Unit-3

Database Design Theory: Fundemental Dependencies, Normal forms

Relational DB Design Algorithm.

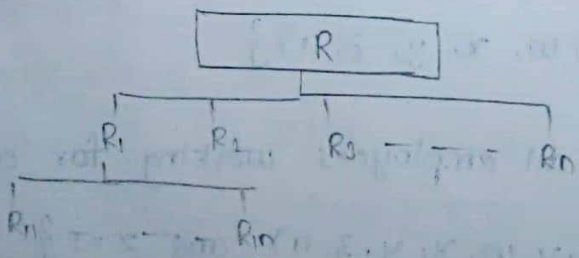
RDB Design approach:

1. ER-to-Relational mapping algorithm
2. Bottom-up Design (RDB design by synthesis)

Based on similarity attributes are grouped into relations if there exists a binary relations between 2 relations they are identified if not separation of attributes takes place

3. top-down Design (RDB design by analysis)

Initially all attributes encountered from requirement are made in single relationcaw called Universal Relation Schema. Again it is decomposed $R_1, R_2 \ldots R_n$ and each relation is decomposed into sub relation



This process continues until leaf node or bottle relations are 3NF or BCNF forms

Design guidelines for RDB Design:

1. Imparting clear scemantics to attributes of relation
2. Reducing the redudancy with relations
3. Reducing null valued attributes within a relation

Scanned with CamScanner

4. This allowing generation of spurious tuples in relations.

① → semantic refers meaniful

meaning ful attributes are to provided in relation

company RDB = { employee, depaitment, project, dependent, works_on, dept_loc }

Acc to this, in a RDB design construct a relation with attributes corresponding to not more than single entity type. or relation ship do not combine attributes belonging to more than one entity type.

② → Acc to this, do not create

Update Anamdies is a problem that is encountered if redundancy problem is not handled propely within relations

1. Insert Anamoly - User faces problems during Insertion of data & ~~adding~~
2. Deletion Anamoly - problem during deletion
3. Modification Anamoly

③ → Acc to this, form a relation in such a way that no null value is allowed

④ → Spurious : meaning lecs

Meaning less tuples even after join.

Allow the join only on relavant attributes

ex primary & foreign keys

Scanned with CamScanner

→ Functional Dependency (FD):

FD is a constraint between two subsets of attributes X and Y in a relation R represented by $X \rightarrow Y$

This constraint holds that all tuples forming a relation state r in a relation R will specify for any 2 tuples t_1 and t_2 if $t_1[X] = t_2[X]$ then $t_1[Y] = t_2[Y]$ holds. means the value for y in t_1 is uniquely determined based on value of x in t_1 then the same must to hold in t_2 also. Here x determines a unique value for y

Relation emp-proj

	ssn	pno	pname	ploc	ename	hours
t_1 →	12345	1	pro-x	newyork	smith	10
$r(R)$ t_2 →	12346	1	pro-x	newyork	franklin	8.5
↑	12345	2	pro-y	chicago	smith	12
!						

F = { ssn → ename

select ename from employee where ssn = 12345 ↵
smith

select ename from employee where ssn = 12346 ↵
franklin

select ~~pname~~ pno from employee where ssn = 12345 ↵

1 ~ 2

so dependency b/w ssn & pno, depedency b/w ssn & ename

Scanned with CamScanner

F = { ssn → ename,
pno → {pname, ploc},
{ssn, pno} → hours }

FD's set

teach	teacher	cource	textbook
	mpp	dbms	navatha
	vss	ds	Allenweis
	nm	dbms	navatha
	mpp	oood	benett

F = { cource → textbook }

teacher → cource X

→ Inference Rules:

IR1: Reflexive Rule: If X is a proper superset ⊇ Y then always X → Y or X → X is a valid FD

IR2: Augmentation Rule: X → Y is valid FD then ⊨ (infer or deduce or derive) XZ → YZ is also a valid FD.

Ex: ssn → ename

{ssn, pno} → {ename, pno}

Transition Rule: IR3 : If X → Y is valid FD & Y → Z is valid FD ⊨ X → Z is valid FD

These three are called Armstrong

Scanned with CamScanner

IR4: Decomposition Rule: $X \to \{A_1\ A_2\ A_3\} \models X \to A_1, X \to A_2, X \to A_3$

IR5: Union Rule: (Reverse of decomposition)

If $X \to A_1, X \to A_2, \ldots X \to A_3 \models X \to \{A_1, A_2 \ldots A_3\}$

IR6: pseudo transitive rule: If $X \to Y$ is available then $WY \to Z \models$ (then we can infer) $WX \to Z$

→ Closure Set of FD: If F is a FD set given on R then closure set of F^+ for F includes all FD's in F along with additional FD's inferred through IR1 to IR3

→ Algorithm for Obtaining Closure Set: This method obtains closure set for the given FD set in F based on each LHS attribute of X

$x^+ = x \quad (x^+ \leftarrow x)$

Repeat

old $x^+ = x^+$

for each FD $Y \to Z$ in F

if $x^+ \supseteq Y$ then $x^+ \cup Z$

until $(x^+ = \text{old } x^+)$

R ⇒ ssn pno pname ploc ename hours

F = { ssn → ename,

pno → {pname, ploc}

{ssn, pno} → hours }

for each LHS attribute x of FD in F,

$x = ssn$

$\{ssn\}^+ = \{ssn, ename\}$

$x = pno$

$\{pno\}^+ = \{pno, pname, ploc\}$

$x = \{ssn, pno\}$

$\{ssn, pno\}^+ = \{ssn, pno, ename, pname, ploc, hours\}$

then

$F^+ = \{ ssno \rightarrow \{ssn, ename\},$

$pno \rightarrow \{pno, pname, ploc\}$

$\{ssn, pno\} \rightarrow \{ssn, pno, ename, pname, ploc, hours\}\}$

→ Trivial FD : A FD $x \rightarrow y$ is trivial FD if $x \supseteq y$ otherwise FD is non trivial FD

→ Equivalance Functional Depending sets:

consides two FD set E & F

i) cover of FD: FD set E is said to be covered if it contains all FDs available in F, in it F^+. Then we say E covers F

ii) equivalent FD sets - E and F are equivalent FD sets if $E^+ = F^+$ (or)

if E covers F and F covers E Then E and F are two equivalent FD

→ Check $F = \{A \to C, AC \to D, E \to AD, E \to H\}$ and

$G = \{A \to CD, E \to AH\}$ equivalent are not.

$R =$

A	B	C	D	E	H

$X = A$

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

$X = AC$

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

$X = E$

$\{E\}^+ = \{E, A, D, H\}$

$X = R$

→ Minimal Cover set: Minimal coverset F for the given FD set E :-

Algorithm to obtain a minimal set for the given FD set E

1. Initially assign E to F ; F = E

2. If F contains FD of the form $X \to A_1, A_2, \dots A_n$ then replace it in F by $X \to A_1, X \to A_2, \dots X \to A_n$

3. for each FD $X \to A$ do

   if $B \in X$ and $\{F - \{X \to A\} \cup \{(X - \{B\}) \to A\}$ is same as F then replace $X \to A$ in F by $\{(X - \{B\}) \to A\}$

4. For each FD $X \to A$ do

   if $F - \{X \to A\}$ is same as F means $X \to A$ is redundant functional dependency may be appeared in F but can be obtained by applying inference rules then remove $X \to A$ from F

→ Compute minimal set F for $E = \{B \to A, D \to A, AB \to D\}$

Method minimal set $F = \{E\} = \{B \to A, D \to A, AB \to D\}$

Step-2 Not applied

Step-3 FD $X \to A$ of F is with more than one attribute in X

$AB \rightarrow D$

$\{F - \{AB \rightarrow D\}\} \cup \{A \rightarrow D\}$ is not as same as F

$\{F - \{AB \rightarrow D\}\} \cup \{B \rightarrow D\}$ is also not same as F

step 3 cannot be applied.

4. Checking for Redundant FD $X \rightarrow A$ in F

① Let $B \rightarrow A$ from F

Then add B to LHS & RHS, according to Augmentation rule

$BB \rightarrow AB \Rightarrow B \rightarrow AB \Rightarrow B \rightarrow D$

so $AB \rightarrow D$ is replaced with $B \rightarrow D$

$F = \{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$.

(If $X \rightarrow Y$, $Y \rightarrow Z$ then replace with $X \rightarrow Z$ according)

By applying $X \rightarrow Y$, $Y \rightarrow Z$ and $X \rightarrow Z$ is available in F then that is redundant so remove that from F

$B \rightarrow D \quad D \rightarrow A \models B \rightarrow A$

$F = \{B \rightarrow D, D \rightarrow A\}$

Scanned with CamScanner

Algorithm to compute key attribute K from Relation R with set of attributes

For each attribute A in R

compute $\{K-A\}^+$ w.r.t FD set F on R

if $\{K \to A\}^+$ contains all the attributes of R

then $K = K - \{A\}$

→ R = | A | B | D |

$F = \{ B \to A, D \to A, AB \to D \}$

Normal forms based on keys & FD's:

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce-Codd Normal Form (BCNF) } minimal normal form
5. Fourth Normal Form (4NF)
6. Fifth Normal Form (5NF) } IN Normal Form

① → concept based on atomic values (simple, composite, multivalued attributes) and Nested relations

② → concept based on fully functional dependency (FFD)

③ → concept of transitive dependency

④ → concept based on super key

⑤ → multivalued dependency (MVD)

⑥ → Join dependency

Scanned with CamScanner

4. prime Attribute :- If attribute is member of candidate key then it is prime attribute

5. Non prime Attributes: If attribute in a relation is not member of candidate

6. Foreign key : If attribute in a relation is refference an attribute in other relation and should have same domain.

Denormalization: It is a process of storing a join of lower normal form relations in order to satisfy higher normal form condition.

→ 1NF: If a relation R is said to be 1NF if doesn contain multivalued attributes, nested relations (and some times composite attributes).

Eg: dept

dno	dname	dmgrssn	dloc
1	research	12345	{Blore, chennai, hyd}
4	sales	12346	hyd

dept contains multivalued attribute dloc, so it is not in 1NF. To make 1NF) for each value store as separate tuple.

then

dno	dname	dmgrssn	dloc
1	research	12345	blore
1	research	12345	chenn

Scanned with CamScanner

1	research	12345	hyd
4	sales	12346	hyd

But due to this redudancy occurs. To avoid that

(2) for multi-valued attribute if all distinct values are known create distinct columns for those values

dno	dname	dmgrssn	dloc1	dloc2	dloc3
1	research	12345	blore	chennai	hyd
4	sales	12346	hyd		

But to this null value problem arises. To avoid this

(3) Remove the multivalued column from that relation and create a new relation along with primary key of old relation. and make that attribute (pk) as foreign key. Combination of those attributes make as key attribute

R_1

dno	dname	dmgrssn
1	research	12345
4	sales	12346

R_2

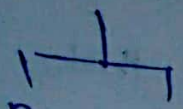
dno (fk)	dloc
1	blore
1	chennai
1	hyd
4	hyd

dno	dloc

Here also redudancy occurs but minimal redundancy is obtained.

This pre is the preferable procedure.

R:-



emp		project, works on	
ssn	ename	pno	hours
12345	smith	1	10
12345	smith	4	15
12345	smith	5	16
12346	~~smith~~ john	1	10
12346	~~smith~~ john	4	10.5

← Nested Relation

↓

(few attribute are taken from emp entity type few from project entity type and some from works_on entity type

R1:

ssn	ename
12345	smith
12346	john

pno	hours	essn
1	10	12345
4	15	12345

Add key attribute of one relation to another relation.

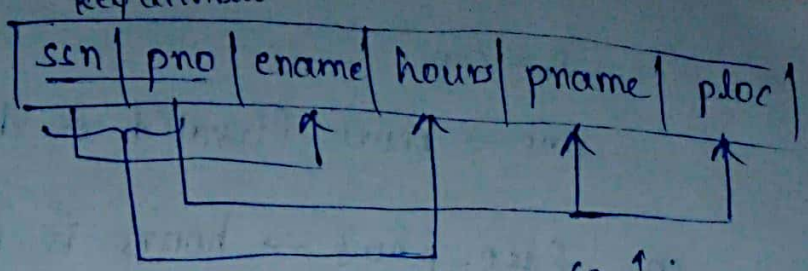
→ 2NF: A relation R is said to be in 2NF if it is INF & non-prime attribute is fully functionally dependent on the primary key of R

→ FFD: If $X \to Y$ is a FD for each attribute B is member of X ($B \in X$), then $X - \{B\} \to Y$ do not holds, then FD is said to be f-fd

Other wise if $X - \{B\} \to Y$ holds a valid FD then $X \to Y$ is partial FD

Scanned with CamScanner

R(emps_project)



(ignore nested relations)

Given FD set F = { ssn → ename,

pno → {pname, ploc}

{ssn, pno} → hours }

key attribute = {ssn, pno}

remove one attribute from key attribute

(ssn, pno) → ename

X → Y

let B = SSN (remove B)

pno → ename

we have to check whether is fd or not

so pno → ename is invalid

let B → pno (remove B)

ssn → ename → valid

Here both should be invalid only then it is in 2NF i.e not ffd (pfd)

so (ssn, pno) → ename is pfd

So this relation is not in 2NF

NPA Y = hours

X = {ssn, pno}

B = pno

ssn → hours is invalid

Scanned with CamScanner

B = ssn

pno → hours (invalid or. do not holds)

so {ssn, pno} → hours is FFD

Y = pname

X = {ssn, pno}

{ssn, pno} → pname

B = pno

ssn → pname (invalid)

B = ssn

pno → pname (valid)

So {ssn, pno} → pname is pfd.

Y = ploc

X = {ssn, pno}

{ssn, pno} → ploc

B = pno

ssn → ploc (invalid)

B = ssn

pno → ploc (valid)

{ssn, pno} → ploc is in pfd

R is not in 2NF bcoz ename, pname, ploc are not fully fd.

Scanned with CamScanner

Take the non-prime attribute which is ffd make a new relation along with key attribute

R_1 = | ssn | pno | hours |

And non p a which is pfd make new relation with valid ffd.

R_2 = | ssn | ename |

R_3 = | pno | pname | ploc |

~~R_4~~ = ~~| pno | ploc |~~

→ 3NF: It is based on concept of transitive dependency

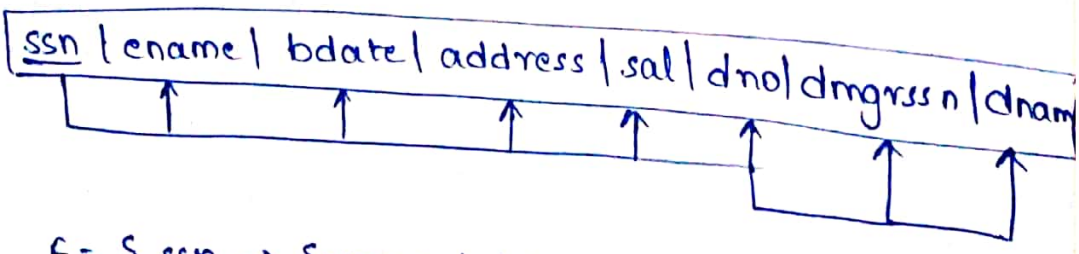
FD $X \rightarrow Y$ is transitive dependent if there $X \rightarrow Z$, $Z \rightarrow Y$ in FD set

Given relation R is in 3NF, if it satisfies 2NF condition, and no non-prime attribute is transitively dependent on key of R, then we say R is in 3NF.

(or)

A relation R is said to be in 3NF for each non trivial FD $X \rightarrow Y$ either X must be super key or Y is a prime attribute and also must be in 2NF.

Ex: emp-dept



not

F = { ssn → {ename, bdate, address, sal, dno}

dno → {dname, dmgrssn} }

Scanned with CamScanner

Given FD is in 1NF and 2NF also but it is not in 3NF because ssn → dname, ssn → dmgrssn are valid transitive dependency (ssn → dno
dno → dname, dmgrssn)

i.e non prime attributes dname, dmgrssn are transitively dependent on SSN key of R.

To satisfy 3NF, decompose R into sub relations identify the attribute A from the set of non-prime attributes by which valid transitive dependencies are inferred. Then all the attributes in a Relation along with A, A is a foreign key include into relation R_1. A along with the non-prime attributes which are transitively dependent include in relation R_2 in which A is primary key of R_2.

R_1 & R_2 are in 3NF condition

R_1 = | ssn | ename | bdate | address | sal | dno | (FK)

R_2 = | dno | dname | dmgrssn |

} 3NF

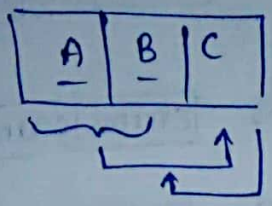
Scanned with CamScanner

Boyce Codd Normal form (BCNF):

A relation in BCNF is also in 3NF but not vice-versa.

If R has FD's of form

Trivial { X – proper subset of Y
Y – superset of Y



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

Then R is always in 3NF

A relation 'R' is said to be in BCNF, for each non-trivial FD $X \rightarrow Y$ in F^+, X must be a super key.

→ Student

sname	cname	instructor
smith	dbms	mpp
smith	os	nn
adams	dbms	vss
scott	os	kgs
tiger	dbms	mpp

Note: 1. Two students with same name cannot register course

2. Instructor is allowed to teach only one course.

$F = \{(sname, cname) \rightarrow instructor, instructor \rightarrow cname\}$

Relation student is in 3NF but not in BCNF because with non-trivial FD instr → cname, instructor is not.

super key. It is in 3NF because of $AB \rightarrow C$ and $C \rightarrow B$

In order to satisfy BCNF condition the above relation can be decomposed into 3 ways.

i, R_1 | sname | cname | R_2 | sname | instructor |

ii, R_1 | cname | instructor | R_2 | cname | sname |

iii, R_1 | instructor | cname | R_2 | instructor | sname |

But the third decomposition $D = \{R_1, R_2\}$ in BCNF and also satisfies non-additive join property means if join is performed on R_1, R_2 based on instructor name, spurious tuples will not be generator.

First and second decompositions are also in BCNF, they will generate spurious tuples on join condition.

Algorithm: To decompose a non-BCNF relation into decomposition containing a set of relations where R_i in 'D' is in BCNF and always that decomposition satisfies non-additive join property.

input R

output $D = \{R_1, R_2, R_n\}$

Method

$D = \{R\}$

repeat

choose a relation R

is not in BCNF

Identify a non-trivial FD $X \to Y$ in which X is not super key, then add sub-relations into D as $X \cup Y$ in one subrelation & R - (Y) into another sub relation until each R_i in D is in BCNF

Fourth Normal form: It is based on multi valued dependency (mvd)

MVD: X and Y are two sets subsets of attributes in R mvd X (multidetermines) $\twoheadrightarrow$ Y holds a valid in a relation state r of R, if there exists two tuples t_1 & t_2 with $t_1[x] = t_2[x]$ then there exists another two tuples t_3 and t_4 such that

i, $t_3[x] = t_4[x] = t_2[x] = t_2[x]$

ii, $t_3[Y] = t_1[Y]$ & $t_4[Y] = t_2[Y]$

iii, $t_3[Z] = t_2[Z]$ & $t_4[Z] = t_1[Z]$

where $Z = R - (X \cup Y)$

Trivial mvd if $X \cup Y = R$

here the combination of all attributes is a super key.

Inference Rule:

IR1 : Complimentation Rule for MVD's:

If $X \twoheadrightarrow Y \models X \twoheadrightarrow R - (X \cup Y)$

IR2 : Augumentation Rule for MVD's:

If $X \twoheadrightarrow Y$ and $W \supseteq Z$ then $WX \twoheadrightarrow YZ$

IR3 : Transitive Rule for MVD's:

If $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\}$ then $\models X \twoheadrightarrow Z - Y$

IR4 : Replication Rule for FD's & MVD's:

If $X \rightarrow Y$ is a valid FD then $X \twoheadrightarrow Y$ & Niceversa.

IR5 : Coalescence rule for FD's & MVD's:

If $X \twoheadrightarrow Y$ and there exists

i, $W \cap Y \neq \phi$

ii, $W \rightarrow Z$

iii, $Y \supseteq Z$ then $X \rightarrow Z$ is valid FD.

→ A relation R is said to be in forth normal form based on a set of FD's and MVD's for every non-trivial MVD $X \twoheadrightarrow Y$

emp :-

ename	pname	dname (dependents)
smith	product x	anna
smith	product y	lena
smith	product x	lena
smith	product y	anna

F = { ename →→ pname,
ename →→ dname }

Relation Emp is in BCNF not in 4NF because all attributes are key attributes in Relation R. The following procedure is adopted to convert non-4NF relation R into sub relations R_1, R_2 to be included into decomposition D.

1. D = R
2. Choose any relation Q in D not in 4NF
3. While there is a relation Q not in 4NF

{
choose MVD $X \twoheadrightarrow Y$ in Q, violates 4NF rule
then
$R_1 = X \cup Y$, $R_2 = Q - Y$
add R_1, R_2 & D
}

The above relation is decomposed in 4NF relations

R_1 =

ename	pname
smith	product x
smith	product y

R_2 =

ename	dname
smith	anna
smith	lena

F_1 = { ename →→ pname } F_2 = { ename →→ dname }

So F_1 & F_2 are in 4NF

(ename ∪ pname) = R - so trivial no need to check for super key

Scanned with CamScanner

Fifth Normal Form:

→ Based on concept of Join Dependency (JD)

→ JD of relation state r of R has to satisfy non-additive join property

for JD $(R_1, R_2 \ldots R_n)$ such that

$$* \text{ (natural join) } (\Pi_{R_1}(r), \Pi_{R_2}(r) \ldots \Pi_{R_n}(r)) = r(R)$$

→ JD $(R_1, R_2 \ldots R_n)$ is trivial if one of R_i is R otherwise non-trivial.

→ A relation R is said to be in 5NF based on FD's, MVD's and JD's for each non-trivial JD of $(R_1, R_2 \ldots R_n)$ each R_i must be super key.

Supply

sname	partname	projectname
smith	bolts	product x
smith	nuts	product y
scott	bolts	product x
scott	bolts	product y

The supply relation has no FD's and no MVD's if it has all key attributes hence we say it is in BCNF not in 5NF. The above supply relation can be decomposed into subrelations:

$$D = \{R_1, R_2, R_3\}$$

with a / constraint each suppliers suppliers parts, each supplier s will supply to project, each project will have atleast one part.

R_1 =

sname	partname
smith	bolts
smith	nuts
scott	bolts

R_2 =

sname	projname
smith	product x
smith	product y
scott	product x
scott	product y

R_3 =

partname	projectname
bolts	product x
nuts	product y
bolts	product x

Each relation R_i in decomposition is in 5NF because if natural join is performed JD $(R_1, R_2, R_3) - r(R)$

Properties of Relational Decomposition:

Universal Relation $R(A_1, A_2 \dots A_n)$ ← holds — $r(R)$

↓

FD set f → has → F^+ (F closure)

Decomposition $D = \{R_1, R_2, R_3 \dots R_n\}$ has to satisfy

(1) Attribute preservation property - $\bigcup_{i=1}^{n} R_i = R$

(2) FD preservation property

$$(\Pi_{R_1}(F), \Pi_{R_2}(F) \dots \Pi_{R_n}(F))^+ = F^+$$

(3) Non-additive join property or loss-less join property

* $(\Pi_{R_1}(r), \Pi_{R_2}(r), \dots \Pi_{R_n}(r)) = r(R)$

Scanned with CamScanner

Algorithm to test Non-additive Join Property:

Input: Universal Relation R, decomposition

$D = \{R_1, R_2, \ldots, R_n\}$

Output: To test the D satisfies non-additive join property or not

Method: ① Create initial matrix S with a seperate row 'i' for each R_i in D and a seperate column 'j' for each attribute A_j in R.

② Initially set $S_{ij} = b_{ij}$.

③ for each row i representing R_i, for each attribute A_j in R { if R_i contains the attributes then set $S_{ij} = a_j$ for all the attributes in R_i in comparision with A_j of R. }

④ Repeat the following until no changes are made with a matrix S.

for each FD $X \rightarrow Y$ in F of R

{

if a row 'i' contains X as LHS attribute and Y with A_j in all the rows set with $S_{ij} = a_j$ for Y

}

⑤ If any row is made with completely a_j's symbols then D satisfies non-additive join property means if join is done on relations in D will not

Scanned with CamScanner

generate spurious tuples.

⑥ If none row is completely made with symbols then D do not satisfies loss-less join pro

Example:

$R = \{ssn, ename, pno, pname, ploc, hours\}$

$F = \{ ssn \rightarrow ename,$

$pno \rightarrow pname, ploc$

$(ssn, pno) \rightarrow hours \}$

$D = \{R_1, R_2, R_3\}$

$R_1 = \{ssn, ename\}$

$R_2 = \{pno, pname, ploc\}$

$R_3 = \{ssn, pno, hours\}$

$S =$

	ssn	ename	pno	pname	ploc	hours
R_1	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}
R_3	b_{31}	b_{32}	b_{33}	b_{34}	b_{35}	b_{36}

$S =$

a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
a_1	b_{32} a_2	a_3	b_{34} a_4	b_{35} a_5	a_6

Row 3 in S is completely with symbols 'a'. So D satisfies loss-less join property.

→ If D contains only 2 relations $D = \{R_1, R_2\}$:

R has F and R_1 has f_1 & R_2 has f_2 if

i, $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ or

ii, $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ if in F^+

then D satisfies NJP

→ If R is decomposed into $D = \{R_1, R_2, -- R_i -- R_n\}$ and R_i is decompored into $D_1 = \{Q_1, Q_2, ---Q_k\}$ if D satisfies NJP and D_1 satisfies NJP then $D_2 = \{R_1, R_2, -- R_{i-1}, Q_1, Q_2 --- Q_k, R_{i+1}, -- R_n\}$ will also satisfy LJP or NJP.

Relational DataBase Design Algorithms:

1. Functional Dependency Preserving 3NF decomposition
2. FDP & NJP 3NF decomposition
3. NJP BCNF Decomposition (next BCNF normal form)

(1.) I/p - Relation R, FD f

output - $D = \{R_1, R_2 -- R_n\}$ each R_i in 3NF and D will satisfy FDP

methode:

① Find minimal set G for f

② If $X \rightarrow A_1$, $X \rightarrow A_2$ --- $X \rightarrow A_n$ in G then create R_i in D with $XUA_1UA_2 --- UA_n\} = a$

③ If any attribute Aj in R is not available in any Ri then include into relevant sub relation.

Ex: R = { ssn, ename, salary, deptno, pno, pname, ploc }

F = { ssn → ename, salary, deptno,
pno → pname, ploc
{ssn, pno} → {salary, ename, pname} }

G = { ssn → ename,
ssn → salary,
ssn → deptno,
pno → pname,
pno → ploc,
redundant { (ssn, pno) → salary,
(ssn, pno) → ename
(ssn, pno) → pname }

if we remove pno
ssn → sala redundant
if we remove pno
ssn → crame redundant
if we remove ssn
pno → pname redundant

Minimal set G = { ssn → ename,
ssn → salary,
ssn → deptno,
pno → pname,
pno → ploc }

So R_1 = { ssn, ename, sal, deptno }

R_2 = { pno, pname, ploc }

satisfy 3NF.

D = { R_1, R_2 } each R_i is in 3NF and always

③ Here D satisfies FD preservation but not NJF so,

②→ Algorithm: Step ① & ② are same.

③ if none of Ri in D contains key attributes then apply key identification algorithm identify set of key attributes and form a relation

④ If any attribute Aj in R is not available in any Ri then include in relevant sub relation

→ Null Values And Dangling Tuples; Alternative Relational DB design with NF's :-

Null Values:

Emp

ssn	ename	sal	sex	dno
12345	xx	10000	m	1
12346	yy	20000	f	2
12347	zz	15000	f	null

dept

dno	dname	dmgrssn
1	R	12345
2	A	12346

if we perform join on two relations emp, dept during natural join then the tuple will be missed.

During aggregate functions also we get inappropriate answers.

Dangling Tuples:

emp1

ssn	superssn	bdate	ol	ssn	dno
12345	12345	01-dec-1980		12345	1
12346	12345	10-dec-1981		12346	2
12347	12346	10-dec-1982		12347	null

select * from emp, emp1 where emp.ssn/=dno emp1.ssn/dno. → kle get 3 rows

emp1

ssn	dno
12345	1
12346	2
12347	null

if we donot want 12347 since no dept then select * from emp', emp1 where emp.dno = emp1.dno → kle get 2 rows

So the tuple 12347 appears some times and not appears some times. This is called as Dangling Tuple:

Ch-2

Introduction to Query Processing & Optimization

Steps involved in processing high level language Query:

Query

↓

scanning, parsing, validate

↓ Intermediate form of query

Query optimization

↓ executive plan

Query code generator

↓ code to execute query

Run time DB processor

↓

Result

Scanning - Table names are availabe or not - If available specified attributes are available or not
key words are scanned

Parsing - Given query is checked if semantically correct or not (Syntax checker)

If the query is valid then it is sent to Query optimizer module then exact tree / plan structure is selected to execute in minimal time.

Scanned with CamScanner

→ Translating SQL Queries into Relational Algebra Expressions

SQL → select ssn, ename
from employee
where sal > (select max(sal)
from employee e
where e.dno = 5);

RA → ① $\mathcal{F}_{max(sal)}(\sigma_{dno=5}(employee)) \rightarrow T$

intermediate query form formed after scanning, passing & validation

② $\Pi_{ssn, name}(\sigma_{sal>T}(employee))$

Algorithm for External Sorting:

External sorting is a sort-merge approach used to sort various DB files available in a large disk or secondary memory. It has 2 phases

1. Sorting phase :-

$i \leftarrow 1$

$j \leftarrow b$ (size of DB file in blocks)

$k \leftarrow n_B$ (size of buffer in main memory in blocks)

$n_R \leftarrow m \leftarrow \lceil j/k \rceil$ ceil

```
while (i <= m)
do {
    read k blocks of data into buffer or less than k
    sort records (internal sort)
    i ← i+1
}
```

2. Merging Phase:

```
i ← 1
p ← log_{k-1} m
j ← m
while (i ≤ p)
do
  {
    n ← 1
    q ← ⌈j / k-1⌉
    while (n ≤ q)
    do
      {
        read k-1 subfiles, one block at a time
        merge and write as new file
        n ← n+1
      }
    j ← q
    i ← i+1
  }
```

- Algorithm for SELECT Operation:

Q_1 – $\sigma_{SSN=123456789}(emp)$

Q_2 – $\sigma_{dno=5 \text{ or } dno=1}(emp)$

Q_3 – $\sigma_{sal<15000 \text{ and } dno=5}(emp)$

Q_4 – $\sigma_{sal>30000}(emp)$

Various Algorithms for 'select' operation:

i. Based on simple condition

S1 – linear search

S2 – Binary search

S3a – using primary index method.

S3b – using hash key

S_4 – using primary index to retrieve multiple records

S_5 – using a clustered index to retreive multiple records.

S_6 – using a secondary tree index (B+ tree) (for equality comparision of tuples)

ii. Complex selection:

S_7 :- Conjuctive selection using an individual index

Scanned with CamScanner

S8: Conjuctive selection using composite index.

S9: Conjuctive selection for intersection of record pointers

Algorithms to implement Join operation:-

Relation R → n number of tuples

Relation S → m number of tuples

$R \bowtie_{A=B} S$ (RA notation)

A=B → natural or equi join

select * from R, S where R.A = S.B (SQL)

1. Nested loop join method:
2. Simple loop join method
3. Partition based hash join method — ① partition phase
   ② probing phase
4. Sort-merge join method.

sort the tuples in R & S on attributes A & B

set i ← 1 , j ← 1

while (i ≤ n) and (j ≤ m)

do {

if R[i].A > S[j].B then

{ output S[j] to T

j ← j+1

}

else if (R[i].A < S[j]) then

{ output R[i] to T

i ← i+1

}

T ← $R \bowtie_{A=B} S$

```
else
    set j ← j+1
{
    (* R[i].A = S[j].B)
    output the concatenated tuple <R[i], S[j]>
    to T
    set l ← j+1
    while l ≤ m and R[i].A = S[j].B
    do {
        output the combined tuple
        <R[i], S[j] to T
        l ← l+1
    }
    k ← i+1
    while k ≤ n and R[i].A = S[j].B
    do {
        output <R[i], S[j]> to T
        k ← k+1
    }
    i ← k, j ← l
}
}
```


Scanned with CamScanner

Algorithm for PROJECT Operation:

$T \leftarrow \pi_{ssn}(emp)$

$T_1 \leftarrow \pi_{salary}(emp)$

i.e $\boxed{T \leftarrow \pi_{A}(R)}$

Implement Aggregate functions:

```
select max(sal)
from employee
```

$T \leftarrow \mathcal{F}_{max, sal}(emp)$

Algorithm for set operations:

$T \leftarrow R \cup S$

$T \leftarrow R \cap S$

$T \leftarrow R - S$

Using Heuristics in Query optimization:

Query tree or Query graph

```
select ssn, fname, dno, dname
from emp, dept
where emp.dno = dept.dno and dno=5;
```


Scanned with CamScanner