint is violated due to, e.g., an insertion of a tuple that violates the constraint. If no name is specified for the constraint, Oracle automatically generates a name of the pattern SYS C<number>.Rules are enforced on data being stored in a table, are called **Constraints**.

Both the Create table & Alter Table SQL can be used to write SQL sentences thatattach constraints.

Basically constraints are of three types :

1) Domain
   - Not Null
   - Check
2) Entity
   - Primary Key
   - Unique
3) Referential
   - Foreign key
   -
4) Not Null:-Not null constraint can be applied at column level only.

We can define these constraints

1) at the time of table creation Syntax :

   CREATE TABLE <tableName> (<ColumnName> datatype(size) NOT NULL,

   <ColumnName> datatype(size),.... **);**

2) After the table creation

ALTER TABLE <tableName>
Modify(<ColumnName> datatype(size)
NOT NULL);

Check constraints

Can be bound to column or a table using CREATE TABLE

or ALTER TABLEcommand.

- Checks are performed when write operation is performed .
- Insert or update statement causes the relevant check constraint.
- Ensures the integrity of the data in tables.

Syntax :

- Check constraints at column levelSyntax :

CREATE TABLE <table Name> (<ColumnName>data type(size)CHECK(column Namecondition),<column name datat ype(size));
CREATE TABLE <table Name>

(<Column Name> data type(size) CONSTRAINT <constraint_name>

CHECK (column Name condition),..

);

- Check constraints at **table levelSyntax :**

CREATE TABLE <table Name>

(<Column Name> data type(size),
<Column Name> data type(size),
CONSTRAINT <constraint_name> CHECK (column Name condition),..);

- Check constraints at table level

Syntax :

CREATE TABLE

<table Name> (<Column Name>data type(size),
<Column Name> data type(size),….,

CHECK (column Name condition));After table

creation
Alter table tablename

Add constraints constraintname

check(condition)The PRIMARY KEY Constraint

A primary key is one or more column(s) in a table used to uniquely identify each row inthe table.

- A table can have only one primary key. Can not be left blank Data must be UNIQUE.
- Not allows null values
- Not allows duplicate values.
- Unique index is created automatically if there is a

primary key.Primary key constraint defined at column level

Syntax:

CREATE TABLE <Table Name>

(<ColumnName1> <Data Type>(<Size>)PRIMARY KEY,<columnname2

<data type(<size>),.....);

- Primary key constraint defined at

Table levelSyntax:

CREATE TABLE <Table Name>

(<ColumnName1><Data Type>(<Size>) ,....,PRIMARY

KEY(<ColumnName1> <ColumnName2>));

- key constraint defined at Table level

Syntax:

CREATE TABLE <Table Name>

(<ColumnName1><DataType>(<Size>)<columnname2 data type<(size)<,<columnname3 data type<size>constraint constraint namePRIMARY KEY(<ColumnName1>));

After table creation

Alter table table name

Add(constraint constraint name primary key(column name));

The Unique Key Constraint

- The unique column constraint permits multiple entries of NULL into thecolumn.
- Unique key not allowed duplicate values
- Unique index is automatically created.
- Table can have more than one unique key.
- UNIQUE constraint defined at

column levelSyntax :

Create table tablename(**<column name> <data type>(<Size>**

UNIQUE),<colum nname> datatype(<size>).....);

UNIQUE constraint defined at
table levelSyntax :

CREATE TABLE tablename (<columnname> <datatype>(<Size>), <columnname> <datatype>(<Size>), UNIQUE(<columnname>, <columnname>));

After table creation

Alter table tablename

Add constraint constraintname unique(columnname);

- The Foreign Key (Self Reference) Constraint Foreign key represents relationshipsbetween tables.

A foreign key is a column(or group of columns) whose values are derived from primarykey or unique key of some other table.

Foreign key constraint defined at
column levelSyntax:
<columnName> <DataType> (<size>) REFERENCES <TableName>[(<ColumnName>)][ON DELETE CASCADE]

- If the ON DELETE CASCADE option is set, a DELETE operation in the
- master table will trigger a DELETE operation for corresponding records in all detailtables.
- If the ON DELETE SET NULL option is set, a DELETE operation in the master table will set the value held by the foreign key of the detail tablesto null.

Foreign key :

ALTER TABLE <child_tablename> ADD CONSTRAINT <constraint_name> FOREIGNKEY (<columnname in child_table>) REFERENCES <parent table name>;

FOREIGN KEY constraint at table level

1) FOREIGN KEY constraint defined with ON DELETE CASCADE FOREIGN KEY(<ColumnName>[,<columnname>]) REFERENCES <TableName> [(<ColumnName>, <ColumnName>) ON DELETE CASCADE

- FOREIGN KEY constraint defined with ON DELETE SET NULL
- FOREIGN KEY(<ColumnName>[,<columnname>]) REFERENCES

<TableName> [(<ColumnName>, <ColumnName>) ON DELETE SETNULL

To view theconstraint

Syntax:

Select constraint_name, constraint_type,search_condition from user_constraints where table_name=<tablename>;
Select constraint_name, column_name from user_cons_columns wheretable_name=<tablename>;

To drop the

constraintsSyntax:-

Drop constraint constraint name;

Describe commands

To view the structure of the table created use the **DESCRIBE** command.Thecommand displays the column names and datatypes

Syntax:-

Desc[ribe]<table_name>

e.g desc student

Restrictions for creating a table:

1. Table names and column names must begin with letter.
2. Table names and column names can be 1 to 30characters long.
3. Table names must contain only the characters A-Z,a-z,0-9,underscore_,$ and #
4. Table name should not be same as the name of another database object.
5. Table name must not be an ORACLE reserved word.
6. Column names should not be duplicate within a table definition.

Alteration of TABLE:-Alter table command

Syntax:-

Case1:-

Alter table <table_name>

Add(colume_name 1
datatype size(),
colume_name 2
datatype size(),

colume_name n datatype size());

Case2:-

```
Alter table <table_name>
Modify(colume_name 1
datatype size(),
        colume_name 2 datatype size(),
        .....
        colume_name n datatype size());
```

After you create a table, you may need to change the table structures because you need to havea column definition needs to be changed. Alter table statement can be used for this purpose.
You can add columns to a table using the alter table statement with the ADD clause.

E.g. Suppose you want to add enroll_no in the student table then we write

Alter table student Add(enroll_no number(10));

You can modify existing column in a table by using the alter table statement with modify clause.

E.g. Suppose you want to modify or chang the size of previously defined field name in the student table then we write

Alter table student modify (name char(25));Dropping a column from a table

Syntax :

ALTER TABLE <Tablename> DROP COLUMN <ColumnName> ;

Drop table command Syntax:-

Drop table <table_name>

Drop table command remnoves the definitions of an oracle table.When you drop a table ,the database loses all the data in the table and all the indexes associated with it.

e.g drop table student;

Truncate table commandSyntax:-

Trunc table<table_name>

The truncate table statement is used to remove all rows from a table and to release the storagespace used by the table.

e.g. Trunc table student;

Rename table commandSyntax:

Rename<oldtable_name> to<newtable_name>

Rename statement is used to rename a table,view,sequence,or synonym.

e.g. Rename student to stud;

Database objects:-Index

An index is a schema object that can speed up retrieval of rows by using pointer. An index provides direct & fast access to rows in a table. Index can be created explicitly or automatically.

Automatically :- A unique index is created automatically when you define a primary key or unique key constraint in a table definition.

Manually :- users can create non unique indexes or columns to speed up access time to the rows.

Syntax:

Create index<index_name> On table(column[, column]...);

Eg. Create index emp_ename_idx On emp(ename);

When to create an index

a) The column is used frequently in the WHERE clause or in a join condition.

b) The column contains a wide range of values.

c) The column contains a large number of values. To display created index of a table

eg.

```
Select ic.index_name, ic.column_name, ic.colun_position col_pos, ix.uniqueness from
user_indexes ix, user_ind_columns ic where
ic.index_name=ix.index_nameand ic.table_name="emp";
```

Removing an Index

Syntax:-

Drop index <index_name>; eg. Dropindex emp_name_idx;

Note: 1) we cannot modify indexes.

2) To change an index, we must drop it and the re-create it.

Views

View is a logical representation of subsets of data from one or more tables. A view takes the output of a query and treats it as a table therefore view can be called as stored query or a virtual table.The tables upon which a view is based are called base tables. In Oracle the SQL command to create a view (virtual table) has the form

Create [or replace] view <view-name> [(<column(s)>)] as

<select-statement> [with check option [constraint <name>]];

The optional clause or replace re-creates the view if it already exists. <column(s)> names the columns of the view. If <column(s)> is not specified in the view definition, the columns of the

view get the same names as the attributes listed in the select statement (if possible).

Example: The following view contains the name, job title and the annual salary of employees working in the department 20:

Create view DEPT20 as select ENAME, JOB, SAL 12 ANNUAL SALARY from EMP where DEPTNO = 20;

In the select statement the column alias ANNUAL SALARY is specified for the expressionSAL*12 and this alias is taken by the view. An alternative formulation of the above view

definition is

12 **from EMP where DEPTNO = 20;**

Create view DEPT20 (ENAME, JOB, ANNUAL SALARY) as select ENAME, JOB, SAL

A view can be used in the same way as a table, that is, rows can be retrieved from a view(also respective rows are not physically stored, but derived on basis of the select statement inthe viewdefinition), or rows can even be modified. A view is evaluated again each time it is accessed. In Oracle SQL no insert, update, or delete modifications on views are allowed that use one of the following constructs in the view definition:

- Joins
- Aggregate function such as sum, min, max etc.
- set-valued sub queries (in, any, all) or test for existence (exists)
- group by clause or distinct clause

In combination with the clause with check option any update or insertion of a row into the view is rejected if the new/modified row does not meet the view definition, i.e., these rows would notbe selected based on the select statement. A with check option can be named using the constraintclause.

A view can be deleted using the command delete <view-name>. Todescribe the structure of a view

e.g. **Describe stud;**

To display the contents of view e.g. Select * from stud

Removing a view:

Syntax:- **Drop view <view_name>**

e.g. Drop view stud

Create [or replace] view <view-name> [(<column(s)>)] as

<select-statement> [with check option [constraint <name>]];

The optional clause or replace re-creates the view if it already exists. <column(s)> names the columns of the view. If <column(s)> is not specified in the view definition, the columns of the view get the same names as the attributes listed in the select statement (if possible).

Example: The following view contains the name, job title and the annual salary of employees working in the department 20:

Create view DEPT20 as select ENAME, JOB, SAL 12 ANNUAL SALARY from EMP where DEPTNO = 20;

In the select statement the column alias ANNUAL SALARY is specified for the expression SAL*12 and this alias is taken by the view. An alternative formulation of the above view definition is

12 from EMP where DEPTNO = 20;

Create view DEPT20 (ENAME, JOB, ANNUAL SALARY) as select ENAME, JOB, SAL

A view can be used in the same way as a table, that is, rows can be retrieved from a view(also respective rows are not physically stored, but derived on basis of the select statement in the viewdefinition), or rows can even be modified.

Sequence:

A sequence is a database object, which can generate unique, sequential integer values. It can be used to automatically generate primary key or unique key values. A sequence can be either in anascending or descending order.

Syntax : Create

```
sequence<sequence_name>
    [increment by n]
```

[start with n]
[{maxvalue n | nomaxvalue}] [{minvalue n|nominvalue}]
[{cycle |nocycle}]
[{cache n| nocache}];

Increment by n	Specifies the interval between sequence number where n is an integer. If this clause is omitted, the sequence is increment by 1.
Start with n	Specifies the first sequence number tobe generated. If this clause is omitted , the sequence is start with 1.
Maxvalue n	Specifies the maximum value, the sequence can generate

Nomax value n	Specifies the maximum value of 10e27-1 for an ascending sequence & -1 for descending sequence. This is a default option.
Minvalue n	Specifies the minimum sequence value.
Nominvalue n	Specifies the minimum value of 1 for an ascending & 10e26-1 for descending sequence. This is a default option.
Cycle	Specifies that the sequence continues to generate values from the beginning

After creating a sequence we can access its values with the help of pseudo columnslike **curval** & **nextval**.

Nextval :

nextval returns initial value of the sequence when reference to for the firsttime. Last references to the nextval will increment the sequence using the increment by clause & returns the new value.

Curval :

curval returns the current value of the sequence which is the value returned by the last reference to last value

Modifyning a sequence:

The sequence can be modified when we want to perform the following :

Set or eliminate minvalue or maxvalue

- Change the increment value.

Change the number of cache sequence number.

Syntax :

Alter sequence <sequence_name>

[increment by n]
[start with n]
[{maxvalue n | nomaxvalue}]
[{minvalue n| nominvalue}]
[{cycle | nocycle}]
[{cache n| nocache}];

Synonym:

A synonym is a database object,which is used as an alias(alternative name)for a table,view or sequence.

Tables

In relational database systems (DBS) data are represented using tables (relations). A query issuedagainst the DBS also results in a table. A table has the following structure:

Column 1	Column 2	. . .	Column n →	Attributes
← Tuple (c	Record)			
…….	…….	…….	……..	

A column is made up of a column name and a data type, and it describes an attribute of the tuples. The structure of a table, also called relation schema, thus is defined by its attributes.The type of information to be stored in a table is defined by the data types of the attributes at table creation time.SQL uses the terms table, row, and column for relation, tuple, and attribute,respectively.

A table can have up to 254 columns which may have di^{ff}erent or same data types and sets ovalues (domains), respectively. Possible domains are alphanumeric data (strings), numbers and date formats.

Datatype	Description	Max Size: Oracle 7	Max Size: Oracle8	Max Size: Oracle 9	Max Size: PL/SQL	PL/SQL Subtypes/ Synonyms
VARCHAR2(size)	Variable length character stringhaving maximum length *size* bytes.	2000 bytes minimum is 1	**4000** bytes minimum is 1	**4000** bytes minimum is 1	32767 bytes minimumis 1	STRING VARCHAR
NVARCHAR2 (size)	Variable length national character setstring having maximum length *size* bytes.	N/A	4000 bytes minimum is 1	4000 bytes minimum is 1	32767 bytes minimumis 1	STRING VARCHAR
VARCHAR	Now deprecated - VARCHAR is a synonym for VARCHAR2 but this usage may change in future versions.	-	-	-		
CHAR(size)	Fixed length character data oflength size bytes. This should be used for fixed length data. Such as codes A100, B102...	255 bytes Default and minimum size is 1 byte.	**2000** bytes Default and minimum size is1 byte.	**2000** bytes Default and minimum sizeis 1 byte.	32767 bytes Default and minimum size is 1 byte.	CHARACTER
NCHAR (size)	Fixed length national character set	N/A	2000 bytes Default	2000 bytes Default and minimum size	32767 bytes Default	

	data of length size bytes. Thisshould be used for fixed lengthdata. Such as codes A100, B102...		and minimum size is1 byte.	is 1 byte.	and minimum size is 1 byte.	
NUMBER(p,s)	Number having precision p andscale s.	The precisionp can range from 1 to 38. The scales can range from -84 to 127.	The precision p can range from 1 to 38. The scale s can range from -84 to 127.	The precisio np can range from 1 to 38. The scale scan range from -84 to 127.	Magnitude 1E-130 .. 10E125 maximum precision of 126 binary digits, which is roughly equivalent to 38 decimal digits The scales can range from -84 to 127. For floating point don't specifyp,s REAL has a maximum precision of 63 binary digits, which is roughly equivalent to 18 decimal its	fixed-point numbers: DEC DECIMAL NUMERIC floating-point: DOUBLE PRECISION FLOAT binary_double binary_float integers: INTEGER INT SMALLINT simple_integer(10g) BOOLEAN REAL
PLS_INTEGER	signed integers PLS_INTEGER values require less storage and	PL/SQL only	PL/SQL only	PL/SQL only	magnitude range is - 21474836	

	provide better performance than NUMBER values. So use PLS_INTEGER where you can!				47 .. 21474836 47	
BINARY_INTEGER	signed integers (older slower version of PLS_INTEGER)				magnitude range is - 21474836 47 .. 21474836 47	NATURAL NATURALN POSITIVE POSITIVENSIGNTYPE
LONG	Character data of variable length (A biggerversion the VARCHAR2 datatype)	2 Gigabytes	2 Gigabytes	2 Gigabytes -but now deprecated	32760 bytes Note this is smalller than the maximum width of a LONG column	
DATE	Valid date range	from January 1, 4712 BC to December 31, 4712 AD.	from January1, 4712 BC to December 31, **9999** AD.	from January 1, 4712 BC to December 31, **9999** AD.	from January 1, 4712 BC to December 31, **9999** AD. (in Oracle7 = 4712 AD)	
TIMESTAMP (fractional_sec onds_precision)	the number of digits in the fractional part ofthe SECOND datetime field.	-	-	Accepted values of fractional_sec onds_precision are 0 to 9. (default = 6)		

YEAR (year_precision) TO MONTH	and months, where year_precision is the number of digits in the YEAR datetime field.			values are 0 to 9. (default =2)		
INTERVALDAY (day_precision)TO SECOND (fractional_seconds_precision)	Time in days, hours, minutes, and seconds. *day_precision* is the maximum number of digits in 'DAY' *fractional_seconds_precision* is the max numberof fractional digits in the SECOND field.	-	-	*day_precision* may be 0 to 9. (default = 2) *fractional_seconds_precision* may be 0 to 9. (default =6)		
RAW(size)	Raw binary data of length size bytes. You must specify size fora RAW value.	Maximumsize is 255 bytes.	Maximum size is**2000** bytes	Maximum size is **2000** bytes	32767 bytes	
LONG RAW	Raw binary data of variable length. (not intrepreted by PL/SQL)	2 Gigabytes .	2 Gigabytes.	2 Gigabytes -but now deprecated	32760 bytes Note this is smalller than the maximum width of a LONG RAW column	
ROWID	Hexadecimal string representing the unique address of a row in its table.	8 bytes	10 bytes	10 bytes	Hexadeci mal stringrepresenting the unique address ofa row in its table. by the ROWID pseudocolumn.)	

UROWID	Hex string representing thelogical address of a row of an index-organizedtable	N/A	The maximum size and default is 4000 bytes	The maximum size and default is 4000 bytes	universal rowid - Hex string representing the logical address of a row of an index-organized table, either physical, logical, or foreign (non-Oracle)	See CHARTOROWID and the package: DBMS_ROWID
MLSLABEL	Binary format of an operating system label.This datatype is used with Trusted Oracle7.					
CLOB	Character Large Object	4Gigabytes	4Gigabytes	4Gigabytes	4Gigabytes	
NCLOB	National Character LargeObject		4Gigabytes	4Gigabytes	4Gigabytes	
BLOB	Binary Large Object		4Gigabytes	4Gigabytes	4Gigabytes	
BFILE	pointer to binaryfile on disk		4Gigabytes	4Gigabytes	The size of a BFILE is system dependent but cannot exceed four gigabytes (2**32 - 1 bytes).	

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

FAQ : Consider relational schema Student(Roll_no, Name, Deptno, Marks,Email_id)
Develop SQL DDL statements.

1. Create table Student;
2. Insert values in student table.
3. Add a new attribute date of birth in student record using alter statement.
4. Drop date of birth attribute from student table.
5. Update a student marks where roll no is 7;
6. Delete a record of student whose roll no is 4;
7. Create view for student table;
8. Create index on Roll no in student table.
9. Create sequence on student table.
10 Create synonym on student table

Assignment No.	**2(b)**
Title	**b. Write at least 10 SQL queries for suitable database application using SQL DML statements: Insert, Select, Update, Delete with operators, functions, and set operator.**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 2 (b)

Title :- Write 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 pdating data in a database. Performing read-only queries of data is sometimes also considered a component ofDML.

Data manipulation language comprises the SQL data change statements, [2] which modify storeddata but not the schema or database objects.

Data manipulation languages have their functional capability organized by the initial word in astatement, 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 ...

The purely read-only SELECT query statement is classed with the 'SQL-data' statements and so is considered by the standard to be outside of DML. The SELECT ... INTO form is considered tobe DML because it manipulates (i.e. modifies) data. In common practice though, this distinction is not made and SELECT is widely considered to be part of DML. Most SQL database implementations extend their SQL capabilities by providing imperative, i.e.procedural languages.

Inserting Data into Table:

To insert data into table, you would need to use SQL **INSERT INTO** command.

You can insertdata into table by using > prompt or by using any script like PHP.

Syntax:

Here is generic SQL syntax of INSERT INTO command to insert data into table:

```
INSERT INTO table_name ( field1, field2,...fieldN )
                VALUES ( value1, value2,...valueN );
```

To insert string data types, it is required to keep all the values into double or single quote, forexample:-**"value"**.

Inserting Data from Command Prompt:

This will use SQL INSERT INTO command to insert data into table tutorials_tbl.Example:

Following example will create 3 records into **tutorials_tbl** table:

```
INSERT INTO tutorials_tbl (tutorial_title, tutorial_author, submission_date)
  VALUES ("Learn PHP", "John Poul", NOW());

INSERT INTO tutorials_tbl (tutorial_title, tutorial_author, submission_date)
  VALUES ("Learn  ", "Abdul S", NOW());
```

```
INSERT INTO tutorials_tbl (tutorial_title, tutorial_author, submission_date)
  VALUES ("JAVA Tutorial", "Sanjay", '2007-05-06');
```

Here, **NOW()** is a function, which returns current date and time.

□ **Fetching Data from Table:**

The SQL **SELECT** command is used to fetch data from database. You can use this command at > prompt as well as in any
script like PHP.Syntax:
Here is generic SQL syntax of SELECT command to fetch data from table:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE Clause]
[OFFSET M ][LIMIT N]
```

- You can use one or more tables separated by comma to include various conditions usinga WHERE clause, but WHERE clause is an optional part of SELECT command.
- You can fetch one or more fields in a single SELECT command.
- You can specify star (*) in place of fields. In this case, SELECT will return all the fields.
- You can specify any condition using WHERE clause.
- You can specify an offset using **OFFSET** from where SELECT will start returningrecords. By default offset is zero.
- You can limit the number of returns using **LIMIT** attribute.

Fetching Data from Command Prompt:

This will use SQL SELECT command to fetch data from table
tutorials_ tbl Example:
Following example will return all the records from **tutorials_tbl** table:

```
> SELECT * from tutorials_tbl
+-------------+---------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+---------------+-----------------+-----------------+
|           1 | Learn PHP     | John Poul       | 2007-05-21      |
|           2 | Learn         | Abdul S         | 2007-05-21      |
|           3 | JAVA Tutorial | Sanjay          | 2007-05-21      |
+-------------+---------------+-----------------+-----------------+
```

The SQL **SELECT** statement returns a result set of records from one or more tables.A **SELECT** statement retrieves zero or more rows from one or more database tables ordatabase views. In most applications, SELECT is the most commonly used Data Manipulation Language (DML) command. As SQL is a declarative programming language, SELECT queries specify a result set, but do not specify how to calculate it. The database translates the query into a "query plan" which may vary between executions, database versions and database software. This functionality is called the "query optimizer" as it is responsible for finding the best possible execution plan for the query, within applicable constraints.
The SELECT statement has many optional clauses:

- WHERE specifies which rows to retrieve.

GROUP BY groups rows sharing a property so that an aggregate function can be appliedto each group.

- HAVING selects among the groups defined by the GROUP BY clause.
- ORDER BY specifies an order in which to return the rows.
- AS provides an alias which can be used to temporarily rename tables or columns.

WHERE Clause

We have seen SQL **SELECT** command to fetch data from table. We can use a conditional clause called **WHERE** clause to filter out results. Using WHERE clause, we can specify a selection criteria to select required records from a table.

Syntax:

Here is generic SQL syntax of SELECT command with WHERE clause to fetch data fromtable:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
[WHERE condition1 [AND [OR]] condition2.....
```

- You can use one or more tables separated by comma to include various conditions usinga WHERE clause, but WHERE clause is an optional part of SELECT command.
- You can specify any condition using WHERE clause.
- You can specify more than one conditions using **AND** or **OR** operators.

- A WHERE clause can be used along with DELETE or UPDATE SQL command also tospecify a condition.

The **WHERE** clause works like an if condition in any programming language. This clause is used to compare given value with the field value available in table. If given value from outside is equal to the available field value in table, then it returns that row.
Here is the list of operators, which can be used with **WHERE** clause.Assume field A holds 10 and field B holds 20, then:

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes thencondition becomes true.	(A = B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of rightoperand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to thevalue of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

The WHERE clause is very useful when you want to fetch selected rows from a

table, especiallywhen you use **Join**.

It is a common practice to search records using **Primary Key** to make search fast.

If given condition does not match any record in the table, then query would not return any row.

Fetching Data from Command Prompt:

This will use SQL SELECT command with WHERE clause to fetch selected data from table tutorials_tbl.

Example:

Following example will return all the records from **tutorials_tbl** table for which author nameis **Sanjay**:

```
> SELECT * from tutorials_tbl WHERE tutorial_author='Sanjay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
```

Unless performing a **LIKE** comparison on a string, the comparison is not case sensitive. You canmake your search case sensitive using **BINARY** keyword as follows:

```
SELECT * from tutorials_tbl  WHERE BINARY tutorial_author='sanjay';
```

LIKE Clause

We have seen SQL **SELECT** command to fetch data from table. We can also use a conditional clause called **WHERE** clause to select required records.

A WHERE clause with equals sign (=) works fine where we want to do an exact match.

Like if "tutorial_author = 'Sanjay'". But there may be a requirement where we want to filter out all the results where tutorial_author name should contain "jay". This can be handled usingSQL **LIKE** clause along with WHERE clause.

If SQL LIKE clause is used along with % characters, then it will work like a meta character (*)in UNIX while listing out all the files or directories at command prompt.

Without a % character, LIKE clause is very similar to equals sign along with WHERE clause.

Syntax:

```
SELECT field1, field2,...fieldN table_name1, table_name2...
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

Here is generic SQL syntax of SELECT command along with LIKE clause to fetch data fromtable:

- You can specify any condition using WHERE clause.
- You can use LIKE clause along with WHERE clause.

- You can use LIKE clause in place of equals sign.
- When LIKE is used along with % sign then it will work like a meta character search.
- You can specify more than one conditions using **AND** or **OR** operators.

A WHERE...LIKE clause can be used along with DELETE or UPDATE SQL commandalso to specify a condition.

Using LIKE clause at Command Prompt:
This will use SQL SELECT command with WHERE...LIKE clause to fetch selected data fromtable tutorials_tbl.

Example:
Following example will return all the records from **tutorials_tbl** table for which author nameends with **jay**:

```
SELECT * from tutorials_tbl  WHERE tutorial_author LIKE '%jay';
+-------------+----------------+-----------------+-----------------+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+-------------+----------------+-----------------+-----------------+
|           3 | JAVA Tutorial  | Sanjay          | 2007-05-21      |
+-------------+----------------+-----------------+-----------------+
```

GROUP BY Clause

You can use **GROUP BY** to group values from a column, and, if you wish, perform calculations on that column. You can use COUNT, SUM, AVG, etc., functions on the grouped column.
To understand **GROUP BY** clause, consider an **employee_tbl** table, which is having thefollowing records:

```
> SELECT * FROM employee_tbl;
+------+------+------------+--------------------+
| id   | name | work_date  | daily_typing_pages |
+------+------+------------+--------------------+
|    1 | John | 2007-01-24 |                250 |
|    2 | Ram  | 2007-05-27 |                220 |
|    3 | Jack | 2007-05-06 |                170 |
|    3 | Jack | 2007-04-06 |                100 |
|    4 | Jill | 2007-04-06 |                220 |
|    5 | Zara | 2007-06-06 |                300 |
|    5 | Zara | 2007-02-06 |                350 |
+------+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table we want to count number of days each employee didwork.
If we will write a SQL query as follows, then we will get the following result:

```
SELECT COUNT(*) FROM employee_tbl;
+----------+
| COUNT(*) |
+----------+
| 7        |
```

But this is not serving our purpose, we want to display total number of pages typed by each person separately. This is done by using aggregate functions in conjunction with a **GROUPBY** clause as follows:
We will see more functionality related to GROUP BY in other functions like SUM, AVG, etc.

COUNT Function

COUNT Function is the simplest function and very useful in counting the number of records,which are expected to be returned by a SELECT statement.
To understand **COUNT** function, consider an **employee_tbl** table, which is having the followingrecords:

```
> SELECT * FROM employee_tbl;
+----+------+------------+--------------------+
| id | name | work_date  | daily_typing_pages |
+----+------+------------+--------------------+
|  1 | John | 2007-01-24 |                250 |
|  2 | Ram  | 2007-05-27 |                220 |
|  3 | Jack | 2007-05-06 |                170 |
|  3 | Jack | 2007-04-06 |                100 |
|  4 | Jill | 2007-04-06 |                220 |
|  5 | Zara | 2007-06-06 |                300 |
|  5 | Zara | 2007-02-06 |                350 |
+----+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to count total number of rows in this table, thenyou can do it as follows:

```
>SELECT COUNT(*) FROM employee_tbl ;
+----------+
| COUNT(*) |
+----------+
|        7 |
+----------+
1 row in set (0.01 sec)
```

Similarly, if you want to count the number of records for Zara, then it can be done as follows:

```
SELECT COUNT(*) FROM employee_tbl  WHERE name="Zara";
+----------+
```

NOTE: All the SQL queries are case insensitive so it does not make any difference if you giveZARA or Zara in WHERE condition.

MAX Function

MAX function is used to find out the record with maximum value among a record set. To understand **MAX** function, consider an **employee_tbl** table, which is having the followingrecords:

```
 > SELECT * FROM employee_tbl;
+-----+-----+------------+--------------------+
| id  | name | work_date  | daily_typing_pages |
+-----+-----+------------+--------------------+
|   1 | John | 2007-01-24 |                250 |
|   2 | Ram  | 2007-05-27 |                220 |
|   3 | Jack | 2007-05-06 |                170 |
|   3 | Jack | 2007-04-06 |                100 |
|   4 | Jill | 2007-04-06 |                220 |
|   5 | Zara | 2007-06-06 |                300 |
|   5 | Zara | 2007-02-06 |                350 |
+-----+-----+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to fetch maximum value ofdaily_typing_pages, then you can do so simply using the following command:

```
SELECT MAX(daily_typing_pages) FROM employee_tbl;
+--------------------------+
| MAX(daily_typing_pages) |
+--------------------------+
|                      350 |
+--------------------------+
1 row in set (0.00 sec)
```

You can find all the records with maximum value for each name using **GROUP BY** clause asfollows:

```
SELECT id, name, MAX(daily_typing_pages)  FROM employee_tbl GROUP BY name;
+----+------+-------------------------+
| id | name | MAX(daily_typing_pages) |
+----+------+-------------------------+
|  3 | Jack |                     170 |
|  4 | Jill |                     220 |
|  1 | John |                     250 |
|  2 | Ram  |                     220 |
|  5 | Zara |                     350 |
+----+------+-------------------------+
5 rows in set (0.00 sec)
```

You can use **MIN** Function along with **MAX** function to find out minimum value as well. Tryout the following example:

```
SELECT MIN(daily_typing_pages) least, MAX(daily_typing_pages) max FROM employee_tbl;
+-------+-------+
| least | max   |
+-------+-------+
|   100 |   350 |
+-------+-------+
1 row in set (0.01 sec)
```

MIN Function

MIN function is used to find out the record with minimum value among a record set. To understand **MIN** function, consider an **employee_tbl** table, which is having the followingrecords:

```
SELECT * FROM employee_tbl;
+----+------+------------+--------------------+
| id | name | work_date  | daily_typing_pages |
+----+------+------------+--------------------+
|  1 | John | 2007-01-24 |                250 |
|  2 | Ram  | 2007-05-27 |                220 |
|  3 | Jack | 2007-05-06 |                170 |
|  3 | Jack | 2007-04-06 |                100 |
|  4 | Jill | 2007-04-06 |                220 |
|  5 | Zara | 2007-06-06 |                300 |
|  5 | Zara | 2007-02-06 |                350 |
+----+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to fetch minimum value of daily_typing_pages,then you can do so simply using the following command:

```
SELECT MIN(daily_typing_pages) FROM employee_tbl;
+--------------------------+
| MIN(daily_typing_pages) |
+--------------------------+
|                      100 |
+--------------------------+
1 row in set (0.00 sec)
```

You can find all the records with minimum value for each name using **GROUP BY** clause asfollows:

```
SELECT id, name, MIN(daily_typing_pages) FROM employee_tbl GROUP BY name;
+------+------+--------------------------+
| id   | name | MIN(daily_typing_pages) |
+------+------+--------------------------+
|    3 | Jack |                      100 |
|    4 | Jill |                      220 |
|    1 | John |                      250 |
|    2 | Ram  |                      220 |
|    5 | Zara |                      300 |
+------+------+--------------------------+
5 rows in set (0.00 sec)
```

You can use **MIN** Function along with **MAX** function to find out minimum value as well. Tryout the following example:

```
SELECT MIN(daily_typing_pages) least, MAX(daily_typing_pages) max FROM employee_tbl;
+-------+-------+
| least | max   |
+-------+-------+
|   100 |   350 |
+-------+-------+
1 row in set (0.01 sec)
```

AVG Function

AVG function is used to find out the average of a field in various records.

To understand **AVG** function, consider an **employee_tbl** table, which is

```
SELECT * FROM employee_tbl;
+----+------+------------+--------------------+
| id | name | work_date  | daily_typing_pages |
+----+------+------------+--------------------+
|  1 | John | 2007-01-24 |                250 |
|  2 | Ram  | 2007-05-27 |                220 |
|  3 | Jack | 2007-05-06 |                170 |
|  3 | Jack | 2007-04-06 |                100 |
|  4 | Jill | 2007-04-06 |                220 |
|  5 | Zara | 2007-06-06 |                300 |
|  5 | Zara | 2007-02-06 |                350 |
+----+------+------------+--------------------+
7 rows in set (0.00 sec)
```

having following records:Now, suppose based on the above table you want to calculate average of all thedialy_typing_pages, then you can do so by using the following command:

You can take average of various records set using **GROUP BY** clause. Following example will take average all the records related to a single person and you will have average typed pages byevery person.

```
SELECT name, AVG(daily_typing_pages) FROM employee_tbl GROUP BY name;| name |
AVG(daily_typing_pages) |
| Jack |              135.0000 |
| Jill |              220.0000 |
| John |              250.0000 |
| Ram  |              220.0000 |
| Zara |              325.0000 |
5 rows in set (0.20 sec)
```

SUM Function

SUM function is used to find out the sum of a field in various records.To understand **SUM** function, consider an **employee_tbl** table, which is having the followingrecords:

```
SELECT * FROM employee_tbl;
+----+------+------------+--------------------+
| id | name | work_date  | daily_typing_pages |
+----+------+------------+--------------------+
|  1 | John | 2007-01-24 |                250 |
|  2 | Ram  | 2007-05-27 |                220 |
|  3 | Jack | 2007-05-06 |                170 |
|  3 | Jack | 2007-04-06 |                100 |
|  4 | Jill | 2007-04-06 |                220 |
|  5 | Zara | 2007-06-06 |                300 |
|  5 | Zara | 2007-02-06 |                350 |
+----+------+------------+--------------------+
7 rows in set (0.00 sec)
```

Now, suppose based on the above table you want to calculate total of all the dialy_typing_pages,then you can do so by using the following command:

```
SELECT SUM(daily_typing_pages)  FROM employee_tbl;
+-------------------------+
| SUM(daily_typing_pages) |
```

You can take sum of various records set using **GROUP BY** clause. Following example will sum up all the records related to a single person and you will have total typed pages by every person.

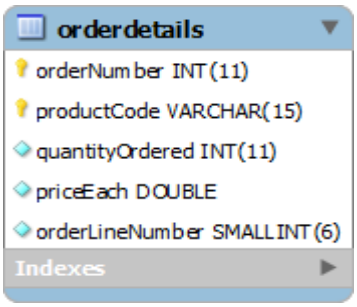
```
SELECT name, SUM(daily_typing_pages) FROM employee_tbl GROUP BY name;
+------+-------------------------+
| name | SUM(daily_typing_pages) |
+------+-------------------------+
| Jack |                     270 |
| Jill |                     220 |
| John |                     250 |
| Ram  |                     220 |
| Zara |                     650 |
+------+-------------------------+
5 rows in set (0.17 sec)
```

HAVING clause

The HAVING clause is used in the SELECT statement to specify filter conditions for group of rows or aggregates. The HAVING clause is often used with the GROUP BY clause. When using with the GROUP BY clause, you can apply a filter condition to the columns that appear inthe GROUP BY clause. If the GROUP BY clause is omitted, the HAVING clause behaves likethe WHERE clause. Notice that the HAVING clause applies the condition to each group of rows, while the WHERE clause applies the condition to each individual row.

Examples of using HAVING clause

Let‟s take a look at an example of using HAVING clauseWe will use the orderdetails table in the sample database for the sake of demonstration.



We can use the GROUP BY clause to get order number, the number of items sold per order andtotal sales for each:

```
SELECT ordernumber,
      SUM(quantityOrdered) AS itemsCount,
      SUM(priceeach) AS total
FROM orderdetails
GROUP BY ordernumber
```

ordernumber	itemsCount	total
10100	151	301.84000000000003
10101	142	352
10102	80	138.68
10103	541	1520.3699999999997
10104	443	1251.8899999999999
10105	545	1479.71

Now, we can find which order has total sales greater than $1000. We use the HAVING clauseon the aggregate as follows

```
SELECT ordernumber,
      SUM(quantityOrdered) AS itemsCount,
      SUM(priceeach) AS total
       FROM orderdetails
      GROUP BY ordernumber
      HAVING total > 1000
```

ordernumber	itemsCount	total
10103	541	1520.3699999999997
10104	443	1251.8899999999999
10105	545	1479.71
10106	675	1427.2800000000002
10108	561	1432.86
10110	570	1338.4699999999998

We can construct a complex condition in the HAVING clause using logical operators such as OR and AND. Suppose we want to find which order has total sales greater than $1000 and contains more than 600 items, we can use the following query:

```
SELECT ordernumber,
      sum(quantityOrdered) AS itemsCount,
      sum(priceeach) AS total
       FROM orderdetails
       GROUP BY ordernumber
       HAVING total > 1000 AND itemsCount > 600
```

ordernumber	itemsCount	total
10106	675	1427.2800000000002
10126	617	1623.71
10135	607	1494.86
10165	670	1794.9399999999996
10168	642	1472.5
10204	619	1619.73
10207	615	1560.08

The HAVING clause is only useful when we use it with the GROUP BY clause to generate the output of the high-level reports. For example, we can use the HAVING clause to answer some kinds of queries like give me all the orders in this month, this quarter and this year that have totalsales greater than 10K.

UPDATE Query

There may be a requirement where existing data in a table needs to be modified. You can do soby using SQL **UPDATE** command. This will modify any field value of any table.
Syntax:
Here is generic SQL syntax of UPDATE command to modify data into table:

```
UPDATE table_name SET field1=new-value1, field2=new-value2
[WHERE Clause]
```

- You can update one or more field altogether.
- You can specify any condition using WHERE clause.
- You can update values in a single table at a time.

The WHERE clause is very useful when you want to update selected rows in a table.Updating Data from Command Prompt:
This will use SQL UPDATE command with WHERE clause to update selected data into table tutorials_tbl.

Example:

Following example will update **tutorial_title** field for a record having tutorial_id as 3.

```
UPDATE tutorials_tbl
  SET tutorial_title='Learning JAVA'
    WHERE tutorial_id=3;
```

DELETE Query

If you want to delete a record from any table, then you can use SQL command **DELETEFROM**. You can use this command at > prompt as well as in any script like PHP.

Syntax:
Here is generic SQL syntax of DELETE command to delete data from a table:

```
DELETE FROM table_name [WHERE Clause]
```

- If WHERE clause is not specified, then all the records will be deleted from the giventable.
- You can specify any condition using WHERE clause.
- You can delete records in a single table at a time.

The WHERE clause is very useful when you want to delete selected rows in a table.Deleting Data from Command Prompt:
This will use SQL DELETE command with WHERE clause to delete selected data into tabletutorials_tbl.
Example:

```
DELETE FROM tutorials_tbl WHERE tutorial_id=3;
```

Following example will delete a record into tutorial_tbl whose tutorial_id is 3.

Create table **location**(location_id numeric(3) primary key,regional_group varchar(15));
Create table **department**(Department_ID numeric(2) primary key,name varchar(20),location_idint, foreign key(location_id) references location(location_id));
Create table **job**(job_ID numeric(3) primary key,function varchar(20));
Create table **employee**(employee_ID numeric(4) primary key,last_name varchar(20),first_namevarchar(20),middle_name varchar(20),job_id numeric(3),manager_id varchar(20), hired_date date, salary numeric(6), comm numeric(4), department_id numeric(2) not null,FOREIGN KEY(job_id) REFERENCES job(job_id),FOREIGN KEY (department_id) REFERENCES department(department_id));

1. List the details about "SMITH"

*Select * from employee where last_name='SMITH';*

2. List out the employees who are working in department 20

*Select * from employee where department_id=20*

3. List out the employees who are earning salary between 3000 and 4500

*Select * from employee where salary between 3000 and 4500*

4. List out the employees who are working in department 10 or 20

   *Select * from employee where department_id in (10,20)*

5. Find out the employees who are not working in department 10 or 30

   Select last_name, salary, comm, department_id from employee wheredepartment_id not in (10,30)

6. List out the employees whose name starts with "S"

   *Select * from employee where last_name like 'S%';*

7. List out the employees whose name start with "S" and end with "H"

   *Select * from employee where last_name Like 'S%H';*

8. List out the employees whose name length is 5 and start with "S"

   *Select * from employee where last_name like 'S__';*

9. List out the employees who are working in department 10 and draw the salaries morethan 3500

   *Select * from employee where department_id=10 and salary>3500*

10. List out the employees who are not receiving commission.

    *Select * from employee where commission is Null*

11. List out the employee id, last name in ascending order based on the employee id.

    Select employee_id, last_name from employee order by employee_id

12. List out the employee id, name in descending order based on salary column

    Select employee_id, last_name, salary from employee order by salary desc

List out the employee details according to their last_name in ascending order andsalaries in descending order

Conclusion: Thus we have studied to use & implement various DML queries.

FAQ :

1. Explain DML.
2. Explain INSERT command with syntax.
3. Explain DELETE command with syntax.
4. Explain UPDATE command with syntax.
5. Explain SELECT command with syntax.
6. Enlist different comparisons operator. Explain with example.
7. Enlist different Logical operator. Explain with example.
8. Explain Order by clause.
9. Enlist different Aggregation function. Explain with example.

.

Assignment No.	**3**
Title	**SQL Queries – all types of Join, Sub-Query and View:** Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 3

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

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

THEORY: SQL – Join The ability of relational„join"operator

is animportant feature ofrelationalsystems. A joinmakes it possible to select data from more than table by means of a single statement. This joiningof tables may be done in a many ways.

Types of JOIN

1) Inner
2) Outer(left, right,full)
3) Cross

1) Inner join :

- Also known as equi join.
- Statements generally compares two columns from two columns with the equivalence operator =.
- This type of join can be used in situations where selecting only those rows that have values in common in the columns specified in the ON clause, is required.

• **Syntax :**

(ANSI style)

SELECT<columnname1>, <columnname2> <columnNameN> FROM

<tablename1>INNER JOIN<tablename2>ON <tablename1>.<columnname> = <tablename2>.<columnname> WHERE <condition> ORDER BY <columnname1>;(theta style)

SELECT<columnname1>, <columnname2> <columnNameN> FROM <tablename1>,

<tablename2> WHERE <tablename1>.<columnname> =

umnname>AND <condition> ORDER BY <columnname1>;

• List the employee details along with branch names to which they belong.

Emp(empno,fname,lname,dept,desig,branchno)Branch(bname,branchno)

Select e.empno,e.fname,e.lname,e.dept, b.bname, e.desig from emp e inner join branch b onb.branchno=e.branchno;

Select e.empno, e.fname, e.lname, e.dept, b.bname, e.desig from emp e, branch b on where b.branchno=e.branchno;

Eg. List the customers along with the account details associated with them.
Customer(custno,fname,lname)
Acc_cust_dtls(fdno,custno)

Acc_mstr(accno,branchno,curbal)Branch_mstr(name,branchno)

- Select c.custno, c.fname, c.lname, a.accno,a.curbal,b.branchno,b.name from customer c innerjoin acc_cust_dtls k on c.custno=k.custno inner join acc_mstr a on k.fdno=a.accno inner join branch b on b.branchno=a.branchno where c.custno like „C%" order by c.custno;
- Select c.custno, c.fname, c.lname, a.accno,a.curbal,b.branchno,b.name from customer c, acc_cust_dtls k, acc_mstr a, branch b where c.custno=k.custno and k.fdno=a.accno and b.branchno=a.branchno and c.custno like „C%" order by c.custno;

Outer Join

Outer joins are similar to inner joins, but give a little bit more flexibility when selecting data from related tables. This type of joins can be used in situations where it is desired, to select all rows from thetable on left(or right, or both) regardless of whether the other table has values in common & (usually)enter NULL where data is missing. Tables

Emp_mstr(empno,fname,lna
me,det)
Cntc_dtls(codeno,cntc_type,c
ntc_ data)

Left Outer Join

List the employee details along with the contact details(if any) using left outer join.

Select e.empno, e.fname, e.lname, e.dept, c.cntc_type, c.cntc_data from emp_mstr e left join cntc_dtls c on e.empno=c.codeno;

- Select e.empno, e.fname, e.lname, e.dept, c.cntc_type, c.cntc_data from emp_mstr e cntc_dtls cwhere e.empno=c.codeno(+);

  All the employee details have to be listed even though their corresponding contact information is notpresent. This indicates all the rows from the first table will be displayed even though there exists no matching rows in the second table.

Right outer join

List the employee details with contact details(if any using right outer join.

- Tables

Emp_mstr(empno,fname,lna
me,dept)
Cntc_dtls(codeno,cntc_type,
cntc_data)

- Select e.empno, e.fname, e.lname, e.dept, c.cntc_type, c.cntc_data from emp_mstr e right joincntc_dtls c on e.empno=c.codeno;
- Select e.empno, e.fname, e.lname, e.dept, c.cntc_type, c.cntc_data from emp_mstr e cntc_dtls cwhere e.empno(+)=c.codeno;

Since the RIGHT JOIN returns all the rows from the second table even if there are no matches in thefirst table.

Cross join

A cross join returns what known as a Cartesian Product. This means that the join combines every row from the left table with every row in the right table. As can be imagined, sometimes

join can be used in situation where it is desired, to select all possible combinations of rows & columns from both tables. The kind of join is usually not preferred as it may run for a very longtime & produce a huge result set that may not be useful.

- Create a report using cross join that will display the maturity amounts for predefined deposits,based on min & max period fixed/ time deposit.
- Tables Tem_fd_amt(fd_amt) Fd_mstr(minprd,maxprd,intre)
- Select fd_amt, s.minprd, s.maxprd, s.intrate,round (t.fd_amt+(s.intrate/100) * (s.minprd/365)

  )) “amount_min_period”,round(t.fd_amt+(s.intrate/100)*(s.maxprd/365))) “amount_max_period” from fd_mstr s cross join tem_fd_amt t;
- Select t.fd_amt, s.minprd, s.maxprd, s.intrate, round(t.fd_amt+(s.intrate/100) * (s.minprd/365)))“amount_min_period”,
  round(t.fd_amt+(s.intrate/100)*(s.maxprd/365))) “amount_max_period”

  from fd_mstr s, tem_fd_amt t;

Self join

- In some situation, it is necessary to join to itself, as though joining 2 separate tables.
- This is referred to as self join

  Example

- Emp_mgr(empno,fname, lname,mgrno)

- Select e.empno,e.fname,e.lname, m.fname "manager" from emp_mgr e, emp_mgr m where e.mgrno=m.empno;

Views

Queries are the principle means of extracting information from data streams and relations. A view represents an alternative selection on a stream or relation that you can use to create subqueries.

A view is only accessible by the queries that reside in the same processor and cannot be exposed beyond that boundary.

Syntax :

Purpose

Use view statement to create a view over a base stream or relation that you reference by identifier in subsequent Oracle CQL statements.

Syntax

You express the a view in a <view></view> element as Example 19-44 shows.

The view element has two attributes:

id: Specify the identifier as the view element id attribute.

The id value must conform with the specification given by identifier::=.

schema: Optionally, specify the schema of the view as a space delimited list of attribute names.
Oracle CEP server infers the types.

Example : View in a <view></view> Element

```
<view id="v2" schema="cusip bid ask"><![CDATA[
   IStream(select * from S1[range 10 slide 10])
]]></view>
```

The body of the view has the same syntax as a query.

Examples

```
<view id="lastEvents" schema="cusip bid srcId bidQty ask askQty seq"><![CDATA[
   select cusip, bid, srcId, bidQty, ask, askQty, seq
   from filteredStream[partition by srcId, cusip rows 1]
]]></view>
<view id="bidask" schema="cusip bid ask"><![CDATA[
   select cusip, max(bid), min(ask)
   from lastEvents
   group by cusip
]]></view>
<view id="bid" schema="cusip bid seq"><![CDATA[
   select ba.cusip as cusip, ba.bid as bid, e.seq
   from bidask as ba, lastEvents as e
   WHERE e.cusip = ba.cusip AND e.bid = ba.bid
]]></view>
<view id="bid1" schema="cusip maxseq"><![CDATA[
   select b.cusip, max(seq) as maxseq
   from bid as b
   group by b.cusip
]]></view>
<view id="BIDMAX" schema="cusip seq srcId bid bidQty"><![CDATA[
   select e.cusip, e.seq, e.srcId, e.bid, e.bidQty
   from bid1 as b, lastEvents as e
   where (e.seq = b.maxseq)
]]></view>
<view id="ask" schema="cusip ask seq"><![CDATA[
   select ba.cusip as cusip, ba.ask as ask, e.seq
   from bidask as ba, lastEvents as e
   WHERE e.cusip = ba.cusip AND e.ask = ba.ask
]]></view>
<view id="ask1" schema="cusip maxseq"><![CDATA[
   select a.cusip, max(seq) as maxseq
   from ask as a
   group by a.cusip
]]></view>
<view id="ASKMIN" schema="cusip seq srcId ask askQty"><![CDATA[
   select e.cusip, e.seq, e.srcId, e.ask, e.askQty
   from ask1 as a, lastEvents as e
   where (e.seq = a.maxseq)

]]></view>
<view id="MAXBIDMINASK" schema="cusip bidseq bidSrcId bid askseq askSrcId ask bidQty
askQty"><![CDATA[
   select bid.cusip, bid.seq, bid.srcId as bidSrcId, bid.bid, ask.seq, ask.srcId as askSrcId, ask.ask,
```

```
bid.bidQty, ask.askQty
   from BIDMAX as bid, ASKMIN as ask
   where bid.cusip = ask.cusip
]]></view>
<query id="BBAQuery"><![CDATA[
   ISTREAM(select bba.cusip, bba.bidseq, bba.bidSrcId, bba.bid, bba.askseq, bba.askSrcId, bba.ask,
bba.bidQty, bba.askQty, "BBAStrategy" as intermediateStrategy, p.seq as correlationId, 1 as priority
   from MAXBIDMINASK as bba, filteredStream[rows 1] as p where bba.cusip = p.cusip)
```

Using this technique, you can achieve the same results as in the subquery case. However, using views you can better control the complexity of queries and reuse views by name in other queries.

Three tire Architecture:

A three-tier architecture is a client-server architecture in which the functional process logic, dataaccess, computer data storage and user interface are developed and maintained as independent moduleson separate platforms. Three-tier architecture is a software design pattern and a well-established software architecture.

In a Three-tier architecture, the client machine acts as merely a front end and does not containany direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a

database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applicationsthat run on the World Wide We

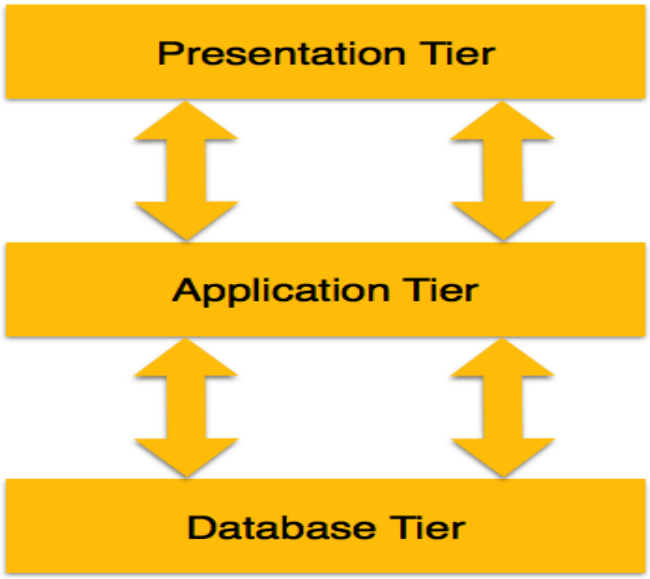


Fig. 1 Three Tire Architecture

Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC and uses a standard graphical user interface with different modules running on the application server. The relational database management system on thedatabase server contains the computer data storage logic. The middle tiers are usually multitiered.

The three tiers in a three-tier architecture are:

1. Presentation Tier: Occupies the top level and displays information related to services available on a website. This tier communicates with other tiers by sending results to the browser and othertiers in the network.
2. Application Tier: Also called the middle tier, logic tier, business logic or logic tier, this tier ispulled from the presentation tier. It controls application functionality by performing detailedprocessing.
3. Data Tier: Houses database servers where information is stored and retrieved. Data in this tier iskept independent of application servers or business logic.

Employee (Eno, Ename, Deptno, Salary) Eno=pk,
Deptno=fkDepartment (Deptno, Dname)
Deptno=pk

Implement all join operation –cross join, natural join ,equi join, left outer ,right outer join etc &
Write

SQL Queries for following questions

i) List of employee names of 'Computer' department.

ii) Find the Employee who"sSalary above 50000 of each department.

iii) Find department name of employee name 'Amit'.

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

FAQ:

1. Explain Join Function.
2. Enlist the different types of join operations.
3. Explain CROSS Join explain with example.
4. Explain Natural join explain with example.
5. Explain Inner join explain with example.
6. Explain Outer join explain with example.
7. What is the use of nested Query. Explain with Example.

Assignment No.	**4**
Title	**Unnamed PL/SQL code block: Use of Control structure andException handling is mandatory.**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 4

Title:- Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory.Write a PL/SQL block of code for the following requirements:-

Schema :

1. Borrower(Rollin, Name, DateofIssue, NameofBook, Status)
2. Fine(Roll_no,Date,Amt)

- Accept roll_no & name of book from user.
- Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.
- If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
- After submitting the book, status will change from I to R.
- If condition of fine is true, then details will be stored into fine table.

Frame the problem statement for writing PL/SQL block inline with above statement.Objective:- Learn the concept of PL/SQL

Theory:

Introduction :-PL/SQL

The development of database applications typically requires language constructs similar to those thatcan be found in programming languages such as C, C++, or Pascal. These constructs are necessary inorder to implement complex data structures and algorithms. A major restriction of the database language SQL, however, is that many tasks cannot be accomplished by using only the provided language elements.

PL/SQL (Procedural Language/SQL) is a procedural extension of Oracle-SQL that offers languageconstructs similar to those in imperative programming languages.

Or

A PL/SQL is a procedural language extension to the SQL in which you can declare and use the variables, constants, do exception handling and you can also write the program modules in the form ofPL/SQL subprograms.PL/SQL combines the features of a procedural language with structured query language

PL/SQL allows users and designers to develop complex database applications that require the usage ofcontrol structures and procedural elements such as procedures, functions, and modules

The basic construct in PL/SQL is a block. Blocks allow designers to combine logically related (SQL-)statements into units. In a block, constants and variables can be declared, and variables can be used to store query results. Statements in a PL/SQL block include SQL statements, control structures (loops),condition statements (if-then-else), exception handling, and calls of other PL/SQL blocks.

PL/SQL blocks that specify procedures and functions can be grouped into packages. A package

is similar to a module and has an interface and an implementation part. Oracle offers

several predefinedpackages, for example, input/output routines, file handling, job scheduling etc. (see directory
$ORACLE HOME/rdbms/admin).

Another important feature of PL/SQL is that it o ffers a mechanism to process query results in a tuple-
oriented way, that is, one tuple at a time. For this, cursors are used. A cursor basically is a pointer
to a
query result and is used to read attribute values of selected tuples into variables. A cursor typically is used in combination with a loop construct such that each tuple read by the cursor can be processedindividually.

In summary, the major goals of PL/SQL are to

- Increase the expressiveness of SQL,
- Process query results in a tuple-oriented way,
- Optimize combined SQL statements,
- Develop modular database application programs,
- Reuse program code, and
- Reduce the cost for maintaining and changing applications

Advantages of PL/SQL:-

Following are some advantages of Pl/SQL

1) Support for SQL :-PL/SQL is the procedural language extension to SQL supports all thefunctionalities of SQL.
2) Improved performance:- In SQL every statement individually goes to the ORACLE server, get processed and then execute. But in PL/SQL an entire block of statements can be sent to ORACLE server at one time, where SQL statements are processed one at atime.PL/SQL block statements drastically reduce communication between the application and ORACLE. This helps in improving theperformance.
3) Higher Productivity:- Users use procedural features to build applications.PL/SQL code is written inthe form of PL/SQL block.PL/SQL blocks can also used in other ORACLE Forms, ORACLE reports. This code reusability increases the programmers productivity.
4) Portability :- Applications written in PL/SQL are portable. We can port them from one environmentto any computer hardware and operating system environment running ORACLE.
5) Integration with ORACLE :-Both PL/SQL and ORACLE are SQL based.PL/SQL variables have data types native to the oracle RDBMS dictionary. This gives tight integration with ORACLE.

Features of PL/SQL:-

1) We can define and use variables and constants in PL/SQL.

2) PL/SQL provides control structures to control the flow of a program. The control structuressupported by PL/SQL are if..Then, loop, for..loop and others.

3) We can do row by row processing of data in PL/SQL.PL/SQL supports row by row processing usingthe mechanism called cursor.

4) We can handle pre-defined and user-defined error situations. Errors are warnings and called asexceptions in PL/SQL.

5) We can write modular application by using sub programs.

The structure of PL/SQL program:-

The basic unit of code in any PL/SQL program is a block. All PL/SQL programs are composed ofblocks. These blocks can be written sequentially.

The structure of PL/SQL block:-

```
DECLARE
        Declaratio
n sectionBEGIN
        Executabl
e section
EXCEPTION
        Exception handling section
END;
```

Where

1) Declaration section
   PL/SQL variables, types, cursors, and local subprograms are defined here.
2) Executable section
   Procedural and SQL statements are written here. This is the main section of the block. This section is required.
3) Exception handling
4) Section Error handling code is written here

Conditional statements and Loops used in PL/SQL

Conditional statements check the validity of a condition and accordingly execute a set of statements.The conditional statements supported by Pl/SQL is

1) IF..THEN
2) IF..THEN..ELSE
3) IF..THEN..ELSIF

1) IF..THN

Syntax1:-

```
If condition
THEN
Statement
list END IF;
```

2) IF..THEN..ELSE

Syntax 2:-

```
IF condition THEN
        Statement list
  ELSE
     Statements
END IF;
```

3) IF..THEN..ELSIF

Syntax 3:-

```
If condition THEN
        Stateme
nt list ELSIF
condition THEN
        Statement list
  ELSE
     Statement list
END IF;END IF;
```

2) CASE Expression :CASE expression can also be used to control the branching logic withinPL/SQL blocks. The general syntax is

```
CASE
WHEN <expression> THEN
<statements>;WHEN
<expression> THEN
<statements>;
.
ELSE
<stat
emen
ts>;
END
CASE;
```

Here expression in WHEN clause is evaluated sequentially. When result of expression is TRUE, thencorresponding set of statements are executed and program flow goes to END CASE.

ITERATIVE Constructs : Iterative constructs are used to execute a set of statements respectively.
Theiterative constructs supported by PL/SQL are follows:

1) SIMPLE LOOP
2) WHILE LOOP
3) FOR LOOP

1) The Simple LOOP : It is the simplest iterative construct and has syntax like:

```
LOOP
   Statements
END LOOP;
```

The LOOP does not facilitate a checking for a condition and so it is an endless loop. To end theiterations, the EXIT statement can be used.

```
LOOP
    <statement list>
        IF
        condition
        THENEXIT;
        END IF;
END LOOP;
```

The statements here is executable statements,which will be executed repeatedly until the conditiongiven if IF..THEN evaluates TRUE.

2) THE WHILE LOOP

The WHILE...LOOP is a condition driven construct i.e the condition is a part of the loop construct and

not to be checked separately. The loop is executed as long as the condition evaluates to TRUE.The syntax is:-

```
WHILE
    condition
    LOOP
    Statements
END LOOP
```

The condition is evaluated before each iteration of loop. If it evaluates to TRUE, sequence of statementsare executed. If the condition is evaluated to FALSE or NULL, the loop is finished and the control resumes after the END LOOP statement.

3) THE FOR LOOP :The number of iterations for LOOP and WHILE LOOP is not known in advance. THE number of iterations depends on the loop condition. The FOR LOOP can be used to have a definitenumbers of iterations.

The syntax is:-

```
For loop counter IN  [REVERSE] Low bound..High
bound LOOPStatements;
End loop;
```

Where

- loop counter –is the implicitly declared index variable as BINARY_INTEGER.
- Low bound and high bound specify the number of iteration .
- Statements:-Are the contents of the loop

EXCEPTIONS:- Exceptions are errors or warnings in a PL/SQL program.PL/SQL implements errorhandling using exceptions and exception handler.
Exceptions are the run time error that a PL/SQL program may encounter.There are two types of exceptions

1) Predefined exceptions
2) User defined exceptions

1) Predefined exceptions:- Predefined exceptions are the error condition that are defined by ORACLE.Predefined exceptions cannot be changed. Predefined exceptions correspond to common SQL errors. The predefined exceptions are raised automatically whenever a PL/SQL program violates an ORACLE rule.

2) User defined Exceptions:- A user defined exceptions is an error or a warning that is defined by theprogram.User defined exceptions can be define in the declaration section of PL/SQL block.
User defined exceptions are declared in the declarative section of a PL/SQL block.
Exceptions have atype Exception and scope**.**

Syntax :

```
DECLARE
        <Exception Name>
EXCEPTION;BEGIN
        ....
        RAISE <Exception Name>
        ...
EXCEPTION
        WHEN <Exception name> THEN
        <Action>
END;
```

Exception Handling
A PL/SQL block may contain statements that specify exception handling routines. Each error or warning during the execution of a PL/SQL block raises an exception. One can distinguish between twotypes of exceptions

System defined exceptions

- **User defined exceptions** (which must be declared by the user in the declaration part of a block wherethe exception is used/implemented)

System defined exceptions are always automatically raised whenever corresponding errors or warnings occur. User defined exceptions, in contrast, must be raised explicitly in a sequence of statements using raise <exception name>. After the keyword exception at the end of a block, user defined exception handling routines are implemented. An implementation has the pattern when <exception name> then <sequence of statements>;

The most common errors that can occur during the execution of PL/SQL programs are handled by system defined exceptions. The table below lists some of these exceptions with their names and a short description.

Oracle Error	**Equivalent Exception**	**Description**
ORA-0001	DUP_VAL_ON_INDEX	Unique constraint violated.
ORA-0051	TIMEOUT_ON_RESOURSE	Time-out occurred while waiting forrecourse
ORA-0061	TRANSACTION_BACKED_OUT	The transaction was rolled back to due to deadlock.
ORA-1001	INVALID_CURSOR	Illegal cursor operation.
ORA-1012	NOT_LOGGED_ON	Not connected to Oracle.
ORA-1017	LOGIN_DENIED	Invalid username/passward
ORA-1403	NO_DATA_FOUND	No data found.
ORA-1410	SYS_INVALID_CURSOR	Conversion to a universal rowed failed.
ORA-1422	TOO_MANY_ROWS	A SELECT…INTO statement matches more than one row.
ORA-1476	ZERO_DIVIDE	Division by zero.
ORA-1722	INVALID_NUMBER	Conversion to a number failed.
ORA-6500	STORAGE_ERROR	Internal PL/SQL error raised ifPL/SQL runs out of memory.
ORA-6501	PROGRAM_ERROR	Internal PL/SQL error.
ORA-6502	VALUE_ERROR	Truncation, arithmetic or conversionerror.
ORA-6504	ROWTYPE_MISMATCH	Host cursor variable and PL/SQL cursor variable have

		incompatible row type
ORA-6511	CURSOR_ALREADY_OPEN	Attempt to open a cursor that is already open.
ORA-6530	ACCESS_INTO_NULL	Attempt to assign values to theattributes of a NULL object.
ORA-6531	COLLECTION_IS_NULL	Attempt to apply collection methods other than EXISTS to a NULL PL/SQL table or varray.
ORA-6532	SUBSCRIPT_OUTSIDE_LIMIT	Reference to a nested table or varray index outside the declared range.
ORA-6533	SUBSCRIPT_BEYOND_COUNT	Reference to a nested table or varrayindex higher than the number of
		elements in the collection
ORA-6592	CASE_NOT_FOUND	No matching WHEN clause in a CASE statement is found
ORA-30625	SELF_IS_NULL	Attempt to call a method on a NULL object instance

Syntax:-

<Exception_name>Exception;

Handling Exceptions:- Exceptions handlers for all the exceptions are written in the exception handlingsection of a PL/SQL block.
Syntax:-

```
Exception
        When exception_name then
                Sequence_of_statements1;
        When exception_name then
                Sequence_of_statements2;
        When exception_name then
                Sequence_of_statements3;
End;
```

Example:

```
Declare
        emp sal
        EMP.SAL%TYPE; emp
        no
        EMP.EMPNO%TYPE;
        too_high_sal exception;

begin

    exception
        when NO DATA FOUND – – no tuple
        selec

        tedthen rollback;
        when too_high_sal then insert into high sal emps
        values(emp no);commit;
    end;
```

After the keyword when a list of exception names connected with or can be specified. The last whenclause in the exception part may contain the exception name others. This introduces the default exception handling routine, for example, a rollback.

If a PL/SQL program is executed from the SQL*Plus shell, exception handling routines may contain statements that display error or warning messages on the screen. For this, the procedure raise application error can be used. This procedure has two parameters <error number> and <message text>.

<error number> is a negative integer defined by the user and must range between -20000 and -20999.

<error message> is a string with a length up to 2048 characters.The concatenation operator "||" can be used to concatenate single strings to one string. In order todisplay Numeric variables, these variables must be converted to strings using the function to char. If the procedure raise application error is called from a PL/SQL block, processing thePL/SQL block terminates and all database modifications are undone, that is, an implicitrollback is performed in addition to displaying the error message.

```
select EMPNO, SAL into emp no, emp sal from EMP
where ENAME = "KING";
```

if emp sal *1.05 > 4000 then raise too high sal
else update EMP set SQL . . .end if

Example:

if emp sal *1.05 > 4000
then raise application error(-20010, "Salary increase for employee with Id "|| to char (EMP no) ||"is too high");

E.g.

```
Declare
    V_maxno number (2):=20;
    V_curno number
    (2);
    E_too_many_emp
    exception;

Select count (empno)into v_curno from empWhere
deptno=10;
If v_curno>25 then Raise
e_too_many_Emp;End if;Exception
    when e_too_many_emp then
    ....
    .....
end
```

Conclusion: Thus we have studied how to use control structure and exception handling.

Lab Exercise

1) Write a PL/SQL block to calculate factorial. Use Exception Handling.
2) Write a PL/SQL block to find prime number for first 30 numbers.
3) Write a PL/SQL block to find Fibonacci series for first 50 numbers.
4) Write a PL/SQL block to find **a** raised to power **b i.e. a^b**
5) Write a PL/SQL block to find the grade of a student. Enter marks for 5 subjects.
6) Write a PL/SQL block to update the table. **Table: ACCT_MSTR ==>**

ACCT_NO	CURBAL
SB1	500
SB5	500
SB9	500
SB13	500

7) Write on your own one PL/SQL block for the problem statement.

FAQ :

1) What is PL/SQL? Explain.
2) What is the difference between "SQL" and "PL/SQL"?
3) What are the different Goals of PL/SQL?
4) What are exceptions? What are the different types of exceptions?
5) What are the different conditional statements used in PL/SQL?
6) What are the different iterative construct used in PL/SQL? Explain in short.
7) What are the features of PL/SQL? Explain.
8) What are the advantages of PL/SQL? Explain
9) How will you stop an infinite loop without closing the program?
10) Why PL/SQL does not support retrieving multiple records?

Assignment No.	**5**
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.**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 5

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

```
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 : Thus we have studied how to calculate area of circle using PLSQL

Lab Exercise :

1. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 6 to 10. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

2. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 10 to 15. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

3. Write a PL/SQL code block to calculate the area of a circle for a value of radius varying from 4 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.

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**.**

Assignment No.	**6**
Title	**Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 6

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>=990then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class. Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class).

Objective :-Learning the concept of procedure, function & package in PL/SQL

Theory :- PROCEDURE:-

A procedure is a subprogram that performs a specific action or task. A procedure has two parts.

1) The procedure specification: The procedure specification specifies the procedure name and theparameters it accepts. It is not necessary to create a procedure that accepts parameters.

2) The procedure body: The procedure body contains the declarative section without DECLAREkeyword, the executable section and an exception section.

Syntax for creating a procedure

```
Create [or replace] PROCEDURE
procedure_name[(argument1 [IN / OUT
/ IN OUT] type), (argument2 [IN / OUT /
IN OUT] type),
....]

IS/AS
```

Procedure_body :

Where

Procedure_name: – is the name of the procedure to be createdArgument:- is the name of the procedure
parameter
Type:- Is the data type of the associated parameter

Procedure_body:-Is a PL/SQL block that makes up the code of the procedure.

IN:-This is default mode. The value of the actual parameter is passed into the procedure. Inside the procedure the formal parameter is considered read only.
OUT:-Any value the actual parameter has when the procedure is called ignored. Inside the procedure ,the formal parameters are considered as write only.
IN OUT:-this mode is combination of IN and OUT

Deleting procedure: - To remove a procedure from the database.

Syntax:-

Drop procedure<procedure_name>;

FUNCTION:-

A function is a subprogram, which is used to compute values. It is similar to a procedure, function alsotakes arguments and can be in different modes. Function also can be stored in the database. It is a PL/SQL block consisting of declarative, executable and exception section. Difference between procedure and function is that the procedure call is a PL/SQL statement by itself,while a function call is called as a part of an expression.
A function can return more than one value using OUT parameter.A function can be called using positional or named notation.

Syntax for creating a function:-

Create [or replace] FUNCTION function_name[(argument1 [IN / OUT / IN OUT] type), (argument2 [IN / OUT / IN OUT] type), ...]

Return return_type IS / AS

Where

Function_body

Function_name: – is the name of the function to be createdArgument: - is the name of the function parameter
Type:- Is the data type of the associated parameter

Function_body:-Is a PL/SQL block containing code for the function.

IN:-This is default mode. The value of the actual parameter is passed into the procedure. Insidethe procedure the formal parameter is considered read only.
OUT:-Any value the actual parameter has when the procedure is called ignored. Inside the procedure ,the formal parameters are considered as write only.
INOUT:-this mode is combination of IN and OUT

Deleting a Function:- To remove the subprogram from the database.

Syntax:-

Drop function<function_name>

Package :

A package is a PL/SQL construct that allows related objects to be stored together. A package has 2 separate parts: the specification and the body. Each of them stored separately in the data dictionary.**Package Specification** :

```
CREATE OR REPLACE PACKAGE package_name

{IS|AS}

type_definition|

procedure_specification |Function
specification|variable_declaration |
exception_dec
laration |
cursor_declar
ation |
pragma
declaration |
end
[procedure_n
ame];
```

Package Body:

The package body is separate data dictionary object from the package header. It cannot be successfullycompiled unless the package header has already been successfully compiled.

Syntax:

```
CREATE OR REPLACE PACKAGE BODY package_name
ASProcedure definition;
Function definition;

.....

End package_name
```

To drop the package(both specification & the body) use the **drop package** command as follows:

Syntax :

Drop package <package_name>;

Conclusion: Thus we have studied how to calculated students marks using stored procedure method.

Lab Exercise

1) Write a procedure on EMP table. It should increase commission of an employee. Employeenumber and commission are passed as parameters to the called procedure.
2) Write a function that returns the number of employees working in a department. Pass departmentnumber as an input to the function.
3) Create table classes with the following fields

   (Deptno, course, cur_student, max_student) Insert 4 or 5 records and

   Write a function which returns true if the specified class is 80 percent full or more, and false otherwise.

   Write a PL/SQL block to call this function and use cursor in PL/SQL block to holdthe records of all department.
4) Write a procedure to update records of classes table and write a PL/SQL block to call thatprocedure.
5) Create a package which consist of procedures for insert ,delete and update the data of classestable.

FAQ :

1) Explain the term procedure and function of PL/SQL in short.
2) What is the difference between "procedure" and "function"?
3) What is the difference between "%type" and "%rowtype"?
4) What is package? Explain.
5) What is the use of package?
6) What are the different modes of argument passing?
7) What is difference between IN & IN OUT?
8) Write a package which consists of cursor, trigger, procedure & function.
9) What are the advantages of procedure & function?
10) Write the syntax to drop function, procedure & package/

Assignment No.	**7**
Title	**Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor) 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.**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 7

Title :- Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor) 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 calledcontext 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 handleor 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 thePL/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
```

PL/SQL Cursor :

When an SQL statement is processed, Oracle creates a memory area known as context area.

A cursor is a pointer to this context area. It contains all information needed for processing

the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information

on a select statement and the rows of data accessed by it.

A cursor is used to referred to a program to fetch and process the rows returned by the SQL

statement, one at a time. There are two types of cursors:

- Implicit Cursors
- Explicit Cursors

1) PL/SQL Implicit Cursors

The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if you don't use an explicit cursor for the statement. These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are: %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

For example: When you execute the SQL statements like INSERT, UPDATE, DELETE then the

cursor attributes tell whether any rows are affected and how many have been affected. If you run a SELECT INTO statement in PL/SQL block, the implicit cursor attribute can be used to find out whether any row has been returned by the SELECT statement. It will return an error if there no data is selected.The following table soecifies the status of the cursor with each of its attribute.

Attribute	Description
%FOUND	Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO statement returned one or more rows. Otherwise it returns FALSE.
%NOTFOUND	Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect no row, or a SELECT INTO statement return no rows. Otherwise it returns FALSE. It is a just opposite of %FOUND.
%ISOPEN	It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements.
%ROWCOUNT	It returns the number of rows affected by DML statements like INSERT, DELETE, and UPDATE or returned by a SELECT INTO statement.

PL/SQL Implicit Cursor Example

Create customers table and have records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	20000
2	Suresh	22	Kanpur	22000
3	Mahesh	24	Ghaziabad	24000
4	Chandan	25	Noida	26000
5	Alex	21	Paris	28000
6	Sunita	20	Delhi	30000

Let's execute the following program to update the table and increase salary of each customer by 5000. Here, SQL%ROWCOUNT attribute is used to determine the number of rows affected:

Create procedure:

```
DECLARE
  total_rows number(2);
BEGIN
UPDATE  customers  SET salary = salary + 5000;
  IF sql%notfound THEN
    dbms_output.put_line('no customers updated');
  ELSIF sql%found THEN
     total_rows := sql%rowcount;
     dbms_output.put_line( total_rows || ' customers updated ');
  END IF;
END;
/
```

Output:

6 customers updated

PL/SQL procedure successfully completed.

Now, if you check the records in customer table, you will find that the rows are updated.

```
select * from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	25000
2	Suresh	22	Kanpur	27000
3	Mahesh	24	Ghaziabad	29000
4	Chandan	25	Noida	31000
5	Alex	21	Paris	33000
6	Sunita	20	Delhi	35000

2) PL/SQL Explicit Cursors

The Explicit cursors are defined by the programmers to gain more control over the context area.

These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Following is the syntax to create an explicit cursor:

Syntax of explicit cursor

Following is the syntax to create an explicit cursor:

```
CURSOR cursor_name IS select_statement;;
```

Steps:`

You must follow these steps while working with an explicit cursor.

1. Declare the cursor to initialize in the memory.

2. Open the cursor to allocate memory.
3. Fetch the cursor to retrieve data.
4. Close the cursor to release allocated memory.

1) Declare the cursor:

It defines the cursor with a name and the associated SELECT statement.

Syntax for explicit cursor decleration

```
CURSOR name IS
 SELECT statement;
```

2) Open the cursor:

It is used to allocate memory for the cursor and make it easy to fetch the rows returned by the

SQL statements into it.

Syntax for cursor open:

```
OPEN cursor_name;
```

3) Fetch the cursor:

It is used to access one row at a time. You can fetch rows from the above-opened cursor as follows:

Syntax for cursor fetch:

```
FETCH cursor_name INTO variable_list;
```

4) Close the cursor:

It is used to release the allocated memory. The following syntax is used to close the above-opened cursors.

Syntax for cursor close:

```
Close cursor_name;
```

PL/SQL Explicit Cursor Example

Explicit cursors are defined by programmers to gain more control over the context area. It is defined

in the declaration section of the PL/SQL block. It is created on a SELECT statement which returns more than one row.

Let's take an example to demonstrate the use of explicit cursor. In this example, we are using the already created CUSTOMERS table.

Create customers table and have records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	23	Allahabad	20000
2	Suresh	22	Kanpur	22000
3	Mahesh	24	Ghaziabad	24000
4	Chandan	25	Noida	26000
5	Alex	21	Paris	28000
6	Sunita	20	Delhi	30000

Create procedure:

Execute the following program to retrieve the customer name and address.

```
DECLARE
  c_id customers.id%type;
  c_name customers.name%type;
  c_addr customers.address%type;
  CURSOR c_customers is
    SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  LOOP
    FETCH c_customers into c_id, c_name, c_addr;
    EXIT WHEN c_customers%notfound;
   dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
  END LOOP;
  CLOSE c_customers;
END;
/
```

Output:

1 Ramesh Allahabad
2 Suresh Kanpur
3 Mahesh Ghaziabad
4 Chandan Noida
5 Alex Paris
6 Sunita Delhi

Drawbacks of Implicit Cursors

Even if your query returns only a single row, you might still decide to use an explicit cursor. Theimplicit cursor has the following drawbacks:

It is less efficient than an explicit cursor
- It is more vulnerable to data errors
- It gives you less programmatic control

The following sections explore each of these limitations to the implicit cursor.

Inefficiencies of implicit cursors

An explicit cursor is, at least theoretically, more efficient than an implicit cursor. An implicit cursor executes as a SQL statement and Oracle's SQL is ANSI-standard. ANSI dictates that a single-row query must not only fetch the first record, but must also perform a second fetch to determine if too many rows will be returned by that query (such a situation will RAISE the TOO_MANY_ROWS PL/SQL exception).Thus, an implicit query always performs a minimum of two fetches, while an explicit cursor only needs toperform a single fetch.
This additional fetch is usually not noticeable, and you shouldn't be neurotic about using an implicitcursor for a single-row query (it takes less coding, so the temptation is always there). Look out for indiscriminate use of the implicit cursor in the parts of your application where that cursor will be executed repeatedly. A good example is the Post-Query trigger in the Oracle Forms.
Post-Query fires once for each record retrieved by the query (created from the base table block and the criteria entered by the user). If a query retrieves ten rows, then an additional ten fetches are needed withan implicit query. If you have 25 users on your system all performing a similar query, your server mustprocess 250 additional (unnecessary) fetches against the database. So, while it might be easier to write an implicit query, there are some places in your code where you will want to make that extra effort andgo with the explicit cursor.

Vulnerability to data errors

If an implicit SELECT statement returns more than one row, it raises the TOO_MANY_ROWS exception. When this happens, execution in the current block terminates and control is passed to theexception section. Unless you deliberately plan to handle this scenario, use of the implicit cursor is adeclaration of faith. You are saying, "I trust that query to always return a single row!"
It may well be that today, with the current data, the query will only return a single row. If the nature ofthe data ever changes, however, you may find that the SELECT statement which formerly identified asingle row now returns several. Your program will raise an exception.

Conclusion: Thus we have studied how to use the concept of cursor in PL/SQL

Lab Exercise

1) Create table with name **student** having the field **rollno**, **first name**, **last name** & **branch**. Insert10 records into table. Write a PL/SQL to create a cursor to hold all the record of student table having branch „**Computer Science**". Display all the records.

2) Write a PL/SQL block to update the record of rollno =100 & set the **branch** to **E and TC'**, if itis not present then insert the record into the student table with the id=100; (use implicit cursor sql%notfound).
3) Write a cursor and use it to raise the employee salaries as follows:
   i) All employees of department 20 get 5% raise
   ii) All employees of department 30 get 10% raise
   iii) Rest of employees get 7.5% raiseUse separate cursor.

FAQ :

1) What is cursor?
2) What are the different types of cursors?
3) What are the different attributes of explicit cursor? Explain in brief.
4) What is implicit cursor?
5) Explain the FOR loop of Cursor.
6) What is difference between simple loop, while loop & for loop?
7) What is difference between Implicit & Explicit Cursor?
8) Explain FOR UPDATE cursor with an example.
9) What is CURRENT OF clause in cursor? Give an example.
10) List all predefined cursor

Assignment No.	**8**
Title	**Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 8

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 thatare being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.
Frame the problem statement for writing Database Triggers of all types, in-line with abovestatement. The problem statement should clearly state the requirements.

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 iscalled 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 forthe manipulation operation that fires the trigger.

2. Row Trigger:-A row trigger is a trigger in which the trigger action is performed

repeatedly for eachrow 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 isused 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 theseoperations 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:-trigger name is the name of the trigger. Table_name :-is thye table name for which ger is defined.Trigger-condition:-The trigger condition in the when clause,if present is evaluates

The bodyof 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;

Conclusion: Thus we have studied how to use open source database .

Lab Exercise :-

1) Create a trigger that audits the operations on
   an Emp table.Steps
   Create table emp_audit

   (id number, operation varchar2(6), Dt date, User_id number, Username varchar2(20));

   If any operation like insert, update, delete done on EMP table then insert into

   EMP_audit tableinformation like the name of the operation with id, user_id and date.
2) Create a table Employee(id, Emp_name, Salary, City)

   Create a trigger to convert the Emp_name into upper case before
   inserting or updating onEmployee table.
3) Create a trigger to check Salary is less than 20000 before inserting or
   updating on Employeetable.
4) Create a trigger (Statement Level Trigger) to display messages after inserting
   or updating ordeleting records on Employee Table.

FAQ :

1) Write a database Trigger
2) Explain Database Trigger Components.
3) Explain Trigger Types with e.g.
4) Explain difference between Row-Level & Statement-Level Trigger.
5) Write a Syntax for Enable & Disable Trigger.
6) Write a Syntax for Displaying Trigger Errors.

Assignment No.	**9**
Title	**Database Connectivity:** Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No: 9

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

In this experiment we are going to learn how to do basic database operations using JDBC (Java Database Connectivity) API. These basic operations are **INSERT, SELECT, UPDATE and DELETE** statements in SQL language. Although the target database system is Oracle Database, but the same techniques can be applied to other database systems as well because of the query syntax used is standard SQL is generally supported by all relational database systems.

Prerequisites :

- JDK
- Oracle Database
- JDBC driver for Oracle Database You
- need to add *ojdbc6.jar* to project library.

Creating a user in Oracle Database and granting required permissions :

Open oracle using cmd. For that type sqlplus in cmd and press Enter.
Create a user-id protected by a password. This user-id is called child user.
create user identified by ;

Grant required permissions to child user. For simplicity we grant database administrator privilege to child user. conn / as sys dba; grant dba to ;

Create a sample table with blank fields :
Principal JDBC interfaces and classes
Let's take an overview look at the JDBC's main interfaces and classes which we'll use in this article. They are all available under the *java.sql* package:

- **Class.forName() :** Here we load the driver's class file into memory at the runtime. No need of using new or creation of object.
  Class. For Name("oracle. jdbc. driver. Oracle Driver");
- **DriverManager:** This class is used to register driver for a specific database type (e.g. Oracle Database in this tutorial) and to establish a database connection with the server via its **getConnection()** method.
- **Connection:** This interface represents an established database connection (session)

from which we can create statements to execute queries and retrieve results, get metadata about the database, close connection, etc.

Statement and **PreparedStatement**: These interfaces are used to execute static SQL query and parameterized SQL query, respectively. **Statement** is the super interface of the **PreparedStatement** interface. Their commonly used methods are:

boolean execute(String sql): executes a general SQL statement. It returns *true* if the query returns a *ResultSet*, false if the query returns an update count or returns nothing. This method can be used with a *Statement* only.

int executeUpdate(String sql): executes an INSERT, UPDATE or DELETE statement and returns an update account indicating number of rows affected (e.g. 1 row inserted, or 2 rows updated, or 0 rows affected).

- **ResultSet executeQuery(String sql):** executes a SELECT statement and returns a *ResultSet* object which contains results returned by the quer

- **ResultSet:** contains table data returned by a SELECT query. Use this object to iterate over rows in the result set using next() method.

- **SQLException:** this checked exception is declared to be thrown by all the above methods, so we have to catch this exception explicitly when calling the above classes' methods.

Connecting to the Database

The Oracle Database server listens on the default port *1521* at *localhost*. The following code snippet connects to the database name **userid** by the user **login1** and password **pwd1**.

Java

```
// Java program to illustrate

// Connecting to the Database

import java.sql.*;

public class connect

{

   public static void main(String args[])
```

```
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
// Establishing Connection
            Connection con = DriverManager.getConnection(
             "jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
            if (con != null)
                System.out.println("Connected");
            else
System.out.println("Not Connected");
            con.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Note: Here ***oracle*** in database URL in getConnection() method specifies SID of Oracle Database. For Oracle database 11g it is ***orcl*** and for oracle database 10g it is ***xe***.

Implementing Insert Statement

- Java

```
// Java program to illustrate

// inserting to the Database

import java.sql.*;

public class insert1

{

    public static void main(String args[])

    {

        String id = "id1";

        String pwd = "pwd1";

        String fullname = "geeks for geeks";

        String email = "geeks@geeks.org";

        try

        {

            Class.forName("oracle.jdbc.driver.OracleDriver");

            Connection con = DriverManager.getConnection("
```

```
            jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");

            Statement stmt = con.createStatement();

// Inserting data in database

            String q1 = "insert into userid values('" +id+ "', '" +pwd+

                                "', '" +fullname+ "', '" +email+ "')";

  int x = stmt.executeUpdate(q1);

            if (x > 0)

                System.out.println("Successfully Inserted");

            else

                System.out.println("Insert Failed")

            con.close();

        }

        catch(Exception e)

        {

            System.out.println(e);

        }

    }

}
```

Implementing Update Statement

Java

```
// Java program to illustrate
// updating the Database
import java.sql.*;
public class update1
{
    public static void main(String args[])
    {
        String id = "id1";
        String pwd = "pwd1";
        String newPwd = "newpwd";
        try
        {
            Class. For Name ("oracle .jdbc. driver. Oracle Driver")
  Connection con = DriverManager.getConnection("
             jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
            Statement stmt = con.createStatement();
```

// Updating database

```
        String q1 = "UPDATE userid set pwd = '" + newPwd +
              "' WHERE id = '" +id+ "' AND pwd = '" + pwd + "'";
        int x = stmt.executeUpdate(q1);
        if (x > 0)
           System.out.println("Password Successfully Updated");
        else
           System.out.println("ERROR OCCURRED :(");
        con.close();
      }
      catch(Exception e)
      {
        System.out.println(e);
      }
   }
}
```

Implementing Delete Statement

Java

```
// Java program to illustrate

// deleting from Database

import java.sql.*;

public class delete

{

 public static void main(String args[])

   {

      String id = "id2";

      String pwd = "pwd2";

      try

      {

         Class .for Name("oracle .jdbc .driver .Oracle Driver");

         Connection con = Driver Manager .get Connection("

          jdbc: oracle :thin:@localhost:1521:orcl", "login1", "pwd1");

         Statement stmt = con.createStatement();
```

// Deleting from database

```
        String q1 = "DELETE from userid WHERE id = '" + id +
              "' AND pwd = '" + pwd + "'";
        int x = stmt .execute Update(q1)
        if (x > 0)
           System.out.println("One User Successfully Deleted");
        else
           System.out.println("ERROR OCCURRED :(");
        con.close();
      }
      catch(Exception e)
      {
        System.out.println(e);
      }
   }
}
```

Implementing Select Statement

Java

```
// Java program to illustrate
// selecting from Database
import java.sql.*;
 public class select
{
    public static void main(String args[])
    {
        String id = "id1";
String pwd = "pwd1";
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("
                  jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
            Statement stmt = con.createStatement();
// SELECT query
            String q1 = "select * from userid WHERE id = '" + id +  "' AND pwd = '" + pwd +
```

```
"  ResultSet rs = stmt.executeQuery(q1);
          if (rs.next())
          {
             System.out.println("User-Id : " + rs.getString(1));
             System.out.println("Full Name :" + rs.getString(3));
             System.out.println("E-mail :" + rs.getString(4));
          }
 Else
          {
             System.out.println("No such user id is already registered");
          }
          con.close();
       }
       catch(Exception e)
       {
          System.out.println(e);
} }}
```

Output:

```
User-id- id1
Full Name -geeks for geeks
E-mail –geeks@geeks.org
```

GROUP B

NoSQL Databases

Assignment No.	**1**
Title	**Study of Open Source NOSQL Database**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 1

Title : Study of Open Source NOSQL Database: MongoDB (Installation, BasicCRUD operations, Execution)

Objectives : Learn the concept of NOSQL

Theory :

Difference between SQL and NOSQL

	SQL	NoSQL
Database Type	Relational Databases	Non-relational Databases / Distributed Databases
Structure	Table-based	• Key-value pairs • Document-based • Graph databases • Wide-column stores
Scalability	Designed for scaling up vertically by upgrading one expensive custom-built hardware	Designed for scaling out horizontally by using shards to distribute load across multiple commodity (inexpensive) hardware
Strength	• Great for highly structured data and don't anticipate changes to the database structure • Working with complex queries and reports	• Pairs well with fast paced, agile development teams • Data consistency and integrity is not top priority • Expecting high transaction load

NOSQL

A **NoSQL** (originally referring to "non SQL" or "non relational") database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Such databases have existed since the late 1960s, but did not obtain the "NoSQL" moniker until a surge of popularity in the early twenty-first century, triggered by the needs of Web 2.0 companies such as Facebook, Google, and Amazon.com. NoSQL databases are increasingly used in bigdata and real-time web applications. [6] NoSQL systems are also sometimes called "Not only SQL" to emphasize that

they may support SQL-like query languages.

Motivations for this approach include: simplicity of design, simpler "horizontal" scaling to clusters of machines (which is a problem for relational databases), [2] and finer control over availability. The datastructures used by NoSQL databases (e.g. key-value, wide column, graph, or document) are different from those used by default in relational databases, making some operations faster in NoSQL. The particular suitability of a given NoSQL database depends on the problem it must solve. Sometimes thedata structures used by NoSQL databases are also viewed as "more flexible" than relational database tables.

Many NoSQL stores compromise consistency (in the sense of the CAP theorem) in favor of availability,partition tolerance, and speed. Barriers to the greater adoption of NoSQL stores include the use of low- level query languages (instead of SQL, for instance the lack of ability to perform ad-hoc joins across tables), lack of standardized interfaces, and huge previous investments in existing relational databases. Most NoSQL stores lack true ACID transactions, although a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs. (See ACID and join support.)

MongoDB

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 filesystem. 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 canhave 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 thatdocuments 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.
Sample Document

Following example shows the document structure of a blog site, which is simply a commaseparated key value pair.

```
{
_id: ObjectId(7df78ad8902c)title: 'MongoDB Overview',
description: 'MongoDB is no sql database',by: 'tutorials point',
url:'http://www.tutorialspoi
nt.com',
```

tags: ['mongodb', 'database', 'NoSQL'],likes: 100,
comments: [{user:'user1',
message: 'My first comment', dateCreated: new Date(2011,1,20,2,15),like: 0
},

{

user:'user2',

message: 'My second comments', dateCreated: new Date(2011,1,25,7,45),like: 5
}]

}

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide _id while inserting the document. If you don‟t provide then MongoDB provides

a unique id

for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id,next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

MongoDB — Advantages:

Any relational database has a typical schema design that shows number of tables and the relationshipbetween these tables. While in MongoDB, there is no concept of relationship.

Advantages of MongoDB over RDBMS

- Schema less: MongoDB is a document database in which one collection holds

different documents. Number of fields, content and size of the document can differfrom one document to another.
- Structure of a single object is clear..0
- No complex joins.
- Deep query -ability.

MongoDB supports dynamic queries on documents using a document-based query language that'snearly as powerful as SQL.
- Tuning.
- Ease of scale -out: MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) workin g set, enabling faster access of data.

Why Use MongoDB?

- Document Oriented Storage: Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability
- Auto -sharding
- Rich queries
- Fast in -place updates
- Professional support by MongoDB

Where to Use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management

Data Hub

MongoDB Help

To get a list of commands, type db.help() in MongoDB client. This will give you a list of commands as shown in the following screenshot.

The use Command

MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of use DATABASE statement is as follows:

use DATABASE_NAME

Example

If you want to create a database with name <mydb>, then use DATABASE statement would be asfollows:

>**use mydb**

switched to db mydb

To check your currently selected database, use the command db dbmb
If you want to check your databases list, use the command show dbs.

>**show dbs**

local
0.78125
GBtest
0.23012
GB
Your created database (mydb) is not present in list. To display database, you need to insert at least onedocument into it.

>**db.movie.insert({"name":"tutorials point"})**

In MongoDB default database is test. If you didn't create any database, then collections will be storedin test database.

The dropDatabase() Method

MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax

db.dropDatabase()

The createCollection() Method

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

Syntax : **db.createCollection(name, options)**

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

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

The drop() Method

MongoDB's db.collection.drop() is used to drop a collection from the database.

Syntax

db.COLLECTION_NAME.drop().

MongoDB supports many datatypes. Some of them are:

- **String**: This is the most commonly used datatype to store the data. String in MongoDB must beUTF-8 valid.

- **Integer**: This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon

your server.

Boolean : This type is used to store a boolean (true/ false) value.

- **Double:** This type is used to store floating point values.
- **Min/Max** Keys: This type is used to compare a value against the lowest and highest BSONelements.
- **Arrays:** This type is used to store arrays or list or multiple values into one key.
- **Timestamp**: ctimestamp. This can be handy for recording when a document has been modified oradded.
- **Object**: This datatype is used for embedded documents.
- **Null:** This type is used to store a Null value.
- **Symbol:** This datatype is used identically to a string; however, it's generally reserved for languagesthat use a specific symbol type.
- **Date**: This datatype is used to store the current date or time in UNIX time format. You can specify

your own date time by creating object of Date and passing day, month, year into it.

- **Object ID:** This datatype is used to store the document‟s ID.
- **Binary** data: This datatype is used to store binary data.

Code: This datatype is used to store JavaScript code into the document.

- **Regular** expression: This datatype is used to store regular expression.

MongoDB — Insert Document

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's insert() orsave()method.
Syntax

The basic syntax of insert() command is as follows −

>db.COLLECTION_NAME.insert(document)

MongoDB — Query Document

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()

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

RDBMS Where Clause Equivalents in MongoDB

To query the document on the basis of some condition, you can use following operation

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

Logical operators

AND in MongoDB

Syntax

In the find() method, if you pass multiple keys by separating them by ',' then MongoDBtreats it as AND condition. Following is the basic syntax of AND −

>db.mycol.find({key1:value1, key2:value2}).pretty()

OR in MongoDB

Syntax

To query documents based on the OR condition, you need to use $or keyword. Followingis the basic syntax of OR −

>**db.mycol.find({ $or: [{key1: value1}, {key2:value2}] }).pretty()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>10AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

db.mycol.find({"likes": {$gt:10}, $or: [{"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" }

MongoDB's 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.Syntax
The basic syntax of update() method is as follows:

>db.COLLECTION_NAME.update(SELECTIOIN_CRITERIA, UPDATED_DATA)

MongoDB Save() Method

The save() method replaces the existing document with the new document passed in the save() method.Syntax
The basic syntax of MongoDB save() method is –

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

The remove() Method

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

Basic syntax of remove() method is as follows:

>db.COLLECTION_NAME.remove(DELLETION_CRITTERIA)

Remove Only One

If there are multiple records and you want to delete only the first record, then set justOne parameter inremove() method.

>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)

Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection.This is equivalent of SQL's truncate command.

```
>db.mycol.remove()
>db.mycol.find()
```

The Limit() Method

To limit the records in MongoDB, you need to use limit() method. The method accepts one numbertype argument, which is the number of documents that you want to be displayed.

Syntax

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

>db.COLLECTION_NAME.find().limit(NUMBER)

MongoDB Skip() Method

Apart from limit() method, there is one more method skip() which also accepts number type argumentand is used to skip the number of documents.

Syntax

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

- **db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)**

The sort() Method

To sort documents in MongoDB, you need to use sort() method. The method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 isused for ascending order while -1 is used for descending order.

Syntax

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

>db.COLLECTION_NAME.find().sort({KEY:1})

Installation steps in Fedora:

->download the software "**mongodb-linux-x86_64-3.4.9.tgz**"

Copy to Download

[root@localhost Downloads]# **tar -xvzf mongodb-linux-x86_64-3.4.9.tgz**[root@localhost Downloads]# **cd mongodb-linux-x86_64-3.4.9/** [root@localhost mongodb-linux-x86_64-3.4.9]# **cd bin**
[root@localhost bin]# **./mongod -dbpath /home/admin/**

[root@localhost bin]# **./mongo**

Installation steps in Windows:

1. Install MongoDB setup

2. Open "C:\Program Files (x86)\MongoDB\Server\3.0\bin"

3. **mongod.exe ==>** Server file || **mongo.exe ==>** Client file

4. Create Folder like **"D:\TE\data\db"**

5. Run MongoDB Server

-> Open Command prompt as Administrator

-> cd C:\Program Files (x86)\MongoDB\Server\3.0\bin

-> mongod.exe --dbpath "D:\TEB\data\db"

6. Run MongoDB Client

-> Open Second Command prompt

-> cd C:\Program Files (x86)\MongoDB\Server\3.0\bin

-> **mongo.exe**

Conclusion : *Thus we have studied new concept NOSQL-MongoDB.*

FAQ :

1. What makes MongoDB the best?
2. If you remove an object attribute, is it deleted from the database? Explain with example.
3. How does MongoDB provide consistency?
4. Define MongoDB.
5. What are the key features of mongodb?
6. Which command is use to create database? Explain with example
7. Which command is use to drop database? Explain with example
8. What is the use of pretty() method? Explain with example
9. Which method is used to remove the document form the collection? Explain with example

Assignment No.	**2**
Title	**Design and Develop MongoDB Queries using CRUDoperations. (Use CRUD operations, SAVE method, logical operators)**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 2

Aim : Design and Develop MongoDB Queries using CRUD operations. (Use CRUDoperations, 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 anddocument.

Database

Database is a physical container for collections. Each database gets its own set of files on the filesystem. 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 canhave 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 thatdocuments 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 specifyconfiguration 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 ofoptions 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 itchecks 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
collecti
ons
mycolle
ction
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.fin
d().pretty()
```

Example

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

{

"_id":
ObjectId(7df78ad8
902c),"title":
"MongoDB
Overview",
"description": "MongoDB is no
sql database","by": "tutorials
point",
```

"url":
"http://www.tutorialspoint.co
m", "tags": ["mongodb",
"database",
"NoSQL"],"likes": "100"
}

>

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

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 theexisting 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 whosetitle is 'MongoDB Overview'.
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New
MongoDBTutorial'}})
>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 thesave() 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 NewTopic",
"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:

Syntax

AND in MongoDB 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. Followingis 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(7df78ad8902c),"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>10AND (by = 'tutorials point' OR title = 'MongoDB Overview')'

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

```
{
"_id":
ObjectId(7df78ad890
2c),"title": "MongoDB
Overview",
"description": "MongoDB is no sql

database","by": "tutorials point",
"url":
"http://www.tutorialspoint.com",
"tags": ["mongodb", "database",
"NoSQL"],"likes": "100" }
```

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

FAQ:-

1. Explain CREATE Operation with example.
2. Explain AND Operator with example.
3. Explain DELETE function in Mongodb.
4. Explain DELETE function in Mongodb.
5. Explain FIND function in Mongodb.
6. Explain OR Operator with example.

Assignment No.	**2**
Title	**Design and Develop MongoDB Queries using aggregation and indexing with suitable example using MongoDB**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 2

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 asingle 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(7df78ad
8902c)title:
'MongoDB

Overview',
description: 'MongoDB is no sql
database',by_user: 'tutorials point',
url: 'http://www.tutorialspoint.com', 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.com', tags: ['mongodb', 'database', 'NoSQL'],likes: 10
},

{

_id:
ObjectId(7df78ad8902e)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 byby_user Pipeline Concept
In UNIX command, shell pipeline means the possibility to execute an operation on some input and usethe output as the input for the next command and so on. MongoDB also supports same concept in aggregation framework. There is a set of possible stages and each of those is taken as a set of documents as an input and produces a resulting set of documents (or the final resulting JSON documentat the end of the pipeline). This can then in turn be used for the next stage and so on.

Following are the possible stages in aggregation framework:

- $project: Used to select some specific fields from a collection.
- $match: This is a filtering operation and thus this can reduce the amount of documents that are givenas input to the next stage.

$group: This does the actual aggregation as discussed above.

- $sort: Sorts the documents.
- $skip: With this, it is possible to skip forward in the list of documents for a given amount of documents.
- $limit: This limits the amount of docu ments to look at, by the given number starting from thecurrent positions.
- $unwind: This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thuswith this stage we will increase the amount of documents for the next stage

Expression	Description	Example
$sum	Sums up the defined value from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$sum : "$likes"}}}])
$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$avg : "$likes"}}}])
$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$min : "$likes"}}}])
$max	Gets the maximum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{$group : {_id : "$by_user", num_tutorial : {$max : "$likes"}}}])
$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$push: "$url"}}}])
$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.mycol.aggregate([{$group : {_id : "$by_user", url : {$addToSet : "$url"}}}])
$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied "$sort"-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", first_url : {$first : "$url"}}}])
$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied "$sort"-stage.	db.mycol.aggregate([{$group : {_id : "$by_user", last_url : {$last : "$url"}}}])

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 highlyinefficient 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 ofensureIndex() 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. Tocreate 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). Following is the list:

Parameter	Type	Description
background	Boolean	Builds the index in the background so that building an index does not block other database activities. Specify true to build in the background. The default value is **false**.
unique	Boolean	Creates a unique index so that the collection will not accept insertion of documents where the index key or keys match an existing value in the index. Specify true to create a unique index. The default value is **false**.

name	String	The name of the index. If unspecified, MongoDB generates an index name by concatenating the names of the indexed fields and the sort order.
dropDups	Boolean	Creates a unique index on a field that may have duplicates. MongoDB indexes only the first occurrence of a key and removes all documents from the collection that contain subsequent occurrences of that key. Specify true to create unique index. The default value is **false**.
sparse	Boolean	If true, the index only references documents with the specified field. These indexes use less space but behave differently in some situations (particularly sorts). The default value is **false**.
expireAfterSeconds	Integer	Specifies a value, in seconds, as a TTL to control how long MongoDB retains documents in this collection.
v	Index Version	The index version number. The default index version depends on the version of MongoDB running when creating the index.
weights	Document	The weight is a number ranging from 1 to 99,999 and denotes the significance of the field relative to the other indexed fields in terms of the score.

weights	Document	The weight is a number ranging from 1 to 99,999 and denotes the significance of the field relative to the other indexed fields in terms of the score.
default_language	String	For a text index, the language that determines the list of stop words and the rules for the stemmer and tokenizer. The default value is **english**.
language_override	String	For a text index, specify the name of the field in the document that contains, the language to override the default language. The default value is language.

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

FAQ : -

1. Enlist various aggregation operations.
2. Explain MIN function with example.
3. Explain PUSH function with example.
4. Explain SUM & AVG function with example.

Assignment No.	**3**
Title	**Implement Map reduces operation with suitable example using MongoDB**
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 3

Aim : Implement Map reduces operation with suitable example using Mongo DB

Objectives : Learn the concept of NOSQL Mongo DB

Theory :

As per the Mongo DB documentation, Map Reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. Mongo DB uses map Reduce command for map-reduce operations. Map Reduce is generally used for processing large data sets.

Map Reduce Command

Following is the syntax of the basic mapReduce command

```
>db.collection.map Reduce (
        function() {  emit(key,value); },
        //map functionfunction(key,values)
        {return reduceFunction},
        {  //reduce function
                out:
                collectio
                n, query:
                docume
                nt,sort:
                docume
                nt, limit:
                number
        } )
```

The map-reduce function first queries the collection, then maps the result documents to emit key-value pairs, which is then reduced based on the keys that have multiple values.In the above syntax −

- map is a javascript function that maps a value with a key and emits a key-value pair
- reduce is a javascript function that reduces or groups all the documents having the same key
- out specifies the location of the map -reduce query result
- query specifies the optional selection criteria for selecting d ocuments
- sort specifies the optional sort criteria
- limit specifies the optional maximum number of documents to be returned Using MapReduce Consider the following document structure storing user posts. The document stores user_name of the user and the status of post.

```
{"post_text": "tutorialspoint is an awesome website for tutorials" ,"user_name":
        "mark",
        "status":"active" }
```

We will use a mapReduce function on our posts collection to select all the active posts, group

them onthe basis of user_name and then count the number of posts by each user using the following code

>**db.posts.mapReduce(**

function() { emit(this.user_id,1); },

function(key, values) {return Array.sum(values)}, {query:{status:"active"}, out:"post_total" })

The above mapReduce query outputs the following result –

```
{
"result" : "post_total", "timeMillis" : 9,"counts" :
{
"input" : 4,
"emit" : 4,
"reduce" : 2,
"output" : 2
},
"ok" : 1,
}
```

The result shows that a total of 4 documents matched the query (status:"active"), the map function emitted 4 documents with key-value pairs and finally the reduce function

grouped mapped documents having the same keys into 2.

To see the result of this mapReduce query, use the find operator –

>**db.posts.mapReduce (function() { emit(this.user_id,1); }, function(key, values) {return Array.sum(values)}, {query:{status:"active"}, out:"post_total"}).find()**

The above query gives the following result which indicates that both users tomand mark have two posts in active states –

```
{ "_id" : "tom", "value" : 2 }

{ "_id" : "mark", "value" : 2 }
```

In a similar manner, MapReduce queries can be used to construct large complex aggregation queries. The use of custom Javascript functions make use of MapReduce which is very flexible and powerful.

Conclusion: *Thus we have studied Map reduce function.*

FAQ : -

1. Define and Explain mapreduce in MongoDB with examples.
2. Why to use Mapreduce in MongoDB
3. Explain the structure of ObjectID in MongoDB.
4. What are NoSQL databases? What are the different types of NoSQL databases?

Assignment No.	**4**
Title	**Database Connectivity:** Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)
Roll No.	
Class	**T.E. (C.E.)**
Date	
Subject	**Database Management System Laboratory**
Signature	

Assignment No. 4

Aim : Database Connectivity:

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

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

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 anddocument.MongoDB is the leading NoSQL database system which has become very popular for recent years due to its dynamic schema nature and advantages over big data like high performance, horizontal scalability, replication, etc. Unlike traditional relational database systems which provide JDBC-compliant drivers, MongoDB comes with its own non-JDBC driver called Mongo Java Driver. That means we cannot use JDBC API to interact with MongoDB from Java. Instead, we have to use its own Mongo Java Driver API.

1. Downloading Mongo Java Driver

Click here to download latest version of Mongo Java Driver (version 2.11.1 as of this writing). The JAR file name is **mongo-java-driver-VERSION.jar** (around 400KB). Copy the downloaded JAR file into your classpath.
Online API documentation for Mongo Java Driver can be found here.

2. Connecting to MongoDB using MongoClient

The MongoClientclass is used to make a connection with a MongoDB server and perform database-related operations. Here are some examples:

Creating a MongoClient instance that connects to a default MongoDB server running on

ocalhost and default

port:

```
1    MongoClient mongoClient = new MongoClient();
```

Connecting to a named MongoDB server listening on the default port (27017)

```
1    MongoClient mongoClient = new MongoClient("localhost");
```

Or:

```
1    MongoClient mongoClient = new MongoClient("db1.server.com");
```

Connecting to a named MongoDB server listening on a specific port:

```
1    MongoClient mongoClient = new MongoClient("localhost", 27017);
```

Or:

```
1    MongoClient mongoClient = new MongoClient("db1.server.com", 27018);
```

Connecting to a replica set of servers:

```
1   List<ServerAddress> seeds = new ArrayList<ServerAddress>();
2   seeds.add(new ServerAddress("db1.server.com", 27017));
3   seeds.add(new ServerAddress("db2.server.com", 27018));
4   seeds.add(new ServerAddress("db3.server.com", 27019));
5
6   MongoClient mongoClient = new MongoClient(seeds);
```

After the connection is established, we can obtain a database and make authentication (if the server is running in secure mode), for example:

```
1    MongoClient mongoClient = new MongoClient();
2    DB db = mongoClient.getDB("test");
3
4    char[] password = new char[] {'s', 'e', 'c', 'r', 'e', 't'};
5    boolean authenticated = db.authenticate("root", password);
6
7    if (authenticated) {
8        System.out.println("Successfully logged in to MongoDB!");
9    } else {
10       System.out.println("Invalid username/password");
11   }
```

By default, MongoDB server is running in trusted mode which doesn't require authentication.

Let's see a complete program:

```
1    package net.codejava.mongodb;
2
```

```
3   import java.net.UnknownHostException;
4   import java.util.List;
5   import java.util.Set


8   import com.mongodb.DB;
9   import com.mongodb.MongoClient;
10
11  public class JavaMongoDBConnection {
12
13      public static void main(String[] args) {
14          try {
15
16              MongoClient mongoClient = new MongoClient("localhost");
17
18              List<String> databases = mongoClient.getDatabaseNames();
19
20              for (String dbName : databases) {
21                  System.out.println("- Database: " + dbName);
22
23                  DB db = mongoClient.getDB(dbName);
24
25                  Set<String> collections = db.getCollectionNames();
26                  for (String colName : collections) {
27                      System.out.println("\t + Collection: " + colName);
28                  }
29              }
30
31              mongoClient.close();
32
33          } catch (UnknownHostException ex) {
34              ex.printStackTrace();
35          }
36
37      }
    }
```

This Java program connects to a MongoDB server running on localhost at default port, then lists all database names available on the server. For each database, it lists all collection names (a collection is equivalent to a table in relational database), and finally closes the connection. This program would produce the following output:

- Database: local
+ Collection: startup_log
- Database: mydb
+ Collection: system.indexes
+ Collection: things
- Database: test
+ Collection: system.indexes
+ Collection: test

3. Using MongoDB connection string URI

It's also possible to use a String that represents a database connection URI to connect to the MongoDB server, for example:

```
1    String dbURI = "mongodb://localhost";
2    MongoClient mongoClient = new MongoClient(new MongoClientURI(dbURI));
```

Syntax of the URI is as follows:

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]

Here are some connection string URI examples:

- Connecting to the MongoDB server running on localhost at the default port:

  mongodb://localhost

- Connecting to the ***admin*** database on a named MongoDB server ***db1.server.com*** running on port ***27027*** with user ***root*** and password ***secret***:

  mongodb://root:secret@db1.server.com:27027

- Connecting to the ***users*** database on server ***db2.server.com***:

  mongodb://db2.server.com/users

- Connecting to the ***products*** database on a named MongoDB server ***db3.server.com*** running on port ***27027*** with user ***tom*** and password ***secret***:

  mongodb://tom:secret@db3.server.com:27027/products

- Connecting to a replica set of three servers:

  mongodb://db1.server.com,db2.server.com,db3.server.com

Or

```
package com.mongodb.quickstart;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import org.bson.Document;

import java.util.ArrayList;
import java.util.List;

public class Connection {

    public static void main(String[] args) {
        String connectionString = System.getProperty("mongodb.uri");
        try (MongoClient mongoClient = MongoClients.create(connectionString)) {
            List<Document> databases = mongoClient.listDatabases().into(new ArrayList<>());


            databases.forEach(db -> System.out.println(db.toJson()));
        }
    }
}
```

As you can see, the MongoDB connection string is retrieved from the *System Properties* so we need to set this up. Once you have retrieved your MongoDB Atlas connection string , you can add the mongodb.uri system property into your IDE. Here is my configuration with IntelliJ for example.

IntelliJ Configuration

Or if you prefer to use Maven in command line, here is the equivalent command line you can run in the root directory:

```
mvn compile exec:java -Dexec.mainClass="com.mongodb.quickstart.Connection" -
Dmongodb.uri="mongodb+srv://username:password@cluster0-
abcde.mongodb.net/test?w=majority"
```

Note: Don't forget the double quotes around the MongoDB URI to avoid surprises from your shell.
The standard output should look like this:

"name": "admin", "sizeOnDisk": 303104.0, "empty": false}

"name": "config", "sizeOnDisk": 147456.0, "empty": false}

{"name": "local", "sizeOnDisk": 5.44731136E8, "empty": false}

{"name": "sample_airbnb", "sizeOnDisk": 5.761024E7, "empty": false}

{"name": "sample_geospatial", "sizeOnDisk": 1384448.0, "empty": false}

{"name": "sample_mflix", "sizeOnDisk": 4.583424E7, "empty": false}

{"name": "sample_supplies", "sizeOnDisk": 1339392.0, "empty": false}
{"name": "sample_training", "sizeOnDisk": 7.4801152E7, "empty": false}

{"name": "sample_weatherdata", "sizeOnDisk": 5103616.0, "empty": false}Insert Operations

Getting Set Up

In the setup part, we created the classes HelloMongoDB and Connection. Now we will work on the Create class.

If you didn't set up your free cluster on MongoDB Atlas, now is great time to do so. You have all the instructions in this
blog post
.

Checking the Collection and Data Model

In the sample dataset, you can find the database sample_training, which contains a collection grades. Each
document in this collection represents a student's grades for a particular class. MongoDB Enterprise Cluster0-shard-0:PRIMARY> db.grades.findOne({student_id: 0, class_id: 339})

```
{
    "_id" : ObjectId("56d5f7eb604eb380b0d8d8ce"),
    "student_id" : 0,
    "scores" : [
        {
            "type" : "exam",
            "score" : 78.40446309504266
        },
        {
            "type" : "quiz",
            "score" : 73.36224783231339
        },
        {
            "type" : "homework",
            "score" : 46.980982486720535
        },
        {
            "type" : "homework",
            "score" : 76.67556138656222
```

```
        }
    ],
    "class_id" : 339
}



{
    "_id": {
        "$oid": "56d5f7eb604eb380b0d8d8ce"
    },
    "student_id": {
        "$numberDouble": "0"
    },
    "scores": [{
        "type": "exam",
        "score": {
            "$numberDouble": "78.40446309504266"
        }
    }, {
        "type": "quiz",
        "score": {
            "$numberDouble": "73.36224783231339"
        }
    }, {
        "type": "homework",
        "score": {
            "$numberDouble": "46.980982486720535"
        }
    }, {
        "type": "homework",
        "score": {
            "$numberDouble": "76.67556138656222"
        }
    }],
    "class_id": {


        "$numberDouble": "339"
    }
}
```


Here is the JSON representation of a document in the

```
package com.mongodb.quickstart;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
```

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import org.bson.types.ObjectId;
import java.util.Random;
import static java.util.Arrays.asList;

public class Create {

public static void main(String[] args) {
try (MongoClient mongoClient = MongoClients.create(System.getProperty("mongodb.uri"))) {

 MongoDatabase sampleTrainingDB = mongoClient.getDatabase("sample_training");
 MongoCollection<Document> gradesCollection = sampleTrainingDB.getCollection("grades");

 Random rand = new Random();
 Document student = new Document("_id", new ObjectId());
 student.append("student_id", 10000d)
       .append("class_id", 1d)
       .append("scores", asList(new Document("type", "exam").append("score",
 rand.nextDouble() * 100),
                        new Document("type", "quiz").append("score", rand.nextDouble() * 100),
                                new Document("type", "homework").append("score",

 rand.nextDouble() * 100),
                        new Document("type", "homework").append("score", rand.nextDouble()
 * 100)));
  gradesCollection.insertOne(student);
} } }
```

GROUP C

Mini Project