Date

Expt.No. ----------

Page No. ----------

Name – Ritu Bachle

Subject – DBMS (practical)

Subject code – CS-(308)

Semester – 3rd Sem.

Branch – CSE

Teacher's Signature :

INDEX

Kay-Bee

Sl. No.	Name of Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
1.	Create tables and Specify the Given Queries in SQL	2	25/10/21		
2.	To Manipulate the Operation on the table.	4	25/10/21		
3.	To Implement the restrictions on the table	6	1/11/21		
4.	To Implement the structure of the table.	10	1/11/21		
5.	To implement the Concept of joins	12	8/11/21		
6.	To implement the concept of grouping of Data.	14	8/11/21		
7.	To implement the concept of SubQuestionecies	15	22/11/21		
8.	To implement the concept of Indexes and views.	17	29/11/21		
9.	To implement the basics of PL/SQL	20	6/12/21		

INDEX

Kay-Bee

Sr. No.	Name of Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
10.	To implement the concept of Cursor and Trigger	23	20/12/21		

Experiment No. 1

Objective :- Create table and specify the Queries in SQL.

Introduction about SQL - SQL (Structed Questionary language) is an nonprocedural language, A block structed format of English key words is used in this Questionry language.

DDL (Data Definition language) - The SQL DDL provides command for defining relation schemas, deleting relations and modifying relation schema.

DML (Data Manipulation Language) - It includes commands to insert tuples into, delete tuples from and modify tuples in the database.

DDL statement for creating a table.

Syntax -

Create table tablename
Columnname datatype (size), Columnname datatype (size));

Creating a table from a table.

Input

Create table student (Roll_num Int, Name varchar(30) Branch varchar(30), Semester Int);

Output - 1

Select * from student

Rollnum	Nam	Branch	Semester

Experiment - 2

Objective :-
To Manipulate the Operations on the table.

DML (Data Manipulation language) Data manipulation is

- The retrieval of information stored in the database.
- The insertion of new information into the database.
- The deletion of information from the database.
- The modification of information stored by the appropriate data model. There are basically two types.

(i) Procedural DML :- require a user to specify what data are needed ~~without~~ and how to get those data.
(ii) Non procedural DML :- require a users to specify what data are needed without specifying how to get those data.

Updating the content of a table :

In creation situation we may wish to change a value in table without changing all values in the tuple. For this purpose the update statement can be used.

Update table name

Set columnname = expression, columnname = expression....
where columnname = expression;

Input

1). Select

Select * from student;

2). Insert

Insert into student values (104, "Ram", "CSE", 3);

3). Update

Update student set Name = "Raman",
where Roll_Num = 103;

4). Delete

Delete From student where
Roll_num = 104;

Experiment Name .....

Output

1).

Roll_num	Name	Branch	Semester
101	Raj	CSE	3
102	Rahul	CSE	3
103	Karan	CSE	3

2).

Roll_num	Name	Branch	Semester
101	Raj	CSE	3
102	Rahul	CSE	3
103	Karan	CSE	3
104	Ram	CSE	3

3).

Roll_num	Name	Branch	Semester
101	Raj	CSE	3
102	Rahul	CSE	3
103	~~Karan~~ Raman	CSE	3
104	Ram	CSE	3

4).

Roll_num	Name	Branch	Semester
101	Raj	CSE	3
102	Rahul	CSE	3
103	Raman	CSE	3

Teacher's Signature : _____

Experiment No.- 3

Objective :- To Implement the restrictions on the table.

Data Constraints : Besides the cell name, cell length and cell data type there are other parameters i.e., other data constreains that can be passed to the DBA at check creation time. The constraints can either be placed at column level or at the table level.

i). Column level Constraints : If the constraints are defined along with the column definition, it is called level constraint.

ii). Table Level Constraints : If the data constraint attached to a specify cell in a table reference, the contents of another cell in the table then the user will have to use table level constraints.

Null Value Concepts : While creating table if a row lacks a data value for particular column that value is said to be null. Column of any data type may contain null values unless the column was defined as not null when the table was created.

1 Syntax:

```
Create table Student (
Name varchar (30) not null,
Branch varchar (30) not null,
Semester Int not null);
```

Primary key: primary key is one or more columns in a table used to uniquely identify each row in the table. Primary key value must not be null and must be uniQuestion across the column. A multicolumn primary key is called composite primary key.

2 Syntax: primary key as a Table constraint

```
Create table student (
Roll_No int, not Null,
Name varchar (30) not NULL,
Branch varchar (30) not NULL,
Semester int not NULL,
Primary key (Roll_No));
```

UniQuestion key concept :- UniQuestion is similar to a primary key except that the purpose of a uniQuestion key is to ensure that information in the column for each record is uniQuestion as with telephone or devices license numbers. A table may have many UniQuestion keys.

Experiment Name .....

MySQL > Desc Student ;

1).

Field	TYPE	NULL	Key	Default	Extra
Name	Varchar(30)	No		NULL	
Branch	varchar(30)	No		NULL	
Semester	Int	No		NULL	

2).

Field	Type	NULL	Key	Default	Extra
Roll_No	Int	No	PRI	NULL	
Name	Varchar(30)	No		NULL	
Branch	Varchar(30)	No		NULL	
Semester	varchar(30)	No		NULL	

Teacher's Signature : _____

3 Syntax :- Uniquestion as a column constraint

Create table student (Roll_No. int, Name varchar (30), Branch varchar(30), phone-no int, Unque key ('roll_no', 'phone_no'));

Default value Concept :- At the line of cell creation a default value can be assigned to it. When the user is loading a record with values and leves this cell empty, the DBA will automatically load this cell with the default value specified. The data type of the default value should match the data type of the column.

Syntax:
Create table student (Roll_no. int not NULL, name varchar (20) not NULL, Semester int NOT NULL Defaullt '3', Unique key ('roll_no'));

Foreign key Concept :- Foreign key represents relationship between table. A foregin key is column whos values are derived from the primary key of the same of some other table the existence of foregin key implies that the table with foreign key is related to the primary key table from which the foreign key is derived. A foregin key must have corresponding primary key value in the

Experiment Name .....

9).

Field	TYPE	NULL	KEY	Default	Extra
Roll_no	int	No	UNI	NULL	
Name	varchar (30)	No		NULL	
Branch	varchar(30)	No		NULL	
Phone_No	int	No	UNI	NULL	

Teacher's Signature :_____

primary key table to have meaning.

Foreign key as a column constraint

Syntax:

Table-1 Create table student (roll_no int not NULL, name varchar(20) NOT NULL, sem int not NULL, primary key ('roll-no'));

Table-2 Create table student_data (roll_no. int NOT NULL, Branch varchar(30), foreign key (roll-no) references student (roll-no));

Experiment Name .....

mysql> Desc student;

Field	Type	NULL	Key	Default	Extra
roll_no	int	No	UNI	NULL	
name	varchar(20)	No		NULL	
Semester	Int	No		3	

Table-1 >> Student

Field	TYPE	NULL	Key	Default	Extra
roll_no	int	No	PRI	NULL	
Name	varchar(20)	No		NULL	
Sem	int	No		NULL	

Table-2 >> Student_data

Field	Type	NULL	Key	Default	Extra
roll_no	int	No	MUL	NULL	
Branch	Varchar (30)	No		NULL	

Teacher's Signature :

Experiment No - 4

Objective :- To Implement the structure of the table

Modifying the structure of Tables - Alter table command is used to changing the structure of a table. Using the alter table clause you cannot perform the following tasks :

(i) Change the name of table
(ii) change the name of column
(iii) drop a column
(iv) decrease the size of a table if table exists.

1. Syntax:

Create table student (roll_No int not NULL, Name varchar(30) Not NULL, Branch varchar(30) NOT NULL, Sem int NOT NULL);

Add new columns;

2. Syntax

Alter tables student ADD (Branch varchar(30) not NULL);

Experiment Name .....

MySQL> Desc student;

1).

Field	Type	Null	Key	Default	Extra
roll_no	int	No		NULL	
Name	varchar(30)	No		NULL	
Sem	int	No		NULL	

2).

Field	Type	Null	Key	Default	Extra
roll_no	int	No		NULL	
Name	varchar (30)	No		NULL	
Sem	int	No		NULL	
Branch	Varchar(30)	No		NULL	

Teacher's Signature : _____

Removing / Deleting Tables:

3. Syntax:

DROP Table student;

4 Add Primary key.

Syntax:

ALTER Table student Add primary key (roll-no);

Add Foreign key.

Syntax:
ALTER Table student add constraint Constraintname
Foreign key (columnname) References table name;

5. DROP the Primary key
Syntax:
ALTER Table student Drop Primary key;

Drop Foreign key
Syntax
ALTER Table tablename
DROP CONSTRAINT Constraintname;

Experiment Name .....

3)

Empty Set

4).

Field	Type	NULL	KEY	Default	Extra
roll_no	int	No	PRI	NULL	
Name	varchar(30)	No		NULL	
Sem	int	No		NULL	
Branch	varchar(30)	No		NULL	

5).

Field	Type	Null	Key	Default	Extra
roll_no	int	No		NULL	
name	vcorchar(30)	No		NULL	
class	int	No		NULL	
Sem	varchar(30)	No		NULL	

Empty Set

Teacher's Signature :_____

Experiment No - 5

Objective :- To implement the concept of Joins

Joint Multiple Table (Equi Join): Some time we requires to treat more than one table as though manipulate data from all the tables as though the tables were not separate object but one single entity. To achieve this we have to join tables Tables are join on column that have same data type and data with in tables.

The tables that have to be joined are specified in the FROM clauses and the joining attributes in the WHERE clause.

Algorithm for JOIN in SQL:

1. Cartesian product of tables (specified in the FROM clause)
2. Selection of rows that match (predicate in the WHERE clause)
3. Project column specified in the SELECT caluse

1. Cartesian product :

Consider two table student and course

```
Select B.*, P.*
FROM student B, course P;
```

2. Inner Join :

Cartesian product followed by selection

Experiment Name .....

Empty.

Table-1

Studen ID	Branch
101	CSE
102	CSE
103	CSE

Table-2

Name	Semester	Contact_No	DOB
A	3	4443628411	22/10/03
B	3	6633441 12	03/5/02
C	3	89890213 1	7/6/02

Inner join

> Select Table1. StudentID, Table2. Name, Table1, Branch from Table1 Inner join Table2 On Table1. DOB = Table2. DOB;

Output

StudentID	Name	Branch
101	A	CSE
102	B	CSE
103	C	CSE

Teacher's Signature :

```
Select B.*, P*
FROM Student B, Course P
WHERE B, Course # P, Course #;
```

3. Left Outer Join : left outer join = Cartesian product + selection but include rows from the left table which are unmatched pat pulls in the values of attributes belonging to the second table.

Exam:

```
Select B*, P*
FROM STudent B left join course P
ON B. Course # P. Course #;
```

4. Right Outer Join : Right outer join = Cartesian product + selection but include rows from right table which are unmatched which are unmatched

Exam:

```
Select B*, P*
FROM student B Right JOIN course P
B. Course # = P Course #;
```

5. FULL Outer join

Exam:

```
Select B*, P*
FROM Student B FULL JOIN course P.
On B. Course # = P Course #;
```

Experiment Name .....

Left Outer join :-

> Select Table1.StudentID, Table1.Branch, Table2.Name From table1 left outer join Table2 on Table1.DOB = Table2.DOB;

StudentID	Branch	Name
101	CSE	A
102	CSE	B
103	CSE	C

Right Outer join

> Select table2.Name, table1.StudentID, from table2 Right join Table1 on table2.DOB = table1 DOB;

Output:-

Name	StudentID
A	101
B	102
C	103

Full Outer Join

>Select table2.Name, Table1.StudentID From Table2 Full outer join Table1 ON Table2.DOB = Table1.DOB order By Table2.Name;

Output:-

Name	StudentID
A	101
B	102
C	103

Teacher's Signature :

Experiment No - 6

Object :- To implement the concept of grouping of Data.

Grouping Data From tables: There are circumstances where we would like to apply the aggregate function not only to a single set of tuples, but also to group of sets of tuples, we specify this wish in SQL using the group by clause. The attribute or attributes given in the group by clause are used to form group. Tuples with same value on all attributes in the group by clause are placed in one group.

Syntax:

```
Select class, sem from student
group by sem;
```

Syntax

```
Select columnname, columnname
From tablename Group By columnname.
Having searchcondition;
```

Experiment Name .....

mysql> select * from student;

roll_no	name	class	sem
1	Ram	8	3

After grouping

class	sem
8	3

Teacher's Signature :_____

Experiment No - 7

Objective :- To implement the concept of SubQueries.

SubQueries :- A SubQueries is a form of an SQL statement that appears inside another SQL statement. It also termed as nested Query. The statement containing a subQuery called a parent statement. The rows returned by the SubQuery are use by the following statement.

1. To insert records in the target table.
2. To create tables and insert records in this table
3. To update records in the target table
4. To create view
5. To provide values for the condition in the WHERE, HAVING IN, SELECT, UPDATE and DELETE statements.

1) Union clause

```
Select branch from table1 Union
Select roll-no from Table 2;
```

Experiment Name .....

Table - 1

name	Branch	roll-no
Poonam	CSE	NULL

Table - 2

roll-no	name	class	Sem
1	Ram	B	3

1) Union

Branch
CSE 1

Teacher's Signature :

2) Intersect clause

```
Select branch from table 1 intersect
Select roll_no from table 2;
```

3) Minus clause

```
Select branch from table 1
  Minus
Select roll_no from Table 2;
```

2) Intersect

Empty set (0.06 sec)

3). Minus

Branch
CSE

Experiment No - 8

Object :- To implement the Concept of Indexes and views.

Indexes :- An index is an ordered list of content of a column or group of columns in a table. An index created on the single column of the table is called simple index. when multiple table columns are included in the index it is called composite index.

Creating an Index for a table :-

Syntax (Simple)

```
CREATE INDEX index name
ON tablename (column name);
```

Composite Index :-

```
CREATE INDEX index name
ON tablename (columnname, columnname);
```

Creating an UniQuestion Index :-

```
CREATE UNIQUESTION INDEX indexfilename
ON tablename (columname);
```

Dropping Indexes :-

An Index can be dropped by using DROP INDEX

Syntax :-

```
DROP INDEX indexfilename;
```

Views :- Logical data is how we want to see the current data in our database. Physical data is how this data is actually placed in our database.
Views are masks placed upon tables. This allows the programmer to develop a method via which we can display predetermined data to users according to our desire.

Views may be created for the following reasons.

1. The DBA stores the views as a definition only. Hence there is no duplication of data.
2. Simplifies Questionries
3. Can be Questionried as a base table itself.
4. Provides data security.
5. Avoids data redundancy.

Creation of Views -

Syntax :-

```
CREATE VIEW viewname AS
SELECT columnname columnname
FROM tablename
WHERE columnname = expression list;
```

Renaming the columns of view:-

Syntax:-

```
CREATE VIEW viewname AS
SELECT newcolumnname.--
FROM tablename
WHERE columnname = expression list;
```

Selecting a data set from a view -

Syntax:-

```
SELECT columnname, columnname
FROM viewname.
WHERE search condition;
```

Destroying a view -

Syntax:-

```
DROP VIEW viewname;
```

Experiment No - 9

Objective :- To implement the basics of PL/SQL

Introduction :- PL/SQL bridges the gap between database technology and procedural programming languages. It can be thought of as a development tool that extends the facilities of Oracles SQL database language. Via PL/SQL you can insert, delete, update and retrieve table data as well as use procedural techniQuestions such as writing loops or branching to another block of code.

PL/SQL Block structure -

```
DECLARE
Declarations of memory variables used later
BEGIN
SQL executable statements for manipulating table data.
EXCEPTIONS
SQL and/or PL.SQL code to handle errors
END;
```

Displaying user Messages on the Screen - Any programming tool requires a method through which message can be displayed to the user.

dbms output is a package that includes a number of procedure and functions that accumulate information in buffer so that it can be retrieved later. These functions can also be used to display message to the user.

Conditional control in PL

put line: put a piece of information in the buffer followed by a end of line marker. It can also be used to display message to the user.

SET SERVER OUTPUT ON:

Example: Write the following code in the PL/SQL block to display message to user
DBMS_OUTPUT.PUT_LINE ('Display user message');

Conditional control in PL/SQL -

Syntax:-
IF < condition > THEN
< Action >
ELSEIF < condition >
< Action >

```
ELSE
< Action>
ENDIF ;
```

The WHILE LOOP :

Syntax :-

```
WHILE < condition>
LOOP
<Action>
END LOOP;
```

The FOR LOOP statement :

Syntax :-

```
FOR variable IN [REVERSE] start_ end
LOOP
< Action >
END LOOP ;
```

The GOTO Statement : The goto statement allows you to change the flow of control within a PL/SQL Block.

Experiment No - 10

Object :- To implement the concept of Cursor and Trigger.

Cursor - We have seen how oracle executes an SQL statement. Oracle DBA uses a work area for its internal processing. This work area is private to SQL's operation and is called a cursor. The data that is stored in the cursor is called the Active Data set. The size of the cursor in memory in the size required to hold the number of rows in the Active Data set.

How to Declare the Cursor :-

The General Syntax to create any particular cursor is as follows:-

Cursor < Cursorname > is SQL Statement ;

How to Open the Cursor :-

The General Syntax to Open any particular cursor is as follows:

Open Cursorname ;

Fetching a record From the Cursor :-

The fetch statement retrieves the rows from the active set to the variables one at a time. Each time a fetch is executed. The focus of the DBA ~~cour~~ cursor advances to the next row in the Active set.
One can make use of any loop structure (Loop-End loop along with while. For) to fetch the records from the cursor into variable one row at a time.

The General Syntax to Fetch the records from the cursor is as follows:

Fetch cursorname into variable 1, Variable 2, ——

Closing a Cursor :-

The General Syntax to close the cursor is as follows:-

Close < cursorname > ;

Database Triggers :-

Database triggers are procedures that are stored in the database and are implicitly executed (fired) when the contents of a table are changed.

Use of Database Triggers :- Database trigger support Oracle to provide a highly customized database management system. Some of the uses to which the database triggers can be put to customize management information in Oracle are as follows.

- A Trigger can permit DML statements against a table only if they are issued, during regular bussiness hours or on predetermined weekdays.
- A trigger can also be used to keep an audit trail of a table along with the operation performed and the time on which the operation was performed.
- It can be used to prevent invalid transactions.
- Enforce complex security authorizations.

Types of Triggers :- four types of triggers

1. Before Statement Trigger :- Before executing the triggering statement, the trigger action is executed.
2. Before Row Trigger :- Before modifying the each row affected by the triggering statement and before appropriate integrity constraints, the trigger is executed if the trigger restriction either evaluated to TRUE or was not included.
3. After statement Trigger :- After executing the triggering statement and applying any deferred integrity constraints, the trigger action is executed.
4. After row Trigger :- After modifying each row affected by the triggering statement and possibly applying appropriate

integrity constraints, the trigger action is executed for the current row if the trigger restriction either evaluates to TRUE or was not included.

Syntax For Creating Trigger :-

Create or replace Trigger <Triggername> {Before, After}
{Delete, Insert, Update} on <Table name> For Each
row when Condition
Declare
<Variable declarations>;
<Constant Declarations>;
Begin
<PL/SQL> Subprogram Body;
Exception
Exception PL/SQL block;
End;

How to Delete a Trigger

The Syntax for Deleting the Trigger is as follows:-

Drop Trigger <Triggername.>;