(4) List name of the ~~employees~~ who have salary more than 10000 & Less 25000;

$$\Pi_{ename}\left(\sigma_{salary>10000 \wedge salary<25000}(Emp)\right)$$

(5) Salary more than 10000 or Less than 25000

$$\Pi_{ename}\left(\sigma_{salary>10000 \vee salary<25000}(Emp)\right)$$

04/03/2024

Union ⇒

⇒ $\Pi_{c.name}(depositor) \cup \Pi_{c.name}(loan)$

Customer name from depositor OR loan

→ Arity (no. of columns) of both ~~columns~~ queries must be same

→ Domain of Respective Columns must be same

eg → $\Pi_{c.name}(depositor) \cup \Pi_{balance}(loan)$ (X)

⇒ Set-difference (−)

Q → List all the depositors who do not have any loan.

↳ $\Pi_{C.name}(depositor) - \Pi_{cname}(loan)$

⇒ Cartesian Product (×)

⇒ Relation

Emp

eid	ename	desig
1	A	101
2	B	102
3	C	101

Dept

did	~~dept~~ d.name
101	CS
102	IT

Emp × Dept

eid	ename	desig	did	dname
1	A	101	101	CS
1	A	101	102	IT
2	B	102	101	CS
2	B	102	102	IT
3	C	101	101	CS
3	C	101	102	IT

⇒ Rename $\rho_x(E)$

→ Could be an ~~old~~ Table name or expression/query as well

In case of table name ⇒ (E) will be renamed as (Table) x

In case of expression ⇒ E will be represented as x (query)

Additional Operations

⇒ Intersection (∩)

$$r \cap s = r - (r - s)$$

R

A
B
C
D
E
F

S

G
H
B
A
P
X

R ∩ S

B
A
D

R - S

AC
E
F

R - (R - S)

B
A
D

⇒ Natural Join (⋈)

$(r \bowtie s)$ = ~~xxxxx~~

$$\Pi_{RUS}(\sigma_{r.attr1 = S.attr1 \wedge r.attr2 = S.attr2 \wedge \ldots}(r \times s))$$

Emp

Eid	ename	did
1	A	101
2	B	102
3	C	101

Dept

did	dname
101	CS
102	IT
103	Mech

Emp ⋈ Dept

eid	ename	did	did	dname
1	A	101	101	CS
2	B	102	102	IT
3	C	101	101	CS

⇒ Division operation (÷)

Q⇒ List all the customers who have accounts in all branches of jaipur.

$(r1 \div r2)$ ⇒ r1 ⇒ all the branches of jaipur.

$$r1 = \Pi_{bname}(\sigma_{b\_city = jaipur}(branch))$$

$$r2 = \Pi_{cname, bname}(depositor \bowtie {}^{P}loan)$$

5/3/2024

Extended Relational Algebra Operations

$\Rightarrow \Pi_{ename, basic+da+HRA}(Emp)$

$\Rightarrow \Pi_{ename, (basic+da+HRA)\ as\ salary}(Emp)$

also

We can use Aggregate fiction

$\Rightarrow \mathcal{G}_{sum(salary)}(Emp)$

Outer Joins →

① Left Outer Join (⟕)

② Right Outer Join (⟖)

③ full Outer Join (⟗)

Operation with Null Values →

→ All arithmetic operations with null returns null.

$< \quad <= \quad > \quad >= \quad \neq$ ⇒ Returns Unknown

→ Boolean Logical Operations

AND ⇒

True ∧ Unknown ⇒ Unknown

false ∧ Unknown ⇒ false

Unknown ∧ Unknown ⇒ Unknown

OR ⇒

True ∨ Unknown ⇒ True

false ∨ Unknown ⇒ Unknown

Unknown ∨ or Unknown ⇒ Unknown

Not ⇒

not (unknown) ⇒ Unknown

Assignment Operator ⇒

$(\leftarrow)$

eg → $\{ R \leftarrow \Pi_{ename, salary}(Emp) \}$

R ↓ Temporary table in which we want to store the result

Relation

$(r \leftarrow r - E)$

eg→

$Emp \leftarrow Emp - (\sigma_{ename='RAM'}(Emp))$

$loan \leftarrow loan - (\sigma_{amount>=0 \text{ and } amount<=1000}(loan))$

Insertion ⇒

$\{r \leftarrow r \cup E\}$

eg ⇒

$Emp \leftarrow Emp \cup \{101, 'RAM', 50000\}$

~~Deletion~~ • Updation →

$account \leftarrow \Pi_{a\_number, a\_name, balance * 1.05}(account)$

Operations deals with Null ⇒

⇒ Select

$\sigma_{salary>30000}(Emp)$

1	A	50000
2	B	46000
4	D	35000

Emp

eid	ename	Salary
1	A	56000
2	B	45000
3	C	null
4	D	35000
5	E	null
6	F	30000
7		

$\sigma_p(E)$ ⇒ If P's value is false or unknown then there will be no result.

Join Operations ⇒ $r \bowtie s$

$t_s \in s$

$t_r \in R$

If both are null that will not match

6/3/24

(÷)

Enrolled ÷ Course

sid
1

Enrolled

sid	cid
1	DBMS
2	DBMS
1	OS
3	OS

Course

cid
DBMS
OS

$R(x, y) \div S(Y)$

⇒ Attributes of S must be proper subset of attributes of R.

⇒ For each corresponding value of y above notation will return value of x from tuple <x, y> which exists ~~every~~ where for each y of S.

Database Design

Normalisation

we need to first understand.
⇒ Functional dependencies

eid ⟶ ename

{ If in any two rows eid is same than ename will also be same. }

eid ⟶ desig ✗

eid	ename	desig	gradeP.
1	A	IA	2800
2	B	Prog.	4800
1	A	Prg	4800
3	C	IA	2800

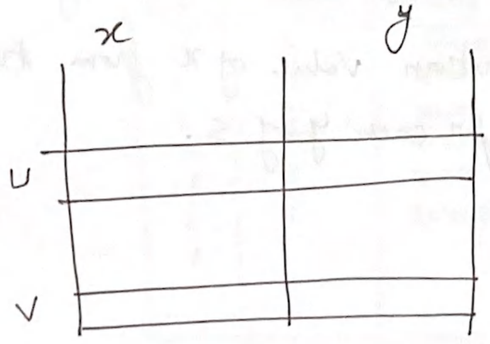
desig ⟶ gradeP. ✓

→ ename is functionally determined by eid.

→ eid functionally determine ename

→ ename is functionally dependent upon eid.

{ $x \to y$
if $U(x) = V(x)$
then $U(y) = V(y)$ }



Inference Rules →

① Reflexive →

if $Y \subseteq X$ (X: superset, Y: subset) than x will determine Y $\{x \to y\}$

② Augmentation →

If $x \to y$ than $xz \to yz$

Emp

cid	ename	dob
1	A	3-2-90
2	B	5-6-85
1	A	3-2-90
4	C	9-3-89
2	B	5-6-85
6	D	2-3-87

@ x = eid, ename

y = ename

Reflexive ⇒ { $x \to y$ because $y \subseteq x$ }

{ eid, dob ⟶ ename then eid, dob ⟶ ename, dob }

⇓

Augmentation.

7/3/24

③ Transitive =>

If $x \rightarrow y$ & $y \rightarrow z$ then

$x \rightarrow z$

eid → ename &

ename → mob then

eid → mob

Emp

eid	ename	mob
1	A	—
2	B	—
3	C	—
4	D	—
5	E	—
2	B	—
3	C	—
1	A	—

~~All those~~

Armstrong Rule

- Reflexive
- Augumentation
- Transitive

④ Additive / Union Rule

$x \rightarrow y$, $x \rightarrow z$ } $x \rightarrow yz$

⑤ Decomposition / Productive Rule

$x \rightarrow yz$ { $x \rightarrow y$, $x \rightarrow z$

⑥ Pseudo Transitive Rule

$x \rightarrow y$, $yz \rightarrow w$ } $xz \rightarrow w$

Fully Functional Dependency

⇒ If $x \rightarrow y$ then y will be functionally dependent on x, but will not be dependent on any proper subset of x.

ename, dob → G.P.

If ename, dob is fully determining GP then

ename ↛ GP

dob ↛ GP

Partial functional Dependency.

⇒ If R is a relation and K is its Candidate key. if X is proper subset of K and $X \longrightarrow A$ means A is a partial functional dependent on K.

Prime Attributes

⇒ If an attribute is part of any Candidate Key, this is known as prime attribute.

⇒ An attribute which is not part of any candidate key, is known as non-prime attributes

Trivial Dependency

$A \longrightarrow B$ is known as trivial dependency

if B is subset of A {same as reflexive rule}

$B \subseteq A$

Closure ⇒

is set of all implied / infered functional dependencies of F is known as closure of F. and represented by F^+.

$$F^+ = \underbrace{F}_{\text{Set of FD}} \cup \underbrace{F'}_{\substack{\text{Set of FD} \\ \text{infered from F}}}$$

Normalization

⇒ Process to analyse existing relations by functional dependencies to minimise reduendency and updation anomalies.

⇒ we break sub divide tables which is known as decomposition.

There are 2 desirable conditions for decompition

① Lossless join decomposition

② dependency preservation.

8/3/24

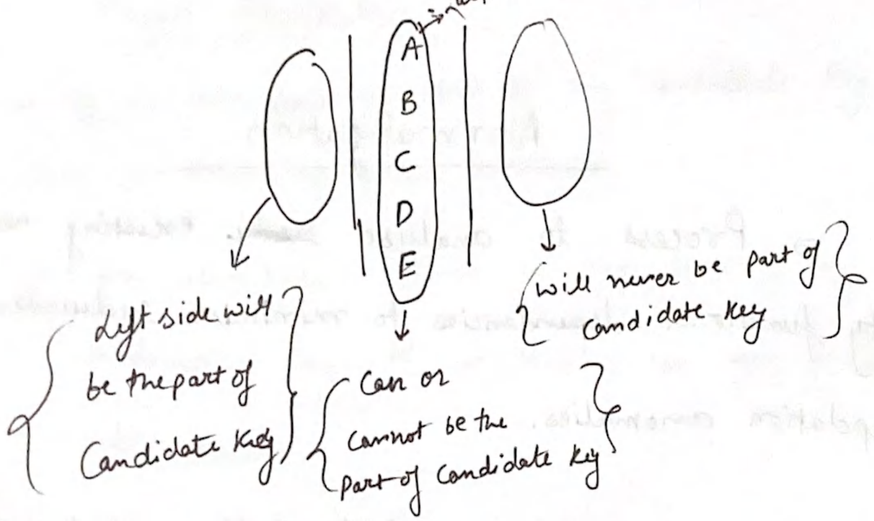
Q→ If R is a relational Schema R(A,B,C,D,E) with the following functional dependencies

$AD \rightarrow C$, $B \rightarrow A$, $C \rightarrow E$, $E \rightarrow BD$

Which of following is not Candidate Key of R.

ⓐ AD ⓑ BE ⓒ CD ⓓ AB

Method 1 →



Method 2→ use options ⇒ ← find closures

$(AD)^+ \Rightarrow \{A, B, C, E, B\}$ ✓

This will be one of the candidate key.

$(BE)^+ \Rightarrow \{B, E, A, D, C\}$ ✓

This one is also candidate key

$(CD)^+ \Rightarrow \{C, D, E, B, A\}$ ✓

$(AB)^+ \Rightarrow \{A, B\}$

Answer ⇒ AB is not a Candidate key of R.

Q→ R(A, B, C, D, E, F, G, H, I, J)

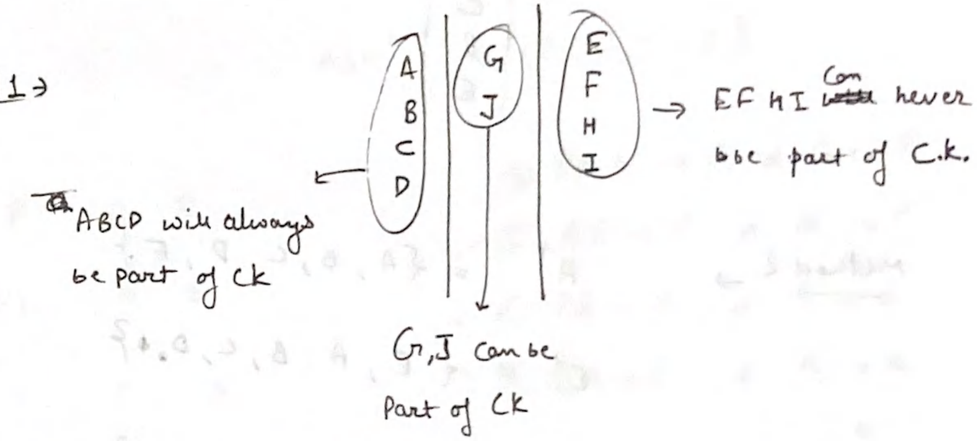
FD ⇒ $ABD \rightarrow E$, $AB \rightarrow G$, $B \rightarrow F$, $C \rightarrow J$, $CJ \rightarrow I$

$G \rightarrow H$

find CK. ⇒

ⓐ ABCD ⓑ ABGF ⓒ ABCI ⓓ ABC

Method 1 →



$ABCD^+ = \{A, B, C, D, E, F, G, H, I, J\}$

If we add G or J with ABCD even though ABCD is enough to determine whole Relation it would become Super key {ABCDG or ABCDJ or ABCDGJ will be Super Key}.

Q→ R(A B C D E)

FD ⇒ A → BC , CD → E , B → D , E → A

CK ?

(A) A, E, CD, BE ✓ (B) AB, CD, EB, BC

✗ (C) A, E, C, B, D (D) A, E, CD, BD

Method 1 :

A
B
C
D
E

Method 2 →

A^+ = {A, B, C, D, E}

E^+ = {E, A, B, C, D}

CD^+ ⇒ {C, D, E, A, B}

BE^+ → {B, D, E, A, C}

⇒

BD^+ {B, D,

Q→ AB → C , CD → E , DE → B

Key ?

(a) AB (b) ABD (c) ABC (d) None of Above

Ans →

A
B
C
D
E

AB^+ {A, B, C,}

ABD^+ {A, B, C, D, E}

P	Q	R
1	4	2
1	5	3
1	6	3
3	2	2

(a) PQR → Q ✗ Q → P

✓ (b) QR → P ✗ Q → P

✗ (c) PQ → R . ✗ R → Q

✓ (d) QR → P ✗ Q → R

Which FD holds.

{Only on option B our FD is not failing
so, it is correct.}