17/11/21

UNIT-III

SQL:

1. Find the name and age of all sailors

select .DISTINCT (To use only one time in relation)

SELECT ⟶ from select list

FROM ⟶ from from list

Qualification ⟶ from where clause

```
select distinct name, age
from sailors
```

2. Find all sailors with rating above 7.

```
select * from sailors
where rating >7
```

3. Find the name of sailors who have reserved boat 103.

```
select s.sname
from sailors S, Reserve R
where S.sid = R.sid AND R.bid = 103
```

4. Find the name of sailors who have reserved red boat.

```
Select s.sname
From sailors S, Reserve R, Boat B
where B.colour = Red AND B.Bid = R.Bid
AND S.sid = R.sid
```

5. Find the colour of boats reserved by lubber.

```
Select   B.colour
From     Sailor S, Reserve R, Boat B
where    S.sname = lubber ! S.sid = R.sid
         AND  R.bid = B.bid.
```

6. Find the name of sailors who have reserved atleast one boat

```
Select   S.sname
From     Sailor S, Reserve R
where    S.sid = R.sid
```

25/11/21

Expression and strings:-

Complete increaments for the rating of person who have sailed two different boats on the same day.

```
select   rating+1 as rating, name
From     sailors, reserve
where    S.sid = R.sid  AND
                Date = 9/8/98
```

Each item in a select list can be a form of expression as column name, where expression is arithmetic or string expression over the column names.

→ The string expression we can use comparsion operators.

→ If we need to sort strings by order other than alphabets.

Like operator:-

SQL supports pattern matching through the like operator along with the use of wilde card (or) % that symbol represent 0 or more characters or arbitary characters and "-" (hyphen) symbol for exactly one arbitary character.

'Like a=%'

'Like 'a%'

Like '-%'

Like '%a'

→ Like 'a%' find any values that states with a character of string.

→ LIKE '%a' find any value that will be last character of string

→ Like '-a%' its means to find the second position of character

→ Like 'a-%' find two characters of string

→ 'a--%' three characters of string

29/11/21

Find the name of sailors who have reserved red or green boat.

```
Select  S.Sname
From    sailors S, reserver boat B
where   S.sid = R.sid AND R.bid = B.bid
        AND colour = 'Red'
```

UNION

```
Select sname
From   ||   where S.sid = R.sid AND R.bid = B.bid
                AND colour = 'Green'
```

[OR]

S.Sid = R.sid AND R.bid = B.bid

Find the name of sailors who have reserved red and green boat

```
Select  S.Sname
From    Sailors S, reserve R, Boat B
where   S.sid = R.sid AND R.bid = B.bid
        AND colour = "Red"
```

INTERSECTION

```
Select  S.Sname
From    sailors S, reserve R, Boat B
where   S.sid = R.sid AND R.bid = B.bid
        AND colour = "Green"
```

Find the sid of all sailors who have reserved red colour boat but not green colour boat

```
Select   snames.sid
From     Sailors S, Reserve R, Boat B
where    S.sid = R.sid AND  R.bid = B.bid
                AND colour "Red"
```

EXCEPT

```
select   S.sid
From     sailors S, Reserve R, Boat B
where    S.sid = R.sid AND B.bid = B.bid
                AND colour = "Green"
```

Nested query's / queries :-

A nested query is the query that has another query embedded with in it. The sub query is applied in the where class

Ex :- Find the name of sailors who have reserved boat 103.

Nested :-

```
select   S.sname
From     sailors
where    S.sid IN( select   R.sid
                   From     reserves
                   where R.bid = 103)
```

General :-

```
select   S.sname
From     sailors, reserve
where    S.sid = R.sid   AND   R.bid = 103
```

Find the name of sailors who have not reserved a red boat.

```
Select   s.sname
From     sailors
where    s.sid  NOT IN [ select  R.sid
                         From    Reserver, Boats
                         where   B.colour = red
                                 R.bid = B.bid ]
```

(s.sid ↓ main query)

↑ nested query (subquery)

30/11/21

Corelated nested query:-

- nested query should be independent

Find the name of sailors who have reserved boat number 103.

~~select * from reserves~~

```
select  sname
From    sailors
where   s.sid IN ( select * from reserves
                   where  R.bid = 103 )
```

set comparision operators:-

The SQL supports IN, EXIST, UNIQUE, ANY AND ALL operations

(all records will display)

Find the sailors whose rating is better than some (any) sailors called horatio

```
select  s.sid
from    sailors
where   s.rating > any ( select rating
                         from sailors
                         where s.name = harratia)
```

Find the sailors with the highest rating:

```
select  s.sid
from    sailors
where   s.rating >= ALL (select rating
                         from sailors)
```

1/12/21

Aggregate operators :- (in select class)

1. COUNT
2. SUM
3. AVG
4. MIN
5. MAX

[(DISTINICT) A] → A represents field of relation

Find the average age of all sailors.

```
select  Avg (s.age)
from    sailors s
```

Find the average age of sailors with rating 10

```
select  Avg (s.age)
from    sailor s
rating
where   s.rating = 10
```

Find the name and age of oldest sailor.

```
select  MAX (age), s.name
from    sailor s
```

Count the no. of sailors → all records

```
select  count (*)
from    sailors s
```

Count the no. of different sailors name

```
select  count [(DISTINCT) s.name]
from    sailors s
```

2|12|21

Group by and having clause

```
select [Distinct] select list
From  from clause
qualification - where class
group of rows - group by
where qualification - having clause
```

Find the age of youngest sailors for each rating level.

```
select  Min [s.age]
From    sailors
where   rating
group by rating
```

→ The select list in the select class consists of a list of column names and list of terms the form aggregate operator of columnn name as new name

→ we already saw as used to rename the output columns

→ every column that appears in group list, the reason is that each row in the result of the query corresponding to one group.

→ The expression appearing in the group qualification is a having clause and they must have a single value for group.

For example:-

Find the age of youngest sailor whose is eligible to vote and for each rating level it atleast two took such sailors.

```
select  s.rating; min(s.age) As newage
From    sailors
where   s.age >=18
group by s.rating
having  count(*) > 1 ;
```

4/12/21

Procedural language/ structural query language :-

- declare (declaration statement) variables
- begin (executable statements)
- expectations (exceptional handling) - run time errors
- end

/ → program will be executed with PL/SQL;

Example:-

Addition of two numbers

```
SQL > Add two number
SQL > Declare
        a   number(10);
        b   number(10);
        c   number(10);
SQL > Begin
        c = a+b;
      dbms_output.put_value("sum of values : " || c);
      end;
      /
```

→ PL/SQL is basically procedural language which provides functionality of decision making.

→ The PL/SQL can execute a number of queries in one block using single command.

→ once can create PL/SQL unit such as procedures, functions, triggers, cursers, which stored in database

for reuse by applications.

→ PL/SQL made up of 4 blocks.

Triggers in active database:- (automatically run when an event occurs in database server)

→ Triggers are a program which is automatically executed when some events occured.

→ A Trigger can be defined as program that is executed by dbms.

→ Trigger is like an event which occurs whenever a change is done to the tables or columns of the table.

→ Triggers are written to be executed in DDL, DML and database operations.

General forms of triggers:-

1. Event
2. Condition
3. Action

→ THE EVENT it describes the modification terms on DB which lead to the activation of the trigger. The events are DDL, DML and data operation.

→ Condition:- This are used to specify the actions to be taken when the corresponding event occurs. The condition evaluates true. The respect action to be taken otherwise rejected.

→ ACTION specifies when the corresponding event occurs and condition evaluates to true.

- An 'action' is collection of sql statements that are executed a part of trigger.

syntax/structure of trigger:-

```
create or update trigger <triggername>
before or after of
insert or update or delete on <table name>
For each row
which<condition for trigger to get execute>
Declare
<declaration part>
Begin
<execution part>
Exception
<exception part>
End;
/
```

7/12/21

Triggers

Following program creates a row level trigger for the customer table that would be for – insert, update, delete operations. Performed on the customer table. This trigger will display the salary differences between old & new salary statements.

```
- create or replace trigger salary
   before on co customer
   for each row
   when(new.ID >0)
Declare
   sal_diff number;
begin
   sal_diff := :new.sal - old sal;
dbms_output.put_values ("old sal:", ||:old sal);
dbms_output.put_values ("new sal:", ||:new sal);
dbms_output.put_values ("new sal:", ||:sal_diff);
end;
/
```

update the table. salary incremented by 1000 Rs:

```
create or replace trigger salary
before update on customer
for each row
```

```
Declare
    Total_rows number;
begin
    update customer;
    set salary = salary + 1000;
If SQL% not found then
    dbms_output.put_values("not updated");
elseif SQL% found
    total_rows = sql% row count;
    dbms_output.put_values("updated");
endif;
end;
/
```

8/12/21

Scheme refinement & Normal forms:-

1. Redundancy (multivalued attri
2. Update anomolies (problem)
3. Insert anomolies
4. Delete anomolies

Employee:-

Eid	Ename	salary	age	hourly_wages	hours_worked
111	AA	25,000	8	10	40
222	BB	30,000	8	10	30
333	CC	35,000	5	7	35
444	DD	40,000	5	7	42
555	EE	45,000	8	10	32

Emp(Eid, Ename, salary, age, hoursworked)

wages(age, hourly_wages)

age	hourly_wages
8	10
5	7

Eid	Ename	salary	age	hour worked
111	AA	25,000	8	40
222	BB	30,000	8	30
333	CC	35,000	5	35
444	DD	40,000	5	42
555	EE	45,000	8	32

Schema Refinement :-

The main cause of schema refinement eliminate the redundant storage by the decompose relation.

- The problem caused by redundancy.
- Redundancy is a method of storing the same information repeatedly. It means storing the same data more than 1 place within the database and it can be lead several problems then the problems are

1. Redundancy
2. Update anomolies
3. Insert anomolies
4. Delete anomolies

1. Redundancy or Redundancy storage:-

It removes the multi-value attributes, It means some tuples or information (values) is stored repeatedly.

2. Update anomolies:-

Suppose if we update 1 row then the dbms will update more than 1 rows.

3) Insertion Anomolies:-

When allows insertion for already exist records again inserted.

4) Deletion Anomolies:-

When more than one record is deleted instead of specified one.

- overcome these problems using decomposition.
- The decomposition is a relation schema R can be decomposed into a collection of smaller relations ($R_1, R_2, R_3 \ldots R_n$) to eliminate redundant information.

$R_i \subseteq R$

9/12/21

Functional dependency:-

The functional dependency is a relationship that exists between 2 attributes. The 2 attributes are prime key, non key attribute in the table.

Ex:- A → B

determine (or) dependency

(→) = determine/ dependency symbol

Emp Id → Empname

Emp Id → known as prime key

Empname → Non-key

Functional dependency → 2 types

1. Trivial functional dependencies
2. Non-trivial "

If A → B has trivial F.D

If B is a subset of A

Ex:- {Emp Id, Empname}

Empname determines emp Id.

$A \rightarrow B$ has non trivial F.D

If B is not a subset of A

Ex:- {Address, Name} $\rightarrow$ DOB Name $\rightarrow$ DOB

FD Inference rule (IR)

$\rightarrow$ Armstrong Axioms

The armstrong axioms are the basic inference role and it is used to conclude functional dependency on the relational database.

$\rightarrow$ The inference rule is a type of assertion and it can be apply to set of functional dependency to derive other functional dependency

$\rightarrow$ In functional dependencies there are 6 types of IR's

1. Reflexive Rule (IR_1)

If $X \geq Y$ then $X \rightarrow Y$

Ex:- $X = \{A, B, C, D, E\}$; $Y = \{a, b, c\}$

2. Augmentation Rule (IR_2)

(partial dependency)

If $X \rightarrow Y$ then $XZ \rightarrow YZ$

$R(ABCD)$, If $A \rightarrow B$ then $AC \rightarrow BC$

3) Transitive rule (IR_3)

$X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

4) union rule (IR_4)

$X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

5) Decomposition rule (IR_5)

$X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

6) pseudo transitive rule (IR_6)

If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$

13/12/21

Normalization:-

- organizing the data
- Reduce the redundancy
- eliminate the undesirable characteristics
- large tables to small table.

1st normal form:-

- Automatic values (not multiple value, single value only).
- cannot use multivalue attribute
- Must use single value attribute in the value
- disallow multivalue attribute & composite key.

2nd normal form:-

SID	Sname	sphone.no	sage
1	A	123456, 78910	10
2	B	11121314	20
3	C	15161718, 19202122	30

After normal applying normal values (Atomic values):-

1	A	123456	10
1	A	78910	10
2	B	11121314	20
3	C	15161718	30
3	C	19202122	30

2nd Normal form:-

- 2NF is in 1st NF
- All non-key attributes are fully functional dependency of primary key.

Teacher id	subject	Tage
20	A	30
20	B	30
30	C	35
40	D	34
40	E	38

T.Id → T.age

T.Id → subject

Decomposition.

- Decompose the table or relation.

T.Id	T.age
20	30
30	35
40	38

T.Id	subject
20	A
20	B
30	C
40	D
40	E

Composite key:-

Composite key is a primary key composed of multiple columns, used to identify record uniquely.

14/12/21

3rd NF:-

$x \to y$ $\quad y \to z$ $\quad x \to z$

1. 3NF, it is in 2NF
2. Not contain any transistive dependency
3. Reduce the data duplication.

non-prime attributes

Eid	Ename	E.zipcode	E.city	E.state
1	A	101	AP	AM
2	B	102	UP	Noida
3	C	103	UP	Noida
4	D	104	MP	Bhopal
5	E	105	TS	Hyd

conditions:-

1) x is a super key
2) Y is a prime key

superkey → {eid}, {eid, ename}, {eid, ename, ezip}.....

C.K → {Eid}

4. A relation is in 3rd NF it Holds at least one of the following condition for non-trivial functional dependency of $X \rightarrow Y$

i) X is a super key

ii) Y is a prime key (part of candidate key)

C.k is superset of Super key

e.id	e.name	e.zipcode
1	A	101
2	B	102
3	C	103
4	D	104
5	E	105

e.zip	e.city	e.state
101	AP	AM
102	UP	noida
103	UP	noida
104	MP	Bhopal
105	TS	hyd

e.id, e.name e.zipcode } super key

e.zip e.city, e.state

{e.id, zip} → candidate key. (prime key)

{e.zip} → {e.city, e.state}

{e.id} → {e.zip}

{e.id} → {e.city, e.state}

Non-prime attributes are functionally dependent on prime key

15/12/21

BCNF :-

- Bcnf - boyce codd normal form is stricter than NF
- Every FD is $X \rightarrow Y$, x is always superkey.

Ex :- There is a company where employees works more than one department.

Eid	E.country	E.dept	Dept-type	Emp-dept-no
200	In	Design	D3	20
200	In	Test	D3	31
300	UK	store	D2	30
300	UK	Devoloping	D2	31

Super key :- Eid $\rightarrow$ E.country

E.dept $\rightarrow$ {Dept-typ, Emp-dept-no}

candidate key :- {Eid, E.dept}

E.dept	Dept-type	Emp-dept-no
Design	D3	20
Test	D3	21
store	D2	30
devoloping	D2	31

{E.dept}

Eid	E.dept
200	Design
200	test
300	store
300	devoloping

{Eid, E.dept}

Eid	E.C
200	In
300	UK

{Eid}

properties of decomposition:-

- lossless join decomposition - $R \Rightarrow R_1$ & $R_2 \Rightarrow R_1 \bowtie R_2 \Rightarrow R$
- dependency preserving decomposition

$R(ABCD) \rightarrow R_1(ABC)$ & $R_2(AD)$

$A \rightarrow BC$ $\quad$ $A \rightarrow D$

$A \rightarrow B$

$A \rightarrow C$

16/12/21

4th normal form:-

- the relation will be in BCNF
- cannot allows multiple value dependency (MVD)

stu-id	subject	Hobby
21	Com	Dance
21	maths	sing
34	chem	dance
74	Bio	cricket
59	phy	Hockey

$A \rightarrow B$

$\downarrow 1$ $\quad$ $\downarrow$ 2,3

PK

2,3 is depended on PK where a) 2,3 are independent of each other (multi value dependency)

stu-id	subject
21	com
21	maths
34	chem
74	bio
59	phy

stu-id	hobby
21	Dance
21	sing
34	dance
74	c.k
59	Hockey

5th normal form:- (or) project join NF (PJNF)

- A relation is in 4th NF
- not contain any join dependency
- Join should be a lossless
- To avoid redundancy.

subject	lecture	semister
com	A	1
com	B	1
math	B	1
math	C	2
chem	D	1

Decompose the table $R \longrightarrow R_1, R_2 \& R_3$

R_1 - {sem, subject}

R_2 - {sub, lecture}

R_3 - {sem, lecture}

$R \longrightarrow R_1 \bowtie R_2 \longrightarrow R$

sub	lecture
com	A
com	B
math	B
math	C
chem	D

sem	subject
1	com
1	math
2	math
1	chem

sem	lecture
1	A
1	B
2	C
1	D