### Background Theory

Oracle workgroup or server is the largest selling RDBMS product. It is estimated that the combined sales of both these oracle database product account for around 80% of the RDBMS systems sold worldwide. These products are constantly undergoing change and evolving. The natural language of this RDBMS product is ANSI SQL; PL/SQL a superset of ANSI SQL oracle 8i and 9i also understand SQLJ.SQL stands for structured query language.PL/SQL is procedural language extension of SQL.

Oracle Corporation has also incorporated a full-fledged java virtual machine into its database engine. Since both executable shares the same memory space the JVM can communicate with the database engine with ease and has direct access to oracle tables and their data.

SQL is structure query language.SQL contains different data types those are

1. char(size)
2. varchar2(size)
3. date
4. number(p,s)
5. long
6. raw/long raw

### Different types of commands in SQL:

1. **DDL commands: -** To create a database objects

Example create table, alter table and drop table statements

1. **DML commands: -** To manipulate data of a database objects. Example update, delete statements.
2. **DQL command: -** To retrieve the data from a database. Example select statement
3. TCL-Transaction control language .example commit, rollback, savepoint statements. E).**DCL commands: -** To control the data of a database…example grant, revoke

### DDL commands:

* 1. **The Create Table Command: -** it defines each column of the table uniquely. Each column has minimum of three attributes, a name, data type and size.

### Syntax:

**Create table** <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>));

### Ex:

create table emp(empno number(4) primary key, ename char(10));

### Modifying the structure of tables.

* + 1. add new columns

### Syntax:

**Alter table** <tablename> add(<new col><datatype(size),<new col>datatype(size));

### Ex:

alter table emp add(sal number(7,2));

### Dropping a column from a table.

**Syntax:**

Alter table <tablename> drop column <col>;

### Example

alter table emp drop column sal;

### Modifying existing columns.

**Syntax:**

Alter table <table name> modify(<col><newdatatype>(<newsize>));

### Ex:

alter table emp modify(ename varchar2(15));

### Renaming the tables Syntax:

**Rename** <oldtable> to <new table>;

### Ex:

rename emp to emp1;

### truncating the tables.

**Syntax:**

**Truncate table** <table name>;

### Example

trunc table emp1;

### Destroying tables.

**Syntax:**

**Drop table** <table name>;

### Example

drop table emp;

### DML commands:

* 1. **Inserting Data into Tables: -** once a table is created the most natural thing to do is load this table with data to be manipulated later.

### Syntax:

insert into <table name> (<col1>,<col2>) values(<exp>,<exp>);

### Delete operations.

* + 1. remove all rows

### Syntax:

delete from <table name>;

* + 1. removal of a specified row/s

### Syntax:

delete from <table name> where <condition>;

### Updating the contents of a table.

* + 1. updating all rows

### Syntax:

Update <table name> set <col>=<exp>,<col>=<exp>;

* + 1. updating seleted records.

Syntax:

Update <table name> set <col>=<exp>,<col>=<exp> where <condition>;

### Types of data constrains.

* + 1. not null constraint at column level.

### Syntax:

<col><data type>(size)not null

* + 1. unique constraint

### Syntax:

Unique constraint at column level.

<col><data type>(size)unique;

* + 1. unique constraint at table level:

### Syntax:

Create table table name(col=format,col=format,unique(<col1>,<col2>);

* + 1. primary key constraint at column level

### Syntax:

<col><datatype>(size)primary key;

* + 1. primary key constraint at table level.

### Syntax:

Create table tablename(col=format,col=format

primary key(col1>,<col2>);

* + 1. foreign key constraint at column level.

### Syntax:

<col><datatype>(size>) references <tablename>[<col>];

* + 1. foreign key constraint at table level

### Syntax:

foreign key(<col>[,<col>])references <tablename>[(<col>,<col>)

* + 1. check constraint

check constraint constraint at column level.

**Syntax:** <col><datatype>(size) check(<logical expression>)

* + 1. check constraint constraint at table level.

**Syntax:** check(<logical expression>)

### DQL Commands:

* 1. **Viewing data in the tables**: - once data has been inserted into a table, the next most logical operation would be to view what has been inserted.
     1. all rows and all columns

### Syntax:

Select <col> to <col n> from tablename;

Select \* from table name;

* 1. **Filtering table data**: - while viewing data from a table, it is rare that all the data from table will be required each time. Hence, sql must give us a method of filtering out data that is not required data.
     1. Selected columns and all rows:

### Syntax:

Select <col1>,<col2> from <table name>;

* + 1. Selected rows and all columns:

### Syntax:

Select \* from <table name> where <condition>;

* + 1. selected columns and selected rows

### Syntax:

select <col1>,<col2> from <tablename> where<condition>;

### Sorting data in a table.

**Syntax:**

Select \* from <table name> order by <col1>,<col2> <[sort order]>;

### DCL commands:

Oracle provides extensive feature in order to safeguard information stored in its tables from unauthorized viewing and damage.The rights that allow the user of some or all oracle resources on the server are called privileges.

* + 1. Grant privileges using the GRANT statement

The grant statement provides various types of access to database objects such as tables,views and sequences and so on.

### Syntax:

GRANT <object privileges> ON <object name> TO<username>

#### [WITH GRANT OPTION];

* + 1. Revoke permissions using the REVOKE statement:

The REVOKE statement is used to deny the Grant given on an object.

### Syntax:

REVOKE<object privilege> ON

FROM<user name>;

### Experiment-01 Aim : Execute single line and group functions for a table

**Description:-Group functions are sum, avg, max, min etc applied on attributes.**

Single-row functions return a single result row for every row of a queried table or view.

They are numeric functions, character functions, date time functions and conversion functions

SELECT POWER (3,2) "Raised" FROM DUAL;

Raised

9

SELECT SQRT (26) "Square root" FROM DUAL;

Square root

5.09901951

### Solution:

SELECT COUNT (\*) FROM employees;

COUNT(\*)

24

SELECT AVG(Salary) average\_sal FROM employees;

AVERAGE\_SAL

15694

SELECT SUM(Salary) total\_sal FROM employees;

TOTAL\_SAL

87472

Select job, sum(sal) from emp group by job;

Select job,sum(sal) from emp group by job having (sal>10000); Select min(salary) from emp;

Select max(sal) from employee;

### Experiment -02

**Aim** :Create Tables in SQLwith necessary integrity constraints ,Keys,Data types for Attributes

**Description:**-integrity constraints allows us to preserve the integrity of the database. Integrity constraints disallow wrong/invalid data into the database.

The various integrity constraints are

1. primary key
2. not null
3. unique
4. default
5. check

Example table with all the above integrity constraints

### Solution:

Create table emp(empno number(3) primary key,

empname varchar2(20) not null, dept varchar2(12) default ‘admin’, phone number(10) unique,

salary number(8,2) check(salary between 3000 and 30000));

### Experiment -03

**Aim:** Implement the Queries in SQL for

1. Record Insertion
2. Data retrieval
3. Update records
4. Deletion of records

**Description:** Insertion,updation and deletion operations are DML operations.DML stands for data manipulation language. The changes are made permanent to the database after commit is executed.

Insert is used to insert rows/tuples into a table Update is used to modify the data in a table Delete is used to delete rows/records from a table Select is used to retrieve data from a table.

### Solution:

#### INSERT INTO EMP VALUES( ---,-----,-----, );

SELECT \* FROM EMP WHERE SAL>50000;

Select \* from emp where salary between 10000 and 40000;

Select \* from emp where job in (‘CLERK’,’MANAGER’,’OPERATOR’); Select \* from emp where sal>1000 and sal<50000;

#### UPDATE EMP SET SAL=45000 WHERE EMPNO=101; DELETE FROM EMP WHERE JOB=’CLERK’;

DELETE FROM EMP; IT WILL DELETE ALL ROWS from emp table

**Experiment -04 Aim: Creating** views for controlled access to a Table

### Description:

An Oracle VIEW, in essence, is a virtual table that does not physically exist. Rather, it is created by a query [joining one or more tables](https://www.techonthenet.com/oracle/joins.php) called base tables.

### Solution:

CREATE or replace VIEW view\_name AS SELECT columns

FROM tables [WHERE conditions];

* Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.

#### EXAMPLE

* Create view *v1 as* select ename,dept,job from emp where job=’CLERK’;

#### CREATE VIEW V2 AS SELECT \* FROM EMP;

* To drop a view
* DROP VIEW view\_name;

**Question:** Can you update the data in an Oracle VIEW?

**Answer:** A VIEW in Oracle is created by joining one or more tables. When you update record(s) in a VIEW, it updates the records in the underlying tables that make up the View.

So, yes, you can update the data in an Oracle VIEW providing you have the proper privileges to the underlying Oracle tables.

**Question:** Does the Oracle View exist if the table is dropped from the database?

**Answer:** Yes, in Oracle, the VIEW continues to exist even after one of the tables (that the Oracle VIEW is based on) is dropped from the database. However, if you try to query the Oracle VIEW after the table has been dropped., you will receive a message indicating that the Oracle VIEW has errors.

If we recreate the tables (the table that you had dropped), the Oracle VIEW will again be fine.

**Experiment -05 Aim :** Execute DCL and TCL statements

**Description:** DCL stands for Data control language .Example Grant and Revoke.TCL stands for transaction control language.Example commit, savepoint, rollback.Grant is used to grant permissions on a database object to other users.Revoke is used to revoke permissions from a user on a particular object.

### Solution :DCL commands:

Oracle allows us to safeguard the information stored in its tables from unauthorized access or damage .The rights that allow the users to access some or all oracle resources on the server are called privileges.

1. Grant privileges using the GRANT statement

The grant statement provides various types of access to database objects such as tables,views and sequences and so on.

### Syntax:

GRANT <object privileges> ON <objectname> TO<username>

#### [WITH GRANT OPTION];

1. Revoke permissions using the REVOKE statement:

The REVOKE statement is used to deny the Grant given on an object.

**Syntax:** REVOKE<object privilege> ON <object name> FROM<user name>; grant all on dept to demo;

grant all on dept to demo with grant option; revoke all on emp from demo;

### TCL statements

commit;

### Undoing the changes

Rollback;

### Insering savepoints

Savepoints are like inserting bookmarks in the transaction.rollback with savepoint Can be used to undo a certain part of the transaction.

### Creating a save point:

Sql> savepoint s1;

### Rollback to a savepoint

Sql>rollback to s1;

### Experiment-06

**Aim:** Write a PL/SQL Procedure for an application using Exception handling

**Description:** Exceptions are pre defined run time errors raised by PL/SQL Example “divide by zero”, “no data found”. And a program is aborted by displaying a message. We have to use exception handler part of the block to handle exceptions.

### Solution :

Declare

A number(2);

B number(2);

C number(2);

Zero\_divide EXCEPTION;

Begin

A:=&a;

B:=&b;

If(B=0) then Raise zero\_divide; else

C:=a/b;

Dbms\_output.put\_line(‘the result is ‘ || C); End if;

Exception

When zero\_divide then Dbms\_output.put\_line(‘b can not be zero’); End;

### Valid Test Data:

Enter the value for a: 10

Enter the value for b:

0

### Expected OUTPUT:

b can not be zero Solution 2:

#### DECLARE

c\_id customers.id%type := &cc\_id; c\_name customers.name%type; c\_addr customers.address%type;

-- user defined exception ex\_invalid\_id EXCEPTION;

#### BEGIN

IF c\_id <= 0 THEN RAISE ex\_invalid\_id;

#### ELSE

SELECT name, address INTO c\_name, c\_addr FROM customers

WHERE id = c\_id;

DBMS\_OUTPUT.PUT\_LINE ('Name: '|| c\_name); DBMS\_OUTPUT.PUT\_LINE ('Address: ' || c\_addr);

#### END IF; EXCEPTION

WHEN ex\_invalid\_id THEN

dbms\_output.put\_line('ID must be greater than zero!');

WHEN no\_data\_found THEN dbms\_output.put\_line('No such customer!');

WHEN others THEN dbms\_output.put\_line('Error!');

#### END;

/

### Experiment -07

Aim: **Write PL/SQL Procedure for an Application using Cursors**

**Description:** Oracle uses a work area to execute SQL commands and processed information. This area can be accessed using cursors. Cursor is a database object used by applications to manipulate data in a set on a row-by-row basis.

**Example:** If you need to insert/update/delete bunch of data one by one, you have to use a cursor with a while loop Cursors can be used when the user wants to process data one row at a time.

Cursors can be implicit or explicit. Oracle automatically creates an implicit cursor for each SQL statement.

It has 4 attributes SQL%ROWCOUNT, SQL%FOUND, SQL%NOTFOUND, SQL%ISOPEN.

EXPLICIT cursors are created & managed by the programmer. It needs every time explicit open,fetch . The data that is stored in the cursor is called active data set. We can open,fetch and close a cursor.

### Life cycle of a cursor:

1. Open cursor :
   1. Server-side private memory of the server process is UGA(user global area).
   2. A memory structure for the cursor is allocated in UGA.
   3. Just a portion of memory is allocated for the cursor and SQL statement is not associated with the memory structure.
2. Parse cursor :
   1. An SQL statement is associated with the cursor.
3. Bind input variables: If the SQL statement uses bind variables, their values must be provided.
4. Execute cursor: The SQL statement is executed.
5. Fetch cursor: If the SQL statement returns data, this step retrieves it. Especially for queries,this step is where most of the processing is performed. In the case of queries, rows might be partially fetched. In other words, the cursor might be closed before fetching all the rows.
6. Close cursor: The resources associated with the cursor in the UGA are freed and consequently made available for other cursors.

**Solution 1**: A Cursor which updates the salaries of an Employee as follows.

1. if sal<1000then update the salary to 1500.
2. if sal>=1000 and <2000 then update the salary to 2500.
3. if sal>=2000 and <=3000 then update the salary to 4000. And also count the no.of records have been updated.\*/

Declare

Cursor my\_cur is select empno,sal from emp;

Xno emp.empno%type;

Xsal emp.sal%type;

C number;

Begin

Open my\_cur;

#### C:=0;

Loop

Fetch my\_cur into xno,xsal;

If(xsal<1000) then update emp set sal=1500 where empno=xno; Else

If (xsal<2000) Update emp set sal=2500 where empno=xno; Else

If ( xsal<3000) then

Update emp set sal=4000 where empno=xno; C:=c+1;

End if; End if; End if;

Exit when my\_cur%NOTFOUND ; End loop;

Close my\_cur;

Dbms\_output.put\_line(c||’records have been successfully updated’); End;

### Expected OUTPUT:

EMPNO ENAME JOB MGR HIREDATE SAL COMMD EPTNO

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7369 SMITH | CLERK | 7902 17-DEC-80 | 2000 | 20 |  | |
| 7499 ALLEN | SALESMAN | 7698 20-FEB-81 | 1600 |  | 300 | 30 |
| 7521 WARD | SALESMAN | 7698 22-FEB-81 | 1250 |  | 500 | 30 |

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7566 JONES | MANAGER | 7839 02-APR-81 | 2975 | 20 |
| 7654 MARTIN | SALESMAN | 7698 28-SEP-81 1250 1400 | | 30 |
| 7698 BLAKE | MANAGER | 7839 01-MAY-81 2850 | | 30 |

### Solution 2:

#### DECLARE

c\_id customers.id%type; c\_name customers.name%type; c\_addr customers.address%type; CURSOR c\_customers is

SELECT id, name, address FROM customers; BEGIN

OPEN c\_customers; LOOP

FETCH c\_customers into c\_id, c\_name, c\_addr; EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr); END LOOP;

CLOSE c\_customers; END;

/

### Experiment-08

**Aim:** Write Row and Statement Triggers for implementing complex Data base constraints**.**

Description: A trigger is a PL/SQL program associated with a specific database table. Triggers are fired/executed automatically whenever a given SQL operation affects the table. The SQL operations may be insert, update,delete etc.

### Advantages./uses of triggers

* Automatically generate derived column values
* Prevent invalid transactions
* Gather statistics on table access
* Maintain synchronous table replicates

### Types of triggers

* Statement triggers
* Row triggers
* Before triggers
* After triggers

### Row Trigger

Create or replace trigger del\_info after delete on student for each row Begin

Insert into del\_dummy values (:old.name, :old.rollno,sysdate); End;

/

### Statement level trigger

create trigger stmt\_level\_trigger after update on emp

begin

update emp

set grade = NULL where grade = 13;

end;

/

**Experiment-09**

**Aim : A PL/SQL program using Procedures to find the minimum of two numbers.**

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS BEGIN

IF x < y THEN

z:= x; ELSE

z:= y; END IF;

END;

BEGIN a:= 23;

b:= 45;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (23, 45) : ' || c); END;

/

**Experiment -10 Aim : A PL/SQL program using functions.**

**Description:**A function is a self contained program segment that carries out some specific task .A function can return only one value at a time.It can be accessed repeatedly from different parts of a program.

### Solution:

/\*create a function which count total no.of employees having salary less than 6000.\*/

/\*function body\*/

Create or replace function count\_emp(esal number)return number as Cursor vin\_cur as Select empno,sal from emp;

Xno emp.empno%type; Xsal emp.sal%type;

C number; Begin

Open vin\_cur; C:=0;

loop

fetch vin\_cur into xno,xsal; if(xsal<esal) then

c:=c+1;

end if;

exit when vin\_cur%notfound; end loop;

close vin\_cur; return c;

end;

/

### Expected output:

Function created.

/\*function specification\*/ Declare

Ne number(3); Xsal number(4); Begin Xsal:=&xsal;

Ne:=count\_emp(xsal);

Dbms\_output.put\_line(‘there are ‘||ne||;employees’); End;

/

### Expected OUTPUT

There are 8 employees.

### Experiment-11

Aim: A **PL/SQL program using concept of Records**

**Description:**One tuple of relation is called one record.The %ROWTYPE attribute enables a programmer to create **table-based**

### Solution:

#### DECLARE

customer\_rec customers%rowtype; BEGIN

SELECT \* into customer\_rec FROM customers

WHERE id = 5;

dbms\_output.put\_line('Customer ID: ' || customer\_rec.id); dbms\_output.put\_line('Customer Name: ' || customer\_rec.name); dbms\_output.put\_line('Customer Address: ' || customer\_rec.address); dbms\_output.put\_line('Customer Salary: ' || customer\_rec.salary);

#### END;

/

### Experiment-12

Aim: **PL/SQL program using Packages**

**Description :** Packages are schema objects that groups logically related PL/SQL types, variables and subprograms.

A package will have two mandatory parts:

* Package specification
* Package body or definition

### Package Specification

The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.

All objects placed in the specification are called **public** objects. Any subprogram not in the package specification but coded in the package body is called a **private** object.

REATE PACKAGE cust\_sal AS

PROCEDURE find\_sal(c\_id customers.id%type); END cust\_sal;

/

### Package Body

The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from code outside the package.

### Solution :

CREATE OR REPLACE PACKAGE BODY cust\_sal AS

PROCEDURE find\_sal(c\_id customers.id%TYPE) IS c\_sal customers.salary%TYPE;

#### BEGIN

SELECT salary INTO c\_sal FROM customers

WHERE id = c\_id; dbms\_output.put\_line('Salary: '|| c\_sal);

END find\_sal; END cust\_sal;

/

### Using the Package Elements

#### ECLARE

code customers.id%type := &cc\_id; BEGIN

cust\_sal.find\_sal(code); END;

/

**Additional Experiments**

**Experiment-01**

**Aim : Program using CASE with Exception statement in PL/SQL Solution :**

#### DECLARE

1. grade CHAR(1);

#### BEGIN

1. grade := 'B'; 5

#### CASE

1. WHEN grade = 'A' THEN DBMS\_OUTPUT.PUT\_LINE('Excellent');
2. WHEN grade = 'B' THEN DBMS\_OUTPUT.PUT\_LINE('Very Good');
3. WHEN grade = 'C' THEN DBMS\_OUTPUT.PUT\_LINE('Good');
4. WHEN grade = 'D' THEN DBMS\_OUTPUT.PUT\_LINE('Fair');
5. WHEN grade = 'F' THEN DBMS\_OUTPUT.PUT\_LINE('Poor');

#### END CASE;

13

#### EXCEPTION

1. WHEN CASE\_NOT\_FOUND THEN
2. DBMS\_OUTPUT.PUT\_LINE('No such grade');

#### END;

1. /

### Expected output

Very Good

PL/SQL procedure successfully completed.

### Experiment-02

Aim:- **A PL/SQL Program using Arrays**

PL/SQL programming language provides a data structure called the VARRAY, which can store a fixed-size sequential collection of elements of the same type. A varray is used to store an ordered collection of data, but it is often more useful to think of an array as a collection of variables of the same type.All varrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

#### DECLARE

type namesarray IS VARRAY(5) OF VARCHAR2(10); type grades IS VARRAY(5) OF INTEGER;

names namesarray; marks grades;

total integer;

#### BEGIN

names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz'); marks:= grades(98, 97, 78, 87, 92);

total := names.count; dbms\_output.put\_line('Total '|| total || ' Students');

FOR i in 1 .. total LOOP dbms\_output.put\_line('Student: ' || names(i) || ' Marks: ' || marks(i));

END LOOP;END;

### Experiment-03

**Aim : PL/SQL Function that computes and returns the maximum of two values**.

**Description:**-It uses a function with two IN paramaters of number type and compares the two using if statement and returns the maximum out of two.

### Solution :

#### DECLARE

a number;

b number;

c number;

FUNCTION findMax(x IN number, y IN number) RETURN number

#### IS

z number; BEGIN

IF x > y THEN

z:= x;

#### ELSE

Z:= y; END IF;

RETURN z; END;

BEGIN a:= 23;

b:= 45;

c := findMax(a, b);

dbms\_output.put\_line(' Maximum of (23,45): ' || c); END;

**Expected output:** Maximum of (23,45): 45