Lec (7) → Schema Refinement

(DM)

## Normalization

Logical DB Design is not very good → as → Conceptual modeling is subjective Process

↓ Thus

we need to refine it. or ensure it is good

for this → a formalism to do → Normalization comes

this

Relational DB Schema } → set of relations

↓

set of attributes

Now we group the attributes to relations is very important.

Normalization = schema refinement } help determine good relations.

## Purpose of Normalization

- > Avoid redundancy
- > relational principle conformed data
- > more able to change
- > avoid certain updating 'Anomalies'
- > intergrity constraints - facilitate to enforce

---

## Redundancy and Data Anomalies

(*) The redundant data could be removed without the loss of information

| NO | job | dep no dep | dep name dname | dep. city city |
|---|---|---|---|---|
| 10 | Sales | 10 | Sales | Kandy |
| 51 | Mng | 20 | Acc | Colombo |
| 40 | Clerk | 40 | Acc | Colombo |
| 45 | Clerk | 30 | operati | Colombo |
| 46 | Clean | 40 | Cleaning | Galle |

Department information (Dep table) ↓ and → These are repeating

anomalies (issues) → happen coz redundancy

Redundancy → (1) Disk space

ProMate

©Sathnindu Kottage - bysatha.com

# Anomalies

→ issues

**Insert Anomaly** } Department info

No (dep)
dep name
dep city } are ready

But we have to add those data with

↓

an employee.

> until we recruit an employee we can't add data of department → dep data depend on employee

| emp No | job | dep No | dep name | dep city |
|---|---|---|---|---|
| NULL | NULL | | | |

Cannot be inserted as emp No is the PK.

---

| | | | | |
|---|---|---|---|---|
| — | — | 20 | Acc. | Barha |
| — | — | 20 | Acc | Barha |
| — | — | 30 | Opr. | Barha |

The same department data (redundant)

**Update Anomaly** → { To update this we need to update all these records

**Deletion Anomaly** } Department information depends with employee

↓

WHAT IF?

we deleted the only emp. of a particular department.

[we loss the department]

(*) Like emp. no 10 was the only worker of girlas dep.

---

Summary

Insert : can't add dep without emp. (Need) (depend)

Update : have to update all (All)

Delete : dep will be deleted if delete the only/last worker. (Depend)

©Sathnindu Kottage - bysatha.com

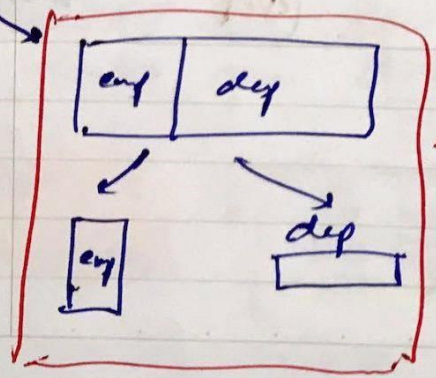
# Repeating Groups

× can't be like this since all attributes have to be atomic (no multivalued)

# Schema Refinement

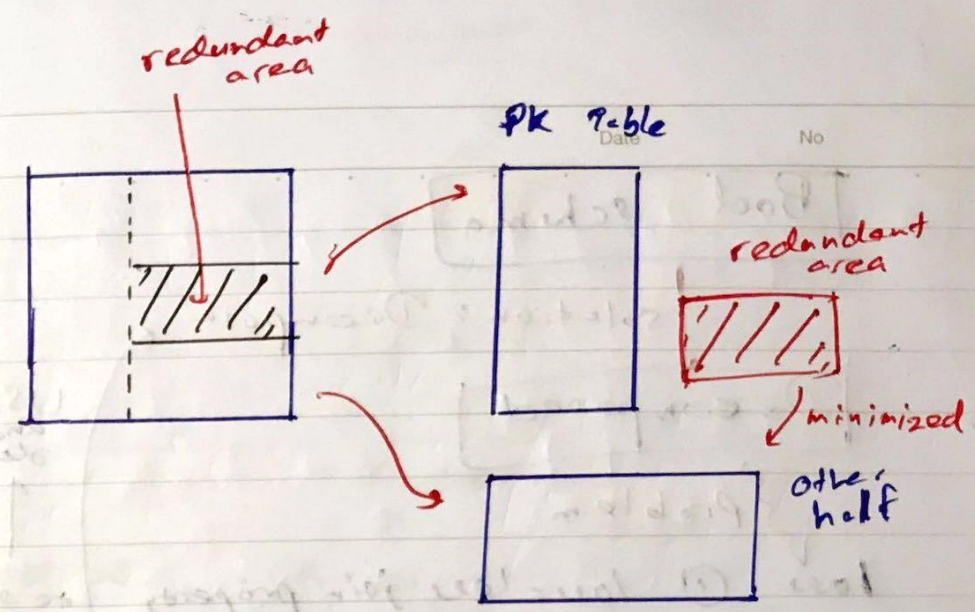
Redundancy → can be refined → by decomposing relations → into smaller relations.

> contain same information → With no redundancy




| emp | | |
|---|---|---|
| 10 | – | |
| 11 | – | |
| 40 | – | |
| 45 | – | |
| 46 | – | |

| dep | | |
|---|---|---|
| 10 | – | – |
| 20 | – | – |
| 30 | – | – |
| 40 | – | – |




Problems related to decomposition.

have to preserve.

① Loss-less join property and
② Dependancy preserving property

Normal forms

~~have been p~~ ~~Ask to~~

to preserve those properties

when we are decomposing

⟹ Data Loss-less (nothing lost) ①

⟹ Relationships didn't loss ②

With Normal form we can verify that we didn't loss any.

if we are using NF ② will not ~~loose~~ let lost

ProMate
©Sathnindu Kottage - bysatha.com
ProMate

Bad schema

solution: Decompose

Decomposed

Problem

loss ① loss-less join property
② Dependency preserving property

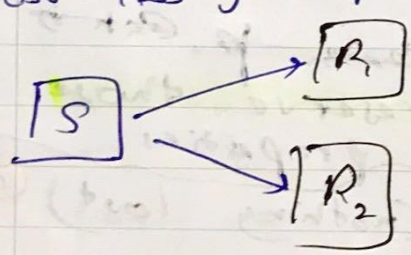
to slove

USE NF and both decomposition

(+) If you use NF to decompose
② Problems won't be happened.

---

Loss-less join property




Decomposed tables

Loss-less means:

if we combine $R_1$ and $R_2$ together → original (S) table should be reformed.

Decomposed

| S | P | D |
|---|---|---|
| $S_1$ | $P_1$ | $D_1$ |
| $S_2$ | $P_2$ | $D_2$ |
| $S_3$ | $P_1$ | $D_3$ |

S

| S | P |
|---|---|
| $S_1$ | $P_1$ |
| $S_2$ | $P_2$ |
| $S_3$ | $P_1$ |

$R_1$

| P | D |
|---|---|
| $P_1$ | $D_1$ |
| $P_2$ | $D_2$ |
| $P_1$ | $D_3$ |

$R_2$

$R_1 \bowtie R_2$ ($R_1$ join with $R_2$)

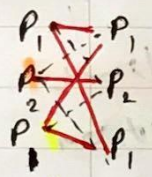
When we are joining consider the equality

common

S P P D

here it is (P)

$R_1 \bowtie R_2$




| S | P | ~~R~~ | D |
|---|---|---|---|
| ✓ $S_1$ | $P_1$ | ~~$R_1$~~ | $D_1$ |
| $S_3$ | $P_1$ | ~~R~~ | $D_3$ |
| ✓ $S_2$ | $P_2$ | ~~$R_2$~~ | $D_2$ |
| $S_3$ | $P_1$ | ~~R~~ | $D_1$ |
| ✓ $S_3$ | $P_1$ | ~~R~~ | $D_3$ |

spurious tuple

not there (original)

→ extra records

9 records

(4) ignored

9 combinations were there.

$(3\times3) = (3+3+3)$

∴ This decomposition is losy. (Not loss-less)

©Sathnindu Kottage - bysatha.com

## (*) Dependancy preserving Property

S (X, Y, Z)

Y functionally determines Z

X functionally determines Y

Decomposed

$S_1$ ( ) $S_2$ ( )

both functional dependancies should exist within the decomposed tables.

---

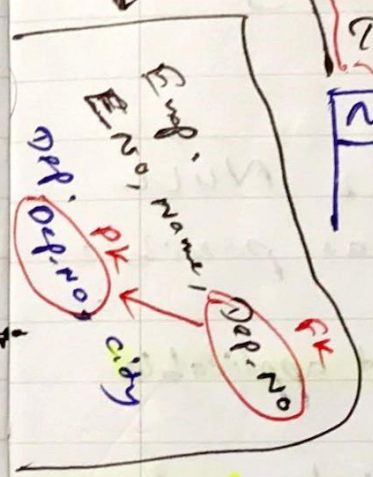
> We can use informal Guidelines, we can do the decomposition, but we can't garantee the (2) Properties.

## Informal Guidelines

### (*) Guideline (1):

Bro, If this table is about employees, wtf are you doing by adding fucking department information ~~into~~ in to that?

make it like this




The very first example. wtf?

| No | job | dep no | dep name | dep city |
|---|---|---|---|---|
| | | | | |

WTF?

if you wanne assign particular employee in to particular department

↓

Use ~~pri~~ foreign keys

and don't do this!

No need of the decomposition anymore than, } but this is informal.

©Sathnindu Kottage - bysatha.com

## * Guideline (2): No redundancy

make ~~a~~ a schema with no
- → insertion
- → update
- → deletion

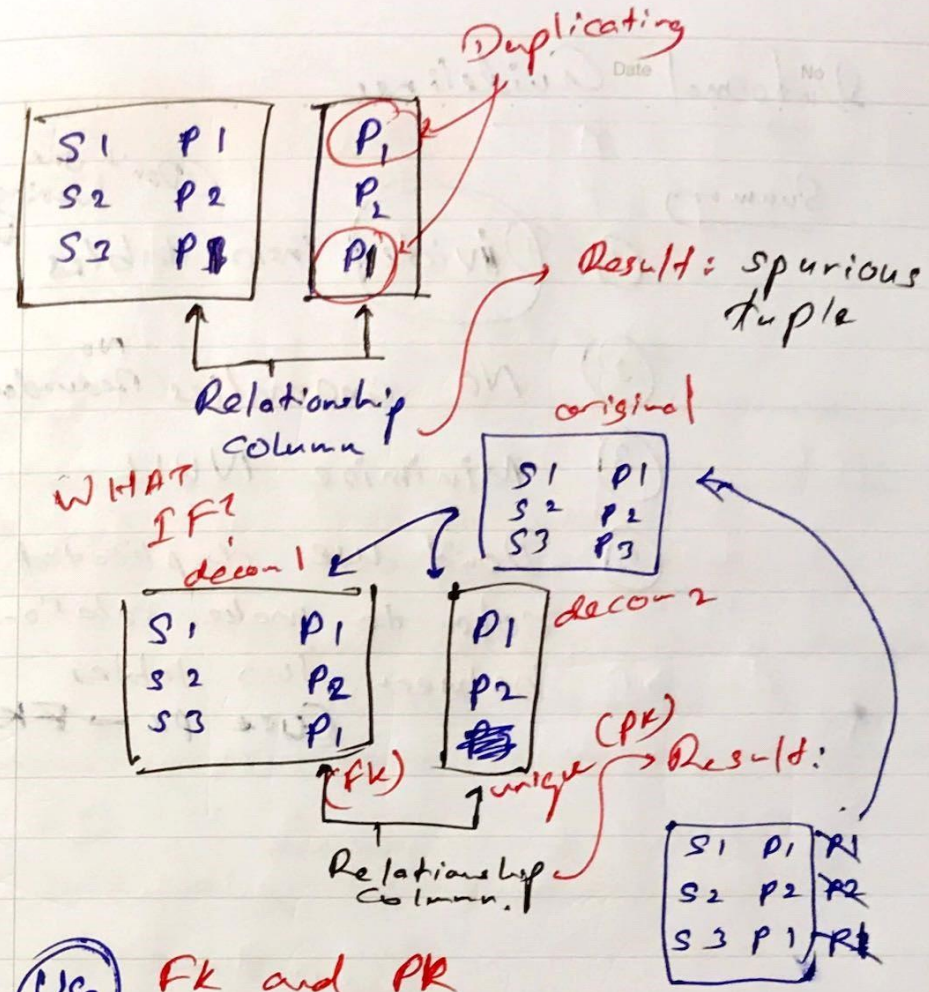
} anomalies

again we (Fk) and we other table

## * Guideline (3): minimize NULL values as possible.

Reasons for NULLS
1. Attribute not applicable or invalid
2. Attribute value unknown
3. Exist, not available

## * Guideline (4): If we used duplicating key ~~(NOT PK)~~ to make the relationship, spurious tuples will be generated when we combine.




Use Fk and PK or some unique feild of one table to resolve this problem

Informal

©Sathnindu Kottage - bysatha.com

# Informal Guidelines

Summary

1. Divide into tables (Don't one fucking table)
2. No anomalies (No Redundancy)
3. Minimize NULL
4. Don't use duplicated cols to make relationship between two tables (use PK → FK)

# Formal Process

→ is Normalization

and it based on functional dependencies (FD)

---

Functional Dependency

FD → used to measure Goodness of relationship

FD → used (also keys) to define (NF) for relation.

---

FD is a constraint between two sets of ~~two sets~~ attributes.

$X \rightarrow Y$

> X ~~do~~ functionally defines Y

> Y functionally dependent on X

when this happend, if

tuple → row

| | X | Y |
|---|---|---|
| $t_1$ | x | Y |
| $t_2$ | x | Y |

this is Y

also this should be Y.

if $t_1[X] = t_2[X]$ then → $t_1[Y] = t_2[Y]$

ProMate

©Sathnindu Kottage - bysatha.com

| | X | Y |
|---|---|---|
| $t_1$ | x | A |
| $t_2$ | x | Z |

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

$\downarrow$

$(A = Z)$

it means,

$t_1[Y] = t_2[Y]$

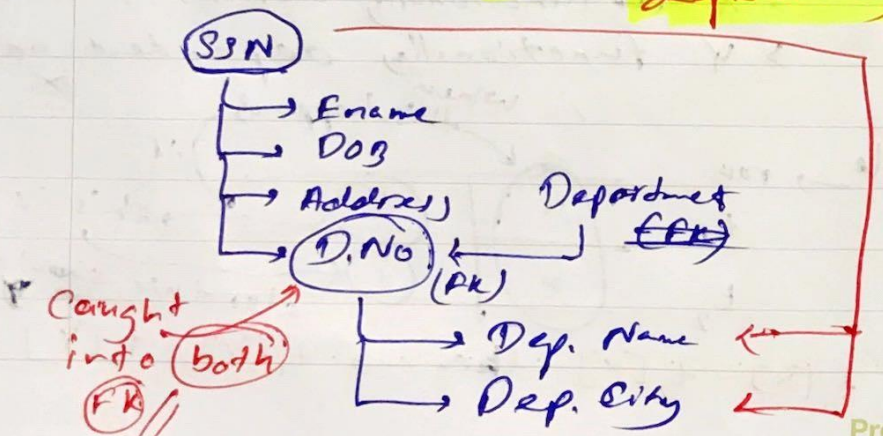
**FD** Describe the relationship between attributes in a relation.

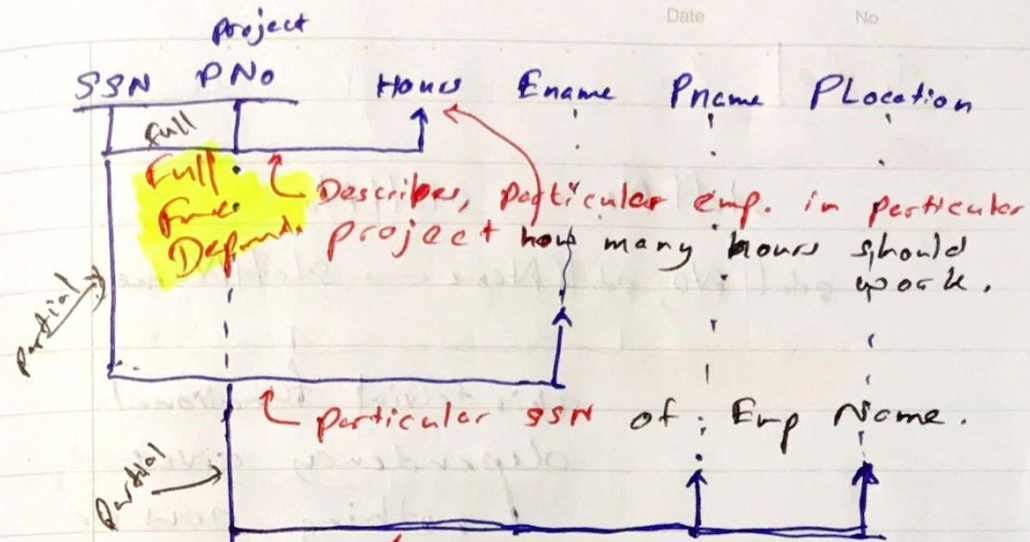
$A \rightarrow B$

↳ dependent

(*) A and B may each consists of one or more attributes.

Transitive functional dependency




if, ~~PNo = 12~~

| | PNo | PName | PLocation |
|---|---|---|---|
| $t_1$ | 12 | x | y |
| $t_2$ | 12 | x | y |

also, if PNo = 12 There should be x & y

$\because$ PNo → (Pname, Plocation)

$\therefore$ $t_1[PNo] = t_2[PNo]$

then $t_1[PName] = t_2[PName]$

and

$t_1[PLocation] = t_2[PLocation]$

should.

ProMate

©Sathbindu Kottage - bysatha.com

## Trivial

staffNo → staffNo

staffNo, staffName → staffName

This trivial functional dependency gives nothing new or insightful.

## Non-Trivial

staffNo → staffName

new info

This non-trivial functional dependency gives new.

## Functional dependancy

→ interface rules (IR)

**IR 1** Reflexive

if, Y subset of X then X → Y

trivial

~~IR 2~~ ~~Augmentation~~

~~if, X → Y then XZ → YZ~~

~~sid → name~~

~~then, using reflexive law~~

~~sid → sid~~

~~sid → name~~

~~sid → sid, name~~

mostly trivial

©Sathnindu Kottage - bysatha.com

## IR2 Augmentation rule

eid → name

↳ then,

eid, address → name, address

## IR3 Transive rule

eid → name
name → dept.

eid → dept

---

Additionally,

① (eid) → (name, dept.)

Decomposition

↳ eid → name.
eid → dept.

② ~~eid~~

eid → name  eid → dept.

Union

eid → name, dept

③ ~~eid → name~~  ~~name, dept →~~

Pseudotransitivity

③ eid → name
city, name → distance

city, eid → distance

---

©Sathnindu Kottage - bysatha.com

# Closure

- Student ~~ID~~ = S~~ID~~
  Course = C
  Instructor = I

---

(SID) → (S_name)

(CID) → (C_name)

(CID) → (IID)

(IID) → (I_name)

IR1 reflexive

> sid → sid (trivial)

> cid → cid (trivial)

> IId → IId (trivial)

IR2 Augmentation

IR3 Transitive

by applying them,

closure can be found.

here,

$\boxed{\text{sid} \rightarrow (\text{S_name}, \text{C_name}, \ldots\ldots)}$

closure.

all the attributed,

©Sathnindu Kottage - bysatha.com

unnormalized

# Normalization

step ①: check multi valued attribu and correct
(check if each and every is atomic)

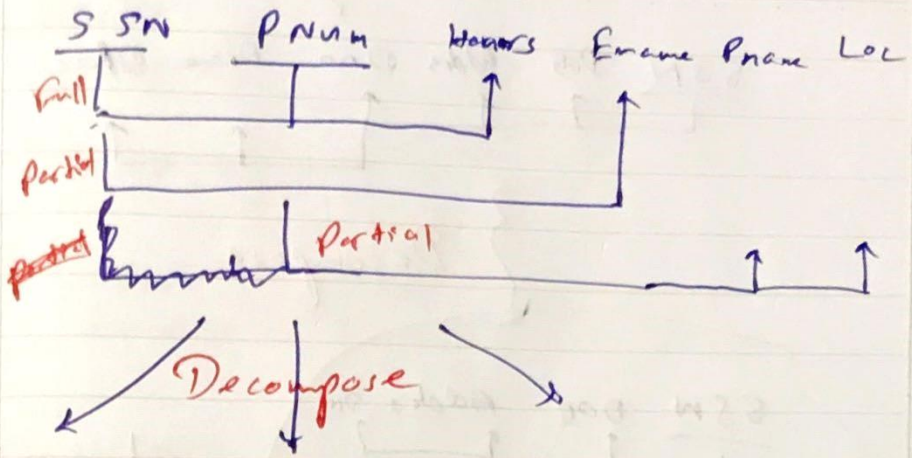
## 1st Normal form

step ②: Partial dependencies can't be hold.

full dependency, assume remove one part of key

now this is a partial dependency

this partial can't be there.




Decompose

① SSN PNum Hours

2nd normal form

② SSN Ename

③ PNum Pname Ploc

step ③: remove transitive dependency

©Sathnindu Kottage - bysatha.com

also non prime

SSN DOB Addrs Dno Dname Dloc

Decompose

SSN DOB Addrs Dno

DNo Dname Dloc

3rd normal form

Full dependancy

entire candidate key →

key attribute a functionality determined by a full key

Partial dependency

→

Subset of candidate key

Transitive

Prime → non-prime

non-prime → non-prime

But

non-prime → prime

cannot be referred to a prime.

to be BCNF

©Sathnindu Kottage - bysatha.com