

BASIC structure of SQL

- * SQL is a structured query language. It is a standard language for communicating with relational database management system. SQL statements are used to retrieve data from the database as well as perform tasks such as adding, updating and deleting data from the database.
- * A typical structure of SQL query is,

```

select A1, A2, A3, ... An
from r1, r2, r3 ... rn
where (p) [predicate]

```

A₁, A₂, ... A_n represents an attributes.

r₁, r₂, ... r_n represents a relation.

p is a predicate (condition).

- * We are using mainly three clauses in a SQL query (select, from, where)

(i) select clause :- It list the attributes desired in the result of query.

- * This clause similar to the projection operation of the relational algebra.

- * SQL names are case-insensitive (we are using upper, lower cases).

- * SQL allows duplicates in relations as well as in query results.

- * Elimination of duplicates, insert the keyword distinct after select clause.

An asterisk (*) in the select clause denotes all attributes.

* select clause used for arithmetic expressions (+, -, *, /) operating on constants (or) attributes of tuples.

Ex: tablename: Loan

loan number	Branch name	amount
L-11	Round Hill	9000
L-14	Down town	15000
L-15	Perry ridge	15000
L-16	Down town	9000
L-17	Red word	10,000

1) Query: find the names of all branches in the loan relation.

Query: SQL > select branchname from loan;

o/p: Displays only branchname.

2) find the names of all branches in the loan relation and remove duplicate names.

Query: SQL > select distinct branchname from loan;

o/p:

branchname
Round Hill
Down town
Perry ridge
Red word

3) find the names of all branches in the loan

Query: SQL > select all branch name from loan;

o/p:

Branch name
Round Hill
Down town
Perry ridge
Red word

4) > select * from loan.

o/p: Displays the loan table.

5) List the tuple of loans of relation with amount multiplied by 100.

o/p:

amount
900000
1500000
1500000
900000
1000000

Query: SQL > select amount * 100
from loan;

Where clause:-

- * It specifies condition that the result must satisfy
- * the where clause is similar to the selection operation in relational algebra.
- * It is the expression that controls the which rows appear in the resulting table.
- * the SQL uses (AND, OR) operators, Comparison operators (<, >, <=, >=, ==) and also used b/w, b/w and, not b/w operators.

Ex:- Find the loan number from the relation loan with loan amount b/w 10,000 to 15000.

SQL > select loan number from loan where amount b/w 10,000 and 15000;

o/p:

Loan number
L-14
L-15
L-16
L-17

From clause:- It specifies the table names that will be queried to retrieve the desired data.

Ex:- Find the all attributes from the loan relation.

Query: > select * from loan;
Display the loan table.

and set operations in SQL:-
 set operators are used to compile the results of two (or) more select statements.
 SQL supports few set operations which can be performed on the table data.
 * These are used to get meaningful results from data stored in the table.

the set operations are,
 union, unionall, Intersect, minus (set difference),
 except, exceptall

union:- It is used to combine the result of two (or) more select statements. It will eliminate duplicate rows from its result set. If you want perform union operation, we must follow the rules.

* same no. of columns should be selected from both the tables.

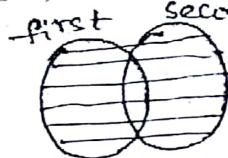
* the same no. of column expressions.

* The same data type, and have them in the same order.

Ex:-

ID	Name
1	abhi
2	adam

ID	Name
1	adam
2	chester



query:- select * from first;
 union
 select * from second;

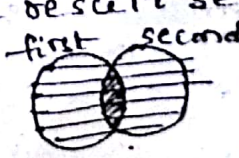
ID	Name
1	abhi
2	adam
3	chester

union all:- This operation is similar to union. But, it allows duplicate rows to the result set.

Ex:-

ID	Name
1	abhi
2	adam

ID	Name
1	adam
2	chester

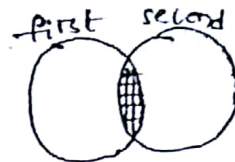


Query ÷ > select * from first;
union all
> select * from second;

ID	Name
1	Abhi
2	Adam
2	Adam
3	Chester

Intersect ÷ It is used to combine two select statements but it only returns the records which are common from both select statement (In case of Intersect the no. of columns and data type must be same in both tables)

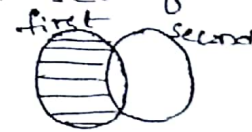
Query ÷ > select * from first;
Intersect
> select * from second;



ID	Name
2	Adam

Minus ÷ The minus operation combines the results of two select statements and return only those in the final result which belongs to the first set of the results (first table data).

Query ÷ > select * from first;
minus
> select * from second;



ID	Name
1	Abhi

Except ÷ It was introduced in 2005. It is used to achieve 'distinct' and 'not in' queries. Except operator returns all distinct rows from Left hand side table which doesn't exist in right hand side table.

Query ÷ > select * from first;
Except
> select * from second;



ID	Name
1	Abhi

Except all ÷ Which returns all records from the first table which are not written in second table. but except all operator doesn't remove duplicate records.

query: $\rightarrow$ select * from first;

except all

$\rightarrow$ select * from second;

first second



Id	name
1	abhi
1	abhi

- Nested / Sub / Inner query :- A sub query (or) inner query (or) a nested query is a query within another SQL query and embedded with in the where clause. The inner query is used to determine the results of the outer query.

Ex: find the name of a student who has minimum age.

query: $\rightarrow$ select name, min(age) from student;

student	
sname	age
A	12
B	19
C	24
D	19
E	30

nested query: $\rightarrow$ select sname, age from student where age $\geq$ (select min(age) from student);

- Aggregate operators :-

Aggregate functions that take a connections of values as input and return a single value to the output. The SQL aggregate operators are.

(1) Avg () :- It returns the Avg value of a numeric column.

syntax: select Avg(columnname) from tablename;

order no	orderid	products	quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

Ex: select Avg(quantity) from order;

(2) min() :- It returns minimum value of a column.

Syntax :- select min(columnname) from tablename;

Ex :- select min(quantity) from order;

(3) max() :- It returns maximum value of a numeric column.

Syntax :- select max(columnname) from tablename;

Ex :- select max(quantity) from order;

(4) sum() :- It returns total sum of a numeric column.

Syntax :- select sum(columnname) from tablename;

Ex :- select sum(quantity) from order;

(5) count() :- It returns the no. of rows that matches a specified criteria.

Syntax :- select count(columnname) from tablename;

Ex :- select count(quantity) from order;

Correlated Query (or) Subquery :-

The subquery is a query that contains inside another query. The inner query is used to determine the results of outer query. In this correlated nested queries, the outer query is executed first.

Difference b/w nested query, correlated nested query :-

Nested query

* Inner query is executed first.

* Inner query is executed only once and result is used by outer query.

Correlated nested query

* Outer query is executed first.

* Inner query is executed for each of the records that outer query returns.

using comparison operators ($>$, $<$, $>=$, $<=$, $=$), in operator, b/w operator.

* Always used in the where clause.

Ex: $\rightarrow$ select sname, age from student where age = (select min(age) from student);

* uses comparison operators, in, b/w, exists and not exist, any.

* It is used in the where clause as well as columns of select statement with operators.

Ex: SQL $\rightarrow$ select sname, age from student where age exists (select min(age) from student);

• Null values:—

* The SQL NULL is a keyword used to missing value in a field. It also represent blank space (not given any variable).

* A field with a null value is a field with no value.

Ex: Create table custor (ID number not null, name varchar(20) not null, age number, address varchar(20), sal number);

customer				
ID	name	age	address	sal
1	Ramesh	32	Hyd	2000
2	Suresh	25	Delhi	1500
3	Kaushal	23	mumbai	3000
4	Komal	25	Pune	-
5	manu	27	Rajasthan	-

Here, not null represents that column should take values, and we do not use not null keyword that column takes null values.

Syntax: SQL $\rightarrow$ select id, name, age, address, sal from customer where sal is not null;

Op:

ID	name	age	address	sal
1	Ramesh	32	Hyd	2000
2	Suresh	25	Delhi	1500
3	Kaushal	23	mumbai	3000

Syntax: SQL $\rightarrow$ select id, name, age, address, sal from

o/p:-

ID	name	age	address	sal
4	Romal	25	pune	-
5	marce	27	Rajasthan	-

Logical connectivity operators:-

there are three logical operators AND, OR, NOT. This operators compare two conditions at a time to determine whether a row can be selected for the output. when retrieving data using a select statement, we can use the logical operators in the where clause, which allows to combine more than one condition.

AND:-

If we want to select rows in a table that must satisfy all the given condition.

Ex:-> select id, name, age, sal
from customer where
age $\geq$ 25 and age $\leq$ 32;

ID	name	age	address	sal
1	Ramesh	32	Hyd	2000
2	aresh	25	Delhi	1500
3	Koushal	23	mumbai	3000
4	Romal	25	pune	4000
5	marce	27	Rajasthan	4500

OR:-

If you want to select rows that satisfying atleast one of the given conditions.

Ex:-> select id, name, age, sal
from customer where name = 'Ramesh'
or name = 'Koushal';

ID	name	age	sal
1	Ramesh	32	2000
3	Koushal	25	3000

NOT:-

If you want to find rows that don't satisfy a condition we can use logical operator NOT. The NOT operator rejects in the result of a condition. That

if a condition is satisfy then the row is not returned.

Ex: select id, name, age, sal from customers where not sal = 7000;

• Complex integrity constraints:

constraints over single table:— complex constraints over a single table (or) specified using a table constraints. which has the form check conditional expression.

syntax:— create table sailors (sid int, sname varchar(10), rating int, age int, primary key (sid), check (rating >= 1 and rating <= 10));

Domain Constraints:

A user can define a new domain using the create domain statement, which makes of check constraints.

syntax:— create domain age int default 18;

syntax:— create domain age int check (age >= 18 and age <= 25);

Integer is the datatype of a age domain that can be restricted by check constrain. In this above example age attribute is the domain.

Assertion:— (ICS over several table)

When a constrain involves two (or) more tables, the mechanism becomes so difficult to overcome this problem the SQL introduced the concept assertion. Assertions are constraints which are not associated with any one table.

Syntax :-

SQL > create table sailors (sid int, sname var^{character}
rating int, age int, primarykey(sid), ⁽¹⁰⁾ check
(rating >= 1 and rating <= 10), check((select
count(s.sid from sailors s) + (select count
(p.pid) from person) < 100));

using Assertion :-

Syntax :- create assertion ^{assertion name} smallclub

(10),
check((select count(s.sid) from student s) +
(select count(p.pid) from person p) < 100);

Triggers in active data base :-

- * Triggers are stored programs, which are automatically executed (or) fired when some events occur.
- * A trigger can be defined as a program that is executed by DBMS whenever updations ~~are~~ specified on database tables.
- * It is like an event which occurs whenever a change is done to the tables (or) columns of the tables.
- * only DBA can specifies the triggers.
- * Triggers are written to be executed in response to any of the following events.
 - (i) A database manipulation ^(DML) statement (Delete, insert, update).
 - (ii) Data definition statement (DDL) -> (Create, Alter, Drop)
 - (iii) A Database operations -> server error, log on, log off, startup, shutdown
- * Trigger could be define on the table, view, schema (or) database with which the event is associated.

General form of the trigger:-

the general form of the trigger includes the following.

i) Event

ii) Condition

iii) Action

Event:- It describes the modifications done on the database which lead to the activation of trigger. The events are DDL, DML, Database operations.

Condition:- These are used to specify the actions to be taken when the corresponding event occurs and the condition evaluates true then respect to action to be taken otherwise, action is rejected.

Action:- Action specifies the action to be taken place at when the corresponding event occurs and the condition evaluates to true. An action is a collection of SQL statements that are executed as a part of trigger activation.

Structure of Trigger:-

create or replace trigger <triggername>
Before or after or Instead of

Insert or update or Delete on <tablename>
for each row

which <condition for trigger to get execute>
declare

<declaration part>

begin

<execution part>

-Exception

<exception part>

end;

The following program creates a row level trigger for the customer table that would be fired for insert (or) update (or) delete operations performed on customer table. This trigger will display the salary difference b/w old and new salary statements.

create or replace trigger SalDiff

Before
delete or insert or update on customer
for each row
when (new.ID > 0)

customer			
ID	name	age	Sal
1	xyz	23	2000
2	PS9	25	5000
3	abc	32	3000

Declare

sal_difference number;

begin

sal_difference := :new.sal - :old.sal;

dbms_output.put_line ('oldsal:', ||:old.sal);

dbms_output.put_line ('newsal:', ||:new.sal);

dbms_output.put_line ('saldifference:', ||:sal_diff);

end;

/

→ Trigger Created.

→ We want to insert a row in a customer table.

Insert Query:

Insert into customer values ('4', 'neha', '33', 5000);

oldsal := —

newsal := 5000

saldiff := —

update the table, the salary incremented by 500

Types of Triggers :-

Triggers are classified based on levels they are two levels of Triggers to be invoked. They are.

statement-level trigger:- These triggers executed only once for multiple rows which are effected by trigger action.

row-level trigger:- These triggers executed by for each row of the table that is effected by the event of triggers. The row level triggers are.

i) Before trigger

ii) After trigger

Before trigger:- It execute before the triggering DML statements.

After trigger:- It execute After the triggering DML statements executed.

Benefits of triggers :-

- * Generating some derived column values automatically.
- * Enforcing referential integrity.
- * Event logging and storing information and table Access.
- * Auditing
- * synchronous replication of tables.
- * Imposing security authorization.
- * preventing invalid transactions.