Module - 3 SQL

Introduction to ORACLE:- ("Oak Ridge Automatic Computer and Logical Engine"). It is one of the RDBMS package. It provides various tools based on a concept called "Client - Server" technologies. Here server performs database activities , that help the user to interact with the database applications .

ORACLE Tools:- The ORACLE provides Two types of tools they are

Front End tools (D2K) i.e Forms, Reports

Back End Tools (SQL, PL/SQL)

Introduction to SQL:- The term SQL means "Structured Query Language". SQL is the standard set of commands used to access data with in the relational database. All tasks related to relational database management system, such as crating tables, modifying the data in the database, granting access to users and so on cab be done using SQL.

History of SQL:- In 1979 IBM(International Business Machine) corporation was developed a communication media called System. But it was not approved ANSI (American national standard institute) and ISO (International Standard Org.) Again in 1981 the same corporation developed a relational software SQL/Ds to interact with RDBMS. But even that was not satisfied by ANSI. In 1986 SQL was adopted by ORACLE corporation. The first name of SQL is SEQUL (Structured English Query Language). In 1989, SQL-89 was released and that is certified by ANSI and ISO. In 1992 SQL-92 was introduced with SQL plus commands, finally in 1999 SQL-99 was introduced by ORACLE corporation with full of tools. SQL playing vital role in software industry to communicate with all RDBMS software industry to communicate with all RDBMS software's like ORACLE, DB2, INGRESS, SYBASE etc.

i) SQL provides index future.

Data types in SQL

The data type specifies what type of data that can be entered into a column of a table. The SQL provides different data types among these, the most commonly used data types are

1) Number
2) Char
3) Varchar (Or) Varhcar2
4) Long
5) Date
6) Large Objects

1) Number:- This data types is used to store number or numeric data in a specified column. The data may be integer or floating point data. **Syntax**: Number (1,d);

Here 'l' specifies the length and 'd' specifies the no of decimal places with in the length. The length should be in the range from 1 to 38. the default length is '38' digits.

Ex:- custno number(10); or price number(8,5);

2) Char:- The Char data type is fixed length character data, This data type allows minimum length is 2000 characters or 2000 bytes.

Syntax: char (size); Here size indicates the maximum no of characters

Ex:- custname char(20);

3) Varchar2:- The "Varchar2" data type is similar to "Char" data type. It is used to store variable length strings. The maximum no of characters that can be entered is 4000

Syntax:- Varchar2(size); Here size indicates the maximum no of characters.

Ex: custname varchar(10);

The only difference between a "char" and "varchar2" is that the "Char" data type used blank spaces when ever a value less than the specified size.

4) Long:- It is similar to "Varchar2" data type. The range of the data type is upto 2GB. Generally this data type is not used, because only one column I a table should have "long" data type i.e A table can have only one long column. And also this data type column can not be indexed. **Syntax:-** long(size);

Ex:-custname long;

5) Date:- This data type is used to store date and time values. It occupies 7 bytes of memory. The default date format is "DD-MM-YYYY" and time format is "HH:MIN:SEC". This date format is called "Julian Date Format". The date values must be in the range from 1-JAN-4712 BC to 31-DEC-4712 AD. To enter time values we have to use built in functions

Ex:- Custdob date;

6) Large Objects :- This data type occupies 4GB memory. These objects are used to store the binary data and character data. These are two types

1) BLOB- Binary Large Objects

2) CLOB - Character Large Objects

Features and SQL Commands

Features:-

1) SQL is a non-procedural language
2) It is a general purpose English like language
3) It is a 4th generation language
4) Every query terminated by (;) semi colon

SQL Commands

SQL having Sub languages i.e called **SQL Commands**

They are
1) Data Definition Language (DDL)
2) Data Manipulation Language (DML)
3) Data Control Language (DCL)
4) Transaction Control Language (TCL)

1) DDL (Data Definition Language)Commands

The term DDL means "Data Definition Language". In SQL the DDl commands are used to define, modify and remove the database objects etc. In SQL the DDL commands are classified into different types. They are.

i) Create ii) Alter iii) Drop iv) Truncate v) Describe vi) Rename

i) Create :- In SQL the database primary object is "table". The table contains rows and columns in which the data is stored. In a table each column is identified by a column name and a given data type with specified width. In SQL the tables are created by using "Create table" command. This command creates only the structure of the table.

Syntax:- Create table <table name>(Columnname1 data type(width), Columnname2 data type(width),....);

Ex:- Create table student(htno number(8), sname varchar2(15), course varchar(5));

ii) Alter:- The alter command is used to modify the structure of the existing table. By using the alter command we can do the following.

→ We can add new columns to an existing table.
→ We can remove the columns from an existing table
→ We can add and drop constraints

Syntax:- Alter table <table name> add/modify(column name data type(width),.....);

Ex 1:- Alter table student add(dob date);

The above command adds new column "dob" to the Student table.

Ex 2:- Alter table student modify(htno varchar(10));

The above command modify the htno "number" data type to "varchar."

Syntax:- Alter table <table name> drop column <column name>;

Ex 1:- Alter table student drop column (course);

The above command removes the existing column "course" from the STUDENT table

Syntax:- Alter table <table name> add constraint;

The above syntax is used to add a table constraint

Ex 2:- Alter table student add primary key(htno);

The above command add a primary key constraint to the 'htno' column on Student table

Ex 3:- Alter table emp add foreign key(deptno) references dept;

The above command add a foreign key constraint to the 'deptno' column to the 'Emp' table.

iii) Drop:- The drop command is used to remove the database tables permanently

Syntax:- Drop table <table name>;

Ex:- Drop table student;

The above example removes the table "Student"

iv) Truncate:- This command is used to remove the all the data permanently from the table, but not structure. **Syntax:-** Truncate table <table name>;

Ex:- Truncate table student;

The above command removes all the data permanently from the table "student" but it does not remove the structure of "Student " table.

v) Describe:- This command is used to display the structure of the table.

Syntax:- Describe <table name>; or Desc <table name>;

Ex:- desc student;

vi) Rename:- This command is used to change the database object name from old name to new

Syntax:- Rename <old file name> to <new file name>;

Ex:- Rename student to stud;

2) DML (Data Manipulation Language) Commands

The DML means "Data Manipulation Language". The DML commands are used to manipulate data in a database. i.e., The DML commands are used to insert data, modify data, delete data and retrieve data in the table. In SQL the DML commands are classified into mainly 4 types. They are i) Insert 2) update 3) Delete 4) Select

i) Insert:- This command is used to add records into a table. This command is also used for inserting data into particular columns of table.

Syntax:- insert into <table name> values (val1,val2.....;

Ex:- insert into student values(101,'venu','B.Tech');

The above insert command is used for inserting values into all the columns. For inserting values for a particular columns we have to use the following systax.

Syn:- insert into <table name> (field1,field2,.....) values (val1,val2.....);

Ex:- insert into student (sno, sname) values (101,'kumar');

The above insert command is inserts only "sno, sname" values to "student" table.

Syntax for inserting Multiple rows

Syntax:- insert into <table name> values ('&field1','&field2',.....);

Ex:- insert into student values('&sno','&sname','&course');

The above insert command takes the values for "Sno, Sname, Phno" at the time of query execution. This insertion is also called "bulk insertion".

ii) Update:- The update command is used to modify the existing data with new data in a table. This command used "Set" clause.

Syntax:- update<table name> set <field1>=val1, <field2>=val2,.....;

Ex:- Update student set course= 'B.Tech';

The above update command modifies the all records(row) in a specified table.

For modifying only the specified records(rows), we have to use "where" clause.

Syntax :- update <table name> set <field>=val1,<field2>=val2,..... where <condition>;

Ex:- update student set course='B.Tech' where sno=101;

The above command modifies only particular records that will satisfy the condition sno=101 from the table "student".

iii) Delete:- The delete command is used to delete the specific records (or) all records from the table. For deleting a particular record in the table, we have to use "Where" clause.

Systax:- delete from <table name>[where <condition>];

Ex 1:- delete from student;

The above command deletes all rows from the table

Ex 2:-delete from student where sno=101;

The above command deletes specific rows that will satisfy the condition sno=101 fro the table "student"

iv) Select : This command is also called "DQL" command. It is used to retrieve (or) display the data from the table. By using this command and we can select all columns (or) specific columns all rows (or) specific rows from the table.

Syntax:- select *from <table name> [Where <condition>];

The above syntax '*' represents all columns in the table. Here 'where' clause is optional.

Ex 1:- Select *from student;

The above command displays all columns and all rows from the "student" table.

Ex 2:- select *from student where sno=101;

The above command displays all columns and specific rows that will satisfy the condition sno=101;

Ex 3:- Select sno, sname from student;

The above command displays specific columns i.e., sno, sname and all rows from the table "student"

Ex 4:- Select sno, sname, course from student where sno=101;

The above command displays particular column values i.e, sno, sname, course and specific rows that will satisfy the condition sno= 101 in the table "student".

3. DCL (Data Control Language) Commands

The term DCL means "Data control language". The DCL commands are used to control user, access to the database objects. These commands are used to giving and canceling privileges to other users. There are two commands under this category.

i) Grant ii) Revoke

i) Grant:- The Grant command is used to giving all (or) particular privileges to other users. The privileges are insert, delete, update and select commands.

Syntax:- grant <privileges> on <table name> to <username>;

Ex:- Grant select on student to smith;

The above command provides "select" privilege on student table to the user "Smith"

After getting privileges "select "from 'scott' user to "Smith" user, we write the following query for display data from the "student" table

Ex:- select *from scott.student;

ii) Revoke:- The revoke command is used to canceling the all or particular privileges that are given by "grant" command.

Syntax:- Revoke <privileges> on <table name> from <user name>;

Ex:- revoke select on student from smith;

The above command cancels the "select " privilege on student table from the user "Smith".

4) TCL (Transaction Control Language) Commands

The term TCL means "Transaction Control Language". A "transaction" is logical unit of work done on the database tables. The transaction may be "simple" or "complex". The simple transaction constraints one DML operation. But the complex transaction contains more than one DML operations. In SQL, the TCL commands are classified into different types they are

i) Commit ii) Roll back iii) Save point

i) Commit:- It is used to end a transaction. It makes the changes permanently to the DB.

Syntax:- Commit; or Commit work;

Ex:-
```
Sql> delete from student where htno=101;
Sql> insert into student values(102, 'jyothi', 'B.Tech');
Sql> commit;
```

ii) Rollback :- it is used to cancel the work done in the current transaction. The roll back command is used to cancel the entire transaction or particular part of the transaction.

To cancel entire transaction:- **Syntax:-** Rollback or Rollback work;

To cancel part of the transaction:- **Syntax:-** Rollback to save point id;

iii) Save point:- The save points are like the markers, which are used to divide a complex transaction into simple transaction. In SQL the save points are created by using save point command.

Syntax:- savepoint Savepoint Id;

Ex:-
```
1) SQL> delete from student where course= 'B.Tech';
   SQL> roll back;
2) SQL> insert into student values(10,'kumar', 'B.Tech');
   SQL> savepoint s1;
   SQL> update student set course = 'B.Tech' where htno=101;
   SQL> roll back to s1;
```

In the above first example, the rollback command cancels the entire transaction. In the second example the rollback command cancels only the update operation.

SQL Functions:

In SQL, the built-in functions are classified into two types. They are

A. Single-Row functions
B. Group (or) Aggregate functions

A. Single-Row Functions: This function returns only one value for each row in the table. The following are the different types of single row functions.

1. Date functions
2. Numeric functions
3. String functions
4. Conversion functions

Note: For checking these functions functionality, we use "dual" table. The "dual" table contains only one dummy column with varchar2(1) data type. The dual table contains only one record "X".

1. **Date functions:** These functions take date value and returns date value. The different date functions are

   a) **Sysdate:** Sysdate means System date. It returns system current date.
   Ex: `select sysdate from dual;`
   Output: 29-Nov-2024

   b) **To_char():** It returns a character string from a date value.
   Syntax: `To_char(date,format);`
   Here format may be YYYY, YY, Y (Number of years)
   DD (Number of the day for month)
   Day (Name of the Day)
   Ex: `Select To_char(sysdate,'YYYY') from dual;`
   Output: 2024

   c) **To_date():** It returns date value using a string value.
   Syntax: `To_date(string,format);`
   Ex: `Select To_date('11/29/1986', 'MM/DD/YYYY') from dual;`
   Output: 29-Nov-1986

   d) **Add_months():** It is used to add number of months to a date value.
   Syntax: `Add_months(date,n);`
   Ex: `select Add_months('29-Nov-1986',2) from dual;`
   Output: 29-Jan-1987

   e) **Last_day():** It returns last date from the given date.
   Syntax: `Last_day(date);`
   Ex: `select Last_day('15-Aug-1984') from dual;`
   Output: 31-Aug-1984

2. **Numeric functions:** These functions take one numeric value and returns numeric value. The different types of numeric functions are

   a) **Abs():** It returns a absolute value of the given number.
   Ex: `select Abs(-29.75) from dual;`
   Output: 29.75

b) **Ceil():** It takes a decimal value and returns the next integer value.

Ex:
```
select Ceil(18.15) from dual;
```
Output: 19

c) **Floor():** It takes decimal value and returns the previous integer value.

Ex:
```
select Floor(18.75) from dual;
```
Output: 18

d) **Round():** It takes decimal value and rounds into integer value.

Ex:
```
select Round(18.75) from dual;
```
Output: 19

e) **Trunc():** It takes decimal value and removes fraction part.

Ex:
```
select Trunc(18.75) from dual;
```
Output: 18

f) **SQRT():** It returns the square root value.

Ex:
```
select SQRT(169) from dual;
```
Output: 13

3. **String functions:** These functions are used to manipulate strings. It takes string values as input and returns either string or numeric values. The different types of string functions are

a) **Initcap():** It returns the initial letter as capital letter in each string.

Ex:
```
select Initcap( 'buchi reddy palem') from dual;
```
Output: Buchi Reddy Palem

b) **Lower():** This function is used to convert the given string into lower case.

Ex:
```
select Lower('BUCHI REDDY PALEM') from dual;
```
Output: buchi reddy palem

c) **Upper():** This function is used to convert the given string into upper case.

Ex:
```
select Upper('buchi reddy palem') from dual;
```
Output: BUCHI REDDY PALEM

d) **Length():** It returns the number of characters in a given string.

Ex:
```
select Length('VENUKUMAR') from dual;
```
Output: 9

e) **Substr():** It returns part of a given string.

Syntax: Substr(string,p,l);

Here p=starting position

L=no.of characters

Ex:
```
select Substr('VENUKUMAR',5,5) from dual;
```
Output: KUMAR

f) **||(String Concatenation):** It is used to combine two or more strings into single string.

Ex:
```
select 'I am studying'||' in GIST' from dual;
```
Output: I am studying in GIST

4. **Conversion functions:** These functions are used to convert a value into one data type to another data type. The different types of conversion functions are

a) **To_char():** It converts a numeric value into a string value.

Ex:
```
select To_char(65) from dual;
```
Output: '65' (character value)

b) **To_number():** It converts string value into numeric value.

Ex: `select To_number('65') from dual;`

Output: 65 (Number value)

c) **NVL():** It is used to replace a null value with string value (or) numeric value.

Ex: `select NVL(16,' ') from dual;`

Output: 16

B. Aggregate Functions (or) Group Functions

The Aggregate function is also called Group function or Statistical functions. The group functions takes multiple value as an input and produces a single output. The group functions supported by SQL are

1) Max()
2) Min ( )
3) Sum ( )
4) Avg ( )
5) Count ( )

1) Max:- It is used to find the maximum or highest value of the specified column in a table.

Ex:- select max(sal) from emp;

2) Min:- It is used to find the minimum lowest value of the specified column in a table.

Ex:- select min(sal) from table;

3) Sum():- It is used to find the total sum of values in a specified column in a table

Ex:- select sum(sal)from emp;

4) Avg:- It is used to find the average of specified column in a table

Ex:- Select avg(sal) from emp;

5) Count():- It is used to count the no.of rows or values. It is used in 3 ways.

i) count (*) ii) count(<column name>) iii) count(distinct<column names>)

The count(*) function counts the values including nulls and duplicate

Ex:- select count(*) from emp;

The count(<column name>) counts the values without including nulls

Ex:- select count(sal)from emp;

The count(distinct <column name>) counts the values without including nulls and duplicates.

Ex:- select count(distinct deptno) from emp;

6) Group by clause:- The group by clause is used to organize rows or values in group wise with the help of aggregate functions.

Syn:- select command group by clause;

Ex:- select sum(sal) from emp group by deptno;

It displays department wise, total salaries in 'emp' table

7) Oder by clause:- By default the select command do not display the data in particular order, i.e sorted order. The "order by " clause is used to display the data in ascending order or descending order.

Syn:- select *from <table name> [where <condition>] order by <column name> [asc/desc]; bydefault the order by clause displays data in ascending order.

Ex:- 1) select *from emp order by ename;

The above command displays all employee names in ascending order.

2) select *from emp order by ename desc;

The above command displays all employee names in descending order.

Views (Or) Virtual tables (Or) Stored queries

A View is a logical window of the physical database i.e, it is a mask of the table. A view takes the output of query. So it is also called as "stored query". A view is one of the database object. A view on which it is created is called a "base table" and that view is called "virtual table".

Generally View is created based on select query. The query can contain columns computed columns, aliases and aggregate functions from one or more tables. These views are called "complex (or) Non-Updatable" views

Advantages of Views:-

- Views provide the security in the Data base. Because the view can restrict users to only specified columns and rows in a table
- Views hides the base table name
- Views requires less storage space i.e It does not contain any data.
- View simplifies queries i.e, it avoid the reconstruction of complex queries.

Creating a View:- In SQL "Create View" command is used to create a view on one or more base tables.

Let us consider the following EMP table we can create a view

EMP_ID	E_NAME	E_SALARY
1	Smith	6500
2	Allen	8500
3	Martin	4500

Syn:- Create view<view name>as select query;

Ex:- 1) Create view v1 as select *from emp;

Now, we can display view v1 in a similar way as we display an actual table.

Eg: Select * from v1;

EMP_ID	E_NAME	E_SALARY
1	Smith	6500
2	Allen	8500
3	Martin	4500

2) Create view V2 as select EMP_ID, E_NAME, from emp;

Now, we can display view v2 in a similar way as we display an actual table.

Eg: Select * from v2;

EMP_ID	E_NAME
1	Smith
2	Allen
3	Martin

3) Create view v3 (eno,empname,empsalary) as select emp_id, e_name, e_salary from emp where e_name= 'Martin';

Now, we can display view v3 in a similar way as we display an actual table.

Eg: Select * from v3;

ENO	EMPNAME	EMPSALARY
3	Martin	4500

Dropping Views:- In SQL, "drop view" command is used to remove the views from the database. When a view is dropped, it does not affect to the base table data. **Syn:-** Drop view<view name>;

Ex:- Drop view v2;

When ever the changes are made in a table, then automatically the changes are made in a view

Updating a View:-

It is a view that can be used to update columns in the base table. i.e., A view that allows DML operators, those views are called "updatable views"

After creating the view we can used the DML command to update the view.

Ex:-Update v1 set e_salary=5000 where emp_id=3;

Some Restrictions/Disadvantages of views:-

1) Aggregate functions can not be used or Group by cannot be used
2) Set operators cannot be used i.e union, intersect and minus etc.
3) The views must be created only a "single table"
4) The "Primary Key" and "Not Null" columns must be included in views
5) "Sub queries" must not be included
6) The "distinct" and "having" clauses can not be included

Joins

A join is a relational operator that combines two or more tables into a single table based on a common column. The joins are used to process multiple tables. A join is actually performed by the "Where " clause. That compares 2 columns of different tables. Such a condition is called "join condition"

Syn:- select column list from table1, table2 where <condition>;

Here column list contains any columns from the specified tables.

Types of joins:- There are several types of joins used in relational data bases. They are

1) Simple Join
2) Natural Join
3) Outer Join
4) Self Join (or) Recursive Join
5) Cartesian Join

1) Simple Join:- It is the most commonly used join type. The simple join is classified into two types. They are

a) Equi Join
b) Non-Equi Join

a) Equi Join:- The join based on exact match between two columns of different tables, will be called as "Equi Join". That means the "where" clause uses comparison operator '=' to perform a join.

Ex:- Select e.ename, e.job, e.deptno, d.deptno from emp e, dept d where e.deptno=d.deptno;

b) Non-Equi Join:- A join which is based on other than equalities is called "Non-Equi join" . This join condition used comparison operators like <, >, <=, >=, < > except '='

Ex:- Select e.ename, e.job, e.deptno, d.deptno from emp e, dept d where e.deptno< >d.deptno;

2) Natural Join:- It is same as Equi Join but it eliminates duplicate columns in the result of the table. This join is also called **"Theta Join"**

Ex:-Select e.ename, e.job, d.deptno from emp e, dept d where e.deptno=d.deptno;

3) Outer Join:- The outer join displays the records that do not have matching values in the common columns. i.e if a row in one table does not have a matching value in another table then also it is included in the output. The symbol for outer join is "(+)". The outer join is extension of Equi or Natural joins. In the syntax of outer join the symbol is '+' shows including to retrieving the second database information The outer joins are 2 types

a) Right Outer
b) Left Outer

a) **Right Outer:-** Select e.ename, e.job, e.deptno, d.deptno from emp e, dept d where e.deptno(+)=d.deptno;

The above example retrieves rows from "dept" table which do not have any matching rows in the "emp" table.

b) **Left Outer:-** Select e.ename, e.job, e.deptno, d.deptno from emp e, dept d where e.deptno=d.deptno(+);

The above example retrieves rows from "emp" table which do not have any matching rows in the "dept" table.

4) **Self Join Or Recursive Join:-** joining of a table to itself is called "Self Join" or Recursive Join. The self join is used to retrieve the data from same table with the internal condition.

Ex:- 1) select e1.ename "employee name", e2.ename "manager name" from emp e1, emp e2 where e1.mgr=e2.empno;

5) **Cartesian Join:-** Joining of multiple tables with out a "join" condition is called Cartesian join. It combines each row of one table with every row of another table. For example If a table "emp" contains 5 rows and other table "dept" contains 4 rows then the cartesion product of "Emp" and "dept" is 2 rows (5 * 4).

Ex:- select e.empno, e.ename, e.deptno, d.deptno from emp e, dept d;

Sub Queries

Sub Query: A query within the query is called "sub query". The sub queries are used to get the information from one or more tables.

Characteristics of sub query:

- The first query in SQL statement is known as "***Outer query***".
- The query inside the SQL statement is known as "***Inner query***".
- At first, the inner query is executed, then after the outer query is executed.
- The inner query must be specified within parenthesis "( )".
- The output of the inner query is used as the input for the outer query.
- The entire SQL statement is sometimes called as nested query.

The Sub queries are classified into 4 types. They are

1. Single row sub query
2. Multiple row sub query
3. Nested sub query
4. Correlated sub query

1. **Single row sub query:** The inner query may return a single value then it is called as "Single row sub query".

   For example, to display employee details who are doing the same job as 'SMITH'.

   Ex: select * from emp where job = (select job from emp where ename = 'SMITH');

2. **Multiple row sub query:** The inner query may return more than one value then it is called as "Multiple row sub query". In this case, we use IN, ALL, ANY special operators.

   Ex (1): select * from emp where job IN (select job from emp where deptno = 20);

   (or)

   select * from emp where job = ANY (select job from emp where deptno = 20);

   It display the employee details whose job is same job in deptno '20'.

   Ex (2): select * from emp where sal > ALL (select sal from emp where deptno = 20);

   It display the employee details whose salary is greater than salary in deptno '20'.

3. **Nested sub query:** A sub query is defined in another sub query is called as nested sub query.

   Ex : select * from emp where sal = (select max(sal) from emp where sal < (select max(sal) from emp);

   It display the employee details, who drawn second maximum salary.

4. **Correlated sub query:** A Correlated sub query is a sub query, that executes once for each row in the outer query. The inner query is depends on the outer query. This process is similar to nested loops in programming languages.

   Ex:

```
for(i=1;i<=2;i++)
for(j=1;j<=3;j++)
    printf("i=%d j=%d",i,j);
```

Operators in SQL

The select command we can use various types of operators like,

i) Arithmetic operators ii) Relational operators (comparison)

ii) Logical operators iv) Special operators

i) Arithmetic operators:- In sql the arithmetic operators are used to perform various arithmetic operators on the data. The different types of arithmetic operates are

OPERATOR	DESCRIPTION
+	Addition
-	Subtraction
*	Multiplication
/	Division
^ (OR) **	Raise to the power of

Ex:- Select empno, ename, sal +10000-4000*3 from emp;

ii) Relational Operators:- These operators are used to compare two values. The operators are also called as comparison operators. The relational operators supported by SQL are

OPERATOR	DESCRIPTION
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> (or) !=	Not Equal to

Ex:- Select empno, ename, sal from emp where job= 'MANAGER';

iii) Logical Operators:- The Logical Operators are used to combine two or more relational expressions. These operators are also called as "compound operators". The logical operators supported in SQL are

AND➔ It returns true, If both expressions are true.

OR➔ It returns false, If both expressions are false

NOT➔ It reverse the value of expression

Ex:- select *from emp where sal>=2000 and job='MANAGER';

The above example retrieve the data from the emp table by using where clause i.e sal>=2000 and job= 'MANAGER';

iv) Special Operators:- The SQL allows special operators with the "where" clause. The special operators supported by SQL are

1) Between 2) Is null 3) Like 4)In

1) Between:- It is used to check whether a column value is with in a range.

Ex:- Select *from emp where sal between 2000 and 5000;

The above example retrieve the data from emp table, whose salary are between 2000 and 5000. The above query is similar to the following

Select *from emp where sal>2000 and sal<5000;

2) Is Null :- It is used to check fro a Null values in a column.

Ex:- select *from emp where comm is null;

3) Like:- It is used to heck whether a column value matches a given character string. It is only used fro character pattern. %(any character) _ (underscore) (one character)

Ex:-1) Select *from emp where ename like 'A%';

This example displays the 'ename' whose employee name must be starts with A and remaining letters should be any.

Ex:-2) select *from emp where ename like '_S%';

This example displays the 'ename' values second character should be 'S' and remaining letter should be of any.

3) In:- It is used to check whether a column value matches any value with in specified list of values.

Ex:- 1) select *from emp where job in ('manager', 'clerk');

2) select *from emp where sal in (2000,3000,5000);

Ex:- select *from emp where sal=2000 or sal=3000 or sal=5000;

Dear one

Don't loss the Patience until you meet the Success in your life

By

DVH. Venu Kumar *M.Sc., B.Ed.,M.Ed., M.Tech.,*

Asst.Professor

Githanjali Institute of Sci.and Technology : Nellore

Integrity Constraints

An integrity constraint is a rule that cannot be violated by the user. The constraints are used to prevent invalid data entry into the table. It enforces rules for the columns in a table. In SQL the integrity constraints are classified into 3 types. They are

1) Domain integrity constraints
2) Referential integrity constraints
3) Entity Integrity constraints

i) Domain integrity constraints:- A domain is a set of values, that may be assigned to a column. The domain integrity constraints enforces valid entries for a given column from the set of possible values. The domain constraints are of 2 types. They are

i) NOT NULL constraints
ii) CHECK constraints

i) NOT NULL:- The value which is absent is called NULL. This constraint is used to avoid the NULL values in a column of a table. i.e if any column is specified as NOT NULL, then we must have to enter a value for the NOT NULL constraint allows the duplicate(redundant) values.

Ex:- 1 Create table student(htno number(5) not null, sname varchar(20), address varchar(20));
In the above example the column htno does not allow NULL values. But other sname, address is allows NULL values. Suppose we can add the NOT NULL constraint after creating the table then we can use Alter table command i.e

Alter table student modify sname not null;

ii) CHECK:- The check constraint is used to specify a condition that should satisfy for each row in a table before it is stored.

Ex:- create table student(htno number(5), sname varchar(20), fee number(7,3) check(fee>0));
Suppose we can add the Check constraint after creating the table then we can use Alter table command i.e

Ex:- alter table student add check(sname like 's%');

2) Entity integrity constraints:- These are also used to prevent invalid data into a column of a table. Mainly entity integrity constraints are divided into 2 types. They are

i) Unique constraint
ii) Primary Key constraint

i) Unique Constraint:- It is used to prevent duplicate values. Any column declared with "unique" constraint in that column does not allow duplicate values. how ever it allows null values into the column

Ex:- Create table student(htno number(8) unique, sname varchar(10),address varchar(20));
Suppose we can add the Unique constraint after creating the table then we can use Alter table command i.e

Alter table student add unique(fee);

ii) Primary Key:- A field or combination of fields used for identifying a single record will be called as "Primary key". The primary key constraint does not allow duplicate and NULL values. Only one primary key constraint can be created for each table.

Ex:-1 Create table student(htno number(5) primary key, sname varchar(15), address varchar(20));

Ex:2:- Create table student(sid number(8), htno number(5), sname varchar(15), address varchar(20), primary key(sid, htno));

3) **Referential Integrity Constraints:-** A field which references to primary key of another table will be called as a **"foreign Key".** It is used to establish a relationship between multiple tables. For specifying the relation ship there must be one common column. The table in which the common column it specified as a primary key will be called as "Master" or "parent" table. Where as the another table will be called "detail" or "child" table.

Ex:- Master or parent table:

Create table dept(deptno number(5) primary key, dname varchar(10), loc varchar(15));

Detail or child table:-

Create table emp(empno number(8) primary key, ename varchar(15),sal number(15), deptno number(8) references dept(deptno));

The records from the Master table will not be deleted until the corresponding records are deleted from the detail table. Similarly new records can not be inserted into the detail table, until the corresponding values are inserted in the master table.

Relational Set Operators

The set operators are used to combine the results of two or more queries into single one. The sql supports 4 types of set operators. They are

1) UNION
2) UNION ALL
3) INTERSECT
4) MINUS

Rules for using set operators:-

1) The queries that are combined by set operator must have some number an order columns.
2) The queries must not contain Long data type
3) The first query labels(headings) are displayed as column headings in the output.

1) UNION:- The Union operator is used to combine two or more Queries. The result of the second query is added to the result of first query, by removing the duplicates.

Ex:- Select empno,ename from emp union select sno,sname from student;

2) UNIONALL:- The "union all" operator works same as "union" operator. The only difference is that the union all does not eliminates the duplicate values, where as the union eliminates the duplicates.

Ex:- Select empno,ename from emp union all select sno,sname from student;

3) INTERSECT:- It returns only rows that are common to both the queries.

Ex:- Select htno,sname from stud1 intersect select htno,sname from stud2;

4) MINUS:- The "Minus" operator returns all rows extracted by the first query which are not in the second query.

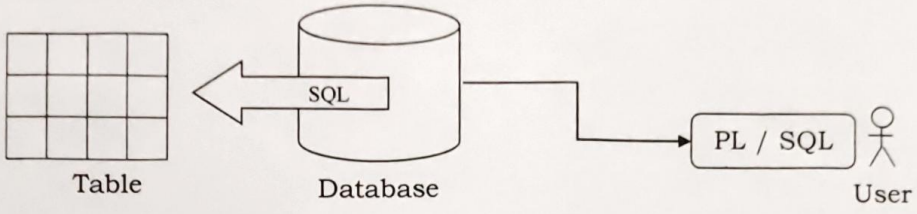
Ex:- Select htno,sname from stud1 minus select htno,sname from stud2;

PL / SQL

PL/SQL (or) Procedural Language

PL/SQL is Oracle's procedural language. It is the extension of SQL. PL/SQL stands for Procedural Language (or) Programming language through SQL statements. PL/SQL is also called Structured Programming Language. It was developed by Oracle Corporation in the late 1980s. It is a case sensitive. By using PL/SQL, we can execute set of statements in ORACLE engine.

PL/SQL allows us to mix SQL statements with procedural statements like IF statement, Looping structures etc. PL/SQL is the superset of SQL. It uses SQL for data retrieval and manipulation and uses its own statements for data processing.



Advantages:

- ✓ PL/SQL is a structural programming language.
- ✓ PL/SQL supports efficiency in calculations, accuracy in results and speed in transactions.
- ✓ Data manipulation and also provides facilities of conditional checking, branching and looping statements.
- ✓ PL/SQL provides user-defined exceptions for handling run time errors.
- ✓ PL/SQL supports various user-defined objects such as function, procedures and triggers etc.
- ✓ Applications written in PL/SQL are portable to any computer system.
- ✓ PL/SQL allows sending an entire block of statements to the database at one time. This reduces network traffic and provides high performance for the applications.

Structure of PL/SQL

Declare Declaration section	Optional
Begin Executable section	
Exception Exception section	Optional
End;	

A PL/SQL blocks can be divided into 3 parts. They are

1. Declaration section
2. Executable section
3. Exception section

A Declaration section appears within the "*declare*" and "*begin*" keywords. In this section, we can declare PL/SQL variables, cursors and local procedures.

Between "*begin*" and "*exception*" keywords, we can write Executable section. In this section, we use SQL statements and PL/SQL statements.

Between "*exception*" and "*end*" keywords, we can write Exception section. PL/SQL program ends with a keyword called "*end;*".

In PL/SQL block, "Declaration section" and "Exception section" are optional. In PL/SQL, we use only DML, TCL and all built-in functions, except group functions.

PL/SQL comment lines:

PL/SQL supports two types of comment lines. Those are

1. Single line comments (- -)
2. Multiple line comments (/* ..... */)

PL/SQL comment lines:

In PL/SQL the input statement is "&".

Ex:

```
a number;
a := &a;
```

The output statement is

```
Dbms_output.put_Line('Message' || value);
```

DATA TYPES

The data type specifies what type of data is stored into a specific variable. The following are the different types of data types.

1. Number
2. Char
3. Varchar2
4. Long
5. Date
6. Boolean

1. **Number:** This data type is used to store integer or real values.
   Syntax: Number(l,d);
   Here l=length and d=no.of decimal places in length
   Ex: (1) sno number(6);
   (2) sal number(7,2);
   sal:=19000.60;
2. **Char:** This data type is used to store only characters or alphabets.
   Syntax: char(size);
   Here size= no.of characters.
   If the size is not specified, then it occurs 1 byte of the memory.
   Ex: gender char;
   gender:='M';
3. **Varchar2:** This data type is used to store alphanumeric vales.
   Syntax: varchar2(size);
   Here size= no.of characters.
   Ex: emp_id varchar2(10);
   emp_id:='E_123';
   The maximum no.of characters size is 4000.
4. **Long:** This data type is similar to varchar2 data type. The range of this data type is "2GB".
   Syntax: long;
   Ex: biodata long;
5. **Date:** This data type is used to store date values. It reserves 7 bytes of memory. But the values are assigned through built-in functions only. The default date format is "DD-MON-YYYY". This date format is called "Julian date format". The date value must be in the range from "01-JAN-4612 BC" to "31-DEC-4612 AD".
   Syntax: date;
   Ex: s_dob date;
   s_dob:=TO_DATE('21-Nov-1993', 'DD-MON-YYYY');
6. **Boolean:** The Boolean data type can store only "True or false". The default value is "false".
   Syntax: boolean;
   Ex: a Boolean;

PL / SQL Attributes

Attributes allows us to refer data types from the database tables. The following types of attributes are supported by PL/SQL.

1. Boolean
2. %type
3. %row type

1. Boolean:- The Boolean data type can holds only True or False.

Syn: Boolean;

Ex:- x Boolean;

2. %type:- The %type is used to inherit the data type from a column of a database table.

Syn:- Variable name table name.column name %type;

Ex:- eno emp.empno%type;

3. %rowtype:- The %rowtype datatype is used to inherit the datatype from a record type that presents a row in a table.

Syn:- Variable name Table name%rowtype;

Ex:- rec emp%rowtype;

Control structures

Control structures are used to change the flow of execution. The following are the different control structures available in PL/SQL.

I. Conditional control statements
II. Repetitive (or) Iterative control statements
III. Unconditional control statements

I. Conditional control statements:

These statements are used to execute group of statements based on condition. There are 3 types of conditional control statements in PL/SQL.

1. If then
2. If then else
3. If then elsif

1. If then:

Syntax:

```
if condition then
        ____
        ____   } Statements
        ____
  end if;
```

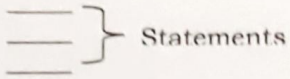
Ex:

```
if a>b then
      Dbms_output.put_Line (a|| 'is biggest' );
end if;
```

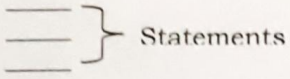
2. If then else:

Syntax:

if condition then



else

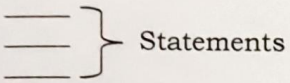


end if;

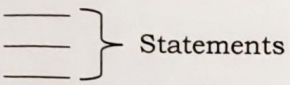
3. If then elsif:

Syntax:

if condition then



elsif condition then



end if;

Here the condition is "true", the statement block will be executed, and otherwise statement block is skipped.

II. Repetitive (or) Iterative control statements:

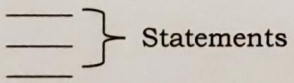
The groups of statements are executed as long as specified condition is satisfied. These control statements are also called looping control statements. They are 3 types of looping control statements.

1. Simple Loop
2. While Loop
3. For Loop

1. Simple Loop:

Syntax:

Loop



Exit when condition;

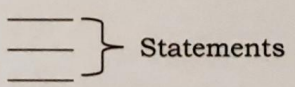
End loop;

2. While Loop:

Syntax:

While condition

Loop



End loop;

3. **For Loop:**

Syntax:

For variable name in [reverse] Lower limit . . Upper limit [step number]
Loop

Statements

End loop;

Ex (1):

For i in 1 . . 10
Loop
Dbms_output.put_Line(i);
End loop;

Output: 1 2 3 4 5 6 7 8 9 10

Ex (2):

For i in 1 . . 10 step 2
Loop
Dbms_output.put_Line(i);
End loop;

Output: 1 3 5 7 9

Ex (2):

For i in reverse 1 . . 10
Loop
Dbms_output.put_Line(i);
End loop;

Output: 10 9 8 7 6 5 4 3 2 1

III. **Unconditional control statements:**

Goto statement:

Syntax:

Goto label:

Statements

<<label>>

Statements

Procedures (or) Stored Procedures

A Procedure is a named PL/SQL block to perform a specific task. The major advantage of procedure is, it can be used to represent business transaction. The procedures are also called "*Sub programs* (or) *Stored procedures*".

Procedure contains two parts. They are

1. Procedure Header
2. Procedure Body

1. **Procedure Header:**

A procedure Header is specified with the help of a keyword "*create or replace procedure*" along with procedure name, followed by list of parameters.

2. Procedure Body:

All the statements that are specified between "*Begin*" and "*End*" keywords will be called as Procedure Body.

Syntax:

Create or replace procedure procedure-name (parameter [IN/OUT] data type, ...) is

------
------ } Local variable

Begin

PL/SQL (or) SQL statement

End;

From the above syntax, "IN/OUT" indicates, whether the parameter is for input (or) output. By default all parameters are "IN" parameters.

Ex:

```
Create or replace procedure fact (n number) is
    f number:=1;
Begin
    for i in 1 . . n
    loop
        f:=f*i;
    end loop;
    Dbms_output.put_line('Factorial=' || f);
End;
```

A procedure can be executed by calling it in another program. For doing, we have to specify the procedure name along with parameter.

Ex (1):

```
Declare
    a number;
Begin
    a:=&a;
    fact(a); → Procedure calling
End;
```

Ex (2):

SQL> Exec fact(5);

A procedure can be deleted by using the following syntax.

Syntax: Drop procedure procedure-name;

Ex: Drop procedure fact;

Advantages of procedures:

- **Modularity:** It allows the user, to divide the program into different sub parts. Because, understanding of program becomes easy also development of program becomes easy.
- **Reusability:** Once the procedure has been created, it can be used many no.of times in many no.of programs.
- **Reduced Redundancy:** The procedure reduces the repetition of code. Hence the redundancy level decreases. Thus we can save the amount of memory.

Functions (or) Stored Functions

A Function is a named PL/SQL block to perform a specific task. The difference between function and procedure is that, the function may return only one value. Whereas the procedure produces multiple values using "OUT" variable.

The functions are also called "*Sub programs* (or) *Stored functions*". Function contains two parts. They are

1. Function Header
2. Function Body

1. Function Header:

A Function Header is specified with the help of a keyword "*create or replace function*" along with function name, followed by list of parameters.

2. Function Body:

All the statements that are specified between "*Begin*" and "*End*" keywords will be called as Function Body.

Syntax:

Create or replace function function-name (parameter [IN] data type, ...)
return data type is

Local variable
Begin
PL/SQL (or) SQL statement
return value;
End;

From the above syntax, "IN" indicates, whether the parameter is for input. By default all parameters are "IN" parameters.

Ex:

```
Create or replace function fact (n number) return number is
    f number:=1;
Begin
    for i in 1 . . n
    loop
        f:=f*i;
    end loop;
    return f;
End;
```

A function can be executed by calling it in another program. For doing, we have to specify the function name along with parameter.

Ex:

```
Declare
    a number;
    f number;
Begin
    a:=&a;
    f:=fact(a); → Function calling
    Dbms_output.put_line('Factorial=' || f);
End;
```

A function can be deleted by using the following syntax.

Syntax: Drop function function-name;

Ex: Drop function fact;

Advantages of procedures:

- **Modularity:** It allows the user, to divide the program into different sub parts. Because, understanding of program becomes easy also development of program becomes easy.
- **Reusability:** Once the function has been created, it can be used many no.of times in many no.of programs.
- **Reduced Redundancy:** The function reduces the repetition of code. Hence the redundancy level decreases. Thus we can save the amount of memory.

CURSORS

The oracle engine uses a work area for its internal processing in order to execute SQL statements. This work area is called Cursor. (OR) A cursor is nothing but a pointer between the contest area and physical data storage. The data can be stored in the cursor is called "Active Data Set". Cursors are used to hold(store) more then one record

The cursors are 2 types they are

1) Implicit
2) Explicit

1) Implicit Cursors:-

- ✓ An Implicit Cursor is automatically crated by Oracle engine.
- ✓ This cursor interacts with only one record.
- ✓ The Implicit Cursor will take a record from the database and stored into a "*Host variable*" using "*into*" keyword.
- ✓ The Implicit Cursor doesn't have declaration, opening, fetching, and closing statements.

Ex:-

```
declare
        rec emp%rowtype;
begin
        select * into rec from emp where empno:=&empno;
        dbms_output.put_line(rec.empno||' '||rec.ename||' '||rec.sal);
end;
```

2) Explicit Cursors:-

- ✓ A cursor can store multiple records at a time will be called as Explicit Cursor.
- ✓ These Cursors are manipulated by the user explicitly.
- ✓ The user must declare, open, fetch, and close the Cursor.

Ex:-

```
declare
        rec emp%rowtpe;
        cursor c1 is select *from emp;
 begin
        open c1;
        loop
        fetch c1 into rec;
        exit when c1%notfound;
        dbms_output.put_line(rec.empno||' '||rec.ename||' '||rec.sal);
        end loop;
        close c1;
end;
```

Cursor Processing Commands:-

1) Open:- The open command open the cursor and executes the SQL commands. Before we can use a cursor, we need to open it

Syn:- open<cursor name>;

2) Fetch:- Once the cursor opend we can used the fetch command to retrieve data from cursor.

Syn:- fetch<cursorname> into <host variable>;

3) Close:- The close command is used to close the open cursor.

Syn:- close <cursor name>;

Cursor attributes:-

1) %isopen:- It returns "True". If the cursor is open other wise it returns "false".

Syn:- cursor name %isopen;

2) %found:- Whether cursor having records or not. If records found it returns "True". Otherwise it returns "False".

Syn:- cursor name %found;

3) %not found:- whether it checks is there any records are there or not. If records not found. It returns "True". Otherwise it returns "False".

Syn:- cursor name %not found;

4) %rowcount:- it returns no of records fetched from the cursor.

Syn:- cursor name %row count;

Triggers

A Trigger is a procedure that is executed automatically, whenever the specified event occurs. These triggers are executed "before/after" performing operation like insert, delete, and update.

The trigger definition contains 4 parts. They are

1. **Trigger timing:** It indicates when the Trigger's code is executed. i.e., Before/After.
2. **Trigger event:** The Trigger event should be Insert, delete, and Update.
3. **Trigger action:** The Trigger's code is called Trigger action. Here the Trigger uses two types of variables. They are *":OLD"* and *":NEW"*. These variables are called "Bind variables".
4. **Trigger level:** The Triggers are used in two levels. They are
   i. **Statement-level Triggers:** The Triggers that will affect multiple rows of a specific table will be called as Statement-level Triggers. These Triggers are also called *Base-level (or) Table-level* Triggers.
   ii. **Row-level triggers:** The Triggers that will affect current row of a specific table will be called as Row-level Triggers.

Creating a Trigger:

A Trigger is created by using the following syntax.

Syntax:

```
Create or replace Trigger Trigger-Name Before/After Insert/Delete/Update
on Table-Name [for each row]
Declare
        Local variable declaration;
Begin
        PL/SQL statements;
End;
```

Ex:

```
Create or Replace Trigger Sun-Trig after insert on emp
Declare
        a varchar2(15);
Begin
        a:=TO_CHAR(SYSDATE, 'DY');
        if a= 'SUN' then
                Raise_Application_Error (-20000, 'Sunday transactions are invalid');
        End if;
End;
```

Enable/Disable a Trigger:

A Trigger can be either *Enable* (ON) or *Disable (OFF)*. By default all triggers are enabled.

Syntax:

Alter Trigger Trigger-Name Enable/Disable;

Ex: *(1). Alter Trigger Sun_Trig Disable;*

(2). Alter Trigger Sun_Trig Enable;

Dropping a Trigger:

The "Drop Trigger" command is used to remove the Trigger from the database.

Syntax:

Drop Trigger Trigger-Name;

Ex: *Drop Trigger Sun_Trig;*

Advantages of Triggers:

- ✓ Triggers are used to provide constraints on database.
- ✓ Triggers are used to prevent invalid transactions.
- ✓ A Trigger can permit transactions only during business hours.
- ✓ A Trigger can also be used to keep audit details of a table.
- ✓ A Trigger can also be used to create backup copies.

Dear one

If U have a mistake, u never answer to any one in the world, but u must answer to your inner soul

By

DVH. Venu Kumar *M.Sc., M.Ed., M.Tech.,*

Asst.Professor

GIST.