# UNIT II note 1

## Relational Database design:-

→ For the Conceptual design - ER diagrams
↓
EER.

→ Conceptual Analysis - for requirement Analysis.

Relational MODEL - most popular.
↳ DBM s/w available
- IBM first first started
- System R.

## CODDS RULES:-

↳ Edgar Frank Codd

According to Codd - database must obey these rules in order to be regarded as a true relational database.

Rule 0 → Foundation rule
- it must be able to manage database
  3 - database - to store data
  - Management system (to manage data + software)
  - relational (tables - relations)

Row & col's have relations.
They must be related.

Rule 1:- Information Rule:-

Information (Data + Metadata)
(Book index)

The information must be represented in the form of a table
and exactly one way.

| Roll no | Name | Address | Cont no | Age ← metadata |
|---|---|---|---|---|
| | | | | ↙ actual Information |

Rule 2:- Guaranteed Access Rule.

each & every data in a relational data base is guaranteed to be logically accessible
- Combination of table name, Primary key value & column name.

Select name from table where
name = "abc"

Rule 3:- Systematic treatement of null value

The NULL values in a database must be given a systematic & uniform treatment. This is very important rule because a NULL can be interpreted as one of the following:
- data is missing
- data is not known or why we are allowing NULL.
- data is not applicable - is it necessary or not!

## 4) Active online Catalog:-

The information / description about table its entities, attributes must be stored somewhere known as data dictionary. which must be accessed by authorized users.

Users can use the same query language to access the catalog which they used to access the database itself.

## 5) Comprehensive Data Sub-language Rule

→ A database can only be accessed using language having linear syntax that supports data definition, data manipulation and transaction management operations.

- This language can be used directly or by means of some applications.
- If the database allows access to data without help of this language, then it is considered as a violation

Query language - Database design
- manipulation
- transaction

- SQL
- not access to database
- Data defin
- manipulation
- security
- integrity
- definition

— all using same language.

- one well defined language to provide all manners of access to data

6) View updating rule:-

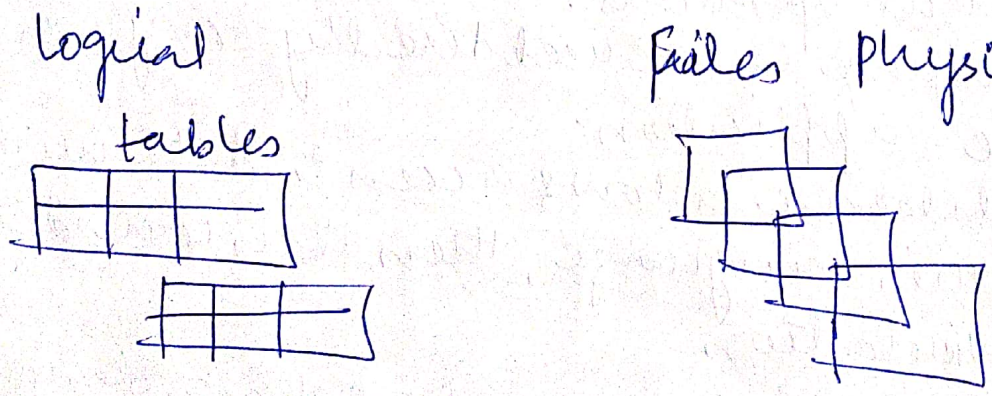
- All the views of the database, which can be theoretically be updated, must also be updatable by the system.

View - Virtual table.

7) High level insert, update, delete rule:-

A database must support high-level insertion, updation or deletion. This must not be limited to a single row, that is, it must support union, intersection or minus operation to yield set of data records.

8) Physical Data Independence.




not necessary no of tables = no of files

physical storage should not effect applications

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

any change in physical ≠ affect application

9) Logical Data Independence :- → how logically tables are related.

any change in table, no of table, should not effect the application.

The logical Data in a database must be independent of the user's view (application). Any change in logical data must not affect the applications using it.

→ For examples, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application.

→ This is one of the most difficult rule to apply.

10) Integrity constraints :→

A database must be independent of the applications that uses it. All its integrity

constraints can be independently modified without the need of any change in the application. This rule makes a database independent of the front end application & its interface.

11) Distributed Interface

Distribution of data should not effect user.

The end user must not able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database.

12) Non-subversion Rule:-

If the system has an interface that provides access to low-level records, then the interface must not be able to subvert the system & by pass security & integrity constraint.

# Functional Dependency:-

$f: \alpha \rightarrow \beta$.

$f(\alpha) \rightarrow \beta$ ✗ it is not like we will give alpha we compute beta.

| | $\alpha$ | $\beta$ |
|---|---|---|
| $t_1 \rightarrow$ | a | 1 |
| $t_2 \rightarrow$ | b | 2 |
| | c | 3 |
| | d | 4 |

it is a matter of search.

e.g $\alpha$ column has all different values a, b, c, d,

$a \rightarrow \beta$

1

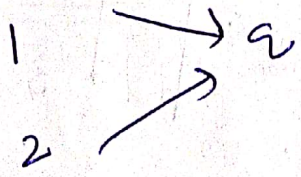
given value of a can → we find $\beta$. no here not the case.

| | |
|---|---|
| a | 1 |
| a | 2 |
| c | 3 |
| d | 4 |

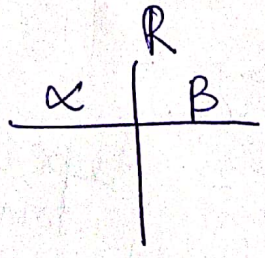
$1 \rightarrow a$

$1 \rightarrow b$ ✗

$2 \rightarrow a$




| | $\alpha$ | $\beta$ |
|---|---|---|
| $t_1 \rightarrow$ | a | 1 |
| $t_2 \rightarrow$ | b | 1 |
| | c | 3 |
| | d | 4 |




$\alpha \subseteq R$, $\beta \subseteq R$

f.D: $\alpha \rightarrow \beta$.

If $t_1[\alpha] = t_2[\alpha]$.

↳ $t_1[\beta] = t_2[\beta]$.

$\alpha \rightarrow \beta$

$\alpha$ ↙ determinant, $\beta$ ↘ dependent

Types

1) trivial FD

$AB \rightarrow A$

$\alpha \rightarrow \beta$

if $\beta \subseteq \alpha$.

2) non-trivial FD.

$\beta \not\subseteq \alpha$.

$AB \rightarrow AC$.

a) $A \rightarrow BC$ ✓

b) $DE \rightarrow C$ ✓

c) $C \rightarrow DE$ ✗

d) $BC \rightarrow A$ ✓

R

| A | B | C | D | E |
|---|---|---|---|---|
| a | 2 | 3 | 4 | 5 |
| 2 | a | 3 | 4 | 5 |
| a | 2 | 3 | 6 | 5 |
| a | 2 | 3 | 6 | 5 |

$\alpha \rightarrow \beta$.

a − 1

a − 2

$\alpha \rightarrow \beta$

$A \rightarrow C$

$BD \rightarrow C$

$ABDE \rightarrow C$.

If different value of $\alpha$ − dependency true

Same value of $\beta$ − dependency true.

Sid → Sname

if Sid is same means

Sname should be same.

Let R be the relational schema with X, Y as attribute sets.

$X \rightarrow Y$ exists in R only if $T_1, T_2$ tuples $\in R$ such that if $T_1.X = T_2.X$ then $T_1.Y = T_2.Y$.

If $X \rightarrow Y$ is always true, when X is a super key.

## Properties of Functional Dependency:-

1) **Reflexive FD:-** If $X \supseteq Y$ then $X \rightarrow Y$ is reflexive (trivial)

2) **Transitivity:-** If $X \rightarrow Y$ & $Y \rightarrow Z$ then $X \rightarrow Z$.

3) **Augmentation:-** If $X \rightarrow Y$ then $XZ \rightarrow YZ$

4) **Splitting rule:-** If $X \rightarrow YZ$ then $X \rightarrow Y$ & $X \rightarrow Z$

SUPER

## CANDIDATE KEY:-

→ Minimal set of attributes used to differentiate records of the table

→ If candidate key forms a single attribute, then candidate key is called simple ~~attribute~~ candidate key, otherwise compound candidate key.

→ attributes belonging to any candidate key are prime attributes of the relation.

## Primary Key:-

- one of the candidate key.
- primary key attribute set is not allowed to have NULL value.
- Atmost one primary key allowed.

## Alternate Key (Secondary Key):-

- All candidate keys except primary key.
- NULL values are accepted
- two records with same value of alternative key are not allowed.
- more than one alternate key are possible.
- There should be atleast one candidate key with NOT NULL.

## SUPER KEY:-

- set of attributes used to differentiate records (min. set not a constraint)

Super key attribute set = Candidate key attribute set + 0 or more other attributes.

→ Every Candidate Key is a super key, but every super key may not be the candidate key.

Q Let R(A1, A2, ... An) be a relation, how many super keys are possible.

(i) only one candidate key {A1}.

No of subsets possible from A2 to An = $2^{n-1}$

A1 can combine with all these subsets,

total super key = $2^{n-1}$

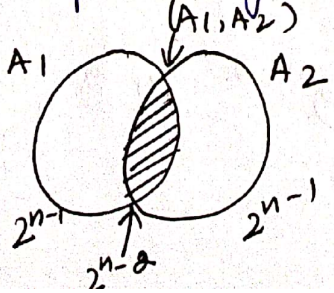
(ii) Candidate Keys {A1, A2}

A1 combined with all others → $2^{N-1}$

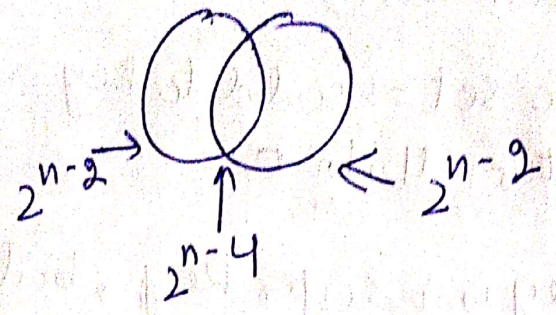
A2 " " " " → $2^{N-1}$

together combined with rest 0 times → $2^{N-2}$.

total super keys = $2^{N-1} + 2^{N-1} - 2^{N-2}$

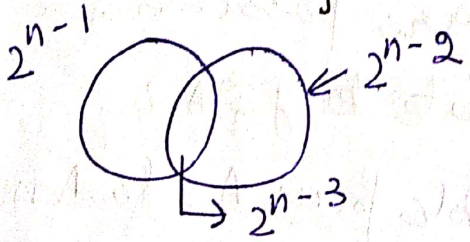


(iii) $\{(A_1, A_2), (A_3, A_4)\}$ → Candidate Keys.



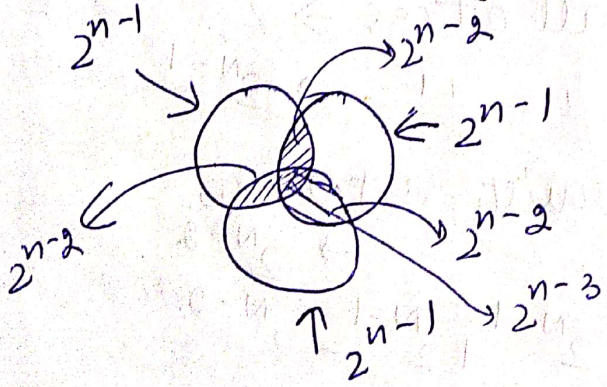

$= 2^{n-2} + 2^{n-2} - 2^{n-4}$

(iv) $\{A_1, (A_2, A_3)\}$




$= 2^{n-1} + 2^{n-2} - 2^{n-3}$

(v) $\{A_1, A_2, A_3\}$




$2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}$

(vi) $R(A_1, A_2 \cdots A_N)$

no. of super Keys formed.

R(ABC)

A
B
C — $2^n - 1$
AB — $2^3 - 1$ = 8
AC
BC
ABC

Note:- If every attribute of relation is candidate key, then max no. of superkeys are $2^n - 1$

NORMALIZATION:-

- eliminates / reduce redundancy in relations

Redundancy
- wastage of memory
- Two or more independent relations in same table, then redundancy is always possible.

- Normalization is the process which try to eliminate redundancy.
- Reduce

## Student

| Roll no | name | age | br-id | br-name | HOD-name | HOD-Phone |
|---|---|---|---|---|---|---|
| 1 | A | 20 | 101 | CS | XYZ | 123 |
| 2 | B | 19 | 101 | CS | XYZ | 123 |
| 3 | C | 21 | 101 | CS | XYZ | 123 |

- new information of student is entered we have to repeat – br id, br name, hod name, hod phone.

Problem 1 Insertion problem

new student — repeat branch, hod, hod phone.

Problem 2:- Deletion problem

delete all tuples → branch information also gets deleted.

Problem 3:- Modification problem.

like hod changes → it needs to be changed in every tuples.

Redundancy

Sometimes ~~Modification~~ leads to inconsistency.

e.g if we forget to change HOD name in some of columns - lead to inconsistency.

## What causes redundancy?

→ We tried to combine all data in a single table.

Paragraph - single idea
idea changes - paragraph changes.
every table must contain a single idea.

| Rollno | name | age | branch id |
|---|---|---|---|
| 1 | A | 20 | 101 |
| 2 | B | 19 | 101 |
| 3 | C | 21 | 101 |

| br-id | br-name | hod-name | p-no |
|---|---|---|---|
| 101 | CS | XY2 | 123 |

2 ideas in single table
for one idea - we have to repeat the other.

Normalization is a technique of organizing the data into multiple related tables to minimize data redundancy

## Data redundancy

repetition of data at multiple places.

- Repetition of data increases the size of database

| | | | |
|---|---|---|---|
| Row1 | — | — | X |
| Row2 | — | — | X |
| Row3 | — | — | X |

Issues

- insertion
- deletion
- updation

Normalization follows rule of divide & rule.

Logical, independent but related data

# NOTE 2

## Functional Dependency :-

Consider the relation schema R, & let $\alpha \subseteq R$ & $\beta \subseteq R$. The functional dependency $\alpha \rightarrow \beta$ holds on schema R. If, in any legal relation $r(R)$ for all pairs of tuples $t_1$ & $t_2$ in $r$ such that $t_1[\alpha] = t_2[\alpha]$, it is also the case that $t_1[\beta] = t_2[\beta]$.

$f: \alpha \rightarrow \beta$ is not like we are given $\alpha$ we can compute $\beta$. It is a matter of search.

Given $\alpha$ we can find $\beta$.

e.g

| | $\alpha$ | $\beta$ |
|---|---|---|
| $t_1$ | a | 1 |
| $t_2$ | b | 2 |
| | c | 3 |
| | d | 4 |

$t_1 \neq t_2$ functional dependency holds

$t_1 \neq t_2$ from $\alpha$ we can find $\beta$.

| | $\alpha$ | $\beta$ |
|---|---|---|
| $t_1$ | a | 1 |
| $t_2$ | b | 1 |
| | c | 3 |
| | d | 4 |

$t_1[\alpha] \neq t_2[\alpha]$

$\Downarrow$

$t_1[\beta] = t_2[\beta]$

functional dependency holds.

$\alpha \rightarrow \beta$

determinet dependent.

R

| | A | B | C | D |
|---|---|---|---|---|
| $t_1$ | $a_1$ | $b_1$ | $c_1$ | $d_1$ |
| $t_2$ | $a_1$ | $b_2$ | $c_1$ | $d_2$ |
| $t_1$ | $a_2$ | $b_2$ | $c_2$ | $d_2$ |
| $t_2$ | $a_2$ | $b_2$ | $c_2$ | $d_3$ |
| | $a_3$ | $b_3$ | $c_2$ | $d_4$ |

$A \rightarrow B$

$t_1[A] = t_2[A] \rightarrow t_1[B] \neq t_2[B]$

functional dependency doesnot hold.

| A | B | C | D | E |
|---|---|---|---|---|
| a | 2 | 3 | 4 | 5 |
| 2 | a | 3 | 4 | 5 |
| a | 2 | 3 | 6 | 5 |
| a | 2 | 3 | 6 | 5 |

$A \rightarrow BC$ ✓

$DE \rightarrow C$ ✓

$C \rightarrow DE$ ✗

$BC \rightarrow A$ ✓

## Types :→

1) Trivial - FD

$AB \rightarrow A$

$\alpha \rightarrow \beta$

if $\beta \subseteq \alpha$.

2) Non-trivial - FD

$\beta \not\subseteq \alpha$

e.g. $AB \rightarrow AC$.

## Attribute closure ($X^+$)

Set of attributes determined by X

Q. $R(A\ B\ C\ D)$

$\{A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D\}$

$(A)^+ = \{A, B, C, D\}$

$A \rightarrow ABCD$

$A \rightarrow A$ - trivial

$A \rightarrow A,\ A \rightarrow B,\ A \rightarrow C,\ A \rightarrow D$.

$(C)^+ = \{C, D\}$

$C \rightarrow C,\ C \rightarrow D$

Q. $R\{AB \rightarrow CD,\ AF \rightarrow D,\ DE \rightarrow F,\ C \rightarrow G,\ F \rightarrow E,\ G \rightarrow A\}$

a) $(CF)^+ = A\ C\ D\ E\ F\ G$

$(CF)^+ = (C, F, G, E, A, D)$ — True

(b) $(BG)^+ = \{B, G, A, C, D\}$

(c) $(AF)^+ = \{A, F, E, D,$
AC DEF G X

$(AB)^+ = \{A, B, C, D, G, \}$

## Super-Key :-

Let R be the relational schema & X be some set of attributes over R, if $X^+$ (closure of X) determines all attributes of R then X is said to be super key of 'R'

$(X)^+ = \{$ All attributes of R $\}$

↓
Super key.

1) R (ABC)

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

$(A)^+ = \{A, B, C\}$

↓
super key

$A \to A$, $A \to B$, $B \to C$

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

↓
super key.

$(ABC)^+ = \{A, B, C\}$

## Candidate Key (minimal Superkey)

If (X is superkey of R & no proper set of X is superkey)
then X is the Candidate key.

(AB): Superkey.

$A^+ = \{$ Not all attributes $\}$

$B^+ = \{$ Not all attributes $\}$

then AB: Candidate key.

If Superkey with one attribute is always a Candidate key.

Q R(ABCDE)

$\{AB \to C, B \to E, C \to D\}$

$(AB)^+ = \{A, B, C, E, D\}$

check whether AB is Candidate key

$A^+ = \{A\}$ → not super key

$B^+ = \{B, E\}$ → not super key.

proper subset of AB are not superkeys

∴ AB is Candidate key.

Take all attributes

$(ABCDE)^+ = \{A, B, C, D, E\}$

but $C \to D$

∴ D is not required in L.H.S.

$(ABCE)^+ = \{ABCE, DE\}$

but $B \to E$

∴ E is not required in L.H.S

$(ABC)^+ = \{A, B, C, D, E\}$

$AB \rightarrow C$

$\therefore$ C is not required in L.H.S

$(AB)^+ = \{A, B, C, D, E\}$

$\therefore$ AB is Candidate key.

Q. R (A B C D E)

$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E, E \rightarrow A\}$

$(AB)^+ = \{A, B, C, D, E\}$

$A^+ = \{A\}$

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

$\therefore$

proper subset of $(AB)^+$ is a super key

$\therefore$ it is not a candidate key.

B is candidate key.

Q. R (A B C D)

$\{AB \rightarrow CD, D \rightarrow A\}$

$(AB)^+ = \{A, B, C, D\}$

$A^+ = \{A\}$

$B^+ = \{B\}$

$\therefore$ AB is super key & also candidate key.

if we replace A by D.

$(DB)^+ = \{A, B, C, D\}$

$\therefore$ DB is also candidate key

Q R {A B C D}

{AB → CD, C → A, D → B}

$(AB)^+ = \{A, B, C, D\}$

replace A by C

$(CB)^+ = \{C, A, B, D\}$

replace B by D

$(AD)^+ = \{C, A, B, D\}$

$(CD)^+ = \{C, D, A, B\}$

Candidate Keys → AB, CB, AD, CD, )

Q R {A B, CD, E, F}

{AB → C, C → D, D → E, E → F, F → A}

$(AB)^+ = \{A, B, C, D, E, F\}$

F → A ∴ we can replace A by F.

$(FB)^+ = \{F, B, A\ C\ D\ E\}$

E → F ∴ we can replace F by E.

$(EB)^+ = \{E, F, A\ B, C, D\}$

D → E ∴ we can replace E by D.

$(DB)^+ = \{A, B, C, D, E, F\}$

C → D ∴ replace D by C.

$(CB)^+ = \{C, B, D, E, F, A\}$

$C^+ = \{C, D, E, F, A\}$

$B^+ = \{B\}$

Q. $R(A, B, C, D, E, F)$

$\{AB \to C, C \to D, D \to E, E \to BF, F \to A\}$

$(AB)^+ = \{A, B, C, D, E, F\}$

$A^+ = \{A\}$

$B^+ = \{B\}$

$E \to BF$

replace B by E.

$(AE)^+ = \{A, E, B, F, C, D\}$

$A^+ = \{A\}$

$E^+ = \{E, B, F, A, C, D, E\}$

∴ AE is super key but not candidate key.

$F \to A$ ∴ replace A by F in AB.

$(FB)^+ = \{F, B, A, C, D, E\}$ → super key & candidate key.

$D \to E$

∴ replace E by D.

$D^+ = \{A, B, C, D, E, F\}$

$C \rightarrow D$

$C^+ = \{A, B, C, D, E, F\}$ → Candidate key.

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

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

$AE^+ = \{A\ B\ C, DE\}$ AE

$DE^+ = \{A, B, C, D, E\}$

$CE^+ = \{A, B, C, D, E\}$

$BE^+ = \{A, B, C, D, E\}$

Q R(A B C D E H)

$\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$

Find its candidate keys.

## MEMBERSHIP TEST :-

To check whether $X \rightarrow Y$ F.D is member of F.D set F or not

$F = \{\} \models X \rightarrow Y$

$\{A \rightarrow B, B \rightarrow C\} \models A \rightarrow C$

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

$A \rightarrow C$

$A \rightarrow A$
$A \rightarrow B$
$A \rightarrow AB$
$A \rightarrow ABC$.

$\{A \to B, B \to C\} \models C \to B$.

$C^+ = \{C\}$

$\therefore C \to B$ is not a member.

$C \to B$ is not a members.

**check following membership valid or not :-**

1) $\{X \to Y, Y \supseteq Z\} \not\models X \to Z$

↓

Y is superset of Z

$Y \to Z$

$X^+ = \{X, Y, Z\}$ True.

2) $\{XY \to Z, Y \to W\} \not\models WX \to Z$

$(WX)^+ = \{W, X\}$ False

3) $\{XY \to Z, Z \to X\} \not\models Z \to Y$

$Z^+ = \{Z, X\}$ False

4) $\{X \to Y, Y \to Z\} \models X \to YZ$

$X^+ = \{X, Y, Z\}$ True

5) $\{XY \to Z, Z \to W\} \not\models X \to W$

$X^+ = \{X\}$

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

$A^+ = \{ABC\}$

## EQUALITY OF FD Sets :-

F , G

F equal to G only if

(i) F cover G

All G functional dependencies should be implied in F.

(ii) G covers F

All F FD's should be implied in G.

Q $F = \{A \to BC, B \to C, AC \to B\}$

$G = \{AB \to C, A \to B, A \to C\}$

(i) if G covers G. X

$A \to BC$ ✓

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

$B \to C$ X

G doesn't cover F

$AC \to B$ X

$(AC)^+ = \{A, C\}$

(ii) F covers G.

$AB \to C$ $\{A \to BC\}$

$A \to B$ $\{A \to BC\}$

$A \to C$ $\{A \to BC\}$

F covers G

∴ G is subset of F

$G \subset F$

Q $F_1 = \{A \to B, AB \to C, D \to AC, D \to E\}$

$F_2 = \{A \to BC, D \to AE\}$

(i) check if $F_1$ covers $F_2$

$A \to BC \quad \{A^+ = \{A\ B\}$ ✗ ✓

$D \to AE \quad \{D^+ = \{D\ A\ C\ E\}$ ✓

$F_1$ ~~does not~~ cover $F_2$

(ii) check if $F_2$ covers $F_1$

$A \to B = \{A \to BC\}$ ✓

~~$AC \to D \quad \{(AC)^+ = \{A\ C, D\}$ false~~ . X

$AB \to C = \{ABC\}$ ✓

$D \to AC \quad \{D\ A\ E\ BC\}$ ✓

$D \to E \quad \{D \to AE\}$ —

$F_1 \equiv F_2$

Q R(ABCD) {AB→CD, D→A, C→B}
What are the Candidate Keys of R1(BCD)

1) Find FD set of the sub relation R1.

for this take all subsets (Proper) of R1

$B^+ = \{B\}$ {B→B}

$C^+ = \{C, B\}$ (C→C, C→B)

$D^+ = \{D, A\}$ but A not in R1

∴ no need to add.

$(BC)^+ = \{B, C\}$ {BC→BC}

$(CD)^+ = (C\ C\ D, B)$

CD→CD CD→A, CD→B

$(BD)^+ = \{B, D, A, B\}$

~~CD→CD, CD→A CD→B~~

~~initial~~ no ~~no~~

BD→BD, BD→A, BD→C.

$(BD)^+, (CD)^+$ = Candidate Keys.

Q R(ABCDEF)

{AB → C, B → D, BC → A, D → EF}

Candidate keys of R1(ABCD)?

$A^+ = \{A\}$

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

$C^+ = \{C\}$

$D^+ = \{D, E, F\}$

$(BC)^+ = \{BCADEF\}$

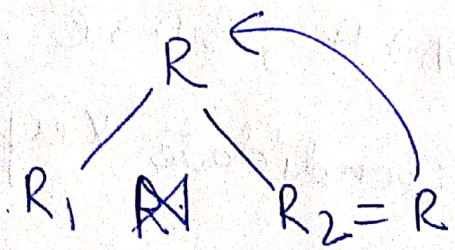
$(AB)^+ = \{ABCDEF\}$.

## Properties of Decomposition:-

1) Lossless join decomposition
2) dependency preserving decomposition.

## Lossless Join decomposition:-

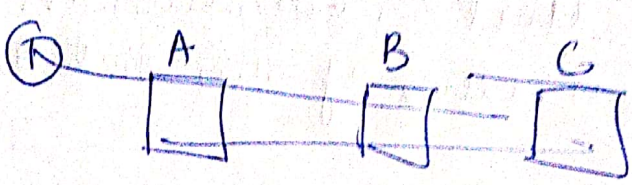
We have a relation R & we decompose this relation into $R_1$ & $R_2$




Let R be a relation Schema and F be a set of FD's over R & the decomposition of R into $R_1$ & $R_2$ is said to

lossless decomposition w.r.t F iff
$R_1 \bowtie R = R$.

| A | B | C |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_2$ | $b_1$ | $c_2$ |
| $a_1$ | $b_2$ | $c_2$ |




P

| A |
|---|
| $a_1$ |
| $a_2$ |

S

| B | C |
|---|---|
| $b_1$ | $c_1$ |
| $b_2$ | $c_2$ |

$2 = 2 \times 2 = 4$

1) A , BC ✗

2) ~~AB~~ , C 4 ✗ 2

3) ~~AC~~ , B 4 ✗

4) AB , BC lossless ] common attribute is the key attribute in atleast one of the table.

5) AC , BC lossless ]

6) AB , AC lossy.

decompose in such a way we should be able to get original relation. no extra or less tuples → then decomposition is lossless.

P ⋈ S $P(A) \cap B(BC) \neq \phi$.

no common tuples then cross product.

$P(A) \bowtie B(BC)$ = lossy (9)

The decomposition of R into two Relations $R_1$ & $R_2$ is said to be lossless if the attribute that is common in $R_1$ & $R_2$ is a key in either of the relations.

or

- Let R be a relation & F be a set of FD's on R. The decomposition of R into $R_1$ & $R_2$ is lossless if $F^+$ contains either the FD

$$R_1 \cap R_2 \rightarrow R_2 - R_1$$

$$R_1 \cap R_2 \rightarrow R_1 - R_2$$

Q R (A, B, C) with FD (A → B) is decomposed into $R_1$ (AB) & $R_2$ (BC) check whether the decomposition is lossy or lossless.

$$R_1 \cap R_2 \rightarrow R_1 - R_2 \Rightarrow \quad B \rightarrow A$$

or

$$R_1 \cap R_2 \rightarrow R_2 - R_1 \Rightarrow \quad B \rightarrow C$$

$$R_1 \cap R_2 = B$$

$$R_1 - R_2 \qquad AB - BC = A$$

$$BC - AB$$

- Hence decomposition is lossless
- Common attribute is not key attribute

$R(ABC)$ $F:(A \to B)$

$R_1(AB)$, $R_2(AC)$

$R_1 \cap R_2 \to R_1 - R_2$

$A \to B$ ✓ $\quad AB - AC$

or

$A \to C$ ✗ $\quad AC - AB$

True lossless

Note:- If $R_1 \bowtie R_2 = R$ {lossless

$R_1 \bowtie R_2 \supset R$ {lossy join}

$R \subset R_1 \bowtie R_2$ {not possible}

Q $R\{A, B, C, D\}$

$\{A \to B, B \to C, C \to D, D \to A\}$

lossy join

$D = \{(AB), (BC), (CD)\}$

↓ A ↓ B ↓ C

$AB \cap BC \to B$

$AB \cap BC \cap CD = \phi$ lossy.

2) ~~AB~~ ~~AB~~ ~~BC~~ A

$(AB \cap BC) \to$

↓

$B \to A$

or

$B \to C$ ✓

$BC \cap CD$

$C \to B$

or

$C \to D$

Q $R(ABC) = \{A \to B, A \to C\}$

$\{AB, BC\}$ ($R_1$, $R_2$)

$R_1 \cap R_2 = B$ is not a superkey of any so it is lossy join

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

or $B \to A$

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

$B \to C$

] lossy join

Q $R(ABCDE)$

$\{AB \to C, C \to D, B \to E\}$

1) $D = \{ABC, CD\}$ ($R_1$, $R_2$) lossy.

$R_1 \cap R_2 = C$ is a key of $R_2$.

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

$C \to AB$

or

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

$C \to D$ ✓

lossless join ∴ $R_1 \cup R_2 \neq R$ ↓ Missing

2) $D = \{ABC, CDE\}$

Ⓒ lossy.

3) $D = \{ABC, DE\}$ lossy.

4) $\{ABCD, BE\}$ lossless.

Q R(A B C D E G)

a) R{ AB → C , AC → B, AD → E , B → D , BC → A, E → G }

1) D = { AB , BC, ABDE, EG }

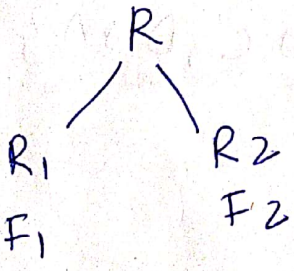
⇒ AB ∩ BC ∩ BDE ∩ EG = ϕ. Lossy.

2) D = { ABC , ACDE, ADG }

ABC ∩ ACDE ∩ ADG

A - Lossy join.

## 2) Dependency Preserving Decomposition:-




$F_1 \cup F_2 = F$ (dependency preserving)

$F_1 \cup F_2 \subset F$ (not dependency pres'ip

$F_1 \cup F_2 \supset F$ (not possible)

- The decomposition of a relation R with FD's 'F' into $R_1$ & $R_2$ with FD's $F_1$ & $F_2$ respectively is said to be dependency preserving.

iff $(F)^+ = (F_1 \cup F_2)^+$

Q R(ABC), F.D{ A→B, B→C, C→A }
is decomposition into
R1(AB) & R2(BC) is this decomposition
is dependency preserving or not.

R1 (AB) {A→B}
R2 (BC) {B→C}
{C→A} X.

{A→B B→C}

??

dependency preserving

Q R(ABCD) with FD {AB→C, D→A}

R1 (AD) R2 (BCD)
D→A BD→C

(AB)+ = ABC
(D)+ = (DA)

Not FD preserving

{F1 ∪ F2}+ = F+
↓↓
{ D→A, BD→C }+ = { AB→C, D→A }.

## Dependency preserving Decomposition :-

The decomposition of relation 'R' with FDs F into $R_1$ & $R_2$ with FD's $F_1$ & $F_2$ resp. is said to be dependency preserving if.

$$(F^+) = (F_1 \cup F_2)^+$$

Q1 :- $R(A, B, C)$ ; $F = \{A \to B, B \to C\}$

Decomposition of R: $R_1 = (A, C)$ ; $R_2 = (B, C)$.

FD's hold in $R_1$

$(A^+) = \{A\ BC\}$

$A \to C$

FD do hold in $R_2$

$(B)^+ = \{B\ C\}$

$B \to C$

$$\left\{ \begin{array}{l} F_1^+ = \{ A \to A, C \to C, A \to C, AC \to AC \} \\ F_2^+ = \{ B \to B, B \to C, C \to C, BC \to BC \} \end{array} \right\}$$

find set of non-trivial F.D's.

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

$\neq F^+$.

Not Functional dependency preserving

Q1:- $R(A,B,C) = \{A \to B, B \to C\}$

$R_1$ (A B)

$R_2$ (B C)

$F_1^+ = (A)^+ = \{A \to B, A \to B \& AB \to AB, B \to B\}$

$F_2^+ = (B)^+ = \{B \to B, B \to C$

$C \to C, BC \to BC$

$A \to B, B \to C$.

Dependency preserving

Q $R(C, S, Z)$

$F = \{CS \to Z, Z \to C\}$

$R_1(S, Z)$ $R_2(CZ)$

$F_1^+ = \{Z \to Z, S \to S, Z \to Z, SZ \to SZ\}$

$F_2^+ = \{C \to C, C \to Z, Z \to Z,$

$CZ \to CZ\}$

$\{Z \to C\}$.

Not dependency preserving.

R ( A, B, C, D )

{ AB → C , C → D , D → A }

$R_1$ { A, B, C } $R_2$ { C, D }

$F_1^+$ { A → A , AB → C , B → B }

$F_2^+$ { ~~D → A~~ , C → D }

not dependency preserving.

Q R ( A B C D )

{ A → B , B → C , C → D , D → B }

$R_1$ $R_2$ $R_3$

( A B ) , ( B, C ) , B , D )

↳ $A^+$

$B^+$ =

is dependency preserving

Q R ( A, B, C, D )

A → B B → C , C → D D → B.

( A B ) , B C )

Q R(ABC) F.D{A → B, B → C, C → A} is decomposed into $R_1(AB)$ & $R_2(BC)$ Is this decomposition D.P or not

$F_1^+ = \{ A \to A, A \to B, B \to C, AB \to AB$

$R_1(AB)$

$F_1 \begin{cases} A \to B \\ B \to A \end{cases}$

$R_2(BC)$

$F_2 \begin{cases} B \to C \\ C \to B \end{cases}$

$\{F_1 \cup F_2\}^+ = F$

$\{ A \to B, B \to A \quad B \to C, C \to B \}$ ~~B → C~~

C → A

$C^+ \{ C, B\ A$

$C \to A \}.$

# IRREDUCIBLE SET OF FD (canonical cover).

{AB → C, D → E, E → C} is the minimal cover of

{AB → C, D → E, AB → E, E → C}

First we check equality then minimality.

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

Q R(W X Y Z)

X → W

WZ → XY

Y → WXZ

Step 1:- Apply decomposition rule

X → W ✓

WZ → XX

WZ → Y ✓

Y → WX

Y → X ✓

Y → Z ✓

→

X → W

WZ → Y

Y → X

Y → Z

$X^+ = \{X\}$ ignore X → W

$X^+ = \{X, W\}$

$(WZ)^+ = \{W, Z\}$ ignore

$(WZ)^+ = \{W, Z, Y\}$ ignore

$(Y)^+ = \{Y$

X → W
WZ → Y
Y ⇒ XZ

irreducible set

Q R(A, B, C, D)

$A \to B$

$C \to B$

$D \to ABC$

$AC \to D$

Step 1 :- decomposition

imp

$A \to B$ ✓ $A^+ = \{AB\}$

$C \to B$ ✗ ✓ $A^+_B = \{A\}$

$C^+ = \{C\ B\}$

$C^+_B = \{C\}$

$D \to A$ ✓

$D \to B$ ✗

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

$D \to C$ ✓

$D^+ = \{D\ A$

$AC \to D$ ✓

$A \to B$

$D \to ABC$

$AC \to D$

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

$D^+ = \{D\ ABC$

$(AC)^+ = \{AC\ D.$

right side no redundancy

left side redundancy.

$(AC)^+ = \{A, C, D$ ~~$A, C$~~ $, B\}$

$A^+ = \{A, B$

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

$\{A \to B, C \to D, D \to AC, AC \to D.\}$

minimal cover

Q R(VWXYZ)

$V \to W$, $VW \to X$, $Y \to VXZ$

Step 1 :- Decomposition

$V \to W$ ✓ | $(V)^+ = \{V, W, X\}$

| $(V)^+_W = \{V\}$

$VW \to X$ ✓ | $(VW)^+ = \{V, W, X\}$

| $(VW)^+_X = \{V, W\}$

$Y \to V$ ✓ | $Y^+ = \{Y\ V\ X\ Z\ W\}$

| $Y^+_V = \{Y\ X\ Z\}$

~~$Y \to X$~~ ✗ | $Y^+_X = \{Y\ V\ W, X\}$

$Y \to Z$ ✓ | $Y^+_Z = \{Y\ V\ W, X\}$

$V \to W$, ~~$VW \to X$~~ $Y \to V$, $Y \to Z$

$(VW)^+ = \{V\ W\ X\}$

$V^+ = \{V\ W\ X\}$

$W^+ = \{W\}$

$\{V \to W, VW \to X, Y \to VZ\}$ min cover

Q $R = \{A\ B\ C\ D\}$

$\{A \to B, C \to B, D \to ABC, AC \to D\}$

Step 1 :- Decomposition

✓ $A \to B$ $A^+_B = \{A, B\}$

$A^+_B = \{A\}$

$C \to B$ ✓

$D \to A$ ✓

~~$C \to B$~~ ✗

$D \to C$ ✓

$AC \to D$ ✓

$A^+ = \{$

# CLOSURE SET OF ATTRIBUTES:-

↳ 2 methods.

R (A, B, C)

A → B , B → C

A → BC

F = $F_1$ + $F_2$

↑A → B → A → C

$F_1$ = direct visible

$F_2$ = not direct visible but it was not visible in a single look.

closure set of attribute is a set of all attributes that can be computed from that attribute.

For Normalization we need key and for key we need closure set of attributes

R (A, B, C)

A → B

B → C

$A^+$ = (A, B, C)

$B^+$ = {B, C}

$C^+$ = (C)

## Normalization :-

It is systematic way of creation of tables

watchman - area

| wid | wname | wife |
|---|---|---|
| NULL | NULL | NULL |

we have to recruit

| | | |
|---|---|---|
| 1 | Raja | Rani |
| 1 | Raja | Rani |
| 2 | Rakesh | Sita |
| 3 | Venu | Swati |

↳ training

| aid | aname | asize |
|---|---|---|
| 1 | GN | 1000 |
| 1 | GN | 1000 |
| 2 | XYZ | 2000 |
| 3 | ABC | 3000 |
| NULL | NULL | NULL. |

↓ GN 1000 ✗.

not allowed · forcing.

wid → NOT NULL

aid → NOT NULL

But we have a constraint that wid cannot be NULL so we are forcing some value this creates an insertion anomaly.

If some employee wants to leave then we have to delete the whole · that leads to loose information.

updation Anamoly

Raju — wife → Maharaji

Raju M. Rani.

Raju M. Rani.

update at multiple places.

# EQUIVALENCE OF FUNCTIONAL DEPENDENCY:-

R ( A C D E H )

F: $A \to C$
$AC \to D$
$E \to AD$
$E \to H$

G: $A \to CD$
$E \to AH$

$F \subseteq G$

$(A)^+ = \{ACD\}$
$(AC)^+ = \{ACD\}$
$(E)^+ = \{E, ADH\}$

$(A)^+ = \{A, C, D\}$
$(E)^+ = \{E\ ADH\}$

$G \subseteq F$

$F \subseteq G, G \subseteq F \to F = G$

## Irreducible Set of Fd (canonical cover)

→ R ( W X Y Z )
$X \to W$
$WZ \to XY$
$Y \to WXZ$

→ find redundant element whether its absence effects the functional dependency.

$\alpha \to \beta \gamma$
$\alpha \to \beta , \alpha \to \gamma$

Step 1 :- apply decomposition rule
decomposition only on R.H.S.

$X \to W$
$WZ \to XX$
$WZ \to Y$
$Y \to WX$
$Y \to X$
$Y \to Z$

$\alpha\beta \to \gamma$

$\alpha \to \gamma$
$\beta \to \gamma$

ignoring considering.

$X^+ = \{X, W\}$

$X^+ = \{X\}$ ——— without $X \to W$.

$(WZ)^+ = (WZXY)$

$(WZ)^+ = \{WZYX\}$ — without $WZ \to X$ ignore

$(WZ)^+ = (WZ)$ without $WZ \to Y$

$Y^+ = (YWXZ)$

$Y^+ = (YXZ)$ without $Y \to W$.

$Y^+ = YZ$

$Y^+ = YXW$

$X \to W$

$\boxed{WZ \to Y}$ check here redunda

$Y \to X$

$Y \to Z$

$(WZ)^+ = \{W, Z, Y \& X\}$

$W^+ = \{W\}$

$Z^+ = \{Z\}$

---

$X \to W$

$WZ \to Y$

$Y \to XZ$

A → B

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 1 | 2 | 3 |
| 2 | 3 | 4 |
| 2 | 4 | 4 |
| 3 | 5 | 7 |

A → B ✗

T → F = F

B → C ✓

AB → C

X → Y  X functionally determines Y

for every 2 tuples $t_1, t_2$

if $t_1[X] = t_2[X]$

then

$t_1[Y] = t_2[Y]$

| Sid | Sname | fname | marks |
|---|---|---|---|
| | | | |

Sid → Sname

Codd Boyce → proposed rules.
normalization is actually imposing some systematic rules.

1) 2 3 BCNF:

Functional Dependency :-

| eid | Cname | Parker |
|---|---|---|
| 1 | nano | C |
| 2 | Benz | A |
| 3 | BMW | B. |
| 4 | nano | C |

C → B mistake.

| | |
|---|---|
| A | AC |
| B | NON |
| C | Longsted |

Cname → Parking

| | |
|---|---|
| nano | C |
| nano | C |

$A \rightarrow B$

$t_1[A]$

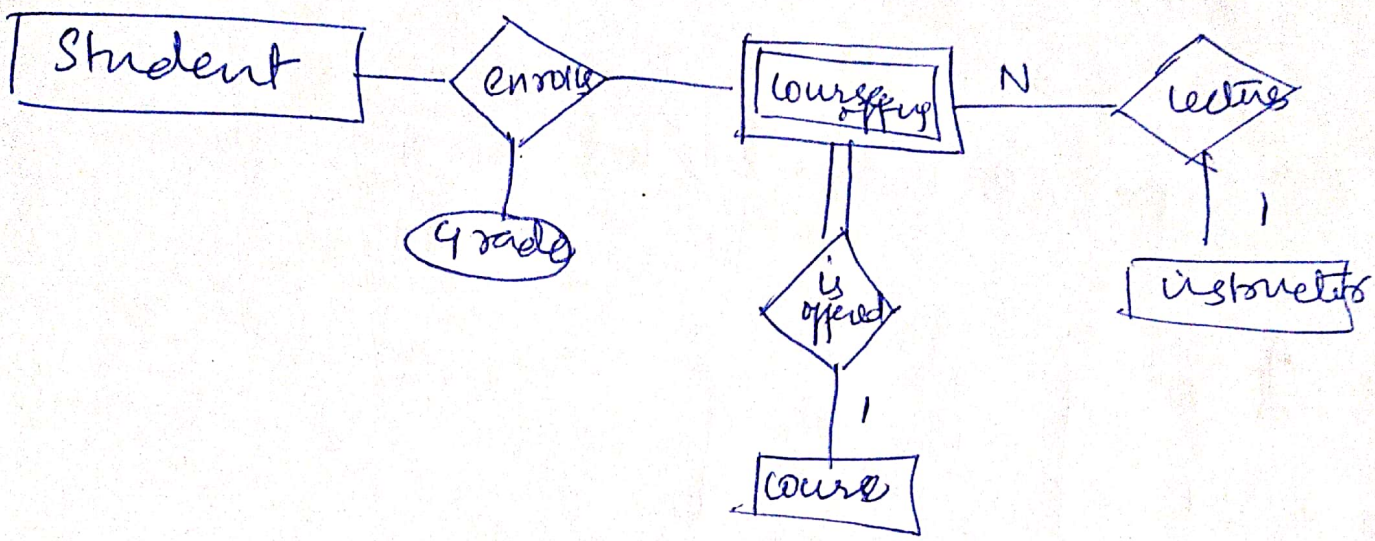
$t_2[A]$

If $t_1[A] = t_2[A]$ then

$t_1[B] = t_2[B]$.

(2)

Relational Model - Representation.




lectures 2) labs.

1:N ✓ dependent rel relationships only.

R(ABCDEF)

{ A → BCD, A → EF, BC → AD, BC → E, BC → F, B → F, D → E }

D = { ABCD, BF, DE }

| ABCD | BF |
| --- | --- |
| A → BCD | B → F |
| BC → AD | |