Page No.
Date

Topics	Page
1. File System & Architecture of DBMS	01
2. Types of keys	03
3. Entity-Relationship model	06
4. Normalization (1NF, 2NF, 3NF, BCNF)	10
(Decomposition)	20
5. Joins	27
6. Relational Algebra	30
7. SQL	35
8. Queries Examples (Nested Query, Sub queries)	38

Database Management Systems →

Database System

- **Database** (collection of related data)
  - RDBMS ← Structured
    - IRCTC
    - University
    - Relation (Table)
  - Unstructured (90%)
    - Webpages
- **DBMS** (operations collection) (Insert, Delete, Update)
  - SQL Server
  - Oracle 9i, 11, 12c
  - mysql
  - DB2 (IBM)

→ Aisa system jaha ham Insert Delete Update operations perform karenge on Relation.

- File System vs DBMS →

> File System → user hi manage and access karega.

> DBMS → client server (centralized).

(i) fast searching (avoids total data transfer and a query only helps to fetch the required data)

(ii) Do not require meta data (data info about data) (just access the data – No Attribute required)

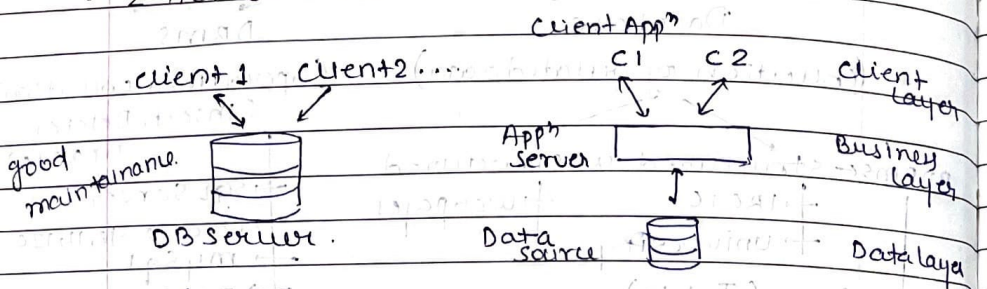
(iii) Concurrency → multiple people access data at same time.
RR
RW
WR
WW

(iv) Security → Role based security. (Role based access control)

(ex) Data → Student, Faculty, Dean

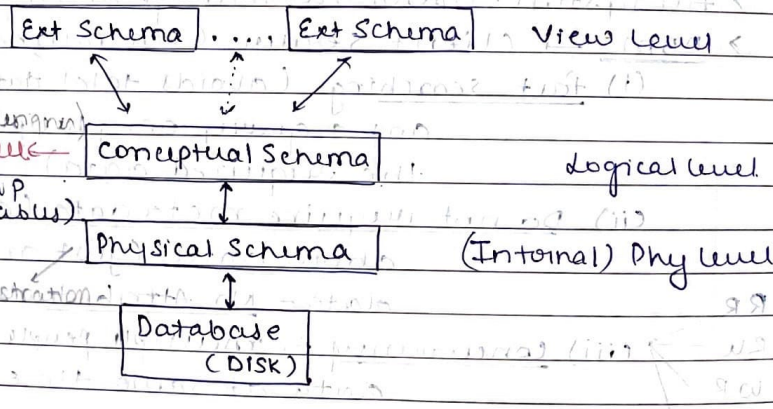
(v) Redundancy (Duplicacy) (Avoids duplicacy)

- 2 tier and 3 tier Architecture →

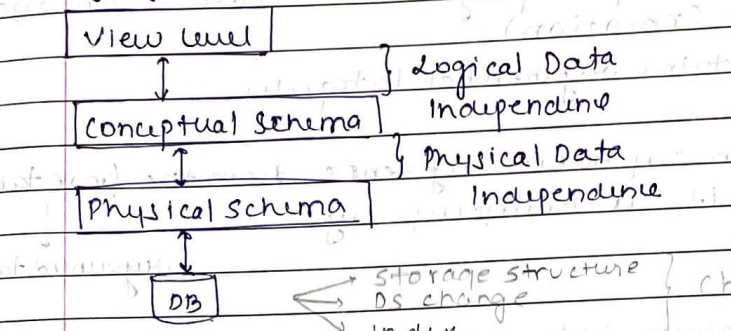


- Schema → logical representation of DB.
  ↳ structure / table / collection of tables

3 schema Architecture →



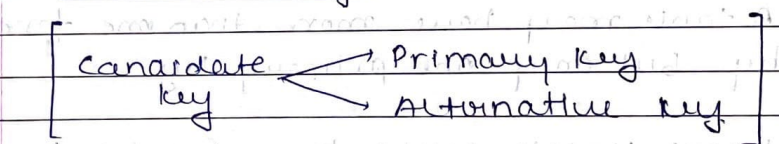
- Data Independance →



- CANDIDATE KEY →

Key → attribute in table
→ uniquely identify any 2 tuples in table

→ Single / group of multiple keys that uniquely identifies rows in a table.
Primary key is chosen from that set of Candidate key and rest are called as Alternative keys



- PRIMARY KEY → (Unique + NOT NULL)

Let us take example of Student table. →
[Candidate keys]
< Ph.no ; Adhaar Card ; PAN ; (Reg NO) ; Roll no. >
most appropriate
(admin give) ← Primary key.

- **FOREIGN KEY** ⟶

attribute or set of attributes that references to primary key of same table or another table (relation)

** → maintains referential integrity

foreign key takes reference from the base table (referenced table) where it is primary key.

Syntax →

FK attribute int references tablename (FK attribute)

↓ attribute primary in one and foreign in other.

↓ referenced table

- ALTER table <referencing_table> ADD Constraint FK foreign key (<FK-attribute>) references <referenced table> (<FK-attribute>)

→ when table already created and we need to add baadme.

→ A table may have more than one foreign key but only one primary key

- **Insert, Update, Delete from foreign key table. : Referential Integrity** ⟶

Integrity → same value (when database connected)

FK referenced table se multiple referencing table ko jaate hai.

Referenced table :

(i) Insert ⟹ no violation

(ii) Delete ⟹ may cause violation

on delete cascade (dono table mein auto-matically)

on delete set Null (value in referencing table : FK = Null)

on delete No Action (delete hi na ho; error aa jaye)

Referencer

(iii) Updation ⟹ may cause violation

Referencing table :

(i) Insert ⟹ may cause violation

(ii) Delete ⟹ no violation

(iii) Updation ⟹ may cause violation

- **Superkey in RDBMS** ⟶

Superkey is a combination of all possible attributes which can uniquely identify two tuples in a table.

If A_1 → candidate key ⟶ 2^{n-1}

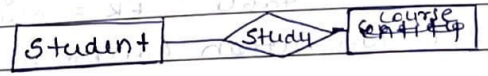
A_1A_2 → candidate keys ⟶ $2^{n-1} + 2^{n-1} - \underbrace{2^{n-2}}_{\text{dono sath}}$

$\underbrace{A_1A_2}_{CK_1}, \underbrace{A_3A_4}_{CK_2}$ ⟶ $2^{n-2} + 2^{n-2} - 2^{n-4}$

thing / object

- **Entity - Relationship model** → (ER model)
  (used for logical representation)
  only designing and no implementation

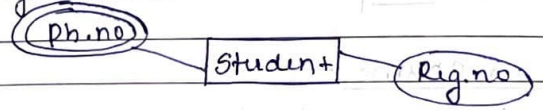
Ex Student (Rollno, age, address)
Course ( )



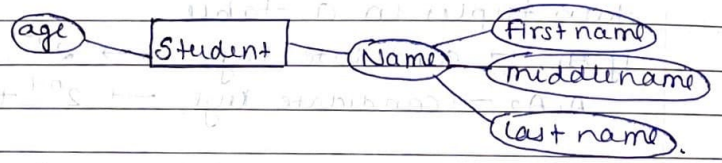
Entity → ▭
Attribute-types → ⬭
Relationships → ◇

- **Types of Attributes in DB or ER-model** →

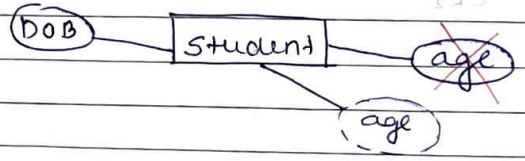
1. Single vs multivalued Attributes :



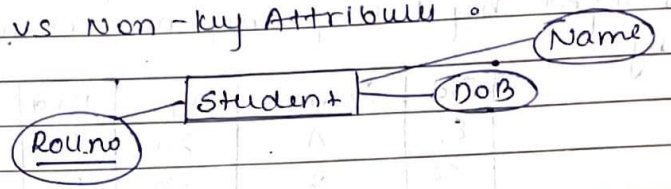
2. Simple vs composite attributes :



3. Stored vs Derived Attributes :



(unique)
↑
*4. key vs Non-key Attributes :



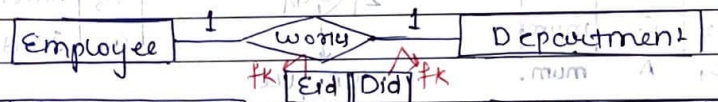
5. Required vs optional Attributes :
(Not in ER but SQL)

6. Complex attribute → (composite + multivalued)

- **Degree of Relationship (Cardinality)** →
  How entities are related to each other.

(a) one-to-one (1-1)
(b) one-to-many (1-m)
(c) many-to-one (m-1)
(d) many-to-many (M-N)

(a) one-to-one relationship →



Eid	Ename	age
E_1	A	20
E_2	B	25
E_3	C	28
E_4	A	24
E_5	B	25

fk

Eid	Did
E_1	D_1
E_3	D_2
E_2	D_3

fk

Did	Dname	LOC
D_1	IT	Bang
D_2	Prod.	Delhi
D_3	HR	Delhi

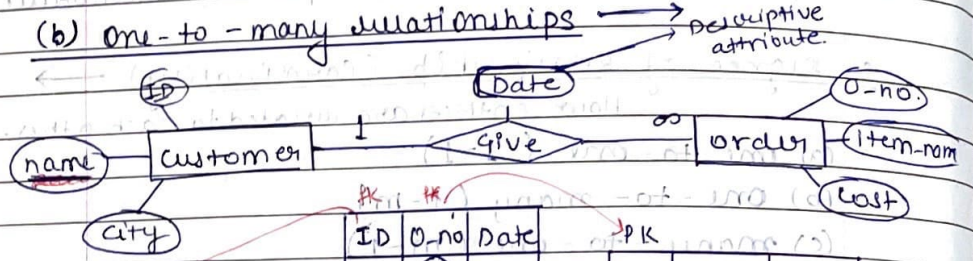
No values can be duplicated

> what will be primary key? → Since no data is repeated, all values in both columns are unique, any of them can be primary key.
{PK → Either one}

When one-to-one relationship, table can be merged by taking any one columns primary key.

Eid	Ename	age	Did
E_1	A	20	D_1
E_2	B	25	D_3
E_3	C	28	D_2
E_4	A	24	–
E_5	B	25	–

(b) One-to-many relationships →



PK

ID	Name	city
C_1	A	Jal
C_2	B	Delhi
C_3	C	mum
C_4	A	mum.

PK FK

ID	O-no	Date
C_1	O_1	
C_1	O_2	
C_3	O_3	
C_2	O_4	

unique
can't be ordered by another customer.

PK

O-no	Item-name	Cost
O_1	Bullet	1000
O_2	Shoes	2000
O_3	Shirt	2000
O_4	Jeans	1500

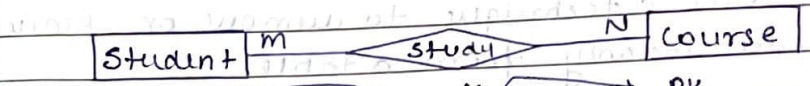
∴ Primary key → O-no.

* Reduce table → (taking O-no as primary key)

ID	O-no (PK)	Item-name	cost
C_1	O_1		
C_1	O_2		
C_3	O_3	–	

- Customer & give → can't be merged
- Give & order → can be merged

(c) many-to-many relationships →



PK FK FK PK

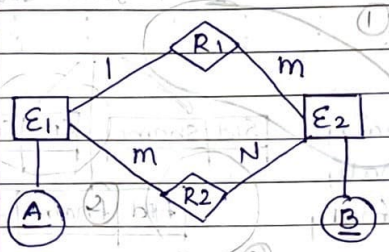
Roll-no	name	age
1	A	16
2	B	17
3	A	16
4	C	17
5	D	15

Roll-no	C-id
1	C_1
2	C_2
1	C_2
2	C_1
3	C_3

C-id	name	credit
C_1	maths	4
C_2	Phy	4
C_3	Chem	4
C_4	Hindi	4

(1) Composite key → Roll no + C-id
(2) Tables can't be reduced as combined primary key

(Ex)



Total 3 tables minimum possible (after reducing)

• NORMALIZATION →

It is a technique to remove or Reduce the redundancy from a table.

Duplicacy → Row level (Solved using Primary key)
→ column level

Row level ↓

SID	Sname	Age
1	Ram	20
2	Varun	25
1	Ram	20

Column level

SID	Sname	Cid	Cname	Fid	fname	Salary
1	Ram	C_1	DBMS	F_1	John	30000
2	Ravi	C_2	C++	F_2	Bob	40000
3	Nitin	C_1	DBMS	F_1	John	30000
4	Amrit	C_1	DBMS	F_1	John	30000

* Anomalies →
(a) Insertion Anomaly
(b) Deletion Anomaly
(c) Updation Anomaly.

> Anomaly is a flaw in databases which occurs because of poor planning and storing everything in a flat database.
> Insertion anomaly is the inability to add data to the database due to the absence of other data (Course without any student reg. can't be added)
> Deletion anomaly is unintended loss of data due to deletion of other data (Delete Sid=2 and we also registered him to course detail this data)
> If we does not realize that data is stored redundantly updation will not be done properly

• First Normal form (1NF) →

Acc to EF codd (father of DBMS);
The table should not contain any multivalued Attributes

RollNo	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS

(Not in 1NF)

RollNo	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Onkar	C
3	Onkar	DBMS

Primary key → RollNo + Course
(Composite)

PK FK

RollNo	Name
1	Sai
2	Harsh
3	Onkar

Base table
Primary key: RollNo

RollNo	Course
1	C
1	C++
2	Java
3	C
3	DBMS

Referencing table
Primary key: RollNo + Course
Foreign key: RollNo

PK

RollNo	Name	Course1	Course2
1	Sai	C	C++
2	Harsh	Java	Null
3	Onkar	C	DBMS

• finding Closure of functional Dependencies →

Helps to find all candidate keys in the table.

(Ex) R(A B C D)
Prime attribute → {A}
Non prime attribute → {B C D}
FD {A → B ; B → C ; C → D}

→ $A^+ = BCDA$; $B^+ = BCD$; $C^+ = CD$; $D^+ = D$
$CK = \{A\}$ ↳ A is determining all attributes.

Prime attribute : An attribute, which is a part of the primary key.

If we check for $(AB)^+ \rightarrow ABCD$
but candidate key is minimal hence
$(AB)^+$ can't be CK
$AB \rightarrow$ Superkey.

(Ex) R(ABCD)
FD = {A → B, B → C, C → D, D → A}
→ $A^+ = BCDA$
$B^+ = BCDA$
$C^+ = CDAB$
$D^+ = ABCD$
CK → {A B C D}
Prime attributes → {A B C D}
Non prime att → {ϕ}

(Ex) R(ABCDE)
FD = {A → B, BC → D, E → C, D → A}
→
Check for the attributes on right side
ABCD
E is on left side hence it will be used to make candidate key
$E^+ = EC$.
$AE^+ = ABECD$
CK = {AE}
$DE^+ = DEABC$
$BE^+ = BECD$
$CE^+ = CE$
$DE^+ = DEABC$
$BE^+ = BEDC$
$CE^+ = CE$

CK = {AE, DE, BE}
Prime att = {A, B, D, E}
Non prime att = {C}

X → Y → Dependend
Determinant Attribute

X Determines Y
or, Y is determined by X

(Ex) Sid → Sname
1 Amisha
2 Amisha → Do alag student hai (determined using Sid)

Sid → determinant
Sname → Dependend att.

- There are 2 types of functional dependencies
  (i) Trivial
  (ii) Non-Trivial

Trivial FD: X → Y → Always valid and true.
then $Y \in X$ (subset) → $LHS \cap RHS \neq \phi$

Non Trivial FD: X → Y & $X \cap Y = \phi$.

- Properties of functional dependencies →

1. Reflexivity: if $y \in x$, then $x \rightarrow y$
2. Augmentation: if $x \rightarrow y$ then $xz \rightarrow yz$
3. Transitive: if $x \rightarrow y$ & $y \rightarrow z$ then $x \rightarrow z$
4. Union: if $x \rightarrow y$ & $x \rightarrow z$ then $x \rightarrow yz$
5. Decomposition: if $x \rightarrow yz$ then $x \rightarrow y$ & $x \rightarrow z$
6. Pseudo Transitivity: if $x \rightarrow y$ & $wy \rightarrow z$ then $wx \rightarrow z$
7. Composition: if $x \rightarrow y$ and $z \rightarrow w$ then $xz \rightarrow yw$.

Partial dependency : Proper subset of CK → non prime attribute

Date

- **Second Normal form (2NF)** →

(Table in 1NF & table should not contain partial dependency)

→ Table or relation must be in 1st NF

→ All the non-prime attribute should be fully functional dependent on candidate key.

(customer)

Customer-id	Store-id	Location
1	1	Delhi
1	3	mumbai
2	1	Delhi
3	2	Banglore
4	3	mumbai

Candidate key : (customer ID, Store ID)

Prime att : Customer ID, Store ID

Nonprime att : Location.

(Ex) R(A B C D E F)

FD = { A → B , B → C , C → D , D → E }.

Super key $AB CDEF^* = \{A B C D E F\}$

(Because all these can be determined by A.)

$AF^+ = \{\quad\} \rightarrow$ candidate key.

$= A^+ \neq A B C D E$ — X

not superkey ← $F^* = F^X$ → trivial FD

If proper subset is also superkey, then that would not be candidate key.

∀Y ← X, F not prime attributes.

∃ ← X If prime att are present on right hand side of any FD, then there are more candidate key

∃Y ← X not ∃Y ← X ≠ I : not in diagram

∃ ← X not ∃ only one candidate key with AF

∃Y ← ∃X not W ← ∃ and Y ← X ≠ I : not in diagram

Page No 15

Date

prime att : A, F

non prime att : B, C, D, E

Partial dependency exists as A → B

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

FD = { AB → CD, C → A, D → B }

$ABCD^+ = \{A B C D\}$

$AB^+ = \{C D A B\}$ → ∴ candidate keys

$A^+ = \{A\}$ → not superkey

$B^+ = \{B\}$

A & B present on right side

∴ more candidate keys possible.

C → A

D → B

AB → AD → candidate key

↓ → CB → candidate key.

CD

~~$CD^+ = \{A B C D\}$~~

$C^+ = \{A, C\}$

$D^+ = \{B, D\}$

∴ Prime att = A, B, C, D

~~AB~~ → C

~~AB~~ → D

CB → AB

AB

AD → AB

AB

∴ 3 candidate keys : AB, CB, AD.

Prime att : A B C D

All relation prime

∴ Already in 2NF.

(Ex) R(A B C D)
FD : {A → B , B → C , C → D}

A B C D⁺ = {A B C D}
A⁺ = {B, C, D, A} . A → superkey, candidate key

prime att → A
non prime att → B, C, D
No subset of A as it is already minimal

∴ Partial dependency wali situation yaha arise hi nahi hogi.
∴ Already in 2NF.

(Ex) R(A B C D)
FD { A → B , B → D }

A B C D⁺ = {A B C D}
AC⁺ = { A → candidate key.
A⁺ = { A B D }
C⁺ = { C }
Prime att : A, C
Non prime att : B, D

→ To convert in 2NF ; we decompose the table

lossy lossless

• Third Normal form (3NF) →

→ table or relation must be in 2NF
→ there should be no transitive dependency in table.

[X → Y & Y → Z ⇒ X → Z

(Ex) R(A B C D)
FD : AB → C C → D
CK : AB
PA : A, B
NPA : C, D

AB → D

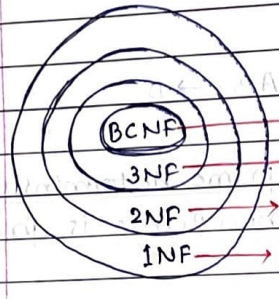
* Non-prime determining a non prime is problem

(Ex) R(A B C D)
FD : AB → CD D → A
AB⁺ = {A B C D}
DB⁺ = {D B C A}
CK : {AB, DB}
PA : {A, B, D}
NPA : {C}

→ To check transitive dependency ; for each FD → LHS must be a candidate/super key
(OR) RHS is a prime attribute.
In above que : AB → CD AB → CK
D → A D → prime att.

• Boyce Codd Normal form (BCNF) ⟶
(Special case of 3NF)

↓ more restricted version of 3NF.
Here LHS should be only candidate key.
(सारे dependencies CK पर ही होने चाहिए). OR
a super key



BCNF → strict transitive dependency.
3NF → transitive dependency
2NF → partial dependency
1NF → no multivalued attributes.

※ Third normal form always ensures 'Dependency Preserving decomposition', but not in BNCF
Both third and BNCF ensures lossless decomposition
not always

(Ex) R(ABCD)
FD: AB → CD, D → A

~~AB~~ $B^+ = B$
$AB^+ = ABCD$ ✓
$DB^+ = DBAC$ ✓
Also check for $CB^+ = CB$. ×

AB → CD (AB → CK)
D → A
PA but not CK
∴ in 3NF but not BCNF.

CK = {AB, BD}
ABD → super key
PA = {A, B, D}
NPA = {C}.

$D^+ \rightarrow DA$

ABCD → DA, BCD

(R) → (R_1), (R_2)

Lossless: when common att is CK of both or either one table.

ABCD → DA, BCD
(D → A)
CK ← D^+ {AD}

BCD
(AB → CD ?)
find another dependency
$B^+ = B$
$C^+ = C$
$D^+ = D$
$BC^+ = BC$
$BD^+ = BDAC$
New dependency. ← {BD → C}

But here dependency is not preserved as previously using relation AB → CD
$AB^+ = \{ABCD\}$
But using new dependencies i.e
D → A & BD → C
$AB^+ = \{AB\}$ × only.
∴ AB → CD dependency not preserved.

5th Normal form

• Lossless / lossy join decomposition ⟶

> Acc to EF there are 2 rules for decomposition:
(1) Decomposition should be lossless
(2) Dependency preserving decomposition.

R

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

$R_1(A,B)$
$R_2(B,C)$

R_1

A	B
1	2
2	2
3	3

R_2

B	C
2	1
2	2
3	2

(Ex) find the value of C if the value of A is 1.
we'll need to join the table.

Select $R_2.C$ from R_2 Natural join R_1 where $R_1.A = 1$;

Natural join → cross product + equivalence condition

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

cross join → product

A	B	C
1	2	1
1	2	2

(2 values) → Table after join

Spurious tuple (extra tuple) → different than original table → lossy decomposition. (Inconsistency)

When we divide a table into two, the tuple to be kept in common has a criteria of selection.
(i) common attribute should be candidate key or superkey of either R_1, R_2 or Both
(Ex) A in this case can be common att.

∴ correct division of table → $R_1(AB)$ $R_2(AC)$
It will give lossless decomposition

① → $R_1 \cup R_2 = R$
Ex $AB \cup AC \equiv ABC$
② → $R_1 \cap R_2 \neq \phi$
$AB \cap AC \neq \phi$
③ → common att must be CK or SK of R_1, R_2 or both.

1NF	2NF	3NF	BCNF	4NF	5NF
* No multi-valued att.	* In 1NF + * No partial dependency	* In 2NF + * No transitive dependency	* In 3NF + * LHS must be CK or SK	* In BCNF + * No multi-valued dependency	* In 4NF + * Lossless decomposition
* only single valued	* only full dependency $(AB \to C)$	* No non prime should determine non prime.			

Page No. | Date

- minimal cover (Irreducable) ⟶

Q. for the following functional dependencies, find the correct minimal cover:

{ A → B, C → B, D → ABC, AC → D }

⟶ Step 1: RHS attributes should be single
Ex - D → ABC
D → A   D → B   D → C

Step 2: Remove redundant keys / FD

Ignoring (A → B) check kisi aur se B determine ho raha hai Ase
$A^+ = \{A\}$, nope, hence A → B lena padega.

D → B ko hataya
then $D^+ = DABC$ ⟶ hence D → B ko remove kr sakte hai

AC → D hataya, then
$AC^+ \{ACB\}$ ⟶ D nahi aa rha hence rakhna padega.

Step 3: LHS should be single attribute

AC → D

A ko hata kar agar C ke closure mein A aa rh hai and V.V. to A ko hata sakte hai.

∴ final minimal cover dependencies are
{ A → B, C → B, D → AC, AC → D }

(Ex) R (ABCDEF), check the highest Normal form
FD : { AB → C, C → DE, E → F, F → A }

⟶ Step 1: find all candidate keys in Relation

AB^+ = ABCDEF ⟶ CK
$A^+ = \{A\}$ Agr A ya B mei se koi
$B^+ = \{B\}$ CK then AB^+ would have been SK

⟶ But A is on RHS;
↳ ∴ more possible CK

F → A
↳ FB^+ = {FBACDE} ⟶ CK.

⟶ F on RHS ⟶ ∴ more CK
EB^+ = {EBFCDA} ⟶ CK

⟶ E on RHS ∴ more CK
CB^+ = {CBDEFA} ⟶ CK.
C on RHS
AB^+ ⟶ already determined

∴ Total CK ⟶ { AB^+, FB^+, EB^+, CB^+ }

Step 2: Write all prime attributes
PA ⟶ {A, B, C, E, F}
Non PA ⟶ {D}

Now check functional dependencies :

(i) BCNF (LHS of all FD should be CK/SK)
C → DE   C → not CK
E → F   E → not CK   Sol^n : (F) → A

∴ Table not in BCNF

> Check for 3NF (transitive dependency)
LHS → CK / SK
Or, RHS → prime attribute
Violated at C → DE
↳ NPA

∴ Not in 3NF

> Check for 2NF (
condition of partial dependency: [LHS proper subset of any CK & RHS non prime attribute

C → DE → partial dependency

∴ Not in 2NF.

∴ Table in 1NF

(Ex) How to find out the Normal form of a Relation →

R(A B C D E F) {AB → C, C → D, C → E, E → F, F → A}

CK = {AB, FB, EB, CB}

PA {A, B, C, E, F}

NPA {D}

→ Check for 2NF :

C → D

∴ Decompose (to convert in 2NF)

R(A B C D E F)

R1: A B E F C

R2: C D

C → D (C → CK)

C+ : {C D}

Page No. 25
Date

Check R1 (A B C E F) for 3NF.

AB → C
C → E
E → F
F → A

CK {AB, FB, EB, CB}
PA {A B C E F}

∴ R1 → in 3NF

also R2 is in 3NF

∴ Check for BCNF

C → E ✗
E → F ✗
F → A ✗

R1 → not in BCNF

R2 → in BCNF

R1 (A B C E F)

R3 (AB → C) R4 (C → E) R5 (E → F) R6 (F → A)

R3 (A B C) R4 (C E) R5 (E F) R6 (A F)

CK: {A B} {C} {E} {F}

• Cover and equivalence of functional Dependencies:

$X = \{A \to B, B \to C\}$
$Y = \{A \to B, B \to C, A \to C\}$

To check equivalence of FD:
(i) if X covers Y [$X \geq Y$]
(ii) if Y covers X [$Y \geq X$]
then, $\boxed{X \equiv Y}$

> if X covers Y → proof →

Y: $A^+ = ABC$ ← $A \to B$
(Y ka closure X ko dekh'ke) $B^+ = BC$ ← $B \to C$
$A \to C$
$\therefore X \geq Y$

and if it covers all dependancies of Y

> if Y covers X → proof →

$A^+ = ABC$ ← $A \to B$
$B^+ = BC$ ← $B \to C$
$\therefore Y \geq X$

∴ $\boxed{X \equiv Y}$ for above relation.

• Dependency preservation decomposition →

Q. Let R(ABCD) with FD $\{A \to B, B \to C, C \to D, D \to B\}$
R is decomposed into $R_1(AB)$, $R_2(BC)$, $R_3(BD)$

→ Hint: R(ABCD) FD → R_1() FD$_1$, R_2() FD$_2$

$FD_1 \cup FD_2 = FD^+$

16/06/2022

• Joins →

(used in both Relational Algebra & SQL)
↓ mathematical expressions
↓ Implementation (commands)

(Ex) find E_name of emp who is working in HR dept.

E.No	Ename	Address	DepNo	Name	Eno.
1	Ram	Delhi	D_1	HR	1
2	Varun	Chd	D_2	IT	2
3	Ravi	Chd	D_3	MRKT	4
4	Amrit	Delhi	D_4	finance	5.
5	Nitin	Noida			

'Employee' 'Department'

> Table can be joined only if they have common attributes.
> Join is: "Cross Product + (Select Statement)
↓ {condition}

Joins types
① Cross join
② Natural join
③ Conditional join
④ Equijoin
⑤ Self join
⑥ Outer join → left, right, full

- **Natural joins** ⟶

[Join: Cross Product + condition]

In the table on prev page, employee and department. (ENO: 1 to 4 DepNo: 1 to 3)

Find the Emp.Name who is working in a dept.

→ To find employee whose adress = Delhi.

Select E.name from Employee where
Address = 'Delhi'

→ 2 diff tables, hence we need to join on the basis of common attribute.

↳ cross product

Select E.name from (Emp, Dept) where
Emp.Eno = Dept.E.no ;

↓

Select E.name from Emp Natural Join Dept.

- **Self join** ⟶

composite

S.id	C.id	Since
S_1	C_1	2016
S_2	C_2	2017
S_1	C_2	2017

Find student id who is enrolled in atleast 2 courses

→ T_1.Sid cross product

Select Sid from Study as T_1, Study as T_2 where
T_1.Sid = T_2.Sid and T_1.cid <> T_2.cid.

↳ Select T_1.sid from study as T_1, study as T_2 where T_1.Sid = T_2.Sid and T_1.cid <> T_2.cid.

(Natural Join + equal condition)

- **Equi Join** ⟶

① Find the Emp name who worked in a dept having location same as their address.

② (Emp)

Dep.No	locatn	Eno.
D_1	Delhi	1
D_2	Pune	2
D_3	Patna	4

(Dept)

Eno	Ename	Address	Depno
1	Ram	Delhi	
2	Varun	Chd	
3	Ravi	Chd	
4	Amrit	Delhi	

Select Ename from Emp, Dept where
Emp.E_no = Dept.E_no and
Emp.Address = Dept.location

- **Left Outer Join** ⟶ (common value + Left table data.)

It gives the matching rows and the rows which are in left table but not in right table.

Emp_no	E.name	Dept_no
E_1	Varun	D_1
E_2	Amrit	D_2
E_3	Ravi	D_1
E_4	Nitin	-

Dept_no	D_name	Loc
D_1	IT	Delhi
D_2	HR	Hyd
D_3	finance	Pune

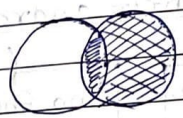
Select emp_no, e_name, dname, loc from emp Left outer join dept on (emp.dept_no = dept.dept_no)

natural join.

final table mein common to ayega saath mein, jaise E_4 match nahi kiya, tab bhi wo ayega.

Page No.
Date

- **Right outer join** ⟶
  It gives the matching rows and the rows which are in right table but not in left table.



RELATIONAL ALGEBRA ⟶

Procedural query language
(or)
formal query language.

→ what to do (access data).
→ How to do

Operators

Basic Operator	Derived Operator
• Projection (Π)	• Join (⋈)
• Selection (σ)	• Intersect (∩)
• Cross product (X)	• Division
• Union (U)	
• Rename (ρ)	
• Set difference (−)	

$\{X \cap Y = X - (X - Y)\}$

- **Projection operator (Π)** ⟶
  (data retriever)

Student

Rollno	Name	Age
1	A	20
2	B	21
3	A	19
⋮	⋮	⋮

query: Retrieve roll_no from table (Student)
→ $\Pi_{Rollno}(Student)$

* Projection operator only projects unique values

→ $\Pi_{name}(Student)$ ⟹

Name
A
B

- **Selection operator** ⟶ (σ)

query: Retrieve name of student whose Rollno = 2.

$\Pi_{name}(\sigma_{Rollno=2}(Student))$

{Π → always work on columns (attribute)
σ → always work on rows (tuples)}

- **Cross Product (×)** ⟶

R_1

A	B	C
1	2	3
2	1	4

R_2

C	D	E
3	4	5
2	1	2

A	B	C	C	D	E
1	2	3	3	4	5
1	2	3	2	1	2
2	1	4	3	4	5
2	1	4	2	1	2

$(R_1 \times R_2)$

total columns in cross product table: $m+n$
$= 3+3 \rightarrow 6$.

total rows (tuple) in cross product table: $x * y$
$2 * 2 = 4$

- **Set difference operators (−)** ⟶

(Ex) $(A - B) = A$ but not B
$= A \cap B'$

$A - B \neq B - A$ {Not commutative}

→ No. of columns must be same
→ Domain of every column must be same.

Query: Name of a person who is a student but not employee.

$(\Pi_{name}(Student) - \Pi_{name}(Employee))$

- **Union Operator (U)** ⟶

S_1: (1, 2, 3) S_2: (3, 4, 5) $S_1 \cup S_2$ ⟶ (1, 2, 3, 4, 5)

→ No. of columns must be same in number
→ Domain of every column must be same.

Non-unique values ko ek baar likhte hai.
Column name will be same as first table.

Rou No	Name
1	A
2	B
3	C

(Student)

Emp No	Name
1	E
1	A

(Employee)

Rollno	Name
1	A
2	B
3	C
7	E

- **Division operator** ⟶ (Derived operator)
(/, ÷, X, −, Π, σ ...) ⟶ use basic operator.

Sid	Cid
S_1	C_1
S_2	C_1
S_1	C_2
S_3	C_2

Enrolled 'E'

Cid
C_1
C_2

Course 'C'

Query: Retrieve Sid of student who enrolled in every course.

→ $A(X,Y) / B(Y)$ = if results x values for that there should be tuple $\langle X, Y \rangle$ for every Y value of relation B

$E(Sid, cid) \mid C(cid)$

$[\Pi_{Sid}(Emp) - ((\Pi_{Sid}(Enrolled)) \times \Pi_{cid}(course)) - (Enrolled)]$

~~S_1 C_1~~
~~S_2 C_1~~
S_3 C_1 − S_1 C_1
~~S_1 C_2~~ S_2 C_1
S_2 C_2 S_3 C_2
~~S_3 C_2~~

S_2 C_2
S_3 C_1

↓
yo disqualified hai

$\Pi_{Sid}(E) - [[\Pi_{Sid}(E) \times \Pi_{cid}(C)] - E]$

↓
yo every mein registered nahi hai.

• Tuple Relational Calculus ⟶ (TRC)
non procedural query language
(what to do ; not how to do)

{ t | P(t) }
t → resulting tuple
P(t) → Predicate [condition]

→ Atomic functions (single variable)

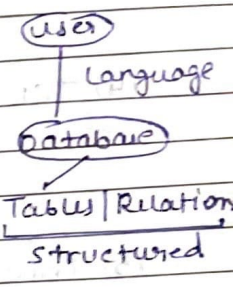
(Ex) write a query in SQL to display Sname of suppliers

{ t.Sname | Supplier (t) }
↳ actual pointer goes through every tuple and get the query.

EF Codd : Relational model (1970) ⟶ Fetched using ⟶ Relational algebra + TRC
then IBM used to implement :
then Oracle and others.

• Structured Query Language (SQL) ⟶

→ Domain specific language
(only relational DB)
→ declarative language
(what to do)
unlike procedural (what & how)
Ex PLSQL



→ DDL, DML, DCL, TCL
→ keys and constraints
→ operators (like, between, In, Not in, Exist | Not exist conditions)
→ Clauses (distinct, order by, group by, from, having)
→ aggregate functions
→ Joins and Nested Queries
→ PLSQL (Triggers, function, cursor, procedures)

• Type of SQL commands ⟶

(1) DDL : Data Definition language
- Create
- Alter
- Drop
- Truncate
- Rename

(2) DML : Data manipulation language
- Select
- Insert
- Update
- Delete

(3) DCL : Data control language
- Grant
- Revoke

(4) TCL : Transaction control language
- Commit
- RouBalk
- Save point

(5) Constraints
- Primary key
- foreign key
- check
- Unique
- Default
- Not Null.

- ALTER command →
- CREATE Table command →

```
create table <table_name>          | create table emp
(                                  | (
  column1 name datatype,           |   id int,
  column2 name datatype,           |   name varchar2(20),
  column3 name datatype            |   Salary number(10)
);                                 | );
desc table_name;                   | desc emp;
```

- ALTER command →
  → Add columns
  → Remove columns
  → modify datatype
  → modify datatypelength.

→ add constraints
→ Remove constraints
→ Remove column | table.

(Ex) Alter table Student Add address varchar(30);
Alter table Student drop column address;
Alter table Student modify id varchar(10);

column → Alter table employee rename column id to eno;
name change

table → Alter table employee rename to emp;
namechange

Alter table employee add primary key(rollno);

- Difference b|w ALTER and UPDATE →

ALTER	UPDATE
→ DDL	→ DML
→ structural change	→ change in data only
→ add | delete | modify	→ update existing records in DB.

(Ex) Update Emp Set
Salary = Salary * 2
where id = 1

- Difference b|w delete, drop and truncate →

Delete	Drop	Truncate
DML	DDL	DDL
Delete from Student (row wise delete)	Drop table Student. No rollback allowed	Truncate Student. (all rows delete. No rollback possible
(by commit) Rollback possible		

• Constraints in SQL →

① unique
② Not Null
③ Primary key : [unique + Not null]
④ check (Ex. check (age > 18))
⑤ foreign key (Referential Integrity)
⑥ Default

Query Examples →

Emp

Eid	Ename	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	30000
5	Varun	IT	50000
6	Sanoly	Testing	Null

(1) Write a SQL query to display maximum salary from Emp table.

```
: Select Max(Salary) from Emp;
```

(2) Write SQL query to display Employee name who is taking maximum salary.

Nested query

```
Select Ename from Emp where
    Salary = (Select max(Salary) from Emp);
```

Inner

* First inner query run ~~hoti~~ then outer query.
→ This is example of nested query.

(3) Write a SQL query to display second highest Salary from Emp table.

```
:
Select max(Salary) from Emp where
    Salary <> (Select max(Salary) from Emp);
```

where → works on complex

(4) Write SQL query to display Employee name who is taking second highest salary.

```
:
Select Ename from Emp where
    Salary = (Select max(Salary) from Emp where
        Salary <> (Select max(Salary) from Emp));
```

(5) Write a query to display all the dept names along with no. of Emps working in that.

Group By

group by ka sathwala attr only.

```
Select dept from Emp group by dept;
```

output ↓

count*

HR	2
MRKT	2
IT	1

```
Select dept, count(*) from Emp group by dept;
```

(6) Write a query to display all the dept names where number of employees are less than 2.

grouped category

```
:
Select dept from Emp group by dept having count(*)<2;
```

(7) Write query to display name of employee working in the dept where number of employees are less than 2.

Nested query

```
:
Select Ename from Emp where dept In (Select dept
    from Emp group by dept having count(*) < 2);
```

(8) Write query to display highest salary department wise and name of employee who is taking that salary.

```
:
Select Ename from Emp where Salary In
    (Select max(Salary) from Emp group by dept);
```

[30000
40000
50000]

IN / NOT IN →

(9) Write query to display detail of emp whose address is either Delhi or Chd or Pune.

```
Select * from Emp where
Address In ('Delhi', 'Chd', 'Pune');
```

multiple value in RHS
I = (1,2,3) ✗ not possible.

Emp

Eid	Ename	Address
1	Ravi	Chd
2	Varun	Delhi
3	Nitin	Pune
4	Robin	Bangalore
5	Ammy	Chd

Project.

Eid	Pid	Pname	Location
1	P_1	IoT	Bangalore
5	P_2	BigData	Delhi
3	P_3	Retail	mumbai
4	P_4	Android	Hyd.

(10) Find name of Emp who are working on a project.

```
Select Ename from Emp where Eid In
    (Select Eid from Project);
```

EXISTS / NOT EXISTS → (Correlated Nested Query)

(11) Find the detail of Emp who is working on atleast one project.

```
Select * from Emp where Eid Exists
    (Select Eid from project where Emp.Eid = project.Eid);
```

Exist / Not Exist → return True / false.

AGGREGATE FUNCTIONS (max, min, count, avg, sum) →

Consider the Emp table used in Q(1) to (8)

(12) Find the max salary from table.

```
Select max(Salary) from Emp;
```

(13) min salary from table

```
Select min(Salary) from Emp;
```

(14) To count total no. of rows in the table.

```
Select count(*) from table;
```



```
Select count (Salary) from Emp;
Output: 5
Select Distinct (count (Salary)) from Emp;
Output: 4
Select Sum (Salary) from Emp;
Output: 10000 + 20000 + 30000 + 30000 + 50000
```

```
Select Avg (Salary) from Emp;
```

$$Avg(Salary) = \frac{Sum(Salary)}{count(Salary)}$$

$$Distinct(Avg(Salary)) = \frac{Distinct(Sum(Salary))}{Distinct(count(Salary))}$$

Correlated Subquery (Synchronized Query)

→ It is a subquery that uses value from outer query.

→ Top to down approach. OQ (1 time)

(15) Find all employees detail who works in a department.

```
Select * from Emp where exists (Select * from
    dept where dept.Eid = Emp.Eid);
```

Emp

Eid	name
1	A
2	B
3	C
4	D
5	E

Dept

Dep.no	name	Eid
P_1	IT	1
D_2	HR	2
D_3	mrkt	3

19/06/2022

Page No.
Date

- Difference between Nested Subquery, Correlated Subquery, Joins ⟶

> Nested subquery ⟶ OQ(IQ)
⟶ Bottom up

```
Select * from Emp where Eid in (
        Select Eid from dept);
```

Output: details of all emp who work in any dept

→* Pehle Inner Query execute, uss result ko bahar compare

> Correlated subquery ⟶ Top down approach

```
Select * from emp where exists (Select id from
        dept where emp.eid = dept.eid);
```

→* (Bahar wali ki ek row, compare to inner wali ki rows)

No. of comparisons: Nested Query ≤ Correlated Query

> Joins: ⟶ Cross product + condition

use more space but less time.

store the temp. (m×n) relation in buffer and make no. of comparisons

join better than correlated, though time comparison ka kaam hai, but join mein tables ko ek saath le diya, correlated mein alag the.

Space: Joins > Correlated
Time: Joins < Correlated

Page No. 43
Date

- Find the Nth Highest Salary using SQL ⟶

```
Select id, Salary from Emp e1 where
    N-1 = (Select count (Distinct Salary)
        from e2.Salary > e1.Salary)
```

Questions:

Q. Display last name of employees who have A as second character in their names. (Like command)
⟶

```
Select last name from emp where lastname
        like '_A%';
```

Q. Command to remove table
⟶ Drop table <table_name>

Q. (Select * from R) Intersect (Select * from R)

1	1		1	1
2	2	⟶	2	2
3	3		3	3
3	3			

[both produce the same result.]

```
Select distinct * from R
```

→ SQL + Programming.
→ Procedural SQL

- PL-SQL in DBMS → (function, Procedure, Trigger, Cursor)

Syntax:

Declaration
↓
Executable code (begin ... end)
↓
Exception handling (error)

```
declare
a int
b int
c int
begin
a:=10;
b:=20;
c:=a+b;
end;
```

→ offers numerous datatypes
→ structured programming through functions and procedures
→ object oriented programming is supported
→ supports development of web application (smart and server pages)

PL SQL code to print "Hello World"
→

```
DECLARE
    message varchar(20):= 'Hello World';
BEGIN
    dbms_output.put_line(message);
END;
```

- Datatype in PL/SQL →
  1. Scalar – Num, Char, Bool, Datetime
  2. LOB (Large Obj) – BFILE, BLOB, CLOB, NCLOB
  3. Composite
  4. Reference.