I N D E X

Name : Rohit Kumar Saket Class & Section : 3rd sem CSE

Subject : DBMS Roll No. : 2182300258

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Teacher's Sign./Remarks
1.	Create tables and specify the Given Queries in SQL.				
2.	To Manipulate the operations on the table.				
3.	To Implement the restrictions on the table.				
4.	To Implement the structure of the table.				
5.	To implement the Concept of joins				
6.	To implement the concept of grouping of Data.				
7.	To implement the concept of Subqueries.				

INDEX

S. No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Teacher's Sign./Remarks
8.	To implement the concept of indexes and views.				
9.	To implement the basics of PL/SQL				
10.	To implement the concept of cursor and Trigger.				

Exp No. 1

Date / /
Page No.
Shivalal

objective :- Create tables and specify the Queries in SQL.

Theory and concepts :-

SQL (Structured Quesing Language) is a nonprocedural language, you specify what you want, not how to get it.

which was to introduced by IBM in 1910.
which is used to communicate with database.

SQL is also called as Squal or CLT (Common language interface)
This only the language which can use to communicate with any RDBMS product.
It has the following components.

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 tuplets from and modify tuples in the database.

View definition :-
The SQL DDL includes commands for defining views.
Transaction control - SQL includes for specifying the beginning and ending of transactions.

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Embedded SQL and Dynamic SQL:-

Embedded and Dynamic SQL define how SQL statements can be embedded with in general purpose programming languages, such as C, C++, JAVA, COBOL, Pascal and Fortran.

Intergrity :-

The SQL DDL includes commands for specifying intergrity constraints that the data stored in the database must specify. Updates that violate integrity constaints are allowed.

Authorization :-

The SQL DDL includes commands for specifying acess rights to relations and views.

Data Definition Language:-

The SQL DDL allows specification of not only a set of relations but also information about each relation, including.

- Schema for each relation.
- The domain of values associated with each attribute.
- The integrity constaints.
- The set of indices to be maintained for each relation.
- The security and authorization information for each relation.
- The physical storage structure of each relation on disk.

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Domain types in SQL -

The SQL standard supports of built in domain types including -

- Char(n) - A fixed length character length string with user specified length.
- Varchar(n) - A variable character length string with user specified maximum length n.
- Int - An integer.
- Small integer - A Small integer.
- Numeric(p, d) - A Fixed point number with user defined precision
- Real, double precision - Floating point and double precision floating point numbers with machine dependent precision.
- Float(n) - A floating point number, with precision of at least n digits.
- Date - A calender date containing a (four digit) year, month and day of the month.
- Time - The time of day, in hours, minutes and seconds Eg. Time '09:30:00'.
- ~~Time - The Time of day, in hours, minutes and seconds Eg. Tim~~
- Number :- Number is used to store numbers (fixed or floating point).

DDL statement for Creating a table :-

Syntax -

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

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Creating a table from table -

Syntax -

```
CREATE TABLE TABLENAME
[(Columnname, columnname,.....)]
AS SELECT columnname, columnname..... from tablename;
```

Insertion of data into tables:-

Syntax -

```
INSERT INTO tablename
[(columnname, columnname, ....)]
values (expression, expression);
```

Inserting data into a table from another table :-

Syntax -

```
INSERT INTO tablename
SELECT columnname, columnname, .....
FROM tablename;
```

Insertion of selected data into a table from another table:-

Syntax -

```
INSERT INTO tablename
SELECT columnname, columnname....
FROM tablename
WHERE columnname = expression;
```

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Retrieving of data from the tables.

Syntax -

```
SELECT * FROM tablename;
```

The retrieving of specific column from a table:-

Syntax -

```
SELECT columnname, columnname
From tablename;
```

Elimination of duplicates from the select statement:

Syntax -

```
SELECT DISTINCT columnname, columnname
FROM tablename;
```

SELECT: Selecting a data set from table data:

Syntax -

```
SELECT columnname, columnname
From tablename
WHERE searchcondition;
```

Teacher's Signature .....

Exp No. 02

Date / /
Page No.
Shivalal

~~Objective:~~

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 and how to get those data.
(ii) Non procedural DML:- require a user 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 the table without changing all values in the tuple, For this purpose the update statement can be used.

Update table name
set columnname = experision, columnname = expression....
where columnname = expression,

Deletion operation:-
A delete request is expressed in much the same way as Questionry. we can delete whole tuple (row) we can delete values on only particulars attributes.

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Deletion of all rows :-

Syntax:
Delete from tablename;

Deletion of specified number of rows

Syntax:-
Delete from table name
where search condition;

Computation in expression lists used to select data

+	Addition	-	Subtraction
*	multiplication	**	exponentiation
/	Division	()	Enclosed operation

Renaming columns used with Expression lists: The default output column names can be renamed by the user if required

Syntax:
Select column name result columnname,
columnname result - columnname,
From table name;

Logical operators:
The Logical operators that can be used in sql sentence are -

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

AND all of must be included
OR any of may be included
NOT none of could be included

Range Searching:- Between operation is used to range searching.

Pattern searching:-
The most commonly used operation on string is pattern matching using the operation 'like' we describe by using two special character.
- percent (%) : the % character matches any substring we consider the following examples.
- 'perry%' matches any string beginning with perry.
- '%idge%' matches any string containing 'idge' as substring.
- '....' matches any string exactly three characters.
- '....%' matches any string of at least of three characters.

Oracle functions:-
Function are used to manupulate data items and return result. function follow the format of function name (argument 1, argument
An arrangement is user defined variable or constant. The structure of function is such that it accepts zero or more arguments.

Examples:
AVg # return average value of n
Syntax:
AVg ([distinct/all] n)
Min - return minimum value of expr;

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Syntax:
MIN ([distict/all] expr)
Count Returns the no of of rows where expr is not null.

Syntax:
Count ([distinct/all] expr)
Count(*) Return the no rows in the table, including duplicates and those with nulls.
Max Return max value of expr.

Syntax:
Max ([distinct/all] expr)
Sum Returns sum of values of n

Syntax:-
Sum ([distinct/all] n)

Sorting of data in table:-

Syntax:
Select Columnname, Columnname
From table
order by column name;

Teacher's Signature .....

Exp No. 03

Date / /
Page No.
Shivalal

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 constrains 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 constraint attached to a are defined along with the column definition, it is called a column 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 constants.

Syntax :-

Create table tablename
(columname data type (size) not null ...)

primary key :- primary key is one or more column is a table used to uniquickly identify each row in the table. primary key values must not be null and must be unique across the column. A multicolumn primary key is called composite primary key.

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivlal

Syntax: primary key as a column constraint
Create table tablename
(columnname datatype (size) primary key,.....)

primary key as a table constraint
Create table tablename
(Columnname datatype (size), columname datatype (size) ...
primary key (columnname, columnname));

Uniquestion key concept:-
A Uniquestion is similar to a primary key expect 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.

Syntax: Uniquestions as a column constraint
Create table table name
(columnname datatype (size) Uniquestion);

Uniquestion as table constraint;
Create table tablename
(columnname datatype (size), columnname datatype (size).....
uniquestion (columnname, columnname));

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Default value concept: At the time of cell creation a default value can be assigned to it. When the user is loading a record with values and leaves 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 tablename
(columnname datatype (size) default value,.....);
```

Foreign key ~~comp~~ concept: Foreign key represents relationship between tables. A foreign key is column whose values are derived from the primary key of the same of some other table. the existence of foreign key implies that the table with foreign key is related to the primary key table from which the foreign key is derived. A foreign key must have corresponding primary key value in the primary key table to have meaning.

Foreign key as a column ~~const~~ constraint

Syntax:

```
create table table name
(columnname data type (size) references another table name);
```

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Foreign key as a table constraint:

Syntax:
Create table name
(columnname datatype (size).....
primary key (columnname);
foreign key (columnname) references table name);

Check Integrity Constraints:-
Use the check constraints when you need to enforce integrity rules that can be evaluated based on a logical expression following are a few examples of appropriate check constants.

- A check constraint name column of the client_master so that the name is entered is upper case.

- A check constraint on the client_no column of the client_master so that no client_no value starts with 'C'

Syntax:
Create table tablename
(columnname datatype (size) CONSTRAINT constraintname)
check (expression));

Teacher's Signature .....

Exp No. 04

Date / /
Page No.
Shivalal

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 able data exists.

The following tasks you can perform thrugh alter table command.

(i) Adding new columns:

Syntax :-

```
ALTER TABLE tablename
ADD(newcolumnname newdatatype(size));
```

(ii) Modifying existing table

Syntax:

```
ALTER TABLE tablename
MODIFY (new columnname newdatatype(size));
```

NOTE:- Oracle not allow constraints defined using the alter table, if the data in the table, violates such constraints.

Exp No. .....

Date / /
Page No.
Shivalal

Removing / Deleting Tables:- Following commands is used for removing or deleting a table.

Syntax:
Drop TABLE tablename;

Defining Integrity constraints in the ALTER TABLE command-

You can also define integrity constraints using the constraint clause in the ALTER TABLE command. The following example show the definitions of several integrity constraints.

(1) Add PRIMARY KEY-
Syntax:
ALTER TABLE tablename
ADD PRIMARY KEY (columnname);

(2) Add FORIEGN KEY:-
Syntax:
ALTER TABLE tablename
ADD CONSTRAINT constraintname
FOREIGN KEY (columnname) REFERENCES tablename;

Droping integrity constraints in the ALTER TABLE command:

You can drop an integrity constraint if the rule that it enforces is no longer true or if the constraint is no

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

needed. Drop the constraint using the ALTER TABLE command with the Drop clause. The following Examples illustrate the droping of integrity constraints.

(1) Drop the primary key:-

Syntax:

```
ALTER TABLE tablename
DROP PRIMARY KEY
```

(2) Drop FOREIGN KEY:-

Syntax:

```
ALTER TABLE tablename
DROP CONSTRAINT constraintname;
```

Teacher's Signature .....

Exp No. 05

Date / /
Page No.
Shivalal

Objective:- To impelement the Concept of Joins

Joint Multiple Table (EQUI JOIN):- Some time we require 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 joined on column that have dame data type and data with in tables.

The tables that have to be joined are specified in the FROM clause 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 clause.

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.

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

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

3. LEFT OUTER JOIN:.

LEFT OUTER JOIN = Cartesian product + selection but include rows from the left table which are unmatched pat nulls in the value 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.

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 # ;

Teacher's Signature .....

Exp No. 06

Date / /
Page No.
Shivalal

Objective:- 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 a group of sets of tuples, we specify this wish in SQL using the group by clause. The attribute of attributes given in the group by clause are used to form group. Tuples with the same value on all attributes in the group by clause are placed in one group.

Syntax:
```
SELECT columnname, columnname
FROM tablename
GROUP BY Columnname;
```

At times it is useful to state a condition that applies to groups rather than to tuples. For example we might be interested in only those branches where the average account balance is more than 1200. This condition does not apply to a single tuple,

Syntax:-
```
SELECT Columnname, Columnname
FROM tablename
GROUP By Columnname:
HAVING search condition;
```

Teacher's Signature .....

Exp No. 07

Date / /

Page No.

Shivalal

Objective:- To implement the concept of subQuestionary.

SubQuestionaries:- A SubQuestionary is a form of an SQL statement that appears inside another SQL statement. It also termed as nested Questionary. The statement containing a subQuestionary called a parent statement. The rows returned by the SubQuestionary are use by the following statement.

~~SubQuell:~~

-It can be used by the following commands:

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, AS SELECT * FROM oldclient_master,

Exam:

Creating clientmaster table from oldclient_master, table

Create table client_master
AS SELECT* FROM oldclient_master;

Using the Union, Intersect and Minus clause:

Union clause: The user can put together multiple Questionaries and combine their output using the Union clause. The Union clause merges the output of two or more Questionaries into a single set of rows and column. The final output of Union clause will be

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

Output :- Records only in Questionary one + Records only in Questionary two + A single set of records with is common in the both Questionnies.

Syntax:
SELECT Columnname, columnname
FROM tablename 1

UNION
SELECT columnname, columnname
FROM tablename 2;

~~Int~~ Intersect clause: The use can put together multiple Questionnies and their output using the interest clause. The final output of the interest clause clause will be:

Output:- A single set of Records which are common in both Questionnies

Syntax
SELECT columnname, columnname
FROM table name 1
INTERSECT
SELECT Columnname, Columnname
From table name 2;

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

MINUS CLAUSE :- The user can put together 2 multiple queries and combine their output = records only in Query one

Syntax:

```
SELECT columnname , columnname
FROM tablename;
MINUS
SELECT columnname , columnname
FROM tablename;
```

Teacher's Signature .....

Exp No. 08

Date / /
Page No
Shivalal

Objective:- To implement the concept of Indexes and Views.

Indexes:- An index is an ordered list of content of a column or group of column 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 (columnname);
```

Dropping Indexes:-

An index can be dropped by using DROP INDEX

SYNTAX:

```
DROP INDEX index filename:-
```

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

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 may be created fore the following reason.-

1. The DBA stores the views as a definition only.
2. Simplifies questionries.
3. Can be questionries 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~~ WHERE columnname = expression_list;
```

Renaming the Column of a View:-

Syntax:

```
CREATE VIEW Viewname AS
SELECT newcolumnname
FROM tablename
where columnname = expression_list;
```

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

~~SELECT~~

Selecting a data set from a view :-

Syntax:

```
CREATE SELECT columnname, columnname
    FROM view name
    WHERE search condition;
```

Destroying a View :-

Syntax:

```
DROP VIEW view name;
```

Teacher's Signature .....

Exp No. 09

Date / /
Page No.
Shivalal

Objective:- To implement the basics of PL/SQL.

Introduction - PL/SQL bridges the gap between database technology and procedural programming language. 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 messages can be displayed to the user.

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

Teacher's Signature .....

Exp No. .....

Date / /
Page No.
Shivalal

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~~:

Teacher's Signature .....

Exp No. .....

Date / /

Page No.

Shivalal

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.

Teacher's Signature .....

Experiment No. - 10.

Objective: ⇒ To implement the concept of Cursor and Trigger.

Cursor:- 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 is the size required to hold the number of rows in the Active Data Set.

Explicit Cursor:

A Cursor declared by the user is called Explicit cursor. For Questionaries that return more than one row, you must declare a cursor explicitly.

The data that is stored in the Cursor is called the Active Data Set. The Size of the Cursor in memory is the size required to hold the number of rows in the Active.

Why use an Explicit Cursor:

Cursor can be used When the user wants to Process data one row at a time.

Explicit Cursor Management :

The steps involved in declaring a cursor and manipulating data in the active set are:-

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the cursor.
- Fetch the data from the cursor one row one row at a time.
- Close the cursor.

Attributes (Explicit cursor) :

Oracle provides certain attributes / cursor variables to control the execution of the cursor. Whenever any cursor is opened and used oracle creates a set of four system variables via which oracle keeps track of the 'Current' status of the cursor you.

- Declare a cursor that specifies the SQL select statement that you want to process.
- Open the cursor.
- Fetch the data from the cursor one row at a time.
- Close the cursor.

Declare the cursor:

Syntax: cursor < cursorname > is SQL Statement;

Open the cursor:

Syntax: 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 Cursor advances to the next row in the Active Set. one can make use of any loop structure to fetch the records from the cursor into variable, one row at a time.

Syntax:

Each cursorname into variable 1, variable 2, ...

Closing a cursor:

Syntax: close < cursorname > ;

Database Triggers:

Database triggers are procedures that are stored in the database and are implicitly executed. When the contents of a table are changed.

Use of Database Triggers:

Database triggers support oracle to provide a highly customized database management system, some of the used to which the database triggers can be put to customize management information in oracle are as follow:

- A Trigger can permit DML statement against a table only if they are issued, during regular business hours or on predetermined weekdays.
- A Trigger can also 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.

How to apply Database Triggers:

A triggers has three basic parts: —

1. A triggering event or statement.

2. A trigger restriction.
3. A trigger action.

Types of Triggers:

Using the various CREATE TRIGGER statement, four types of triggers can be created:-

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 applying 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 possible 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:

Syntax:

```
Create trigger <Triggername> {Before, After}
{Delete, Insert, Update} on <Tablename>
For Each row When Condition.
Declare
    <Variable declarations>;
    <Constant Declarations>;
    Begin
        <PL/SQL> SubProgram Body;
    Exception
    Exception PI/SQL block;
    End;
```

How to Delete a Trigger:

Syntax:

```
Drop Trigger <Triggername>;
```