UNIT-III

STRUCTURED QUERY LANGUAGE(SQL)

Basic Structure of SQL:

- SQL stands for Structured Query Language.
- It is a programming language not a database.
- It can be implemented by using different softwares like db2,mySQL,Oracle 10g,Oracle 11g.
- It is based on set and relational operations with certain modifications.

eid	ename	age	salary
1	*a*	*25*	*30000*
2	*b*	*26*	*35000*
3	*c*	*27*	*32000*
4	*d*	*28*	*30000*
5	*e*	*29*	*35000*

Different clauses in SQL:

1.SELECT CLAUSE: It is used to retrieve the information from a relation(or) displays the information.

➔ It is also equivalent to the projection( )in relational algebr***a.***

SYNTAX: select A1,A2,..,An from r1,r2,...,rn where P;

where, A1,A2,..,An are attributes,
r1,r2,...,rn are relation (or) table,
P is predicate(condition).

- ➔ Select from employee;   here ‘*’ indicates “all attribute”.
- ➔ Select clause allow duplicates in a relation as well as query result.

2.DISTINCT CLAUSE: To remove duplicates in a relation,we use distinct keyword after select statement.

- ➔ It retrieve Unique values from a table.

SYNTAX: select distinct column_name from table_name;
Ex: select distinct salary from employee;

salary
30000
35000
32000

3. WHERE CLAUSE: The where clause is used to specify a condition while fetching data from single or multiple tables(joining).

- ➔ If the given condition is satisfied then it returns a specific value from a table.
- ➔ We should use where class to filter the records and fetching only necessary records.
- ➔ The where clause not only used in select but also used in Update,delete statements.

SYNTAX: select column_name1,.....,column_name n from table_name where condition;

EX:select ename,age from employee where salary>32000;

ename	*Age*
b	*35000*
e	*35000*

4.FROM CLAUSE: It produce the tabular structure.It is followed by select statement.

5.GROUP BY CLAUSE: It is used in combine with select statement to arrange identical data into groups.

- ➔ It is followed by select statement.
- ➔ It is used to group differentiate rows of data together based on any one column.

SYNTAX: select column_list from table_name group by column_name;

EX: select salary,sum(salary) from employee group by salary;

salary	*sum(salary)*
30000	*60000*
35000	*70000*
32000	*32000*

6.ORDER BY CLAUSE: It is also used with select statement and used to sort the data in ascending or descending order.

SYNTAX: select column_list from table_name order by column name desc;

ex: select *from employee order by salary desc;

salary
35000
35000
32000
30000
30000

7.HAVING CLAUSE: The having clause must be followed by group by clause in SQL query.

SYNTAX: select column_list from table_name group by column_name having(condition);

EX: select salary,sum(salary) from employee group by salary having sum(salary)>45000;

OUTPUT:

salary	*sum(salary)*
30000	*60000*
35000	*70000*

SQL FUNCTIONS:

- ➔ All SQL functions are inbuilt functions.
- ➔ These are classified as two types:

1.Single row function
2.Multiple row function

1.SINGLE ROW FUNCTION:

- ➔ There are the one who works on the single row and return one output for row.

EX: Conversion Function,Character Function (or) String Function,Numeric Function.

Conversion Function:

upper( ): This function convert a string to Uppercase.
Syn: select upper(string);

Eg: select upper("dbms") ;
O/p: DBMS

lower( ): This function convert a string to lowercase.
Syn: select lower(string);

Eg: select upper("Dbms") ;
O/p: dbms

String Functions:-

These are accept character as input and return number or character value.

1.concat():- This function is used to combine two strings.
Syn: select concat(string1,string2);

Eg: select concat("cse","world");
O/p: cse world

2.strcmp( ):-This function is used to compare two strings.
Syn: select concat(string1,string2);
Eg:select strcmp("man" , "mom");
O/p: 1

3.length( ): This function is used to count the length of the string.
Ex: select length("cse");
O/P: length("cse")
3

4.substr( ):This function is used to return a portion of string from given start point to end pount.
Ex: select substr("world",2);
O/p: substr("world",2)
orld

5.instr( ): This fuction is used to return a numeric position of a character (or) string.
Ex: select instr("world", "l")

O/P: instr(“world”, “l”);
4

6.lpad( ): This fuction is used to inert the symbol with the actual length of the string from left side.

Ex: select lpad(“world”,10, “*”);

O/P: lpad(“world”,10, “*”)
*****world

7.rpad( ): This function is used to insert the symbol with the actual length of the string from rpad side.

Ex: select rpad(“world”,10, “*”);

O/P: rpad(“world”,10, “*”)
world*****

8.ltrim( ): This function is used to remove leading spaces in a given string from leftside.

Ex: select ltrim(“   world”);

O/P: world

9.rtrim( ): This function is used to remove leading spaces in a given string from rightside.

Ex: select ltrim(“world   ”);

O/P: world

3.NUMERIC FUNCTIONS:

1.truncate( ):

Ex: select trunc(28.7)   #removes decimal part.

O/P: 28

2.round( ):

Ex: select round(27.6)

O/P: 28

3.mod( ):

Ex: select mod(27,6)

O/P: 3 #remainder

4.least( ):

Ex: select least(-27.5,-28.5)

O/P: -28.5

5.greatest( ):

Ex: select greatest(-27.5,-28.5)

O/P: -27.5

6.sqrt( ):

Ex: select sqtr(25)

O/P: 5

7.ceil( ):

Ex: select ceil(27.2)

O/P: 28

select ceil(-27.2)

O/P: -27

8.floor( ):

Ex: select floor(27.2)

O/P: 27

select floor(-27.2)
O/P: -28

9.power( ):

Ex: select power(8,2)
O/P: 64

MULTIPLE ROW FUNCTIONS:

- These are works upon group of rows and return one result for the complete set of rows.
- These are also called as “group function” (or) “aggregate fuctions”.
- The following are the aggregate functions:
  (a) sum( )
  (b) avg( )
  (c) count( )
  (d) min( )
  (e) max( )
  (f) first( )
  (h) last( )

(a) sum( ): This function is used to get the sum of numeric column.
Syntax: select sum(column_name) from table_name;

EX: select sum(salary) from employee;
O/p:

sum(salary)

(b) avg( ): This function is used to get the average of numeric column.

Syntax: select avg(column_name) from table_name;

EX:select avg(salary) from employee;

O/p:

avg(salary)

(c) count( ): This function is used to get the no.of rows in table.
Syntax: select count(*) from table_name;

EX:select count(*) from employee;
O/p:

count(*)

➔ This fuction is also allows the where condition;
Ex: select count(*) from employee where name= “a”;
O/P: 1

(d) min( ): This function is used to get the minimum value from a column.
Syntax: select min(column_name) from table_name;

EX:select min(salary) from employee;
O/p: 30000

(e) max( ): This function is used to get the maximum value from a column.
Syntax: select max(column_name) from table_name;

EX:select max(salary) from employee;
O/p: 35000

(d) first( ): This function is used to get the first value of selected column.
Syntax: select column_name from table_name limit 1;

EX:select name from employee limit 1;
O/p: a

(e) last( ): This function is used to get the last value of selected column.
Syntax: select column_name from table_name order by column_name desc limit 1;

EX: select name from employee order by name desc limit 1;
O/p: e

NULL VALUES IN SQL:

- ➔ The SQL NULL is used to represent a Missing value.
- ➔ A NULL value in a table is value in a column that appears to be blank.
- ➔ A column with NULL value is “A Column with no value”.It is very important to understand that a NULL valur is different than 0 value (or) column contains spaces.
- ➔ In general,each NULL value is different from every other NULL value in database.

IMPORTANCE OF NULL VALUE:

NULL values are
(a)Not applicable: Which means when a value doesn’t exist for an entity.

Ex:Some of the students are not contain middle_name.

(b)Unknown:
(i)Missing: Which means that value exist but unknown.

Ex:Just know the names of your friend don’t know the middle_name or last_name.

(ii)Not Known: Which means that no information about the existence.

- ➔ We check NULL value by using IS NULL (or) IS NOT NULL operators.

➔ For example,
create table student(sid int NOT NULL,first_name varchar(10),middle_name varchar(10),last_name varchar(10),marks int);

➔ In above example, NOT NULL specifies that column should always accept value of given datatype.There are one column that contains NOT NULL values, that is sid and remaining 4 column first_name,middle_name,last_name contains NULL values.

IS NOT NULL OPERATOR:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
3	g	h	i	NULL
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
8	r	s	t	NULL
9	NULL	NULL	NULL	95

Select sid ,first_name,middle_name,last_name,marks from student where marks IS NOT NULL;

OUTPUT:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
9	NULL	NULL	NULL	95

IS NULL OPERATOR:

select sid ,first_name,middle_name,last_name,marks from student where marks IS NULL;

OUTPUT:

sid	first_name	middle_name	last_name	marks
3	g	h	i	NULL
8	r	s	t	NULL

Replace NULL Values:-

There are different ways to replace NULL values.

(a)IF NULL function
(b)case statement
(c)COALESCE function

(a)IF NULL function:

```
select  first_name, middle_name, last_name,
       IFNULL(marks,0) as marks from student;
```

OUTPUT:

sid	first_name	middle_name	last_name	marks
1	a	b	c	70
2	d	e	f	75
3	g	h	i	0
4	NULL	j	k	78
5	l	NULL	m	80
6	n	o	p	85
7	NULL	NULL	q	90
8	r	s	t	0
9	NULL	NULL	NULL	95

(b)case statement:

```
select first_name,last_name,case when marks IS NULL
then 0 else marks end as marks from student;
```

OUTPUT:

first_name	last_name	marks
a	c	70
d	f	75
g	i	0
NULL	k	78
l	m	80
n	p	85
NULL	q	90
r	t	0
NULL	NULL	95

(c)COALESCE function:

```
select
sid,COALESCE(first_name,middle_name,last_name) as
name,marks from student;
```

(OR)

```
 select  sid,COALESCE(first_name , middle_name ,
last_name ,  ’no name’) as name,marks from student;
```

OUTPUT:

sid	name	marks
1	a	70
2	d	75
3	g	0
4	NULL	78
5	l	80

6	n	85
7	NULL	90
8	r	0
9	NULL	95

Nested Queries in SQL:-

- In Nested Quries ,A query is written inside a Query.
- The Nested Query is also called as “Subquery” also called as “Innerquery”.

Rules of Nested Queries:-

1.The result of Inner Query is used in execution of Outer Query.
2.A subquery must always appear within pair of parenthesis.

3.A sub-query must return only one column with multiple rows, that means you cannot use “ select * ” in sub-query,but main query contain multiple columns with multiple rows.

4.You can use IN or not IN along with sub-query.

5.Sub-Query can be used with select,update,delete,insert statement along with operators like > , < , >= , <= , = , IN , BETWEEN.

Syntax for nested query:

select column_list from table_name where column_name operator (select column_name from table_name where condition);

eid	name	age	salary
1	ram	25	10000
2	raj	27	8000
3	rakesh	24	12000
4	ramesh	28	13000
5	harish	29	11000

SubQuery with Select Statement:-

Eg: select * from employee where age in(select age from employee where age>=27);

Output:-

eid	name	age	salary
2	raj	27	8000
4	ramesh	28	13000
5	harish	29	11000

SubQuery with UpdateStatement:-

```
update employee set salary=salary*0.5 where age
in(select age from employee where age>27);
```

OUTPUT:

eid	name	age	salary
1	ram	25	10000
2	raj	27	8000
3	rakesh	24	12000
4	ramesh	28	6500
5	harish	29	5500

SubQuery with Delete Statement:-

```
delete from employee where ahe in(select age from
employee where age>27);
```

OUTPUT:

eid	name	age	salary
1	ram	25	10000
2	raj	27	8000
3	rakesh	24	12000

SubQuery with Insert Statement:-

The SQL sunquery can be also used with insert statement.In insert statement,the data return from subquery is used to insert into another table(that is new table).

SYNTAX:-

insert into employee_new select *from employee where eid in(select eid from employee);

employee_new:(new table)

eid	name	age	salary
5	ravi	23	1000
6	raji	22	9090

OUTPUT:

eid	name	age	salary
5	ravi	23	1000
6	raji	21	9090
1	ram	25	10000
2	raj	27	8000
3	rakesh	24	12000

TYPES OF NESTED QUERIES:

There are of two types:

1. Co-related Nested Query
2. Independent Nested Query

1. Co-related Nested Query:

The output of inner query depends on the row which is being executed in outer query is called " Co-related Nested Query ".

2.Independent Nested Query:

The execution of innermost query is independent on outer query but the result of inner query is used in execution of outer query is called as " Independent Nested Query ".

GENERAL CONSTRAINTS IN SQL:

- ➔ SQL constraints are predefined rules and restrictions in a single column or multiple columns.
- ➔ These are provide accuracy and the integrity of the data inside the table.
- ➔ These are 2 types:

  1. Table level constraints: Means it limits table data.

2.Column level constraints: It limits column data.

The following are mostly used constraints in SQL:

(1)NOT NULL:

- ➔ This constraints desribes a column without NULL value.Once not NULL constraint is applied to column,we cannot pass NULL value to that column.

  NOTE: NOT NULL constraint cannot be defined at column level.

  Ex: create table student(sid int NOT NULL,sname varchar(10),marks int,age int);

In above query sid column will not take NULL values.

(2)UNIQUE:

- ➔ This constraint describe a column having UNIQUE values that means column not contains duplicate data.

  Table Level:

- ➔ create table student(sid int NOT NULL UNIQUE,sname varchar(20),marks int);

  In above query sid column contains unique values and won’t take NULL values.

Column Level:

- ➔ alter table student add unique(sid);

(3)CHECK:

- ➔ This constraint describe a value of column between range.It performs check on the values before storing the data into the database.

Table Level:

create table student(sid int NOT NULL check(sid>0),name varchar(20),age int);

In above query sid column is greater than 0. sid column values are greater than 0.

Column Level:

alter table student add check(sid>0);

(4)DEFAULT:

- ➔ This constraint describe insert default values to a column.The default values will be added to all new records,if no other values are specified.

Table Level:

create table student(sid int,name varchar(20),age int default 20);

In above query the age column contain default value 20 and won’t accept other values.

Column Level:

alter table student alter age set default 20;

(5)Primary key

(6)Foreign key

OPERATORS IN SQL:

1.Arithmetic operator : +,-,*,/,%
2.Relational operator : ==,=!,>=,<=,<,>
3.Logical operator :
OR,AND,IN,BETWEEN,NOT,ALL,ANY,LIKE, EXISTS

employee Table

eid	name	age	salary
1	raj	22	5000
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

AND:

select *from employee where age>=24 and salary>=6000;

eid	name	age	salary
3	rakesh	24	7000
6	rupesh	27	9000

OR:

select *from employee where age>=24 or salary>=6000;

eid	name	age	salary
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

NOT:

select *from employee where age is not null;

eid	name	age	salary
1	raj	22	5000
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

IN:

select *from employee where age in(24,27);

eid	name	age	salary
3	rakesh	24	7000
5	rajesh	24	4000
6	rupesh	27	9000

BETWEEN:

select *from employee where age between 24 and 27;

eid	name	age	salary
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

ALL:

select *from employee where 29>all(select age from employee);

eid	name	age	salary
1	raj	22	5000
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

Select *from employee where 24>all(select age from employee);

OUTPUT: Empty set

ANY:

select *from employee where 24>any(select age from employee);

eid	name	age	salary
1	raj	22	5000
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

LIKE:

select *from employee where name like “ram%”;

eid	name	age	salary
2	ram	23	6000
4	ramesh	25	5000

EXISTS:

select *from employee where exists(select age from from employee age>25);

eid	name	age	salary
1	raj	22	5000
2	ram	23	6000
3	rakesh	24	7000
4	ramesh	25	5000
5	rajesh	24	4000
6	rupesh	27	9000

KEYS IN SQL:

- ➔ A key can be a single attribute or group of attributes,where combination may act as key.
- ➔ Keys are plays major role in Relational_Database(RD).

Different types of keys:

(1)Super key
(2)Candidate key
(3)Composite key
(4)Secondary key
(5)Surrogate key
(6)Primary key
(7)Foreign key

sid	name	phonenumber	age
1	a	9123456780	20
2	b	9876543210	21
3	a	8123467890	20
4	a	7123456890	21
5	c	3456788990	20

(1)Super Key:

- ➔ A set of attributes within a table that can be uniquely identified each record within a table.
- ➔ Super key is superset of candidate key.
- ➔ In above table, {sid},{sid,name},{phonenumber},{sid,age,name},{name,phonenumber}...etc all are keys.
- ➔ Here,sid is unoque for every row of data.Hence,it can be used as identify each row uniquely.
- ➔ {sid,name},here name of two students can be same but sid’s are cannot be same.hence,this combination acts as a key.
- ➔ At the same time phonenumber for every student will be unique.hence,again phonenumber can be a key.

(2)Candidate Key:

- ➔ The minimal set of attributes which can be uniquely identified in a table.

- ➔ It is an attribute or set of attributes that can be as primary key for a table to uniquely identify each record in a table.
- ➔ They can be morethan one candidate keys in a table.
- ➔ In above tabke,sid,phonenumber both are candidate keys for the student table.
- ➔ A candidate key can never be null or empty and its value should be unique.
- ➔ There can be morethan one candidate keys in a table.

(3)Composite Key:

- ➔ If any single attribute of a table is not capable to being a key i.e., it can not identify each record uniquely.So,we combine two or more attributes to form a key is known as “Composite key”.

(4)Secondary Key:

- ➔ The candidate key which is not selected as primary key is known as “secondary key” (or) “alternate key”.

(5)Surrogate Key:

A key which can be unique in nature,not null and upadatable is called “Surrogate Key”.

Ex: phone_number.

(6)Primary Key:

- ➔ Primary key contains unique values and never contains new values.
- ➔ It is unique column in a table.
- ➔ A table can have only one primary key which consists of one or more columns.

(7)Foreign Key:

- ➔ It means it links two different tables together and column in one table that can be pointing to primary key in another table.
- ➔ They act as cross reference between tables.

INTRODUCTION TO PL/SQL:

- ➔ It is a combination of SQL with Procedural Language(PL).
- ➔ It was developed by Oracle Corporation in 1990's.
- ➔ It is extension of SQL and it allow programmer to write code in a Procedural Format.
- ➔ PL/SQL means gives instructions to the compiler what to do with SQL and how to through Procedural way.

Features of PL/SQL:

- ➔ It Support different Datatypes.
- ➔ It Support extensive error checking.
- ➔ It Support variety of programming structures.
- ➔ It Support Functions and Procedures.
- ➔ It Support OOP(Object Oriented Programming).
- ➔ It Support in development of web application and Server pages.

PL/SQL Structures:

(a)PL/SQL Block
(b)Procedures
(c)Functions
(d)Packages
(e)Triggers
(f)Cursors

(a)PL/SQL Block: The Block structure of PL/SQL contain 3 Sections/Parts.They are:

(1)declare
(2)executable statements or commands
(3) exception handling

(1)Declare:

- ➔ This section enclosed between keywords BEGIN/ begin and END/end.
- ➔ It is optional section and define all variables.

(2)executable statements or commands:

- ➔ This Section enclosed between ketwords BEGIN/ begin and END/end.
- ➔ It is mandatory section.
- ➔ It consists of executable statements of the program.
- ➔ It should have atleast one executable line of code,which may be just a null command to include that nothing should be executed.

(3) exception handling:

- ➔ This Section is start with keyword “exception”.
- ➔ It is optioal section contains exceptions that handles errors in the program.

Syntax:

```
declare
< declaration section >
begin
< executable statements >
exception
< exception handling >
end;
```

Example Program:

```
declare
message varchar(20):= ‘cse world’;
begin
dbms_output.put_line(message);
end;
```

(b)Procedures in PL/SQL:

- ➔ It is a subprogram unit consists of group of PL/SQL statement.each procedure in PL/SQL contains their own name and also it contain nested blocks to execute the process.
- ➔ It also contains declaration(optional),executable(mandatory) and exceptions(optional) sections.
- ➔ The values are can paused into the procedure from calling program and also pass the values to the calling program from procedure.
- ➔ It can return a statement to calling program but it can’t return any values to return statement.
- ➔ Procedures are cannot be called directly from select statement but they called from execute keyword or calling program.

Syntax:

```
create or replace procedure procedure_name
[(parameter_name[IN/OUT/IN OUT] type[.....])]
{IS/AS}
< procedure body >
end procedure_name;
```

replace:

It means modification (or) manipulation of an existing procedure.

Parameter_name:

It is a name of the variable which contains the parameter with IN,OUT,IN OUT with datatypes.

IN:

It takes the values from calling program and it is a read only parameter.

These Parameter is pass by reference.

OUT:

It return the value to the calling program from procedure.here OUT parameter act as variable you can also change the value.

IN OUT:

It pass initial value to the subprogram and return updated value to the calling program.

It can be assign a value and that can be read.

Procedure_body:

It contains set of executable statement.

Example:

```
create or replace procedure message
as
begin
dbms_output.put_line('Hello World');
end;
```

➔ In above example procedure is cannot called directly,it can be called with help of execute keyword.

Syntax for calling a procedure:

```
    execute procedure_name;
    ‘Hello World’
```

Drop a Procedure:

```
    drop procedure procedure_name;
```

Ex-2:

```
    declare
        a number;
        b number;
        c number;
procedure minimum( x in number, y in number, z out
number) as
    begin
    if x<y then
        z:=x;
    else
        z:=y;
    end if;
    end;
    begin
        a:=25;
        b:=40;
        minimum(a,b,c);
dbms_output.put_line(‘minimum of (25,40) is’ || c);
    end;
```

Output:

```
statements processed.
Minimum of (25,40) is 25.
```

Ex-3:

```
declare
a number;
procedure square(x in out number) is
begin
x=x*x;
end;
begin
a:=25;
square(a);
dbms_output.put_line(‘square of 25 :’ || a);
end;
```

Output:

```
statement processed.
Square of 25: 625
```

Functions in PL/SQL:

A PL/SQL function is same as a procedure except that it returns a value.

Syntax:

```
create or replace function function_name
[(parameter_name [IN/OUT/IN OUT] type[....])]
return return_datatype
```

```
{IS/AS}
BEGIN
      < function_body >
END[function_name];
```

Function_Name: It specifies name of the function.
Or replace: It allows modifying an existing function.
IN: It represent that value will be passed from outside.
OUT: It represent that this parameter will be used to return a value outside of the procedure.
RETURN:
It specifies that datatype you are going to return from the function.The function must contain a return statement.
Function_Body:
It contains the executable part.
AS:
This keyword is used instead of the IS keyword for creating a standalone function.

Example 1:
(finding maximum number among two numbers)

```
      declare
            a int;
            b int;
            c int;
```

```
    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:=50;
        c:=findmax(a,b);
    dbms_output.put_line(‘max of (23,50) is:’ || c);
    end;
```

Output:

```
    statement processed.
    Maximum of (23,50) is 50.
```

Example 2:
(checking whether the given num is palindrome or not)

```
declare
    x number;
```

```
        y number;
        z number;
function palin(n in out number)
return number is
        temp number;
        rem number;
begin
        temp:=0;
        m:=n;
        while(n>0) loop
                rem:=mod(n,10);
                temp:=(temp*10)+rem;
                n:=n/10;
        return m;
end;
begin
        x:=12321;
        z:=x;
        y=palin(x);
        if y=z then
        dbms_output.put.line("given num is palindrome");
        else
        dbms_output.put.line("given num is not palindrome");
        end if;
end;
```

OUTPUT:

```
        statements processed.
        Given num is palindrome
```

TRIGGERS:

- ➔ It is a procedure that start automatically if specified changes occur to the database.
- ➔ The Oracle execute(fired) automatically when given SqL operations like insert,update,delete that can be effect on the table.
- ➔ It contains 3 parts:

(1)Trigger Event:

Which contains events of DML operations.

(2)Condition:

It is optional and test the trigger is run or not.

(3)Trigger Action:

It performs what type of changes are made to the database table.

- ➔ When an event occur,the database trigger is fired and predefined PL/SQL statements with necessary action.

Syntax:

```
create or replace trigger trigger_name
{before/after}
insert or update or delete on table_name
for each row
when condition
declare
<declarative statements>
begin
```

```
<executable statements>
exception
<exception handling>
end;
```

Example:

employee table

eid	name	age	salary
1	a	20	2000
2	b	21	3000
3	c	22	4000
4	d	23	5000

```
Create or replace trigger changes
before
insert or update or delete on employee
for each row
when (new.eid>0)
declare
sal_diff number;
begin
sal_diff := :new.salary - :old.salary;
dbms_ouput.put_line(‘old salary’|| :old.salary);
dbms_ouput.put_line(‘new salary’|| :new.salary);
dbms_ouput.put_line(‘salary differ’|| sal_diff);
end;
```

Output:

```
    Trigger created.
```

Upadate:

```
update employee set salary=salary+1000 where
eid=2;
select *from employee;
```

Output:

```
        old salary: 4000
        new salary: 5000
        salary differ:1000
```

eid	name	age	salary
1	a	20	2000
2	b	21	4000
3	c	22	4000
4	d	23	5000

Insert:

```
insert into employee values(5, ‘e’,24,6000);
select *from employee;
```

Output:

```
        old salary
        new salary: 6000
        salary differ
```

eid	name	age	salary
5	e	24	6000
1	a	20	2000
2	b	21	4000
3	c	22	4000
4	d	23	5000

Delete:

```
delete from employee where eid=3;
select *from employee;
```

Output:

```
1 row(s) deleted.
```

eid	name	age	salary
5	e	24	6000
1	a	20	2000
2	b	21	4000
4	d	23	5000

CURSORS:

- A cursor is a temporary work area created in the system memory when a SQL statement is executed.

- A cursor contains information on a select statement and the rows of data accessed by it.
- This temporary work area is used to store the data retrieved from the database, and manipulate this data.
- A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the *active* set.
- There are two types of cursors in PL/SQL:

  1.Implicit Cursors

  2.Explicit Cursors

(1)Implicit Cursors:

- These are created by default when DML statements like, INSERT, UPDATE, and DELETE statements are executed. They are also created when a SELECT statement that returns just one row is executed.
- Oracle provides few attributes called as implicit cursor attributes to check the status of DML operations. The cursor attributes available are %FOUND, %NOTFOUND, %ROWCOUNT, and %ISOPEN.

Attribute	Description	Example
%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, it returns FALSE.	**SQL%FOUND**
%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.	**SQL%NOTFOUND**
%ISOPEN	Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.	**SQL%ISOPEN**
%ROWCOUNT	Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.	**SQL%ROWCOUNT**

Example for implicit cursors:

student table

sid	name	age	marks
1	a	20	80
2	b	21	90
3	c	22	85
4	d	23	95

```
declare
total_rows number;
begin
update student set marks=marks+10 where marks>90;
if sql%notfound then
dbms_output.put_line(‘no students is updated’);
elsif sql%found then
    total_rows := sql%rowcount;
dbms_output.put_line(total_rows || ‘updated’);
end if;
end;
```

Output:

```
    statement processed.
    1 updated.
```

```
select *from student;
```

sid	name	age	marks
1	a	20	80
2	b	21	90
3	c	22	85
4	d	23	105

(2)Explicit Cursors:

- They must be created when you are executing a SELECT statement that returns more than one row. Even though the cursor stores multiple records, only one record can be processed at a time, which is called as current row.
- When you fetch a row the current row position moves to next row.

Syntax for explicit cursor:

CURSOR cursor_name IS select_statement;

where,

- *cursor_name – A suitable name for the cursor.*
- *select_statement – A select query which returns multiple rows.*

There are four steps in using an Explicit Cursor:

- **DECLARE:** the cursor in the declaration section.
- **OPEN:** the cursor in the Execution Section.
- **FETCH:** the data from cursor into PL/SQL variables or records in the Execution Section.
- **CLOSE:** the cursor in the Execution Section before you end the PL/SQL Block.

These are the three steps in accessing the cursor:

1) Open the cursor.
2) Fetch the records in the cursor one at a time.
3) Close the cursor.

General Syntax to open a cursor is:

```
OPEN cursor_name;
```

General Syntax to fetch records from a cursor is:

```
FETCH cursor_name INTO record_name;
```

OR

```
FETCH cursor_name INTO variable_list;
```

General Syntax to close a cursor is:

```
CLOSE cursor_name;
```

General Form of using an explicit cursor is:

```
 DECLARE
     variables;
     records;
     create a cursor;
 BEGIN
   OPEN cursor;
   FETCH cursor;
    process the records;
   CLOSE cursor;
 END;
```

Example for explicit cursors:

```
declare
c_sid  student.sid%type;
c_name  student.name%type;
c_age  student.age%type;
c_marks  student.marks%type;
cursor c_student is
select sid,name,age,marks from student;
begin
open c_student;
loop
fetch c_student into c_sid,c_name,c_age,c_marks;
exit when c_student%notfound;
dbms_output.put_line(c_sid || ‘’ || c_name || ‘’ || c_age || ‘’ || c_marks);
end loop;
```

```
close c_student;
end;
```

Output:

```
    statements processed.
    1   a   20  80
    2   b   21  90
    3   c   22  85
    4   d   23  95
```

~~~~~~~~~~~****ALL THE BEST****~~~~~~~~~~~~~
~~~~~~~~~~~