UNIT-III

SQL: QUERIES, CONSTRAINTS, TRIGGERS.

- form of basic SQL query
- UNION, INTERSECT and EXCEPT
- Nested querries, aggregation operators, NULL values
- Complex integrity constraints in SQL
- triggers and active databases.

SCHEMA REFINEMENT:-

- Problems caused by redundancy
- decompositions
- Problems related to decomposition
- reasoning about functional dependency.
- FIRST, SECOND, THIRD normal forms,
- BCNF, (Boyce-codd Normal form)
- lossless join decomposition
- multi-valued dependencies
- FOURTH Normal form
- Fifth Normal form.

Form of basic SQL query:-

The basic form of SQL query is
↳ 3 clauses.

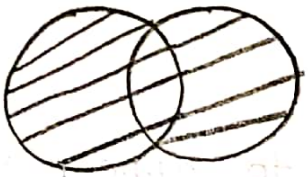
SELECT [DISTINCT] {*/ column_name (, column_name, ...)}
FROM table_name [alias] (, table_name)
[WHERE condition]
[GROUP BY column_list] [HAVING condition]
[ORDER BY column_list].

3 clauses { select A_1, A_2, A_3 / from r_1, r_2, r_3 / where P

- SELECT ⇒ which columns are to appear in the output.
  ↳ corresponds to projection operation.
- DISTINCT ⇒ eliminates duplicates
- FROM ⇒ specifies the tables to be used.
  ↳ corresponds to cartesian product operation.
- WHERE ⇒ filters the rows according to condition.
  ↳ corresponds to select predicate
  The where condition is a boolean combination (using AND, OR and NOT) of conditions of the form expression op expression, where op is one of the comparision operators. (<, <=, =, >=, >)
- GROUP BY forms groups of rows with the same column value.
- HAVING filters the group
- ORDER BY sorts the order of the output.

UNION, INTERSECT, EXCEPT (minus)

UNION: It is used to combine the results of two or more SELECT statements. It will eliminate duplicate rows from its resultset. UNION operation is applied when columns and data type must be same in both the tables.



Example:- Table 1

ID	Name
1	abhi
2	adam

Table 2

ID	Name
2	adam
3	chester

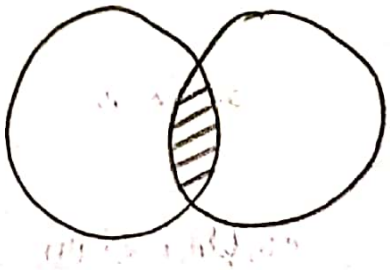
Query:- select * From first UNION select * From second;

Result:-

ID	NAME
1	abhi
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 statements. In case of intersect the number of columns and datatype must be same.



Example:-

ID	Name
1	Abhi
2	adam

ID	Name
2	adam
3	chester

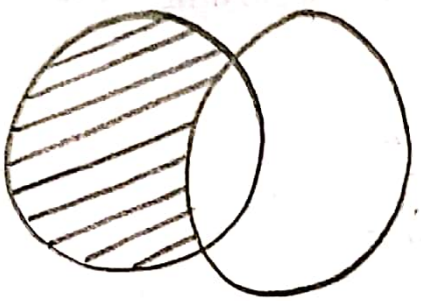
Query:- Select * FROM first INTERSECT select * From second;

out put:-

ID	Name
2	adam

Except:-

It combines results of two select statements and return only those in final result, which belongs to first set of result.



Example:-

ID	Name
1	abhi
2	adam

ID	Name
2	adam
3	chester

Query:-

```
Select * From First MINUS select * from Second;
```

output:-

ID	Name
1	abhi

Triggers:-

A trigger is a SQL procedure that initiates an action. when an event occurs. event is INSERT, DELETE and UPDATE.

It has 3 parts.

① Event (E)

② Condition (C) ⇒ optional (whether the trigger should run or not)

③ Action. (A) ⇒ what changes should be made

Syntax:-

```
Create [or Replace] Trigger trigger_name
{Before/After} trigger event ON tablename
[for each row]                                  } Trigger Procedure.
[when condition] - - - - - - - - - - -
Declare
        declaration statements;
Begin
        executable statements;
Exception
        exception handling stmts;
END;
```

Example:- SQL> select * from customers;

ID	Name	age	address	salary
1	Hari	32	Delhi	20k
2	Priya	35	Hyd	30k
3	Raghu	30	Mumbai	40k
4	Deepu	32	Pune	50k
5	Pinky	37	Madhya	40k.

* Update the salary by triggering table. (update)

⇒ Trigger displays the salary difference b/w old values and new values.

trigger name ↑

```
create or replace trigger display_salary_changes
Before Delete or Insert or update on customers
                                    ↓ tablename
for each row
 when (New.ID > 0)
DECLARE
      sal_ diff number;
BEGIN
      sal_ diff := :New salary - :old salary;
 dbms_output.put_line ('old salary :');
      "     "     "  ("new salary :");
      .     .     .  ("sal difference");
END;
```

o/p:- Trigger created.

Aggregation Operators:-

An aggregate operator in the relational model is an operator that derives a single value from the set of values appearing within some relation.

- Avg – calculates the average of set of values.
- COUNT – counts rows in a specified table or view.
- MIN – Gets the minimum value in a set of values.
- MAX – Gets the maximum value in a set of values.
- SUM – calculates the sum of values.

NULL value:-

It is the term used to represent a missing value. A field with a NULL value is a field with no value.

- It is important to understand that NULL value is different from zero.
- Null value has three different interpretations.

> value unknown (value exists but is not known)

> value not available (exists but is purposely withheld)

Eg:-

Fname	Mname	Lname
Amit	NULL	Kumar

> value may not be applicable for that particular attribute.

Complex integrity Constraints in SQL:-

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into table.

constraints could be either on a column level or a table level.

① NOT NULL Constraint - Ensures that a column cannot have NULL value.

② DEFAULT constraint - Provides a default value for a column when none is specified.

③ UNIQUE constraint - Ensures that all values in a column are different.

④ PRIMARY key - Uniquely identities each row/record in a database table.

⑤ FOREIGN key :- Uniquely identities a row/record in any of the given database table.

⑥ INDEX :- Used to create and retrieve data from database very quickly.

Complex Integrity Constraints in SQL:

Integrity constraints

→ Check
→ NOT NULL
→ UNIQUE
→ Default.

Schema refinement in dbms:

Schema refinement (Polishing) is the process that re-defines the schema of a relation.

→ Refinement approach is based on decomposition (dividing)
→ Decomposition can eliminate the redundancy.

Schema-refinement addresses the following problems:

① Problems caused by redundancy
② Problems caused by decomposition

R (A B C D)

student (sid, sname, sage)

* TO avoid redundancy.

Decomposition: Let R be a relational schema.

A set of relational schemas $(R_1, R_2, R_3 \cdots, R_n)$ is a decomposition of R if

$R = R_1 \cup R_2 \cup \cdots R_n$

Eg 1- Relation $R(x, y, z)$

$R_1(x, y)$ & $R_2(y, z)$

Now we get

$R = R_1 \cup R_2$

Problem with decomposition:-

R

Model name	Price	category
a_{11}	100	canon
s_{20}	200	Nikon
a_{20}	150	canon

R_1

Modelname	category
a_{11}	100
s_{20}	200
a_{20}	150

R_2

Price	category
100	Canon
200	Nikon
150	Canon

$R_1 \cup R_2$

Modelname	Price	category
a_{11}	100	Canon
a_{11}	150	canon
s_{20}	200	Nikon
a_{20}	100	Canon
a_{20}	150	Canon

$R_1 \cup R_2 \neq R$

Functional dependency :-

The attributes of the table are said to be dependent on each other when an attribute of a table uniquely identifies another attribute of same table.

Eg:- We have student table with attributes : stu_id, stu_name, stu_age. Here stu_id attributes uniquely identifies student name attribute of a student table because if we know stu_id we can say name of the student easily. It can be written as stu_id → stu_name.

* If column A of table uniquely identifies the column B then it can be represented as $A \rightarrow B$ (B is dependent on A)

A	B
1	1
2	1
3	2
4	3

symbol →

A → B
↓ ↳ dependent
determinant

If A = 2 then B = 1

- If in two tuples $t_1 \cdot A = t_2 \cdot B$ then $t_1 \cdot B = t_2 \cdot B$.
  ↓
  If repetetion is there

Eg:

R.NO	student	Marks	dept	course
1	a	78	CS	C_1
2	b	60	EE	C_1
3	a	78	CS	C_2
4	b	60	EE	C_3
5	c	80	IT	C_3
6	d	80	EC	C_2

R.NO → Name ✓

Name → R.NO ✗

R.NO → marks ✓

Dept → course ✗

Problems Caused by redundancy:-

Redundancy means duplication of data. Problems caused due to redundancy are

1. Insertion anamoly.
2. Deletion anomaly.
3. Updation anomaly.

Eg:-

emp_id	emp_name	emp_adress	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	chennai	D900
166	Glenn	chennai	D004

Insert anamoly:- Suppose a new employee joins a company who is under trainining and currently not assigned to any department then we would not be able to insert the data into table if emp_dept does not allow null value

Update anamoly:- If I want to update the address of Rick i need update two tuples i.e two depaetments. If I update only in one row then Rick will have 2 addresses which leads to inconsistency.

Delete anamoly:- suppose if the company closes the dept D890 then the information of that employee will also be deleted

* Normalization:

A process of organizing the data in database to avoid data redundancy, insertion anamoly, updation anamoly and deletion anamoly.
↓ difficulty.

* It has 6 parts. (1NF, 2NF, ... 5NF)
* A process of organizing data into tables such a way that the results of using database are always clear.

Types of Normalization:-

① First Normal Form (1NF)
② Second Normal form (2 NF)
③ Third Normal form (3 NF)
④ Boyce-codd Normal form (BCNF)
⑤ Fourth Normal form (4 NF)
⑥ Fifth Normal form (5 NF)

Empid	name	dept	add
1	n_1	d_1	a_1 → a11
1	n_1	d_3	a_1 → a11
2	n_2	d_2	a_2
3	n_3	d_4 ✗	a_3
4	n_4	d_2	a_4
5	t_1	?	?

} update anamoly

delete - d4 ✗

↳ trainee - Insertion anamoly.

First Normal form:-

It is a property of a relation in relational database. A relation is in first normal form if and only if the domain of each attribute contains only atomic values, and the value of each attribute contains only single value from that domain.

atomic values → which can't be divided further.

Rules:-

1. * each attribute should contain atomic values.
   * Composite attributes → take more attributes
   * Multivalued " → take more tuple.

Eg:- student

cid	sname	s_addr	P.no
1	Jenny	Haryana	P_1, P_2
2	Jiya	Punjab	P_3
3	Payal	Rajastan	P_4, P_5

⇒ 1 NF ✓

sid	sname	state	country	Phno
1	Jenny	HR	INDIA	P1
1	Jenny	HR	INDIA	P2
2	Jiya	Punjab	.	P3
3	Payal	Rajash	.	P4
3	Payal	Rajastan	.	P5

2. * A column should contain value of same domain.

   Int → Integer value

   string → "

sname	id
shanvi	4

3. * each column should have unique name (P.no)
   * No ordering to rows & columns.
   * No duplicate rows.

Second Normal form (2NF):-

Candidate key:- It is a minimal set of attributes which can uniquely identify a tuple is known as candidate key.

→ The value of candidate key is unique and not null.

[stud_id, stud_name], stud_address

↓ Prime attribute → Part of candidate key.

↓ Not a part of candidate key → Non-prime attribute

* Rules:-

① * It should be in 1 NF

② * It (relation) must not contain any partial dependency.

Partial dependency:- Proper subset of candidate key → non prime attribute

No partial dependency.

Eg:- R (A, B, C, D, E, F)

Functional dependency = { A → B, B → C, C → D, D → E }

A can determine B so B is discarded

AB/~~C~~/~~D~~/~~E~~F = { A B C D E F }

↓

We can't discard F.

AF = candidate super key.

A, F are primary attributes

Non-prime attributes - B, C, D, E

Partial dependency exist so it is not in 1 NF

Eg - (2)

R (A, B, C, D)

FD = { AB → CD, C → A, D → B }

candidate keys = ABCD = { ABCD }

S.K → AB⁺ ⊄ ∅ = { ABCD }

↳ A⁺ (closure of A) = A ✗

B⁺ (closure of B) = B ✗

candidate key = AB.

↓

replace A with C

↓

CB ← replace B with D

↓

AD

Eg:- customer table:-

customer_id	Store ID	location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai

PK

candidate keys = customerID, store ID

Prime attributes = customerID, store ID.

Non-prime attribute = location

Location is dependent on store id.

cust_id	store id
1	1
1	3
2	1
3	2
4	3

Store id	location
1	Delhi
2	Banglore
3	Mumbai

Example:-

R (ABCDEF) Not in 2NF due to Partial dependency

Functional dependency $\{C \to F, E \to A, EC \to D, A \to B\}$

candidate key : $EC = FADB$

$EC^+ = ECFADB$

1. CK = {ec}

2. Prime attributes : E, C → used to form C.K

3. Non-prime attributes : {A, B, D, F}

FD $\{C \to F, E \to A, EC \to D, A \to B\}$

$C \to F$: PD → Proper subset of Prime attribute

$E \to A$: PD

$EC \to D$: set → NO subset → FD

Condition for Partial dependency

L.H.S should be proper subset of candidate key

R.H.S " " a non-prime attribute

{AB} → subset

A, B → proper subset.

C → F {PD}

E → A {PD}

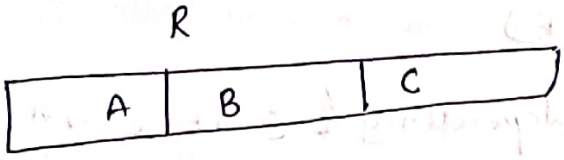
EC → D {FD}

A → B {FD}

Reflexive Rule :-

~~R(ABCD)~~

~~ABC → AB.~~



R(A, B, C)

$F = \{A \to B, B \to C\}$

Reflexivity. ⟶

$$\left.\begin{array}{l} A \to B \\ A \to C \end{array}\right\} \to \text{union}$$

$$A \to BC$$

Augmentation ⟶ $A \to B$

$CB \to D$

↓

$CA \to D.$

Transitivity ⟶ $A \to BC$

$$\left.\begin{array}{l} A \to B \\ A \to C \end{array}\right\} \text{Decomposition}$$

R(A, B, C, D, E)

$F\{A \to BC, CD \to E, B \to D, E \to A\}$

Computing F^+ ↳ set of all FD's that are logically implied from given FD set

R(AB) $F = \{A \to B, B \to A\}^+$

$x \to y$ func. dep.

F^+ -

$A \to$	A
$B \to$	B
$AB \to$	AB
ϕ	ϕ

3NF:-

It should be in 2 NF

There should be no transitive dependency.

Eg:-

Rollno	state	city
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

Transitive dependency

$A \to B$ & $B \to C$

$\Downarrow$

$A \to C$

NPA NPA

candidate key = {Roll NO}

Prime attribute = Roll NO → state

Non-prime attribute = {state → city}

Rollno → state

State → city } Roll no → city.

* Non-prime attribute determines non-prime attributes

↓

This should not be there.

Eg-①

R (ABCD)

FD : (AB → C, C → D)

CK = A, B

PA = A, B

NPA = C, D

Eg-②

R (ABCD)

FD AB → CD, D → A

CK AB^+ = ABCD

~~BCNF~~:-

Non-trivial functional dependency.

① L.H.S is super key

② R.H.S is prime attribute

Eg - ③

R (A, B, C, D) ——— This is not in 3rd NF. due to transitive dependency

FD - (A → B, B → C, C → D)
Transitive dependency TD

PA → A, CK = A

NPA = BCD

$X \to Y$, $Y \subseteq X$ → Trivial functional dependency

BCNF:- (It is also called as 3.5 NF)

Rules:

~~Drawbacks~~ ~~in~~ ~~3rd NF~~

It should be in 3NF

for every functional dependency A → B

~~A~~ for every functional dependency $x \to y$, x should be superkey of table.

eg:-

Roll No	Name	voterid	age
1	Ravi	123	20
2	varun	456	21
3	Ravi	789	23
4	Rahul	1011	21

CK = {Roll no, voter id}

FD: Roll No → name

" → voterid

voterid → age

voterid → Rollno

L.H.S of each functional dependency should be a candidate key or super key.

eg for BCNF

Std	Course	Teacher
sai	DBMS	Priya
Bhanu	DBMS	Madhu
Deepu	Java	Ramya
shiva	Java	Ramya
Ram	DBMS	Madhu

⇒ key : {std, course}

Non-key attribute = Teacher

⇓

is not a super key

Now, we decompose

std	course
sai	DBMS
Bhanu	DBMS
Deepu	Java
shiva	Java
Ram	DBMS

$x \rightarrow y$.

↓

super key

course	Teacher
DBMS	Priya
DBMS	Madhu
Java	Ramya.

$x \rightarrow y$

↓

Super key.

4NF

2NF
3NF
BCNF
} on keys & FD's.

It depends on keys
and
Multi-valued dependencies.

Rules:-

4NF eliminates independent many to one relationships b/w columns.

* A relation must be in BCNF
* A given relation may not contain more than one multivalued attributes.

eg:-

stdid	subject	activity
100	Music	swimming
100	Accounting	
100	Music	Tennis
100	Account	Tennis
150	Maths	Jagging.

→ Many student id have same subject

→ Now decompose to convert to 4NF.

schema → {stdid, subject, activity}

New schema → {stdid, subject}

new " = {stdid, activity}

stid	subject
100	Music
100	accounting
150	Maths

stid	activity
100	swimming
100	tennis
150	Jogging

It is in 4NF

4NF:-

↳ depends on keys and Multivalued dependencies.

2NF
3 NF
BCNF
} on keys & functional dependencies

⟹ Rules:

1. Relation must be in BCNF.
2. A given relation may not contain more than one multivalued attribute.

BCNF (✓)
NO Multivalued dependencies

Multivalued dependency

Trivial Multivalued dependency
⇓
$x \to y$
if
$y \subseteq x$
⇓
subset
(or)
R.H.S of FD is subset of L.H.S of FD
Eg:- $AB \to A$

Non-trivial multivalued dependency
⇓
if there is atleast one attribute in R.H.S that is not a part of L.H.S
Eg:- $AB \to C$
$AB \to AC$

5NF (fifth Normal form) or Project Join normal form:

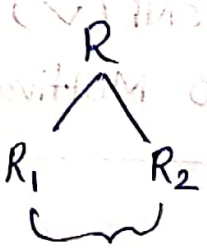
Rules:-

1. R should be in 4NF
2. It cannot be further decomposed.

No lossless ↓ non loss

* NO lossless join
Decomposition:- Dividing.

Decomposition



If i join these relations we should get same as R

$$R = R_1 + R_2$$

Lossy decomposition: Contains extra tuples.

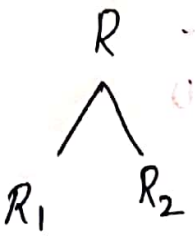
→ by learning JOIN dependency.

→ basic concept of breaking down a table.

* Lossless join decomposition:

R (A, B, C)

1	1	1
2	1	2
3	2	1
4	3	2



$R_1(A, B)$

A	B
1	1
2	1
3	2
4	3

$R_2(B, C)$

B	C
1	1
1	2
2	1
3	2

$R1 \times R2$

A	B	B	C
1	1	1	1
1	1	1	2
1	1	2	1
1	1	3	2
2	1	1	1
2	1	1	2
2	1	2	1
2	1	3	2
3	2	1	1
3	2	1	2
3	2	2	1
3	2	3	2

$3 \times 3 = 9$ . . . . . .

cartesian product

$R1 \times R2$

$= 11\ 11$

$11\ 12$

$11\ 21$

$21\ 11$

This is same in both R_1 & R_2

$$R_1.B = R_2.B$$

→ Natural Join

$R1 \bowtie R_2$

A	B	B	C
1	1	1	1
1	1	1	2
2	1	1	1
2	1	1	2
3	2	2	1

remove it