DBMS_Lab_Manual

Group A

Assignment - 1

Class: T.E. Computer Name:

Roll No:

TITLE: Conceptual design using ER Model, Reducing ER into tables, normalization.

OBJECTIVE Propose a Conceptual Design using ER features using tools like ERDplus, ER Winetc. Convert the ER diagram into tables on paper and propose a normalize Relational data model.

PROBLEM STATEMENT:

<< Write your mini Project Problem Statement here>>

THEORY:

- All ER Notations
- Brief rules to reduce ER diagram into tables
- Brief about 1NF, 2NF, 3NF and BCNF

E-R Diagram (for your Problem Statement):

Reduced Table Structure from above E-R Diagram:

Normalized Tables at least in 3NF:

CONCLUSION

Assignment No: 2(A)

Class: T.E. Computer Name:

Roll No:

Title - Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.

Objective: To learn all type Data Definition Language commands and their uses.
Introduction:
SQL stands for Structured Query Language
•SQL lets you access and manipulate databases
•SQL is an ANSI (American National Standards Institute) standard

Commands of SQL are grouped into four languages.
1>DDL
DDL is abbreviation of Data Definition Language. It is used to create and modify the structure of database objects in database.
Examples: CREATE, ALTER, DROP,RENAME, TRUNCATE statements
2>DML
DML is abbreviation of Data Manipulation Language. It is used to retrieve, store, modify, delete, insert and update data in database.
Examples: SELECT, UPDATE, INSERT,DELETE statements
3>DCL
DCL is abbreviation of Data Control Language. It is used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it.
Examples: GRANT, REVOKE statements
4>TCL
TCL is abbreviation of Transactional Control Language. It is used to manage different transactions occurring within a database.
Examples: COMMIT, ROLLBACK statements

Data Definition Language (DDL)
1. Data definition Language (DDL) is used to create, rename, alter, modify, drop, replace, and delete tables, Indexes, Views, and comment on database objects; and establish a default database.
2. The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables.

The most important DDL statements in SQL are:
•CREATE TABLE- Creates a new table
•ALTER TABLE- Modifies a table
•DROP TABLE- Deletes a table
•TRUNCATE -Use to truncate (delete all rows) a table.
•CREATE INDEX- Creates an index (search key)
•DROP INDEX- Deletes an index

1.The CREATE TABLE Statement
The CREATE TABLE statement is used to create a table in a database.
Syntax:
CREATE TABLE tablename (attr1_name attr1_datatype(size) attr1_constraint, attr2_name attr2_datatype(size) attr2_constraint,….);

SQL Constraints

Constraints are used to limit the type of data that can go into a table.
Constraints can be specified when a table is created (with the CREATE TABLE statement) or after the table is created (with the ALTER TABLE statement).

Following are the constraints:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

Add constraint after table creation using alter table option

Syntax - Alter table add constraint constraint_name constraint_type (Attr_name)
Example - Alter table stud add constraint prk1 primary key(rollno);
Drop constraint:
Syntax- Drop Constraint Constraint_name;
Example - Drop constraint prk1;

2. The Drop TABLE Statement

Removes the table from the database
Syntax- DROP TABLE table_name;

3. The ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
Syntax-
To add a column in a table, use the following syntax:
ALTER TABLE table_name
ADD column_name datatype;

To delete a column in a table,

Syntax-
ALTER TABLE table_name DROP COLUMN column_name;

To change the data type of a column in a table,

Syntax-
ALTER TABLE table_name MODIFY COLUMN column_name datatype;

4. The RENAME TABLE Statement:

Rename the old table to new table;
Rename old_tabname to new_tabname;

5. The TRUNCATE TABLE Statement

The ALTER TABLE Statement is used to truncate (delete all rows) a table.
Syntax -
To truncate a table,
syntax : TRUNCATE TABLE table_name;

6. CREATE VIEW Statement-

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Syntax- CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition;

7. SQL Dropping a View –

Syntax- DROP VIEW view_name;

8 . Create Index Statement

1. Index in SQL is created on existing tables to retrieve the rows quickly. When there are thousands of records in a table, retrieving information will take a long time.
2. Therefore indexes are created on columns which are accessed frequently, so that the information can be retrieved quickly.
3. Indexes can be created on a single column or a group of columns. When a index is created, it first sorts the data and then it assigns a ROWID for each row

Syntax-
CREATE INDEX index_name ON table_name (column_name1,column_name2...);

- index_name is the name of the INDEX.
- table_name is the name of the table to which the indexed column belongs.
- column_name1, column_name2.. is the list of columns which make up the INDEX

9. Drop Index Statement

Syntax: DROP INDEX index_name;

Conclusion:

AssignmentNo:2(B)

Class: T.E. Computer **Name:**

Roll No:

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

Objective: To learn and understand DML statements in MySQL.

Hardware requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software requirements:

Ubuntu 14 Operating System, MySQL

Theory:

DML command

Data Manipulation Language (DML) statements are used for managing data in database. DML commands are not auto-committed. It means changes made by DML command are not permanent to database, it can be rolled back.

1) INSERT command

Insert command is used to insert data into a table. Following is its general syntax,

INSERT into *table-name* values(data1,data2,..)

Lets see an example,

Consider a table **Student** with following fields.

S_id	**S_Name**	**age**

INSERT into Student values(101,'Adam',15);

The above command will insert a record into **Student** table.

S_id	**S_Name**	**age**
101	Adam	15

2) UPDATE command

Update command is used to update a row of a table. Following is its general syntax,

UPDATE *table-name* set column-name = value *where* condition;

Lets see an example,

update Student set age=18 where s_id=102;

Example to Update multiple columns

UPDATE Student set s_name='Abhi',age=17 where s_id=103;

3) Delete command

Delete command is used to delete data from a table. Delete command can also be used with condition to delete a particular row. Following is its general syntax,

***DELETE** from table-name;*

Example to Delete all Records from a Table

DELETE from Student;

The above command will delete all the records from **Student** table.

Example to Delete a particular Record from a Table

Consider **Student** table

DELETE from Student where s_id=103;

SQL Functions

SQL provides many built-in functions to perform operations on data. These functions are useful while performing mathematical calculations, string concatenations, sub-strings etc. SQL functions are divided into two catagories,

- Aggregrate Functions
- Scalar Functions

Aggregrate Functions

These functions return a single value after calculating from a group of values.Following are some frequently used Aggregrate functions.

1) AVG()

Average returns average value after calculating from values in a numeric column.

Its general Syntax is,

*SELECT **AVG**(column_name) from table_name*

e.g.

*SELECT **avg(salary)** from Emp;*

2) COUNT()

Count returns the number of rows present in the table either based on some condition or without condition.

Its general Syntax is,

*SELECT **COUNT**(column_name) from table-name;*

Example using COUNT()

Consider following **Emp** table

eid	name	age	salary
401	Anu	22	9000
402	Shane	29	8000

SQL query to count employees, satisfying specified condition is,

SELECT **COUNT(name)** from Emp where salary = 8000;

3) FIRST()

First function returns first value of a selected column

Syntax for FIRST function is,

*SELECT **FIRST**(column_name) from table-name*

SQL query

SELECT FIRST(salary) from Emp;

4) LAST()

LAST return the return last value from selected column

Syntax of LAST function is,

SELECT **LAST**(column_name) from *table-name*

SQL query will be,

SELECT LAST(salary) from emp;

5) MAX()

MAX function returns maximum value from selected column of the table.

Syntax of MAX function is,

SELECT **MAX**(column_name) from *table-name*

SQL query to find Maximum salary is,

SELECT MAX(salary) from emp;

6) MIN()

MIN function returns minimum value from a selected column of the table.

Syntax for MIN function is,

SELECT MIN(column_name) from table-name

SQL query to find minimum salary is,

SELECT MIN(salary) from emp;

7) SUM()

SUM function returns total sum of a selected columns numeric values.

Syntax for SUM is,

SELECT SUM(column_name) from table-name

SQL query to find sum of salaries will be,

SELECT SUM(salary) from emp;

Scalar Functions

Scalar functions return a single value from an input value. Following are soe frequently used Scalar Functions.

1) UCASE()

UCASE function is used to convert value of string column to Uppercase character.

Syntax of UCASE,

SELECT **UCASE**(column_name) from *table-name*

Example of UCASE()

SQL query for using UCASE is,

SELECT UCASE(name) from emp;

2) LCASE()

LCASE function is used to convert value of string column to Lowecase character.

Syntax for LCASE is:

SELECT **LCASE**(column_name) from *table-name*

3) MID()

MID function is used to extract substrings from column values of string type in a table.

Syntax for MID function is:

SELECT **MID**(column_name, start, length) from *table-name*

4) ROUND()

ROUND function is used to round a numeric field to number of nearest integer. It is used on Decimal point values. Syntax of Round function is,

SELECT **ROUND**(column_name, decimals) from *table-name*

Operators:

AND and **OR** operators are used with **Where** clause to make more precise conditions for fetching data from database by combining more than one condition together.

1) AND operator

AND operator is used to set multiple conditions with *Where* clause.

Example of AND

SELECT * from Emp WHERE salary < 10000 **AND** age > 25

2) OR operator

OR operator is also used to combine multiple conditions with *Where* clause. The only difference between AND and OR is their behaviour. When we use AND to combine two or more than two conditions, records satisfying all the condition will be in the result. But in case of OR, atleast one condition from the conditions specified must be satisfied by any record to be in the result.

Example of OR

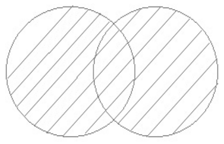
SELECT * from Emp WHERE salary > 10000 **OR** age > 25

Set Operation in SQL

SQL supports few Set operations to be performed on table data. These are used to get meaningful results from data, under different special conditions.

3) Union

UNION is used to combine the results of two or more Select statements. However it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables.



Example of UNION

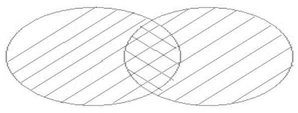
select * from First

UNION

select * from second

4) Union All

This operation is similar to Union. But it also shows the duplicate rows.



Union All query will be like,

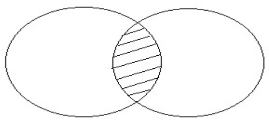
select * from First

UNION ALL

select * from second

5) Intersect

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same. MySQL does not support INTERSECT operator.



Intersect query will be,

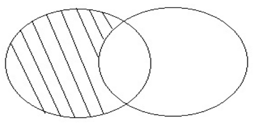
select * from First

INTERSECT

select * from second

6) Minus

Minus operation combines result of two Select statements and return only those result which belongs to first set of result. MySQL does not support INTERSECT operator.



Minus query will be,

select * from First

MINUS

```
select * from second
```

Conclusion:

Assignment No: 3

Class: T.E. Computer **Name:**

Roll No:

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

Objective:

1. To learn and understand DML statements in MySQL
2. To learn SQL Joins, Subqueries & Views.

Hardware requirements:

Any CPU with Pentium Processor or similar, 256 MBRAM or more, 1 GB Hard Disk or more.

Software requirements:

Ubuntu 14 Operating System, MySQL

Theory:

SQL JOIN

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.Let's look at a selection from the "Orders" table:

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, look at a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico

3	Antonio Moreno Taquería	Antonio Moreno	Mexico

The "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column. Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values inboth tables:

Example:

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

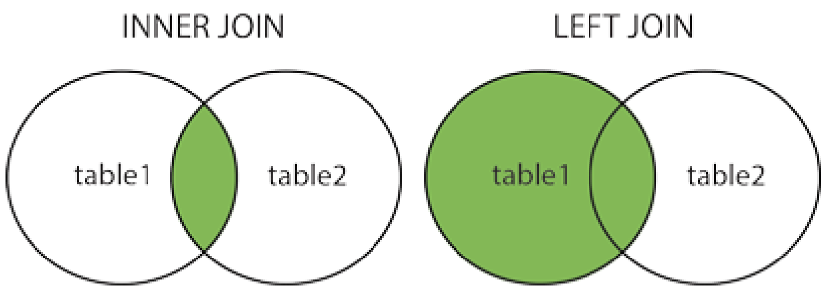
and it will produce something like this:

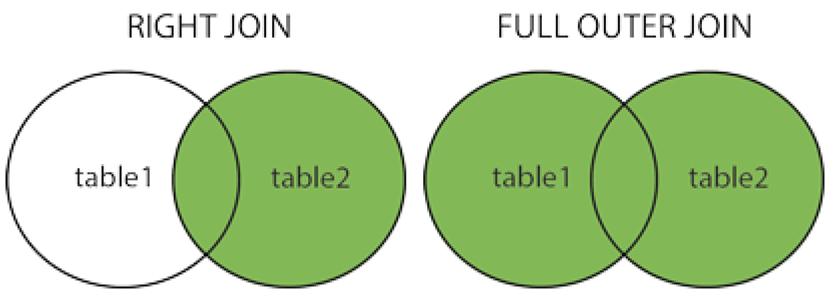
OrderID	**CustomerName**	**OrderDate**
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Different Types of SQL JOINs:

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table
- **SELF JOIN**: (As the name signifies, in SELF JOIN a table is joined to itself.)





SQL Views:

SQL CREATE VIEW Statement

In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax:

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

SQL CREATE VIEW Examples:

The view "Product List" lists all active products (products that are not discontinued) from the "Products"table. The view is created with the following SQL:

CREATE VIEW ProductList

AS SELECT ProductID,

ProductNameFROM Products

WHERE Discontinued = No;

Then, we can query the view as

follows:SELECT * FROM

ProductList;

SQL Updating a View:

You can update a view by using the following syntax:

SQL CREATE OR REPLACE VIEW Syntax

CREATE OR REPLACE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Now we want to add the "Category" column to the "Product List" view. We will update the view with thefollowing SQL:

CREATE OR REPLACE VIEW Product List AS

SELECT ProductID, ProductName, Category

FROM Products

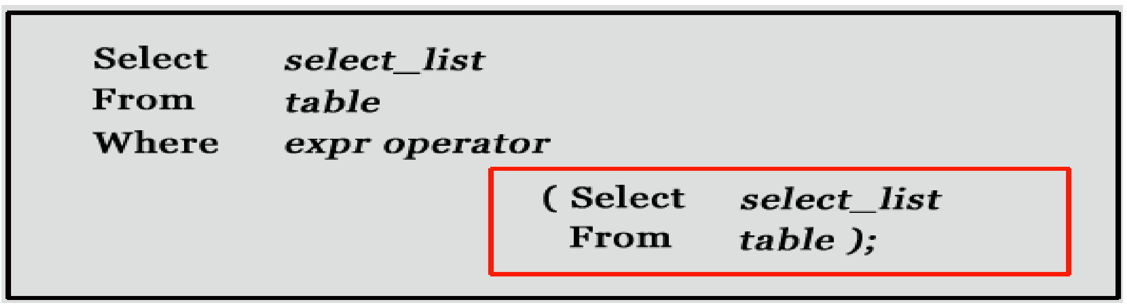
WHERE Discontinued = No;

Subqueries:

A subquery is a SQL query nested inside a larger query.

- A subquery may occur in :
  - - A SELECT clause
  - - A FROM clause
  - - A WHERE clause
- In MySQL subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL.
- A subquery can be treated as an inner query, which is a SQL query placed as a part of another query called as outer query.
- The inner query executes first before its parent query so that the results of the inner query can be passed to the outer query.

Subquery Syntax :



```
Select      select_list
From        table
Where       expr operator
                        ( Select    select_list
                          From      table );
```

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

Subquery syntax as specified by the SQL standard and supported in MySQL

```
DELETE FROM t1
  WHERE s11 > ANY
   (SELECT COUNT(*) /* no hint */ FROM t2
WHERE NOT EXISTS
 (SELECT * FROM t3
 WHERE ROW(5*t2.s1,77)=
    (SELECT 50,11*s1 FROM t4 UNION SELECT 50,77
      FROM(SELECT * FROM t5) AS t5)));
```

A subquery can return a scalar (a single value), a single row, a single column, or a table (one or more rowsof one or more columns). These are called scalar, column, row, and table subqueries.

Subqueries: Guidelines

There are some guidelines to consider when using subqueries :

- A subquery must be enclosed in parentheses.

- Use single-row operators with single-row subqueries, and use multiple-row operators with multiple-row subqueries.

- If a subquery (inner query) returns a null value to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

Types of Subqueries

- The Subquery as Scalar Operand
- Comparisons using Subqueries
- Subqueries with ALL, ANY, IN, or SOME
- Row Subqueries
- Subqueries with EXISTS or NOT EXISTS
- Correlated Subqueries
- Subqueries in the FROM Clause

Conclusion:

Thus,we have successfully studied Joins, Subqueries and views and implemented SQL Queries.

Assignment No: 4

Class: T.E. Computer **Name:**

Roll No:

Aim: Write a PL/SQL block to calculate fine for a library book by accessing borrower information from the database.

Problem Statement:

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

Objective:

1. To learn and understand PL/SQL in Oracle

Hardware requirements:

Any CPU with Pentium Processor or similar, 256

MBRAM or more, 1 GB Hard Disk or more.

Software requirements:

Windows 7 Operating System, Oracle 11g, SQL developer

Theory:

PL/SQL stands for Procedural Language extension of SQL. PL/SQL is a combination of SQL along withthe procedural features of programming languages.It was developed by Oracle Corporation in the early 90’s to enhance the capabilities of SQL.

Architecture of PL/SQL:

The PL/SQL architecture mainly consists of following 3 components:

1. PL/SQL block
2. PL/SQL Engine
3. Database Server

PL/SQL block:

- This is the component which has the actual PL/SQL code.
- This consists of different sections to divide the code logically (declarative section for declaring purpose, execution section for processing statements, exception handling section for handling errors)
- It also contains the SQL instruction that used to interact with the database server.
- All the PL/SQL units are treated as PL/SQL blocks, and this is the starting stage of the architecture which serves as the primary input.
- Following are the different type of PL/SQL units.
    - Anonymous Block
    - Function
    - Library
    - Procedure
    - Package Body
    - Package Specification
    - Trigger
    - Type
    - Type Body

PL/SQL Engine

- PL/SQL engine is the component where the actual processing of the codes takes place.
- PL/SQL engine separates PL/SQL units and SQL part in the input (as shown in the image below).
- The separated PL/SQL units will be handled with the PL/SQL engine itself.
- The SQL part will be sent to database server where the actual interaction with database takes place.
- It can be installed in both database server and in the application server.

Database Server:

- This is the most important component of Pl/SQL unit which stores the data.
- The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.
- It consists of SQL executor which actually parses the input SQL statements and execute the same.

Advantage of Using PL/SQL

1. Better performance, as sql is executed in bulk rather than a single statement
2. High Productivity
3. Tight integration with SQL
4. Full Portability
5. Tight Security
6. Support Object Oriented Programming concepts.

Basic Difference between SQL and PL/SQL

SQL	PL/SQL
• SQL is a single query that is used to	• PL/SQL is a block of codes that used to write the entire program blocks/ procedure/

perform DML and DDL operations.	function, etc.
• It is declarative, that defines what needs to be done, rather than how things need to be done.	• PL/SQL is procedural that defines how the things needs to be done.
• Execute as a single statement.	• Execute as a whole block.
• Mainly used to manipulate data.	• Mainly used to create an application.
• Interaction with Database server.	• No interaction with the database server.
• Cannot contain PL/SQL code in it.	• It is an extension of SQL, so it can contain SQL inside it.

PL/SQL Block Structure:

Basic Syntax of PL/SQL which is a block-structured language; this means that the PL/SQL programs aredivided and written in logical blocks of code. Each block consists of three sub-parts −

S.No	Sections & Description
1	Declarations This section starts with the keyword DECLARE. It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.
2	Executable Commands This section is enclosed between the keywords BEGIN and END and it is a mandatory section. It consists of the executable PL/SQL statements of the program. It should have at least one executable line of code, which may be just a NULL command to indicate that nothing should be executed.
3	Exception Handling This section starts with the keyword EXCEPTION. This optional section contains exception(s) that handle errors in the program.

Every PL/SQL statement ends with a semicolon (;). PL/SQL blocks can be nested within other PL/SQLblocks using BEGIN and END. Following is the basic structure of a PL/SQL block −

```
DECLARE
   <declarations
section>BEGIN
   <executable
command(s)>
EXCEPTION
   <exception
handling>END;
```

The 'Hello World'

```
ExampleDECLARE
   message  varchar2(20):= 'Hello, World!';
BEGIN
   dbms_output.put_line(message);
END;
/
```

The end; line signals the end of the PL/SQL block.

PL/SQL Placeholders

Placeholders are temporary storage area. PL/SQL Placeholders can be any of Variables, Constants and Records. Oracle defines placeholders to store data temporarily, which are used to manipulate data duringthe execution of a PL SQL block.

Define PL/SQL Placeholders

Depending on the kind of data you want to store, you can define placeholders with aname and a datatype. Few of the datatypes used to define placeholders are as given below.
Number (n,m) , Char (n) , Varchar2 (n) , Date , Long , Long raw, Raw, Blob, Clob, Nclob, Bfile

PL/SQL Variables
These are placeholders that store the values that can change through the PL/SQL Block.

General Syntax to declare a variable is

variable_name datatype [NOT NULL := value];

- variable_name is the name of the variable.
- datatype is a valid PL/SQL datatype.
- NOT NULL is an optional specification on the variable.
- value or DEFAULT valueis also an optional specification, where you can initialize a variable.
- Each variable declaration is a separate statement and must be terminated by a semicolon.

For example, if you want to store the current salary of an employee, you can use a variable.

DECLARE salary number (6);

The below example declares two variables, one of which is a not null.
DECLARE
salary number(4);

dept varchar2(10) NOT NULL := “HR Dept”;

The below example declares two variables, one of which is a not null.
DECLARE
salary number(4);

dept varchar2(10) NOT NULL := “HR Dept”;

The below example declares two variables, one of which is a not null.
DECLARE
salary number(4);

dept varchar2(10) NOT NULL := “HR Dept”;

Conclusion:
Thus we have suucessfully implemented PL/SQL block to retrieve fine for issued library book byreading borrower information from the database.

Assignment No: 5

Class: T.E. Computer **Name:**
Roll No:

Aim: To Study and implement PL/SQL programming along with Procedures and Functions.

Problem Statement:

Write and execute simple PL/SQL programs and apply this knowledge to execute PL/SQL procedures and functions.

Hardware requirements:

- Any CPU with Pentium Processor or similar, 256 MB
- RAM or more, 1 GB Hard Disk or more.

Software requirements:

- Windows 7 Operating System, Oracle 11g, SQL developer

Theory:
PL/SQL (Procedural Language/Structured Query Language)It is Oracle Corporation's proprietary procedural extension to the SQL database language, used inthe Oracle database. Some other SQL database management systems offer similar extensions to the SQL language. PL/SQL's syntax strongly resembles that of Ada, and just like some Ada compilers of the 1980s, the PL/SQL runtime system uses Diana as intermediate representation. The key strength of PL/SQL is its tight integration with the Oracle database.

Basic code structure in PL/SQL

```
DECLARE
TYPE / item / FUNCTION / PROCEDURE declarations
BEGIN
Statements
EXCEPTION
EXCEPTION handlers
END;
```

The DECLARE and EXCEPTION sections are optional.

Simple Example:

```
DECLARE
number1 int;
number2 int:= 17; -- value default text1
VARCHAR(12) := 'Hello world';
BEGIN
```

```
SELECT street_number
INTO number1
FROM address
WHERE name = 'Sahil';END;
```

FUNCTIONS

A function can be called inside the statement. It can return a value with the help of return statement and it returns only one value.

A function returns any single value, which can be a table. It can be used in SQL query and isn't based on precompile.

SYNTAX:

```
CREATE FUNCTION <function_name> [(input/output variable declarations)] RETURN
return_type
<IS|AS>
BEGIN
[declaration block]
<PL/SQL block WITH RETURN statement>
[EXCEPTION
EXCEPTION block]
END;
```

Example:

```
CREATE FUNCTION add(a in int,b out int,c in out int) return int IS
BEGIN
SELECT CONCAT ('a = ', a , ' b = ', b , ' c = ' , c);
SELECT CONCAT ('Addition Result = ');
return (a+b+c);
END;
```

PROCEDURES

In MySQL, a stored procedure can be called with the help of call statement. A stored procedure returns more than one value.

A stored procedure returns 0 by default. It cannot be used in SQL query and is based on precompile.

SYNTAX:

```
CREATE PROCEDURE <procedure_name> [(input/output variable declarations)]BEGIN
[declaration block]
<PL/SQL block statements>
[EXCEPTION
EXCEPTION block]
END;
```

Example:

Create Procedure to check stock of items whose quantity is less than particular number and display result in temporary table 'stockcheck' and drop temp table after display.

```
CREATE PROCEDURE check_stock()
BEGIN
DECLARE SNO INT;
DECLARE ITEM CHAR(30);
DECLARE PRICE INT
DECLARE TR INT;
DECLARE QA INT;
DECLARE C1 INT;
Declare C2 INT;
DECLARE CS cursor for select * from stock;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET C1=1;
set C2=22;
create table stockcheck(stock_no int, item char(30), quantity_available int);
OPEN CS;
lable1: loop
FETCH CS INTO SNO,ITEM,PRICE,TR,QA;
if C1=1 thenleave
lable1;end if;
if QA <10 thenset
C2=11;
insert into stockcheck values(SNO,ITEM,QA);end
if;
end loop;
if C2=22 then
Select "Enough stock for all items";
ELSE
select * from stockcheck;drop
table stcokcheck; end if;
close CS;
END;//
```

Conclusion: Thus successfully implemented procedures and function in PL/SQL.

Assignment No: 6

Class: T.E. Computer **Name:**
Roll No:

Aim: Write a PL/SQL block to create cursor to copy contents of one table into another. Avoid redundancy.

Problem Statement:

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 availablein the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Objective:

1. To learn and understand PL/SQL in Oracle.
2. To learn and understand cursors.

Hardware requirements:

- Any CPU with Pentium Processor or similar, 256 MB
- RAM or more, 1 GB Hard Disk or more.

Software requirements:

Windows 7 Operating System, Oracle 11g, SQL developer

Theory:

Oracle creates a memory area, known as the context area, for processing an SQL statement, which contains all the information needed for processing the statement; for example, the number of rows processed, etc.

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

You can name a cursor so that it could be referred to in 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

Implicit Cursors

Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control theimplicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursoris associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, we can refer to the most recent implicit cursor as the SQL cursor, which always has attributes such as %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK_ROWCOUNT and %BULK_EXCEPTIONS,designed for use with the FORALL statement. The following table provides the description of the most used attributes −

S.No	Attribute & Description
1	%FOUND Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, itreturns FALSE.
2	%NOTFOUND The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
3	%ISOPEN Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
4	%ROWCOUNT Returns the number of rows affected by an INSERT, UPDATE, or DELETE

	statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as sql%attribute_name as shown below in the example.

Example

```
Select * from customers;



+......+..............+.......+...............+..............+

|  1 |            |  32 | Ahmedabad |
Ramesh            |  25 | Delhi        |  1500.00
                                       |
| 2 | Khilan      |  23 | Kota
                                       |  2000.00
|  3 |            |  25 |              |
kaushik           Mumbai
+......+..............+.......+...............+..............+
```

The following program will update the table and increase the salary of each customer by 500and use the SQL%ROWCOUNT attribute to determine the number of rows affected −

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE customers
  SET salary = salary + 500;
  IF sql%notfound THEN dbms_output.put_line('no
    customers selected');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers selected ');
  END IF;
END;
/
```

Result −

```
6 customers selected

PL/SQL procedure successfully completed.
```

If you check the records in customers table, you will find that the rows have been updated −

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
| 1 | Ramesh   | 32 | Ahmedabad |  2500.00 |
|  2 | Khilan    | 25  | Delhi      |  2000.00 |
|  3 | kaushik   | 23 | Kota       |  2500.00 |
|  4 | Chaitali  | 25 | Mumbai     | 7000.00 |
|  5 | Hardik    | 27 | Bhopal     | 9000.00 |
|  6 | Komal     | 22 | MP        |  5000.00 |
+----+----------+-----+-----------+----------+
```

Explicit Cursors

Explicit cursors are programmer-defined cursors for gaining more control over the context area. An explicit cursor 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.

The syntax for creating an explicit cursor is −

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor includes the following steps –

Declaring the cursor for initializing the memory

- Opening the cursor for allocating the memory
- Fetching the cursor for retrieving the data
- Closing the cursor to release the allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement.For example −

```
CURSOR c_customers IS
   SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example,

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example,

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example,

```
CLOSE c_customers;
```

Example

Example of explicit cursor:

```
DECLARE
  c_id customers.id%type;
  c_name customerS.No.ame%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;
/
```

Conclusion: Thus we have successfully implemented PL/SQL block to retrieve fine for issued library book byreading borrower information from the database.

Assignment No: 7

Class: T.E. Computer **Name:**
Roll No:

Aim: Write a PL/SQL block to create trigger on Library table to keep track of updation and deletion of records.

Problem Statement:

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 ofthe records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

Objective :

1. To learn and understand PL/SQL in Oracle.
2. To learn and understand triggers.

Hardware requirements:

- Any CPU with Pentium Processor or similar, 256 MB
- RAM or more, 1 GB Hard Disk or more.

Software requirements:

Windows 7 Operating System, Oracle 11g, SQL developer

Theory:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the followingevents −

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event isassociated.

Benefits of Triggers

Triggers can be written for the following purposes −

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

Types of PL/SQL Triggers

- There are two types of triggers based on the which level it is triggered.
  1) Row level trigger - An event is triggered for each row upated, inserted or deleted.
  2) Statement level trigger - An event is triggered for each sql statement executed.

Creating Triggers

The syntax for creating a trigger is −

```
CREATE [OR REPLACE ] TRIGGER trigger_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col_name]

ON table_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

   Declaration-statements

BEGIN

   Executable-statements

EXCEPTION
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger_name − Creates or replaces an existing trigger with the trigger_name.
- {BEFORE | AFTER | INSTEAD OF} − This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} − This specifies the DML operation.
- [OF col_name] − This specifies the column name that will be updated.
- [ON table_name] − This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] − This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] − This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) − This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

Example

```
Select * from customers;

 ---- ---------- ----- ------------------ ----------

+____+__________+_____+__________________+__________+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
   2 | Khilan      25 | Delhi     |
|  3 | kaushik  |                        1500.00 |
   4 | Chaitali |  23 | Kota      |
|  5 | Hardik   |                        2000.00 |
   6 | Komal       25 | Mumbai    |
|                                        6500.00 |
+----+----------+-----+------------------+----------+
```

The following program creates a row-level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values −

```
CREATE OR REPLACE TRIGGER display_salary_changes

BEFORE DELETE OR INSERT OR UPDATE ON customers
```

```
FOR  EACH  ROW
WHEN (NEW.ID > 0)
DECLARE

   sal_diff number;
BEGIN

   sal_diff := :NEW.salary - :OLD.salary;
   dbms_output.put_line('Old salary: ' || :OLD.salary);
   dbms_output.put_line('New salary: ' || :NEW.salary);
   dbms_output.put_line('Salary difference: ' || sal_diff);

END;
```

When the above code is executed at the SQL prompt, it produces the following result –

```
Trigger created.
```

The following points need to be considered here –

OLD and NEW references are not available for table-level triggers, rather you can use them for record-level triggers.
The above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using the DELETE operation on the table.

Triggering a Trigger

Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table –

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Kriti', 22, 'HP', 7500.00 );
```

When a record is created in the CUSTOMERS table, the above createtrigger, **display_salary_changes** will be fired and it will display result –

```
Old salary:

New salary: 7500
Salary difference:
```

Trigger for Library updation and deletion:

```
CREATE OR REPLACE TRIGGER BOOKS_AUDIT
BEFORE DELETE OR UPDATE ON library
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH
ROWBEGIN

INSERT INTO library_audit
   VALUES
   ( :old.id,
     :old.book,
     :old.author
      ,sysdate);
END;
```

Conclusion: Thus we have successfully implemented trigger to keep track of update and delete operationperformed on library table.

Assignment No:8

Class: T.E. Computer **Name:**
Roll No:

Aim: Implement MYSQL/Oracle database connectivity with PHP/ python/Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

Description:

Two-Tier Architecture : The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.For example now we have a need to save the employee details in database.
The two tiers of two-tier architecture is:

1. Database (Data tier)
2. Client Application (Client tier)

* Steps for connection with database :

1.Register driver with Driver Manager.
Class.forName(”com.mysql.jdbc.Driver”);
2.Create connection with database.
Connection conn=DriverManager.getConnection(url,username,password);
3.Submit statement to a database.
Statement start=conn.CreateStatement();
Result res=start.executeQuery(select * from Account);
4.Process the result.

```
while(res.next())
{ (System.out.println(res.getInt(”I }
  D”));
```

5.Close the connection.

* Procedure :

1. **Syntax for create :**

create table <tablename>(in/out <variable name>type,in/out <variable name> type...);

begin statement 1;
statement 2;
...
Statement
n end
Example :

create table Account(in id int,in ac type int,acc balance int);

end

1. **Syntax for insert :**

insert into <tablename>values(value1,value2,value3...);
begin
statement 1;
statement 2;
...
Statement
n end
Example :

insert into Account values(1,'Saving',1000); end

2. Syntax for update :

update <tablename> set column1=value1,column2=value2,... where some column=some value; begin
statement 1;
statement 2;
...
statement
n end
Example :

update Account set ac balance=ac balance+amt where ac type='saving';
update Account set ac balance=ac balance-amt where ac type='current';

3. **Syntax for delete** :

delete from <tablename> where some column=some value;
begin
statement 1;
statement 2;
...
statement n; end

Example :

delete from Account where id=1;
delete from Account where acc type='current';

4. **Syntax for select :**

select column name,column name from <tablename> where id=value; AND
select * from <tablename>; begin
statement 1;
statement 2;
...
statement n end

Example :

select Account.id,Account.acc type from Account where id=1;
select * from Account;

Conclusion: Thus the JAVA Database connectivity concepts implemented using JDBC-ODBC

Group B

Assignment No: 9(A)

Class: T.E. Computer **Name:**
Roll No:

Aim: To study MongoDB and Implement CRUD Operations.\

Problem Statement:

Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution)

Objective:

1. To learn and understand NOSQL Database.
2. To execute CRUD Operations on MongoDB.

Hardware requirements:

- Any CPU with Pentium Processor or similar, 256 MB
- RAM or more, 1 GB Hard Disk or more.

Software requirements:

Ubuntu 14.04, Mongodb Packages

Theory:

MongoDB is a free and open-source NoSQL document database used commonly in modern web applications. MongoDB works on concept of collection and document.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. Acollection exists within a single database. Collections do not enforce a schema.

Document

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

CRUD Operations:

CRUD operations *create*, *read*, *update*, and *delete* documents.

These operations are considered to be the four basic functionalities of a repository .

CRUD operations can be mapped directly to database operations:

- Create matches insert
- Read matches select
- Update matches update
- Delete matches delete

Create Operations

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

```
db.collection.insert()
```

Syntax:

```
db.collection.insert(
   <document or array of documents>,
   {
     writeConcern: <document>,
     ordered: <boolean>
   }
)
```

e.g.

1) db.products.insert({ item: "card", qty: 15 })

Will store record(document) as

```
{ "_id" : ObjectId("5063114bd386d8fadbd6b004"), "item" : "card", "qty" : 15 }
```

2) db.products.insert({ _id: 10, item: "box", qty: 20 })

Will store record (document) as

```
db.products.insert( { _id: 10, item: "box", qty: 20 } )
```

Read Operations

Read operations retrieves documents from a collection; i.e. queries a collection for documents.MongoDB provides the following methods to read documents from a collection:

- db.collection.find()

You can specify query filters or criteria that identify the documents to return.

Examples

Find All Documents in a Collection

The find() method with no parameters returns all documents from a collection and returns all fieldsfor the documents. For example, the following operation returns all documents in the bios collection:

```
db.bios.find()
```

Find Documents that Match Query Criteria

To find documents that match a set of selection criteria, call find() with the <criteria> parameter. The following operation returns all the documents from the collection products where qty is greaterthan 25:

```
db.products.find( { qty: { $gt: 25 } } )
```

Update Operations

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(SELECTION_CRITERIA, UPDATED_DATA)
```

Example

Consider the mycol collection has the following data.

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

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

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

By default, MongoDB will update only a single document. To update multiple documents, youneed to set a parameter 'multi' to true.

```
>db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

Delete Operation:

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)
```

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"}
>
```

Remove Only One

If there are multiple records and you want to delete only the first record, then set justOne parameter in remove() 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()
>
```

Conclusion:

Assignment No: 9(B)

Class: T.E. Computer **Name:**
Roll No:

Aim: Develop MongoDB Queries using SAVE & Logical Operators.

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

Objective :

1. To learn and understand NOSQL Databas..
2. To execute CRUD Operations using SAVE Method & Logical Operators.

Hardware requirements:

- Any CPU with Pentium Processor or similar, 256 MB
- RAM or more, 1 GB Hard Disk or more.

Software requirements:

Ubuntu 14.04, MongoDB Packages.

Theory:

MongoDB Save() Method

he db.collection.save() method is used to updates an existing document or inserts a new document, depending on its document parameter.

Syntax

The basic syntax of MongoDB save() method is shown below −

```
>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":"Test1",
        "by":"JIT"
   }
)
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New Topic",
   "by":"DBMSTutor"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Test1",”by” : ”JIT”}
>
```

- save()method performs an insert since the document passed to the method does not contain the_idfield. During the insert, the shell will create the _id field with a unique ObjectId value, as verified by the inserted document.
- save()performs an update withupsert:true since the document contains an_idfield.

e.g. db.invoice.save({ _id: 1001,inv_no: "I00001", inv_date: "10/10/2012", ord_qty:200 });

Logical Operators in MongoDB:

Name	**Description**
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause

e.g.

1) AND Queries With Multiple Expressions Specifying the Same Field

```
db.inventory.find( { $and: [ { price: { $ne: 1.99 } }, { price: { $exists: true } } ] } )
```

This query will select all documents in the inventory collection where:

- the price field value is not equal to 1.99 and
- the price field exists.

2) AND Queries With Multiple Expressions Specifying the Same Operator

```
db.inventory.find( {

   $and : [

      { $or : [ { price : 0.99 }, { price : 1.99 } ] },

      { $or : [ { sale : true }, { qty : { $lt : 20 } } ] }

   ]

} )
```

This query will select all documents where:

- the price field value equals 0.99 or 1.99, and
- the sale field value is equal to true or the qty field value is less than 20

Conclusion:

Assignment No: 10

Class: T.E. Computer **Name:**
Roll No:

Title - Aggregation and indexing with suitable example using MongoDB.

Pre-requisites:- Version of 64 Bit Operating Systems Open Source Fedora-19 or Higher equivalent with LAMP tools,8 GB RAM, 500GB/1TB HDD, Latest versions of64-Bit Programming languages such as Microsoft VisualStudio(ver. 12 or Higher),equivalent with latest versions of 64-bit Databases Oracle MySQL, MongoDB, CouchDB.

Objective:-

1) To develop Database programming skill.
2) To develop use data storage devices and related programming and Management skills.
3) Learn the how to reduce data using database mongodb.
4) Problem solving using advanced databases techniques and tools.
5) Ability to handle and programming of storage devices.

Theory :

1) Aggregation :-

1) Aggregations are operations that process data records and return computed results. MongoDB providesa rich set of aggregation operations that examine and perform calculations on the data sets. Running dataaggregation on the mongodb instance simplifies application code and limits resource requirements.
2) Like queries, aggregation operations in MongoDB use collections of documents as an input and returnresults in the form of one or more documents.

Single Purpose Aggregation Operation:-

1) Aggregations are operations that process data records and return computed results.
2) MongoDB provides a rich set of aggregation operations that examine and perform calculations on thedata sets.
3) Running data aggregation on the mongodb instance simplifies application code and limits resource re-quirements.
4) Like queries, aggregation operations in MongoDB use collections of documents as an input and return results in the form of one or more documents Diagram of the annotated distinct operation.

Additional Features:-

1) Both the aggregation pipeline and map-reduce can operate on a sharded collection.

2) Map-reduce operations can also output to a sharded collection.

3) The aggregation pipeline can use indexes to improve its performance during some of its stages.

4) In addition, the aggregation pipeline has an internal optimization phase.
5) For a feature comparison of the aggregation pipeline, map-reduce, and the special group functionality,see Aggregation Commands Comparison.

Syntax:-

$$db.Teacher1_info.Aggregate(group : _id : "Dept_name", Total : sum : 1)$$

$$As[k] < -A[n], \ WhereAs[k]isasublistobtainedafterapplyingoperationAggregate.$$

$$sum = count|ConditionedMatched.$$

$$sum = 0|ConditinedisnotMatched.$$

(2)Index:-

(1) Without indexes, MongoDB must scan every document in a collection to select those documents thatmatch the query statement. These collection scans are inefficient because they require mongod to processa larger volume of data than an1 index for each operation.
(2) Indexes are special data structures that store a small portion of the collections 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.
(3) Fundamentally, indexes in MongoDB are similar to indexes in other database systems. MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents ina MongoDB collection.
(4) If an appropriate index exists for a query, MongoDB can use the index to limit the number of doc- uments it must inspect. In some cases, MongoDB can use the data from the index to determine which documents match a query.

Index Types:-

(1) Default:-
If applications do not specify a value for id the driver or the mongodb will create an id field with an ObjectId value.
The id index is unique, and prevents clients from inserting two documents with the same value for the idfield.
(2) Single Field:-
In addition to the MongoDB-defined id index, MongoDB supports user-defined indexes on a single fieldof a document.
(3) Compound Index:-
MongoDB also supports user-defined indexes on multiple fields. These compound indexes behave like single-field indexes; however, the query can select documents based on additional fields. The order of fields listed in a compound index has significance. For instance, if a compound index consists of userid:1, score: -1 , the index sorts first by userid and then, within each userid value, sort by score.

Index Properties:-

1) Unique Indexes-
The unique property for an index causes MongoDB to reject duplicate values for the indexed field.
To create a unique index on a field that already has duplicate values, see Drop Duplicates for index creation options.
2) Sparse Indexes-

The sparse property of an index ensures that the index only contain entries for documents that have the indexed field.

The index skips documents that do not have the indexed field.

Create An Index(Syntax):-

1) Single Indexes:

To create an index, use ensureIndex() or a similar method from your driver. The ensureIndex() method only creates an index if an index of the same specification does not already exist.

For example, the following operation creates an index on the userid field of the records collection:

$$db.Teacher1info.ensureIndex(”name” \ : \ 1)$$

2) Compound Index:-

To create a compound index use an operation that resembles the following prototype:

$$db.Teacher1_info.ensureIndex(”name” \ : \ 1,^{JJ} \ name^{JJ} \ : \ 2)$$

The value of the field in the index specification describes the kind of index for that field. For example, a value of 1 specifies an index that orders items in ascending order. A value of -1 specifies an index that orders items in descending order.

3) Unique Index:-

MongoDB allows you to specify a unique constraint on an index. These constraints prevent applications from inserting documents that have duplicate values for the inserted fields. Additionally, if you want to create an index on a collection that has existing data that might have duplicate values for the indexed field, you may choose to combine unique enforcement with duplicate dropping.

$$db.Teacher1_info.ensureIndex(”name” \ : \ 1, unique \ : \ true);$$

Get Indexes:-

To view the name of an index, use the getIndexes() method is used.

$$db.Teacher1_info.getIndexes();$$

Drop Indexes:-

To force the creation of a unique index index on a collection with duplicate values in the field you are indexing you can use the dropDups option. This will force MongoDB to create a unique index by deletingdocuments with duplicate values when building the index. Consider the following prototype invocation of ensureIndex():

$$db.Teacher1_info.dropIndex(”name_1”);$$

Mathematical Model :

1) D : Document
collect of fields

D = { name, id, status, dept_name, sal }

2) C [n] collection of document D

3) A [ ] $\xleftarrow{aggregate}$ C [ ]

aggregate 1)
db.<collection_name>.aggregate
({ $group : { _id : ”$dept_name”,Total: { $sum : / } });
where
A [ ] = collection of document dd
= { id,_total }

Total = 1 dept_name present at single time
= 2...n dept_name present more than 1 time.

aggregate 2)
db.<collection_name>.aggregate
($group : { _id: : ”‘$dept_name”’,total : { $sum: $sal } });
where sal = salary of matched dept_name document
Total = sal dept name present only single time.
= sal1 + sal2 + ...sal k ,where k = number of times dept_name is repeated.

4) indexing :
Ci [ ] $\xleftarrow{index}$ c [ ]

db.<collection_name>.ensureindex(
attr{ name : 1 -1 })
1 = assending
2 = descending
Ci [ ] −→ collection of document d.d=
{ attr_name}

5) drop index :
db.<collection_name>.dropindex(”‘index_name”’)
dropindex

sys msg dropindex ←− C [ ]where
sys msg = ok name of index exist
= fail name of index does not exist.

Conclusion : Hence we implemented the aggregation and indexing successfully .

Assignment No: 11

Class: T.E. Computer
Name:

Roll

No:

Title: Map reduce operation with suitable example using MongoDB.

Objective:-

1) To develop Database programming skill
2) To develop Operating Systems programming and administrative skills
3) To develop use data storage devices and related programming and Management skills4) Learn the how to reduce data using database mongodb.

Pre-Requisites:-

Version of 64 Bit Operating Systems Open Source Fedora-19 or Higher equivalent with LAMP tools,8 GB RAM, 500GB/1TB HDD, Latest versions of64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher),equivalent with latest versions of 64-bit Databases Oracle MySQL, MongoDB,CouchDB.

Theory :

1) Map reduce :-

Map-reduce is a data processing paradigm for condensing large volumes of data

into useful aggregated results. For map-reduce operations, MongoDB provides the map Reduce database command.
In MongoDB, map-reduce operations use custom JavaScript functions to map, or associate, values to a key. If a key has multiple values mapped to it, the operation reduces the values for the key to a single object.
In MongoDB, the map-reduce operation can write results to a collection or return the results inline. If youwrite map-reduce output to a collection, you can perform subsequent map-reduce operations on the same input collection that merge replace, merge, or reduce new results with previous results. MongoDB sup- ports map-reduce operations on sharded collections. Map-reduce operations can also output the results to a sharded collection. See Map-Reduce and Sharded Collections.

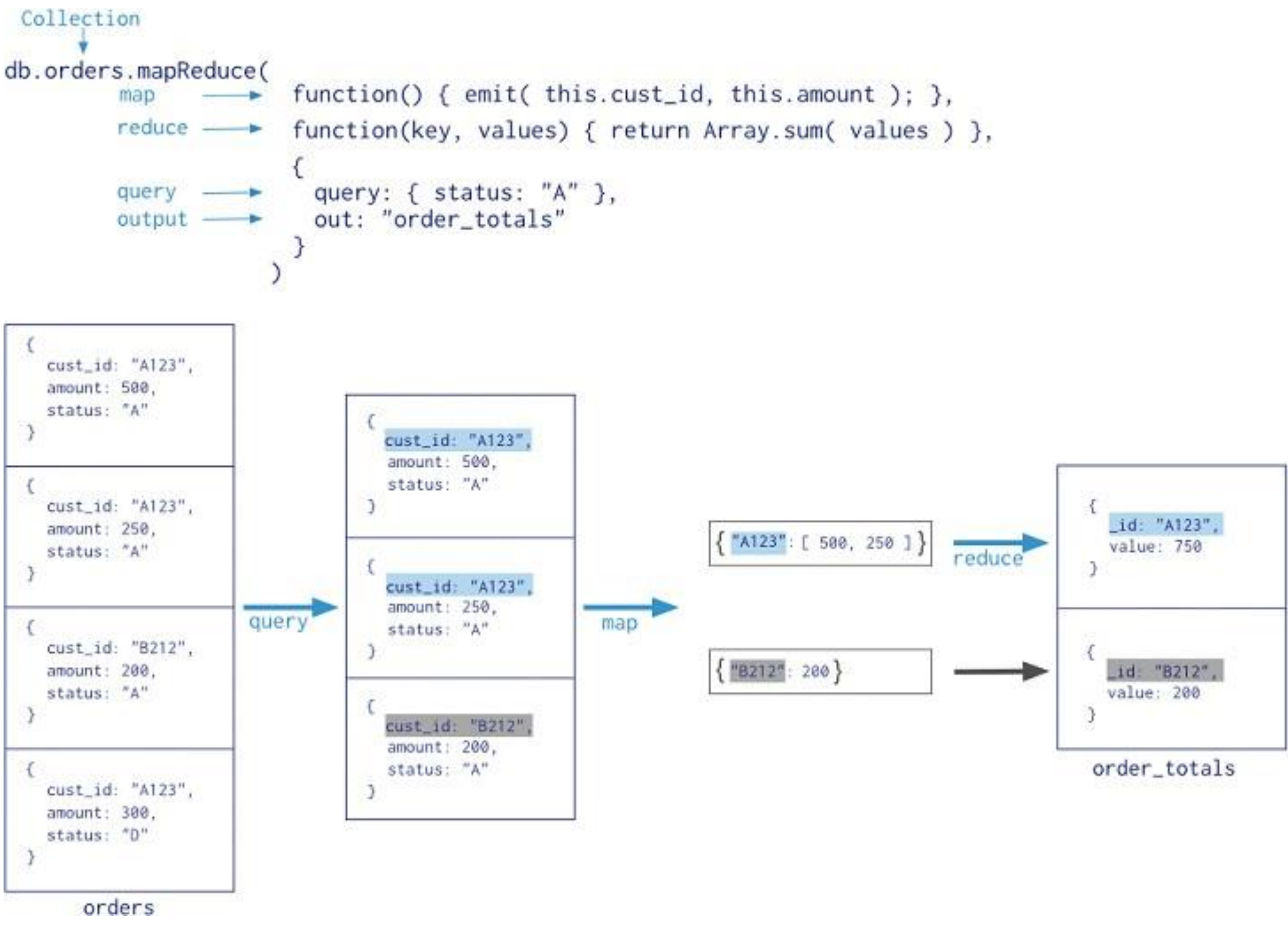
Syntax:-

$db.orders.mapreduce(function()\{emit(this.cust_id, this.amount); \}, function(Key, values)\{returnarray.sum$

$$\{query : \{Status :^{JJ} A^{JJ}\}, out :^{JJ} order_totals^{JJ}\})$$

Consider the following map-reduce operation:

Example:- Diagram of the annotated map-reduce operation. In this map-reduce operation, MongoDB applies the map phase to each input document. The map function emits key-value pairs. For those keys that have multiple values, MongoDB applies the reduce phase, which collects and condenses the aggre- gated data. MongoDB then stores the results in a collection. Optionally, the output of the reduce functionmay pass through a finalize function to further condense or process the results of the aggregation.All map-reduce functions in MongoDB are JavaScript and run within the mongod process. Map-reduce op-erations take the documents of a single collection as the input and can perform any arbitrary sorting andlimiting before beginning the map stage. mapReduce can return the results of a map-reduce operation asa document, or may write the results to collections. The input and the output collections may be sharded.



```
db.orders.mapReduce(
                 function() { emit( this.cust_id, this.amount ); },
                 function(key, values) { return Array.sum( values ) },
                 {
                   query: { status: "A" },
                   out: "order_totals"
                 }
               )
```

Collection
map
reduce
query
output
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
query
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
map
{ "A123": [500, 250] }
{ "B212": 200 }
reduce
{ _id: "A123", value: 750 }
{ _id: "B212", value: 200 }
order_totals

1 Mathematical Model:

1) $R[K] \xrightarrow{map\ reduce} c[n]$

where c[n] = collection consisting n

number of document (D)D = { cust

_

id, status, balance}

K < n

2) $C^* [K] \xleftarrow{map} c[\,]$

K = 1 to n field name = field value

= 0 field name /= field value

3) $E[h] \xleftarrow{emit} C * [k]$

E [h] consist of document with key value pwr which is

mentioned in emit function.D = { key : [<multiple values>

]} key is repeated.

{ key : [<single value>]} key is not repeated.

4) $R[k] \xleftarrow{reduce} E[h]k = h$

where R [k] consist of k no of document D.

D = { key , val1 + val2 + ...} D= { key :[multiple values] }

{ key, val} D = { key : [single value] }

Conclusion:- The framework breaks up large data into smaller parallelizable chunks and handles scheduling. Maps each piece to an intermediate value Reduces intermediate values to a solution. User-specified partition and combiner options.

Assignment No: 12

Class: T.E. Computer **Name:**
Roll No:

Title : Connectivity with MongoDB using any Java application.

Objective :

Using mongodb connectivity java application.

Theory :

What is MongoDB ?

MongoDB is an open-source document database that provides high perfor-mance, high availability, and automatic scaling.

Database

A record in MongoDB is a document, which is a data structure composed offield and value pairs. MongoDB documents are similar to JSON objects.

A MongoDB document.
The advantages of using documents are:

Documents (i.e. objects) correspond to native data types in many program-ming languages. Embedded documents and arrays reduce need for expensive joins. Dynamic schema supports fluent polymorphism.

Key Features High

Performance

MongoDB provides high performance data persistence. In particular, Support for embedded data models reduces I/O activity on database system. Indexes support faster queries and can include keys from embedded doc-uments and arrays. High Availability To provide high availability, MongoDBs replication facility, called replicasets, provide:

automatic failover.

data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability. Automatic Scaling

MongoDB provides horizontal scalability as part of its core functionality. Automatic sharding distributes data across a cluster of machines.

Replica sets can provide eventually-consistent reads for low-latency highthroughput deployments.

Steps in creation document and insert in mongodb :

MongoDB will create a collection implicitly upon its first use. You do not need to create a collection before inserting data. Furthermore, because Mon- goDB uses dynamic schemas, you also need not specify the structure of yourdocuments before inserting them into the collection.

1. From the mongo shell, confirm you are in the mydb database by issu- ing the following:
2. db
3. If mongo does not return mydb for the previous operation, set the contextto the mydb database, with the following operation:
4. use mydb
5. Create two documents named j and k by using the following sequence of JavaScript operations:
6. j = name : ”mongo”
   7. k = x : 3
7. Insert the j and k documents into the testData collection with the follow- ing sequence of operations:
8. db.testData.insert(j)

9. db.testData.insert(k)
10. When you insert the first document, the mongod will create both the mydb database and the testData collection
11. . 11. Confirm that the testData collection exists. Issue the following opera- tion:
12. show collections

    a. The mongo shell will return the list of the collections in the current (i.e. mydb) database. At this point, the only collection is testData. All mongod databases also have a system.indexes collection.

13. Confirm that the documents exist in the testData collection by issu-ing a query on the collection using the find() method:
14. db.testData.find()

This operation returns the following results. The ObjectId values will beunique: ” id” : ObjectId(”4c2209f9f3924d31102bd84a”), ”name” : ”mongo”” id” : ObjectId(”4c2209fef3924d31102bd84b”), ”x” : 3

Functions used

Insert

The following table presents the various SQL statements related to inserting records into tables and the corresponding MongoDB statements.

To add a file first create an object of BasicDBObject and then insert into collection

```
    BasicDBObject doc = new BasicDBObject(”name”, name) .append( ”,”)
.append( ”, ”);
    coll.insert(doc);
```

Mathematical model :

1) D = Document

D = {name, ID, age }

2) C [n] = Collection consistency

documents.3)$C_*[\ ] \xleftarrow{insert} C[\ ]$

where c * [ ] is a collection obtained after operation insertion.

4) find() :

db.collection_name.find()

$c[n] \xleftarrow{find} c[n]$

Conclusion : Hence we successfully implemented program.

Group C
Mini Project

Title of Project:

Group Members: (Name with Roll no.)

Front End Used: (JAVA/PHP/PYTHON/PERL/RUBY/Any other)

Database Used: (MongoDB/MySQL/Oracle)

Abstract (10 lines):

Introduction (20 Lines):

ER Diagram:

Relational Model:

Screenshots of Output:

Testing Results:

Conclusion: