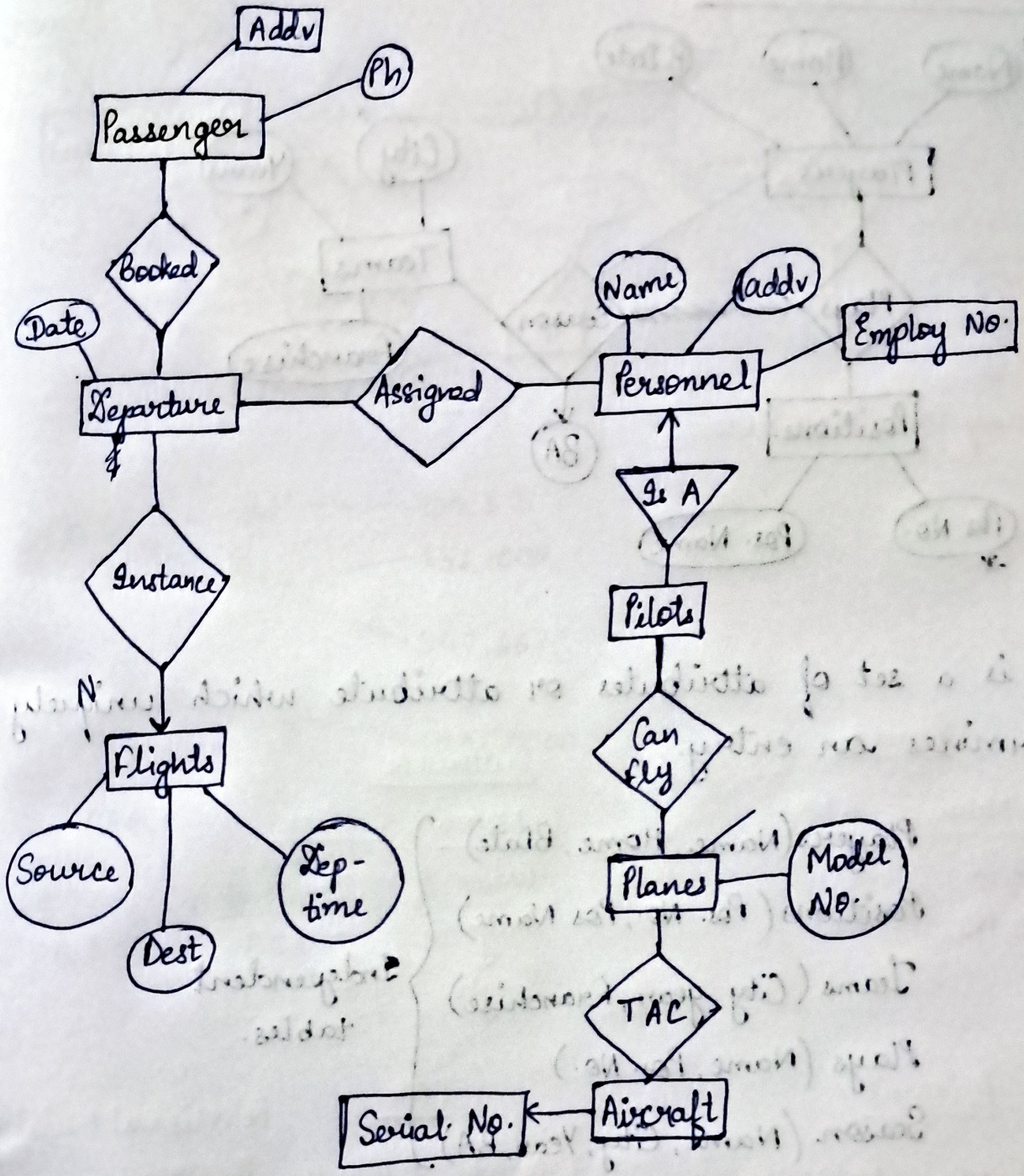
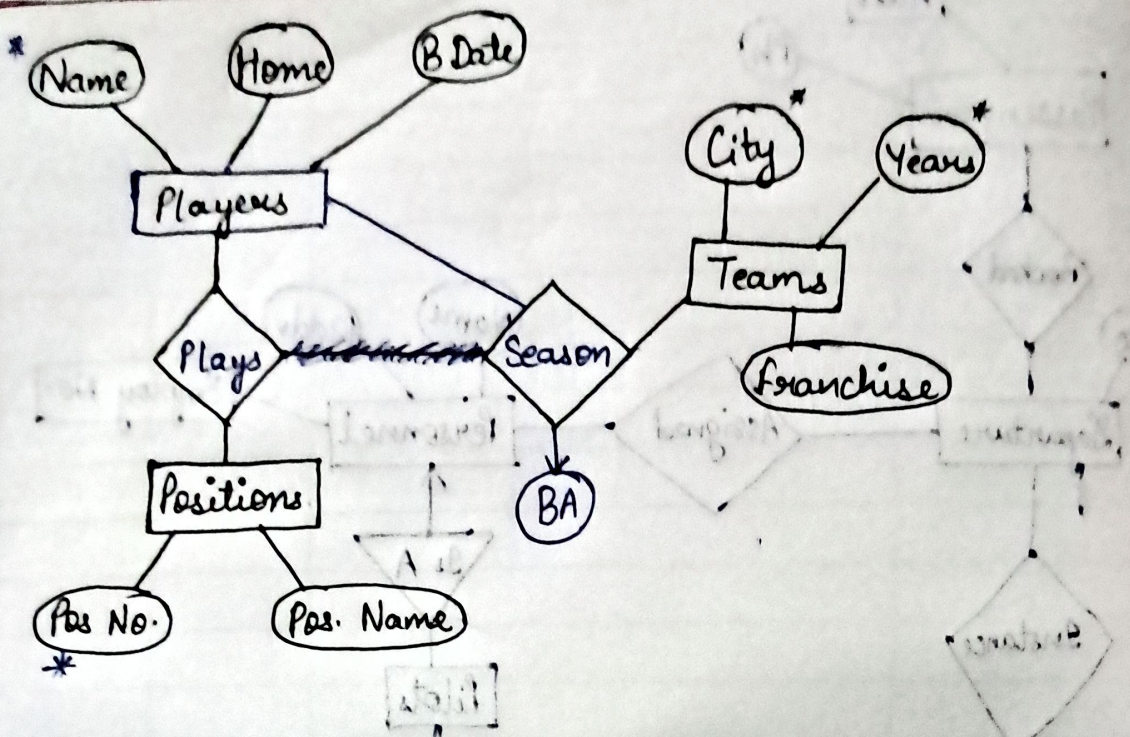

Addr
Ph
Passenger
Booked
Date
Departure
Assigned
Name
Addr
Employ No.
Personnel
Is A
Pilots
Instance
Flights
Source
Dest
Dep-time
Can fly
Planes
Model No.
TAC
Aircraft
Serial No.

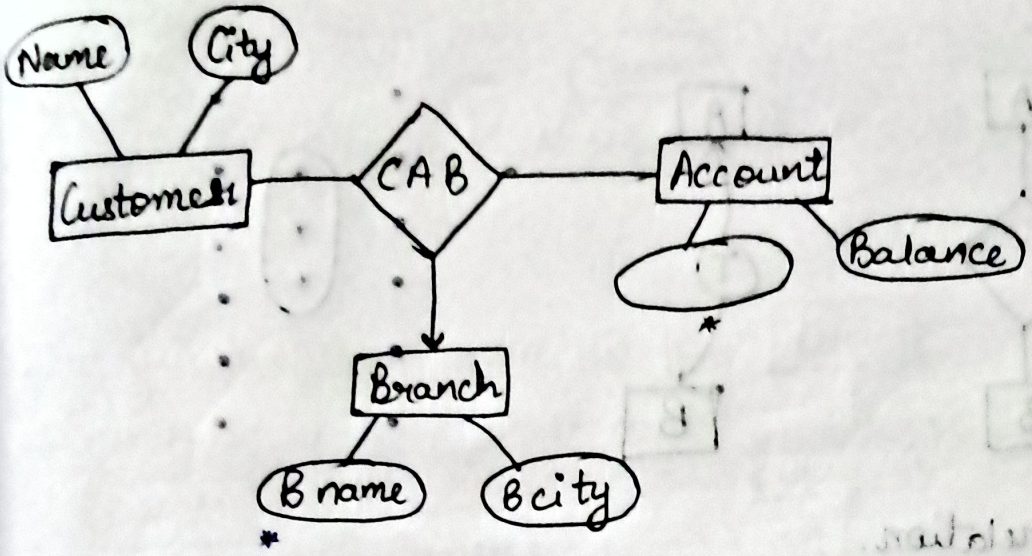
★ Relational Data Model

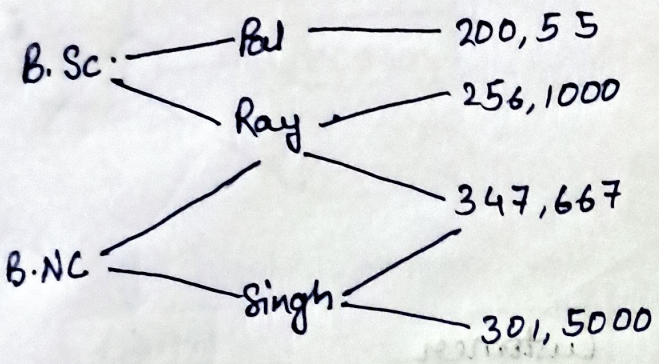


Key is a set of attributes or attribute which uniquely determines an entry.

Players (Name, Home, BDate)
Positions (Pos. No., Pos Name)
Teams (City, year, Franchise)
Plays (Name, Pos. No.)
Season (Name, City, Year, BA)

} Independent tables.

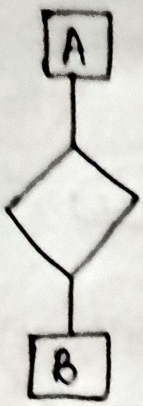




CAB

Name	B. Name	B. No.
Pal	BSc	200

Network Model



Many-one relation.

Branch

B.Name	B.No.
B-Sc	-
B-Nc	-

Customer

C.Name	C.City
Pal	-
Ray	-
Singh	-

Account

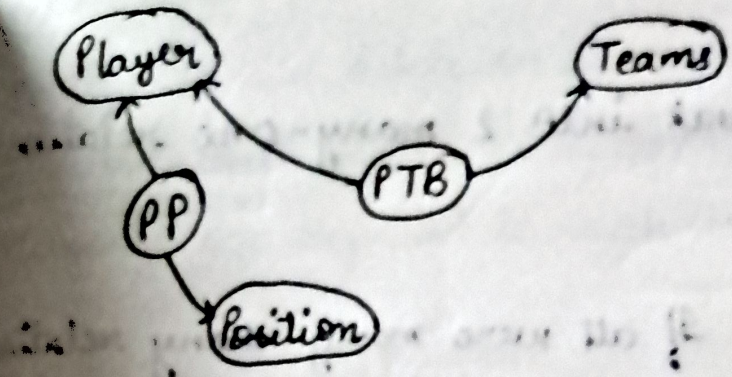
Ac.no.	Balance
200	-
256	-
347	-
301	-

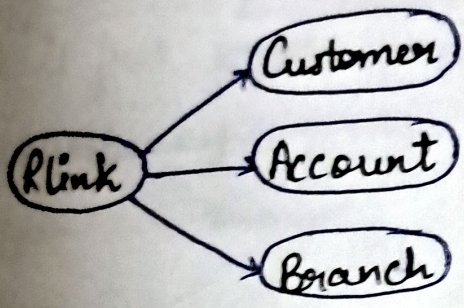
CAB

C.Name	B.Name	Ac.no.
Pal	B-Sc	200
Ray	B-Sc	256
Ray	B-Nc	347
Singh	B-Nc	347
Singh	B-Nc	301

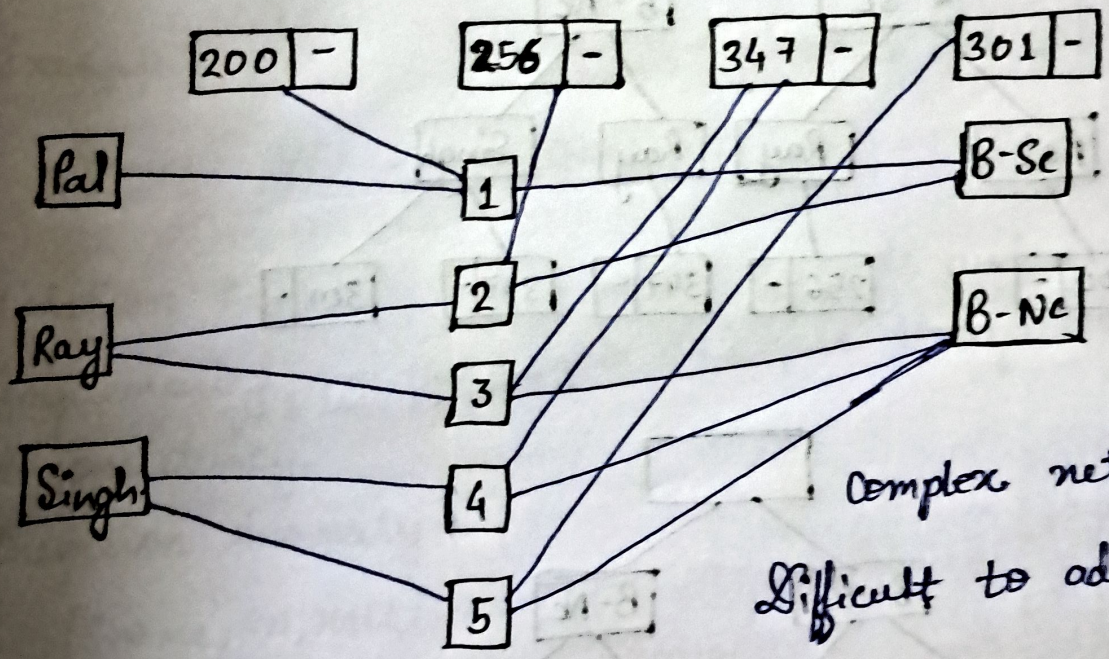
Relational Model

Problem of redundancy





Network model.

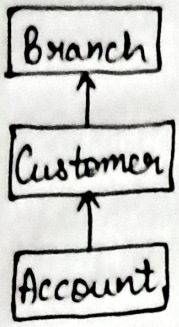


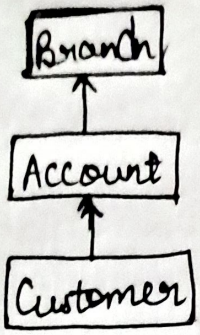
Complex network.

Difficult to add entries.

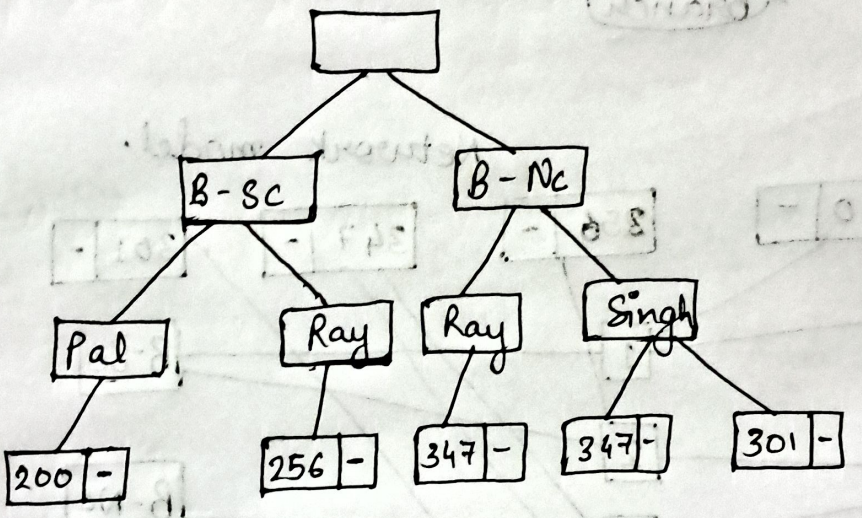
Hierarchial Model

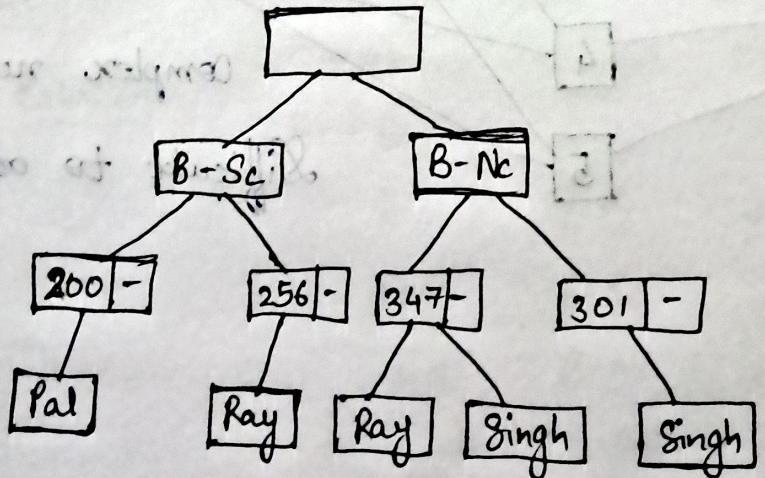
Many-Many relation : Break into 2 many-one relations





If all were many-many relation then there will be 6 branches.





Advantages:

Easier to add, delete or search elements.

Disadvantage:

Memory size required is much large.

★ Relational Database Design.

The features of an ideal database ~~design~~ design are:

i) It is free from:

(i) Redundancy

(ii) Updation anomaly

(iii) Insertion anomalies

(iv) Deletion anomalies

Example:

Sname	RN	HALL	GAME	FEE

Updation Anomaly: If updation is made and is not implemented on the whole database then it leads to inconsistency.

Insertion Anomaly:

(Sname, RN, Hall)

(RN, Game)

(Game, Fee)

★ Functional Dependency (FD)

$U \rightarrow$ Universal set of attributes

$\forall U_j \in U \quad U_j \subset U$

$X \rightarrow Y$

X and Y are the subset of U_j.

X functionally determines Y.

Y is functionally dependent on X.

$R : X \rightarrow Y \quad \mu_1 \& \mu_2$ (tuples)

↓

(one record)

$\mu_1[x] = \mu_2[x] \Rightarrow \mu_1[y] = \mu_2[y]$

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

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_3	d_4

$A \rightarrow C$

$D \rightarrow B$

$C \rightarrow A$

$AB \rightarrow D$

$A \rightarrow A$ {Trivial FD}

$A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$

Closure of a set of FDs

Let F be a FD.

$F^+ = \{X \to Y \mid F \to X \to Y\}$

Example : $R(A,B,C)$; $F = \{A \to B, B \to C\}$,

$F^+ = \{A \to C, A \to ABC, AB \to ABC, \ldots\}$

Primary key

$R(A_1, A_2, \ldots, A_n)$

$X \subseteq A_1, A_2, \ldots, A_n$ is in F^+

1) $X \to A_1, A_2 \ldots A_n$ is in F^+

2) ~~$Y \subseteq X$ & $Y \to A_1, A_2$~~ There ~~should~~ might exist

$Y \subseteq X$ & $Y \to A_1, A_2 \ldots A_n$ is in F^+

Primary key is not unique.

Armstrong's Axioms

1. Reflexivity: If $Y \subseteq X \subseteq U$ then $X \to Y$
2. Augmentation : If $X \to Y$ and $Z \subseteq U$, then $XZ \to YZ$
3. Transitivity: If $A \to B, B \to C$ then $A \to C$

Example: R(CITY, ST, PIN)

$F = \{$CITY ST $\to$ PIN , PIN $\to$ CITY$\}$

CITY ST $\to$ PIN

CITY ST $\to$ PIN CITY ST

PIN $\to$ CITY

PIN ST $\to$ CITY ST

~~PIN ST $\to$ PIN CITY ST~~ PIN ST $\to$ CITY ST PIN

∴ It has 2 primary keys CITYST and PINST

If an attribute does not appear on the RHS of an FD, then it cannot be derived logically. ∴ it must be a part of the primary key.

Armstrong's Axioms and Inference Rules

1. Union Rule: $\{X \to Y, X \to Z\}$ F $X \to YZ$

   $X \to XZ$, $XZ \to YZ$

   ↳ Transitivity: $X \to YZ$

2. Pseudotransitivity Rule: $\{X \to Y, WY \to Z\}$ F $XW \to Z$

3. Decomposition Rule:

   If $X \to Y$ & $Z \subseteq Y$, then $X \to Z$

4. If $X \to YZ$ then $X \to Y$ & $X \to Z$

- **Closure of a set of attributes**

  F $X^+ = \{A \mid X \to A\}$

Closure of a set of attributes is the set of keys which are determined by X using Armstrong's axioms and Inference Rules.

Lemma: $X \to Y$ follows from Armstrong's axioms if and only if Y is a subset of X^+.

Theorem: Armstrong's axioms are sound and complete.

Sound: If a relation is true for an FD, then that relation would be true whenever the FD is true.

Complete/Concrete: No other relation other, th

* X^+ algorithm of Bernstein

1. Let $N = 0$ and $X(N) = X$
2. If $A \to B$ in F, where A is contained in $X(N)$ whereas B is not contained in $X(N)$, then → Iterate for all FDs

$$X(N+1) = X(N) \cup B$$

otherwise, Terminate.

3. $N = N+1$, GO TO Step 2.

Example: $R(A, B, C, D, E, G)$

$F = \{AB \to C, C \to A, BC \to D, ACD \to B, D \to EG, BE \to C, CG \to BD, CE \to AG\}$

Find $(BD)^+ = ?$

$N = 0$; $X(0) = BD$

$D \to EG$; $X(1) = BDEG$

$BE \to C$; $X(2) = BDEGC$

$C \to A$; $X(3) = BDEGCA$

BD is the key of the relation as it determines all other attributes.

Example: R(FI, MOC, SAL, SS, JOB, COR, LUCK, ED, EFF, SEN)

F = {FI → MOC, SALSS → FI, JOB FI → COR,

LUCK ED → JOB, JOB → ED, EFF SEN JOB → SAL}

SAL SS SS EFF SEN LUCK SAL JOB

N=0, X(0) = SS EFF SEN LUCK SAL JOB

FI → M SAL SS → FI, X(1) = SS EFF SEN LUCK SAL FI

JOB → ED, X(1) = SS EFF SEN LUCK JOB ED

LUCK → ED EFF SEN JOB → SAL, X(2) = SS EFF SEN LUCK JOB
ED SAL

SALSS → FI, X(3) = SS EFF SEN LUCK JOB ED SAL FI

FI → MOC

Keys: SS EFF SEN LUCK JOB

SS EFF SEN LUCK ED

Partial Dependency

X → Y, X'CX s.t. X' → Y

↓

Partial dependes

Elementary FD

F $f: X \to A$, $B \in X$

$(X - \{B\}) \to A$ is in F^+

Then B is extraneous attribute and it can be ~~dropped~~ dropped.

Example: $F = \{AB \to DEF, AC \to G, A \to C\}$

$(A)^+ = ?$

$N = 0 \quad X(0) = A$

- $A \to C$, $X(1) = AC$

$AC \to G$, $X(2) = ACG$ $\quad (AC)^+ = ACG$.

$\therefore$ C is the extraneous attribute in $AC \to ACG$ as $A \to ACG$.

$\therefore F' = \{AB \to DEF, A \to G, A \to C\}$

$= \{AB \to DEF, A \to CG\}$

- **Redundant FD**

f in (F) if $(F - f)^+ = F^+$

$f: X \to A$

If an FD is removed from the set because it can be logically derived from other FDs, then that particular FD, f in this case, is called, redundant FD.

- **Minimum FD**

A set of FD, F, is minimum when there is no set of FDs, G, with less no. of FDs than F, such that

$G^+ = F^+$

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

* $G_1 = \{A \to BC\}$

L-Minimum

Set of FDs. F is L-minimum if

1) If F is minimum

2) There is no partial dependency in F.

Example : $\{ABC \to D, A \to B\}$

Minimum but B is redundant.

* → $\{AC \to D, A \to B\}$

→ ~~$\{AC \to BD\}$~~

LR-Minimum

Set of FDs which is minimum on the right-hand side.

$A \to AB$ ✗

Example : R(A, B, C, D, E, F)

$F = \{AB \to E, AC \to F, AD \to B, B \to C, C \to D\}$

$AB \to E$: NR $(AB)^+ = ABCDF$

$AC \to F$: NR $(AC)^+ = ACDBE$

$AD \to B$: NR $(AD)^+ = AD$

} Redundancy in FDs

$AB \to E$: $A^+ =$ $B^+ = BCD$

$AC \to F$: $A^+ =$ $C^+ = CD$

$AD \to B$: $A^+ =$ $D^+ =$

} Partial dependency

Example: $F = \{AB \to C, C \to A, BC \to D, ACD \to B, D \to EG, BE \to C, CG \to BD, CE \to AG\}$

$AB \to C$ N.R.	$BE \to C$
$C \to A$ N.R.	$CG \to B$
$BC \to D$ N.R.	$CG \to D$ (R)
$ACD \to B$ (R)	$CE \to A$ (R)
$D \to E$	$CE \to G$
$D \to G$	

Check for redundancy.

~~RE~~

~~LR-Minimum set~~: Non-redundant set:

$AB \to C$	$BE \to C$
$C \to A$	$CG \to B$
$BC \to D$	$CE \to G$
$D \to E$	
$D \to G$	

Now, check for partial dependency.

There exists no partial dependency.

To minimize the above set, we merge ~~the~~ 2 FDs which have same attributes on the left-hand side. ∴ In this case, we merge $D \to E$ and $D \to G$.

As each FD represents effectively a table, ∴ if we reduce an FD by merging, we would be able to reduce a table.

$AB \to C$	$BE \to C$
$C \to A$	$CG \to B$
$BC \to D$	$CE \to G$
$D \to EG$	

Decomposition of Relational Scheme

$$R = \{A_1, A_2, \ldots, A_n\}$$

$$P = \{R_1, R_2, \ldots, R_k\}$$

$$R = R_1 \cup R_2 \cup \ldots \cup R_k$$

R is a relation of n attributes $A_1, A_2, \ldots, A_n$.
R is decomposed into k sub-relations $R_1, R_2, \ldots, R_k$, such that their union gives back R.

Lossless Join ~~Probability~~ Property

$$R : R_1, R_2, \ldots, R_k$$

D is a set of dependencies.

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \bowtie \ldots \bowtie \Pi_{R_k}(r)$$

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_2	b_3	c_2

$R(A, B, C)$

$\Pi_R(A, C)$

A	C
a_1	c_1
a_2	c_2
a_2	c_2

∴ For each ~~relat~~ sub-relation we consider

Theorem:

If $\rho = (R_1, R_2)$ is a decomposition of R and F is a set of ~~dependies~~ dependencies then ρ is a lossless join decomposition in F if and only if

$$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$$

or

$$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$$

→ Attributes of R_1 are removed from R_2

Example:

$R(A, B, C)$ $F = \{A \rightarrow B\}$

$R_1(A, B)$, $R_2\{A, C\}$

$R_1 \cap R_2 = A$

$R_1 - R_2 = B$ → satisfies the lossless join property

$A \rightarrow B$ maybe in F or F^+

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

Let $r = \{a_1, b_1, c_1, a_2, b_1, c_2\}$

$\Pi_{AB}(r) = \{a_1, b_1, a_2 b_1\}$

$\Pi_{BC}(r) = \{b_1, c_1, b_1, c_2\}$

$\Pi_{AB}(r) \bowtie \Pi_{BC}(r) = \{a_1, b_1, c_1, a_2, b_1, c_1, a_1, b_1, c_2, a_2, b_1, c_2\} \neq r$

Relations are joined through the common attribute i.e. B in this case.

Preservation of Dependencies

$$R \Rightarrow R_1, R_2, \ldots, R_k$$

$$\bigcup_{i=1}^{k} \Pi_{R_i}(F) = F$$

If dependence is not preserved, I may not be able to answer all possible queries.

Example: $R(C, S, P)$

$$F = \{CS \rightarrow P,\ P \rightarrow C\}$$

$R_1(S, P)$ $\quad$ $R_2(C, P)$

$\Pi_{R_1}(F) = \{S \rightarrow S,\ P \rightarrow P\}$

$\Pi_{R_2}(F) = \{P \rightarrow C\}$ Does not preserve dependency.

Example: $R(A, B, C, D)$; $F = \{A \rightarrow B,\ C \rightarrow D\}$

$R_1(A, B)$ & $R_2(C, D)$ Not lossless

$\Pi_{R_1}(F) = A \rightarrow B$

$\Pi_{R_2}(F) = C \rightarrow D$

} Dependency is preserved.

$$\Pi_{R_1}(F) \cup \Pi_{R_2}(F) = F$$

★Candidate key

Individual columns in a table that qualifies for the uniqueness of all rows. They all are candidates for primary key.

★Primary key

Primary keys are the columns that are chosen to maintain uniqueness in a table.

★Foreign key

Collection of key/keys in 1 table that uniquely identifies a row of another table. It is a primary key for some other table.

★Super key

In a primary key, if ~~no~~ one or more columns are added, then it becomes a super key.

Example: (S.name, RN, Hall)

Primary key: RN

Super key: RN, S.name

Normal Forms (NF)

It is a process by which when decomposition is done, it is free of anomalies.

Boyce Codd Normal Form (BCNF)

R: $F: X \rightarrow Y$

If Y is not a subset of X and X is a key of the relation R, then F is a BCNF. If all FDs in F are in BCNF, then R is BCNF.

Example: R (PAN, PI, DI, DRUG, QTY, COST).

$F = \{PAN \rightarrow PI, PI \rightarrow DI, PI \cdot DRUG \rightarrow QTY, DRUG \cdot QTY \rightarrow COST\}$

PAN DRUG → Key

$(PAN\ DRUG)^+ = (PAN\ PI\ DI\ DRUG\ QTY\ COST)$

PAN → PI (Not in BCNF)

Decomposition into BCNF:

$R_1 (PAN \rightarrow PI)$ $R_2 (PAN, DI, DRUG, QTY, COST)$

↓

Expect PI.

$\Pi_{R_2}(F) = \{DRUG\ QTY \rightarrow COST\}$

↳ Not BCNF

$R_3 (DRUG, QTY, COST)$ $R_4 (PAN, DI, DRUG, QTY)$

PAN → DI

R_5 (PAN, DI) R_6 (PAN, DRUG, QTY)

R is R_1, R_3, R_5 and R_6.

Theorem:

BCNF decomposition produces lossless join decomposition always.

~~Theorem~~ May or maynot preserve dependency of relations.

Example: {C → A, AE → B, BF → C, CD → EF, ~~EF~~ EF → AD}.

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

*Prime Attribute

An attribute A of a relation R is known as a prime attribute if it is member of key of R. Otherwise it is a non-prime attribute.

Example: R(A,B,C,D), F={AB→C, B→D, BC→A}

Key: AB, BC

A, B, C → prime attributes

D → Non-prime attribute

Third Normal form (3NF)

When A is not ~~x~~

A relational scheme R is a 3NF whenever there is one FD of the form X→A where A is not in X, then either X is a key or A is a prime attribute.

Bernstein's 3NF Algorithm

Input: Set of attributes in a relation and a set of FDs.

Output: Set of sub-relations such that they are free of anomalies.

1) Rewrite the FDs such that there is only attribute on the ~~outside~~ right-side of the FDs.

2) Rewrite the ~~least~~ list of FDs such that there exists no redundant FD.

3) Rewrite the FDs such that no proper subset of left-hand side functionally determines right-hand side.
4) Combine the dependencies with same left-hand side using the union rule.

If $X \to Y$ is in the list, make a decomposition with attributes $\{X, Y\}$. This will ensure preservation of dependencies. Now, relation is in minimal form.
5) Calculate the keys of the original relation. If no key is included in any of the decomposed sub-relations, then create a new sub-relation with the attributes of key only. This will ensure lossless join property.
6) If any of the sub-relation is a subset of the other, remove the smaller sub-relation.

This algorithm ensures lossless join property as well as preservation of dependencies.

Example : $R(PAN, PI, DI, DRUG, QTY, COST)$

$F = \{PAN \to PI, PI \to DI, PI \cdot DRUG \to QTY, DRUG \cdot QTY \to COST\}$

F is non-redundant as well as free of partial dependencies.

$\{PAN, PI\}, \{PI, DI\}, \{PI, DRUG, QTY\}, \{DRUG, QTY, COST\}$

Key : PAN. DRUG

$\{PAN, DRUG\}$

Example : $R(A,B,C,D,E,F,G,H,I)$

$F = \{A \to BCD, AE \to FG, F \to AEG, C \to HI\}$

(i) $A \to B$ $AE \to G$ → Redundant

$A \to C$ $F \to A$ $C \to H$

$A \to D$ $F \to E$ $C \to I$ } No redundant FA

$AE \to F$ $F \to G$

No partial-dependency.

(ii) $A \to BCD$ $AE \to F$ $F \to AEG$ $C \to HI$

Sub-relations : $\{A,B,C,D\}, \{A,E,F\}, \{A,E,F,G\}, \{C,H,I\}$

Key : AE or F

Sub-relations : $\{A,B,C,D\}, \{A,E,F,G\}, \{C,H,I\}$

3NF : $R_1 = \{A,B,C,D\}, R_2 = \{A,E,F,G\}, R_3 = \{C,H,I\}$

1NF : All relations are 1NF.

2NF : No partial dependency but there may be transitive dependency.

*Multivalued Dependency

For one value of attribute more than one value (multiple) can be obtained by another attribute.

Example:

MATH	T1	B1	MATH1	T1	B1
	T2	B2	MATH1	T2	B2
	T3		MATH1	T2	B1
DSA	T11	D1	MATH1	T2	B2
	T12	D2			
		D3			

$(C, t_1, x_1), (C, t_2, x_2)$ appears then $(C, t_1, x_2), (C, t_2, x_1)$ will also appear.

$X \rightarrow\rightarrow Y$ holds in R

t, s $t[X] = s[X]$ then $t, s \rightarrow$ tuples.

then $\exists$ $u, v \rightarrow$ tuples, such that

1. $u[X] = v[X] = t[X] = s[X]$
2. $u[Y] = t[Y]$ and $u[R-X-Y] = s[R-X-Y]$
3. $v[Y] = s[Y]$ and $v[R-X-Y] = t[R-X-Y]$

X	Y	R-X-Y
t[X]	t[Y]	t[R-X-Y]
s[X]	s[Y]	s[R-X-Y]
u[X]	u[Y]	u[R-X-Y]
v[X]	v[Y]	v[R-X-Y]

- $R(X, Y, Z)$

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

If Y and $R - X - Y$ are interchanged then the relation remains the same, therefore the above holds.

- $\{X \rightarrow\rightarrow Y, Y \rightarrow\rightarrow Z\}$ then $X \rightarrow\rightarrow (Z - Y)$
- $\{X \rightarrow Y\}$ then $\{X \rightarrow\rightarrow Y\}$

★ 4th Normal Form (4NF)

R is a relation with a set of dependencies D. R is in 4NF if there is an MVD of the form $X \rightarrow\rightarrow Y$ where Y is not contained in X and XY does not include all the attributes of R, then X is a key of R.

$X \rightarrow\rightarrow Y$ $\quad R\{X, Y, Z\}$

$Y \nsubseteq X$ and $XY \neq \{X, Y, Z\}$ then X is a key.

Example: $R(A, B, C, D, E, F, G)$

$D = \{A \rightarrow\rightarrow B, B \rightarrow\rightarrow G, B \rightarrow\rightarrow EF, CD \rightarrow E\}$

Key: ACD

$A \rightarrow\rightarrow B$ not in 4NF.

$R_1(A, B)$ ~~$R_2(A, C, D)$~~ $R_2(A, C, D, E, F, G)$

$CD \rightarrow E$

$R_3(C, D, E)$ $\quad R_4(A, C, D, F, G)$

$A \rightarrow\rightarrow G$

$R_5(A, G)$, $R_6(A, C, D, F)$

$A \rightarrow\rightarrow F$

$R_7(A, F)$, $R_7(A, C, D)$

Sub-relations:

(A, B), (C, D, E), (A, G), (A, F), (A, C, D)

SQL

Table z

a	g	d
4	6	3
7	4	6
4	9	1
3	8	4
7	6	3
4	9	5

Query: Select g,a,d,a from z

Project

g	a	d	a
6	4	3	4

Select * from z where a+d>g

Select σ

→ Selects rows from z such that the entries have a+d>g.

a	g	d
4	6	3
7	4	6
7	6	3

Select Max(g) from z ⇒ 9

Select Sum(d) from z ⇒ 17

~~Select a, Max(g), d from z ⇒ Selects the entire row for which e~~

Select * from z where g=9

or

Select * from z where g = (Select Max(g) from z)

a	g	d
4	9	1

Select * from z where g in (9,8,6)

a	g	d
4	6	3
4	9	1
3	8	4
7	6	3

Select a, Max(g) from z group by a

a	Max(g)
4	9
7	6

Select d, Max(g) from z group by a

d	Max(g)
1	9
3	6
4	8

Select d, g from z where g in (a,g) in (Select a, Max(g) from z group by a)

↓

{(4,9), (7,6), (3,8)}

d	g
1	9
5	9
3	6
4	8

Select d,g from r where (a,g) in (Select a, sum(g) from r group by a)

d	g
3	15
6	10
3	10

d	g
4	8

↓

{(4,9), (7,6), (3,8)}

Create table U(a int, k int)

a	k
5	9
3	6
8	4
3	1

insert into U values(5,9)

insert into U values(3,6)

insert into U values(8,4)

Create table U(a int, k int, check, primary key a) ^ k

insert into U values(3,1)

Create table V(p int, h int, m int, primary key p,m,
foreign key m references U(k))

insert into V values(4,3,6)

insert into V values(4,8,1)

insert into V values(6,7,1)

insert into V values(9,1,7) X

insert into U values(8,7)

insert into V values(9,1,7) ✓

V

p	h	m
4	3	6
4	8	1
6	7	1
9	1	7

U

a	k
5	9
3	6
8	4
3	1
8	7

Example:

Players

Name	Age
Anil	48
Dipu	17
Gyan	29
Hari	23
Jalaj	18
Kapil	37

Performance

Name	No	Runs
Gyan	#3	65
Dipu	#4	71
Hari	#3	82
Gyan	#1	73
Gopal		

Matches

No.	Place
#1	Delhi
#2	Bombay
#3	Kanpur
#4	Patna
#5	Delhi

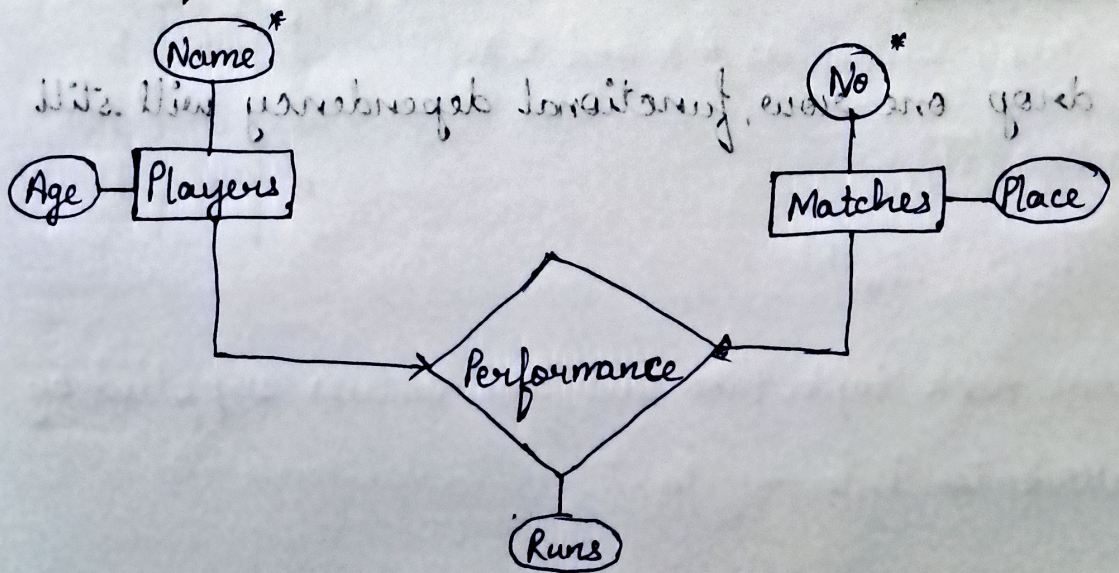
Create table player (Name char(10), age int, primary key Name).

Create table matches (no. int, place char(6), primary key No)

Create table performance (Name char(10), no int, runs int, primary key (Name, No),

foreign key Name references player (Name),

foreign key No references Matches (No))



z

a	g	d
4	6	3
7	4	6
4	9	1
3	8	4
7	6	3
4	9	5

k

b	g
3	6
2	9
8	4

Select * from z, k

↓ Join z ⋈ k

a	z.g	d	b	k.g
4	6	3	3	6
4	6	3	2	9

6 × 3 = 18 rows

Select * from z, k where z.g = k.g

↓ Natural join z ⋈ k

a	z.g	d	b
4	6	3	3
4	6	3	5
7	4	6	8
4	9	1	2
7	6	3	3
7	6	3	5
4	9	5	2

ad → g

(4,3) → 6

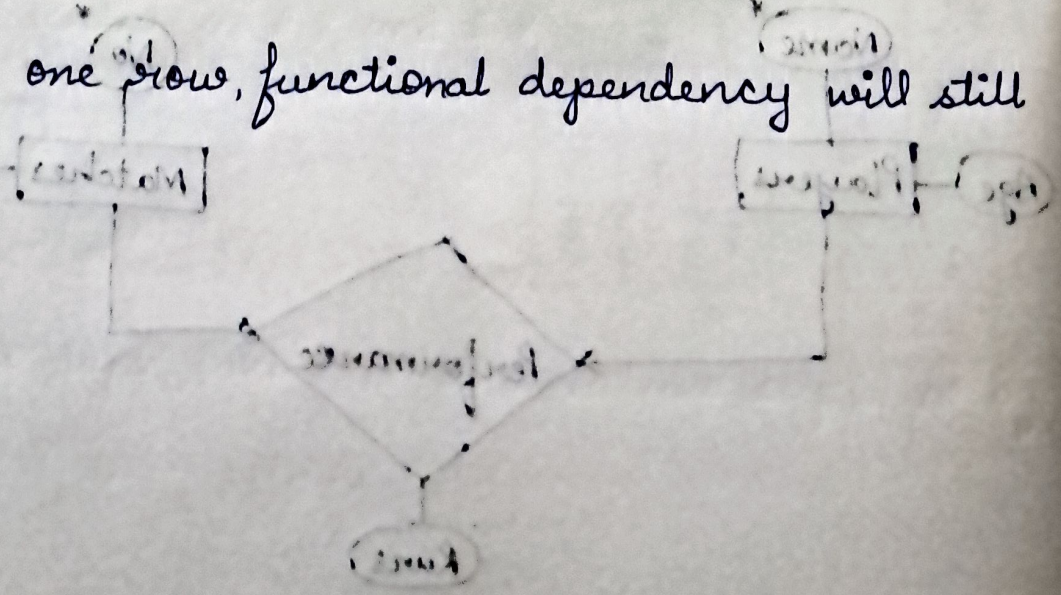
(7,6) → 4

(4,1) → 9

(7,3) → 6

(4,5) → 9

If we drop one row, functional dependency will still hold.



P	Q	R
4	2	8
4	7	6
5	1	9
4	7	8
5	1	3
4	2	6

P = 4 Q = {2, 7}

hence P → Q absent

Q = {2, 7} independent

hence P →→ Q

★RELATIONAL ALGEBRA

1. Union (U)

R ∪ S

A	B	C
a	b	c
d	a	f
c	b	d

R

D	E	F
b	g	a
d	a	f

S

No. of attributes should be same for both R and S.

a	b	c
d	a	f
c	b	d
b	g	a

2) Set Difference

R - S

Set of tuples which will be in R but not in S.

a	b	c
c	b	d

3) Cartesian - Product (X)

R X S

$R \rightarrow k_1$ (No. of attributes)

$S \rightarrow k_2$

∴ A table with tuples $k_1 + k_2$ will be formed.

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	b	g	d
c	b	d	b	g	a
c	b	d	d	a	f

If the name of an attribute is same (say X) then it will be represented as R.X and S.X.

4) Projection (Π)

$\Pi_{i_1, i_2, \ldots, i_m}(R)$

→ From each tuple, take out the values supplied by the attributes $i_1, i_2, \ldots, i_n$.

C	A
c	a
b	d
d	c

$\Pi_{C,A}(R)$

$\Pi_{2,3}(S)$

E	F
g	a
a	b

5) Selection (σ)

F is a formula. operands may be ~~alternative~~ attr. name, col. no. or a constant along with logical & arithmetic operators.

Tuples satisfying the formula F will be chosen.

$\Pi_F(R)$ $\sigma_F(R)$ $\sigma_{2>3}(R)$

$\sigma_{1='s' \vee 2='p'}(R)$

$\sigma_{B='b'}(R)$

A	B	C
a	b	c
c	b	d

Additional Operators

1. Intersection ($\cap$)

$R \cap S \Rightarrow$ Tuples which are present in both.

$R \cap S = R - (R - S)$

d	a	f

2. Quotient ($\div$)

$R^{(r)}$ $S^{(s)}$ $r > s$, $s \neq \emptyset$

$R \div S$ is the set of tuples with $R_t = (r - S)$ such that for all tuples u in S, tu is in R, where t is a tuple of $R \div S$.

R

a	b	c	d
a	b	e	f
b	c	e	f
e	d	e	d
e	d	e	f
a	b	d	e

S

c	d
e	f

R ÷ S

a	b
e	d

3) θ-Join

$R \bowtie_{i\theta j} S$ ~~R×S~~

A	B	C
1	2	3
4	5	6
7	8	9

D	E
3	1
6	2

$\sigma_{(i\theta(r+j))} R \times S$

↓

Column numbers of s change when ~~carte~~ Cartesian product is taken.

$R \bowtie_{B<D} S$

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2
1	2	3	6	2

Natural Join (⋈)

For each attribute A which appears in both R and S, select those tuples from R X S where

$$R.A = S.A$$

or, from R X S, take those tuples for which if the attribute name appears in both R and S, value is same

1. R X S
2. R.A = S.A
3. Remove S.A

R

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b

R X S

A	R.B	R.C	S.B	S.C	D

Select tuples for which R.B = S.B and R.C = S.C

R ⋈ S

A	B	C	D
a	b	c	d
d	b	c	d
c	a	d	b
a	b	c	e
d	b	c	e

Example: Employee (Emp-id, name, salary, M-id)

M-id = 1 for manager
= 0 otherwise

Find the name of employees whose salary is more than any manager.

$$\Pi_2(\sigma_{4\neq'1' \wedge 8='1' \wedge 3>7}(\text{Employee} \times \text{Employee}))$$

Example:

Customer (C-name, street, ~~city~~ C-City)

Deposit (B-name, ac. no., C-name, Balance)

Borrow (B-name, Loan no., C-name, amount)

Branch (B-name*, B-city, Asset)

Client (C-name*, Emp. name)

1. ~~Client~~ Customer names for which name of ~~client~~ customer and emp. is same.

$$\Pi_1(\sigma_{1=2}(\text{client}))$$

2. 'XYZ'

Find customer names and customer cities for which employee name is 'XYZ'

$$\Pi_{1,5}(\sigma_{2='XYZ' \wedge 1=3}(\text{Client} \times \text{Customer}))$$

3. Find all customers who have both account and loan at KGP branch.

$$\pi_3 (\sigma_{1='KGP' \wedge 5='KGP' \wedge 3=7} (Deposit \times Borrow))$$

$$\pi_3 (\sigma_{1='KGP'} (Borrow \bowtie Deposit))$$

4. Find all customers who had account in ~~all account~~ all branches in the city Kolkata.

$$\pi_{C\text{-name}, B\text{-name}} (Deposit) \div \pi_{B\text{-name}} (\sigma_{B\text{-city}='KOL'} (Branch))$$

Example: Lives (p-name, Street, City)

Works (p-name, C-name

1. Find the name, and city for all employees who work for FCI company.

2. Find the name, street and city of all employees who work for FCI company and salary is more than 50000.

3. Find all customers who have both account and loan at KGP branch.

$$\Pi_3 \left(\sigma_{1='KGP' \wedge 5='KGP' \wedge 3=7} (Deposit \times Borrow) \right)$$

$$\Pi_3 \left(\sigma_{1='KGP'} (Borrow \bowtie Deposit) \right)$$

4. Find all customers who had account in ~~all~~ account all branches in the city Kolkata.

$$\Pi_{C\text{-}name, B\text{-}name}(Deposit) \div \Pi_{B\text{-}name}(\sigma_{B\text{-}city='KOL'}(Branch))$$

Example: Lives (p-name, Street, City)

Works (p-name, C-name, Salary)

located_in (C-name, city)

Managers (p-name, m-name)

1. Find the name, and city for all employees who work for FCI company.

$$\Pi_{1,3}(Lives \bowtie \Pi_{p\text{-}name}(\sigma_{C\text{-}name='FCI'}(Works)))$$

2. Find the name, street and city of all employees who work for FCI company and salary is more than 50000.

$$((\sigma_{C\text{-}name='FCI' \wedge Salary>5000}) \Pi_{pname} \bowtie Lives)$$

3. Find all employees who live in the same city and the company they work for.

$$\pi_{1,3}\left(\sigma_{3=8}(\text{Lives} \bowtie \text{Works} \bowtie \text{Localedin})\right)$$

4. Find all employees who live in the same city and same street as their manager.

P = LIVES ⋈ Mangers

$$\pi_1(\sigma(\text{LIVES} \bowtie P)$$

p_name = M_name ∧ 2 = 4 ∧ 3 = 5 ∧ Lives.p_name ≠ Managers.p_name

LIVES.

5. Find all employees whose salary is more than every employee of 'XYZ' company.

$$\pi_{1,3}\left(\sigma_{2='XYZ'}(\text{Works} \times \text{Works}) - \sigma_{3<6 \wedge 3='XYZ'}(\text{Works} \times \text{Works})\right)$$

Or

$$P = \pi_3\left(\sigma_{c\_name='XYZ'}(\text{Works})\right) \qquad R = P - Q$$

$$Q = \pi_1\left(\sigma_{1<2}(P \times P)\right)$$

PROPOSITIONAL LOGIC

$\neg$ (not), $\wedge$ (and), $\vee$ (or),

$\Rightarrow$ (implication or if-then)

$\equiv$ (equivalence or iff)

* Proposition

A statement which uses variables and symbols (atoms)

Atoms: Logical symbols: $\neg, \wedge, \vee, \Rightarrow, \equiv$

* Tuple Relational Calculus

$\{t \mid \psi(t)\}$

t is a tuple variable, of some fixed length and $\psi(t)$ is a formula built from atoms.

$R(s)$, $s \in R$

Here, the representation is in terms of tuple.

$s[i]$ $u[j]$

Set of attributes give tuple.

★ Existential quantifier ($\exists$)

$(\exists x)\, \psi(x)$

★ Universal quantifier ($\forall$)

$(\forall x)\, \psi(x)$

Example: $(\exists s)\, R(s)$ ⇒ There exists atleast one tuple s which is a tuple of relation R.

$(\forall s)(\psi(s))$ → All s are such that they belong to $\psi(s)$.

All expressions which can be expressed in relational algebra can be expressed in tuple relational calculus.

$\{t \mid R(t) \vee S(t)\}$

★Existential quantifier ($\exists$)

$(\exists x)\,\psi(x)$

★Universal quantifier ($\forall$)

$(\forall x)\,\psi(x)$

Example: $(\exists s)\,R(s)$ ⇒ There exists atleast one tuple s which is a tuple of relation R.

$(\forall s)(\psi(s))$ ⇒ All s are such that they belong to $\psi(s)$.

All expressions which can be expressed in relational algebra can be expressed in tuple relational calculus

$$\{t \mid R(t) \vee S(t)\}$$

$R \cup S : \{t \mid R(t) \vee S(t)\}$

$R - S : \{t \mid R(t) \wedge \neg S(t)\}$

$R^{(r)} \times S^{(s)} : \{t^{(r+s)} \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1] = u[1] \wedge t[2] = u[2] \wedge \dots \wedge t[r] = u[r] \wedge t[r+1] = v[1] \wedge \dots \dots t[r+s] = v[s]\}$

First r attributes in t belong to u and the next s attributes belong to v.

$\Pi_{i_1, i_2, \ldots, i_k}(R) : \{t^{(k)} \mid (\exists u)(R(u) \wedge t[1] = u[i_1] \wedge \ldots t[k] = u[i_k])\}$

$\Pi_{A,C}(R) : \{t \mid (\exists u)(R(u) \wedge t[1] = u[A] \wedge t[2] = u[C])\}$

$\sigma_F(R) : \{t \mid R(t) \wedge F'\}$

Example: $\Pi_{1,4}(\sigma_{2=3}(R^{(2)} \times S^{(2)}))$

$= \{t^{(4)} \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1] = u[1] \wedge u[2] = u[2] \wedge$
$t[3] = v[1] \wedge t[4] = v[2] \wedge u[2] = v[1])$

$\{t \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge u[2] = v[1] \wedge t[1] = u[1] \wedge t[2] = v[2]\}$

Q. Find the branch no., loan no, and customer name and amount for loans above ₹10000.

Customer (C-name, street, C-city)

Deposit (B-name, ac. no., C-name, Balance)

Borrow (B-name, Loan no., C-name, amount)

Branch (B-name, B-city, Asset)

Client (C-name, Emp. name)

$\{t \mid t \in Borrow \wedge t[4] > 10000\}$.

Find all customers who have loan more than 10000.

$\{t \mid t \in (\exists u)(Borrow(u) \wedge u[4] > 10000 \wedge t[1] = u[3])\}$

Find all customers who have loan from IIT bran and the cities where they live.

$$\{t \mid (\exists u)(\exists v)(Borrow(u) \wedge Customer(v) \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3] \wedge u[3] = v[1] \wedge t[1] = \{v[1] \wedge t[2] = v[3]\}$$

$$\{t \mid (\exists u)(\exists v)(u \in Borrow \wedge v \in Customer \wedge u[1] = \text{'IIT'} \wedge u[3] = v[1] \wedge t[1] = u[3] \wedge t[2] = v[3]\}$$

$$\{t \mid (\exists u)(u \in Borrow \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3]) \wedge (\exists v)(v \in Customer \wedge u[3] = v[1] \wedge t[2] = v[3])\}$$

Find all customers having a loan or account or both at IIT Branch.

~~$\{t \mid (\exists u)(\exists v) \; R(u) \wedge S(v) \; (\forall u)(\forall v) \; R(u) \vee S(v) \; Deposit(u) \wedge Borrow(v)$~~

$$\{t \mid (\exists u)(Borrow(u) \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3]) \vee (\exists v)(Deposit(v) \wedge v[1] = \text{'IIT'} \wedge t[1] = v[4])$$

$$\{t \mid (\exists u)(Deposit(u) \wedge u[1] = \text{'IIT'} \wedge t[1] = u[3]) \vee (\exists v)(Borrow(v) \wedge v[1] = \text{'IIT'} \wedge t[1] = v[3])$$

Find all customers who had account at all branches in the city 'KGP'.

$$\{t \mid (\forall u)(u \in Branch \wedge u[2] = \text{'KGP'}) \Rightarrow (\exists v)(v \in Deposit \wedge u[1] = v[1] \wedge t[1] = v[3])\}$$

$P \Rightarrow Q \equiv \neg P \vee Q$

$\{t \mid (\exists u)(u \notin \text{Branch} \vee u[2] \neq \text{'KG Rd'}) \vee (\exists v)(v \in \text{Deposit} \wedge u[1] = v[1] \wedge t[1] = v[3])\}$

Midsem syllabus: Till Tuple Calculus

* SQL

Select $A_1, A_2, \ldots, A_n$ from $R_1, R_2, \ldots, R_m$ where P;

$\equiv \Pi_{A_1, A_2, \ldots, A_n}(\sigma_P(R_1 \times R_2 \times \ldots \times R_m))$

union intersect minus

and or not

(i) Find the name of all branches in Deposit relation.

```
select b_name          or    select distinct(b_name)
from Deposit;                from Deposit;
```

(ii) Find all customers who have account at IIT Branch.

```
select c_name
from Deposit
where b_name='IIT';
```

(iii) Find all customers having a loan or account or both at IIT Branch.

~~select c_name~~

```
select c-name
from Deposit
where b-name = 'IIT'
```
union
```
select c-name
from Borrow
where b-name = 'IIT'
```

(iv) Find all customers having a loan, at any branch and their city.

```
select C.c-name, C_city
from Customer C, Borrow B
where C.c-name = B.c-name
and b-name = 'IIT'
```

(v) Find all customers who have both loan and account at IIT Branch.

```
Select c-name
from Deposit
where b-name = 'IIT'
and c_name in
(Select c-name
from Borrow
where b-name = 'IIT')
```