CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")
#ProfGiri @ IIIT Hyderabad

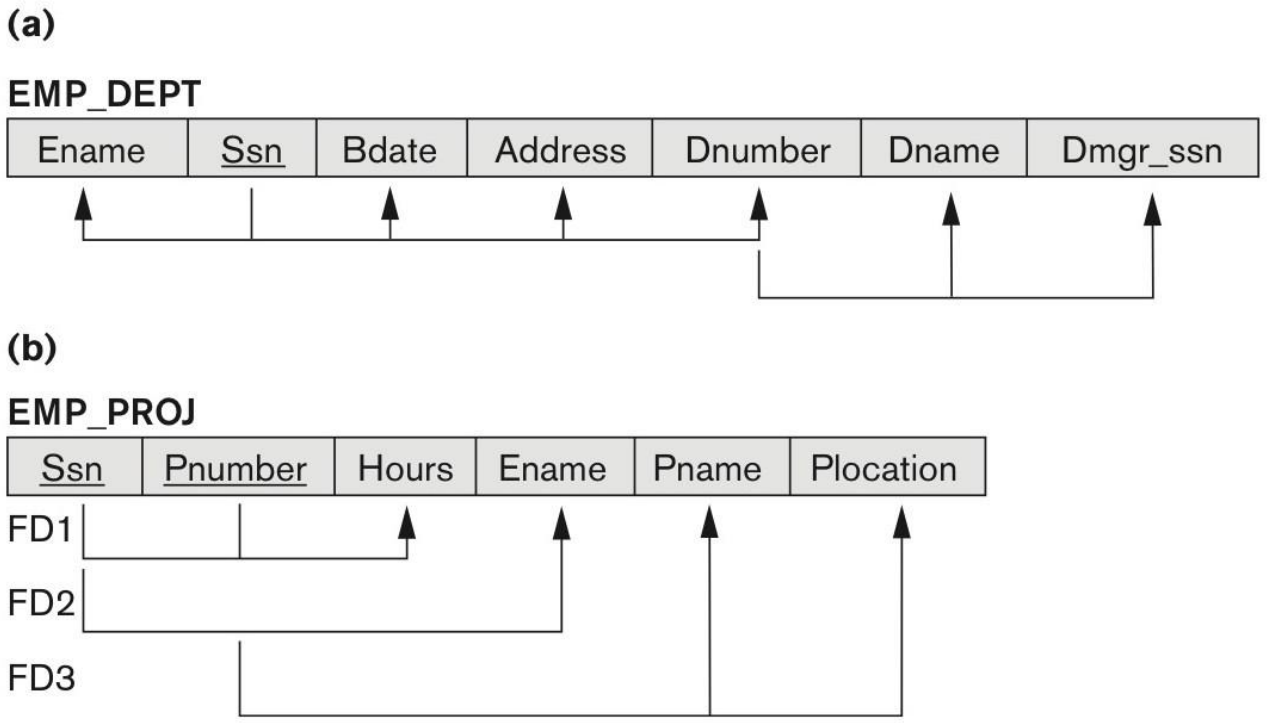
pk.profgiri /in/ponguru @ponguru Ponnurangam.kumaraguru

1.1 Semantics of the Relational Attributes must be clear

GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities

<u>Bottom Line:</u> *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*



Any concerns here?

EMP_DEPT: mixing attributes of employees & departments

EMP_PROJ: mixing attributes of employees, projects & works_on

1.2 Redundant Information in Tuples and Update Anomalies

- Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies

EXAMPLE OF AN INSERT ANOMALY

Consider the relation:

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

Insert Anomaly:

Cannot insert a project unless an employee is assigned to it

Conversely

Cannot insert an employee unless an he/she is assigned to a project

EXAMPLE OF A DELETE ANOMALY

Consider the relation:

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

Delete Anomaly:

When a project is deleted, it will result in deleting all the employees who work on that project.

Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

EXAMPLE OF AN UPDATE ANOMALY

Consider the relation:

EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)

Update Anomaly:

Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

Guideline for Redundant Information in Tuples and Update Anomalies

GUIDELINE 2:

- Design a schema that does not suffer from the insertion, deletion and update anomalies
- If there are any anomalies present, then note them so that applications can be made to take them into account

1.3 Null Values in Tuples

GUIDELINE 3:

- Relations should be designed such that their tuples will have as few NULL values as possible
- Attributes that are NULL frequently could be placed in separate relations (with the primary key)

Reasons for nulls; different meanings for null:

- Attribute not applicable or invalid [visa status to US students]
- Attribute value unknown [DOB of an employee]
- Value is known but absent; it has not been recorded yet [phone # of employee]

1.3 Null Values in Tuples

Poor design:

Employee(EmpID, Name, Department, Salary, Commission, Bonus)

Why? Better design?

Employee(EmpID, Name, Department, Salary)

SalesCommission(EmpID, Commission)

BonusPayment(EmpID, Bonus)

1.4 Generation of Spurious Tuples – avoid at any cost

Bad designs for a relational database may result in erroneous results for certain JOIN operations

GUIDELINE 4:

No spurious tuples should be generated by doing a natural-join of any relations.

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

*
*
*

Additional tuples that were not there in Emp_proj is here, they are called spurious tuples

2. Functional Dependencies

Functional dependencies (FDs)

- Are used to specify *formal measures* of the "goodness" of relational designs
- And keys are used to define **normal forms** for relations
- Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes

A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

2.1 Defining Functional Dependencies

X → Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y

For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]

X → Y in R specifies a *constraint* on all relation instances r(R)

Written as X → Y; can be displayed graphically on a relation schema as in Figures; denoted by the arrow →

FDs are derived from the real-world constraints on the attributes

Examples of FD constraints (1)

Social security number determines employee name

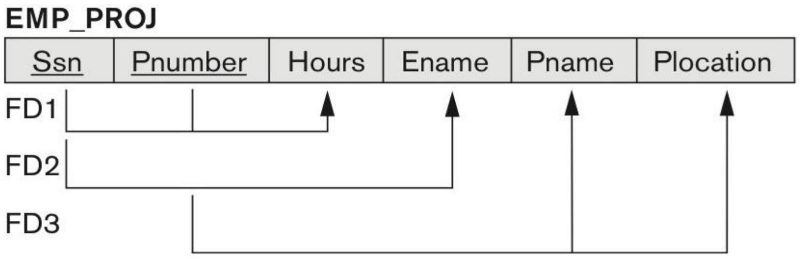
SSN → ENAME

Project number determines project name and location

PNUMBER → {PNAME, PLOCATION}

Employee ssn and project number determines the hours per week that the employee works on the project

{SSN, PNUMBER} → HOURS



Defining FDs from instances

Note that in order to define the FDs, we need to understand the meaning of the attributes involved and the relationship between them.

Given the instance (population) of a relation, all we can conclude is that an FD *may exist* between certain attributes.

What we can definitely conclude is – that certain FDs *do not exist* because there are tuples that show a violation of those dependencies.

What FDs may exist?

A relation *R*(A, B, C, D), which FDs *may exist* in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

B → C; C → B; {A,B} → C; {A,B} → D; {C,D}→ B

How about A → B? B→ A? D → C?

3.1 Normalization of Relations (1)

Normalization:

The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

Normal form:

Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Activity: 10^{th} Nov

Infinium, Develop 2 examples relation where where mixing of attributes exist?

Give an example of INSERT, DELETE, UPDATE anomaly in Infinium DB

Write at least 2 FDs that may exist in Infinium DB
Write at least 2 FDs that may not exist in Infinium DB

This Lecture

Definitions of Keys and Attributes Participating in Keys (2)

A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S

If a relation schema has more than one key, each is called a **candidate** key.

- One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.

A **Prime attribute** must be a member of *some* candidate key

A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

Prime attributes

Consider the relation:

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

and suppose the **candidate keys** are:

- `{A, B}`
- `{C, D}`

Then:

- **Prime attributes:** A, B, C, D (since all are part of some candidate key).
- **Non-prime attributes:** None (since every attribute participates in at least one key).

Prime attributes

Consider the relation:

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

and suppose the **candidate keys** are:

- `{A, B}`
- `{C, D}`

Then:

- **Prime attributes:** A, B, C, D (since all are part of some candidate ke
- **Non-prime attributes:** None (since every attribute participates in a

Another example:

`R(A, B, C, D)` with **candidate key** `{A, B}` only.

Then:

- **Prime attributes:** A, B
- **Non-prime attributes:** C, D

3.4 First Normal Form

Disallows

- composite attributes
- multivalued attributes
- **nested relations**; attributes whose values for an *individual tuple* are non-atomic

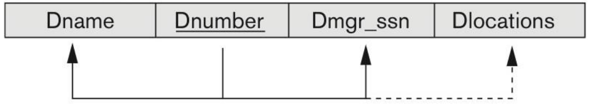
Considered to be part of the definition of a relation

Most RDBMSs allow only those relations to be defined that are in First Normal Form

Normalization into 1NF

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Figure 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT

Ways to make it make it 1NF?

1NF

2 = Redundancy, Dnumber & Dlocation primary key

DEPARTMENT F.K.

Dname	<u>Dnumber</u>	Dmgr_ssn

P.K.

DEPT_LOCATIONS

F.K.

<u>Dnumber</u>	<u>Dlocation</u>

P.K.

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

1 = Two 1NF relations

3 = If the maximum number of values (n) for location is known, replace it with n attributes
e.g. Only 3 locations for the company – Dlocation1, Dlocation2, Dlocation3
Introducing NULL if most departments have fewer than 3 locations
Hard to query, e.g. List the departments that have ‘Bellaire’ as one of the locations
1st option is commonly used one

Normalizing nested relations into 1NF

(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 14.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a nested relation attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Ssn is the primary key, Pnumber is the partial key

Remove the nested relation attributes into a new relation and propagate primary key

This idea can be applied recursively to a relation with multiple-level nesting to unnest

BLOB, CLOB – atomic, single-valued so 1NF

3.5 Second Normal Form (1)

Uses the concepts of **FDs, primary key**

Definitions

- **Prime attribute:** An attribute that is member of some candidate key K
- **Full functional dependency:** a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples:

- {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold
- {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> ENAME also holds

Fully functional dependency

If X and Y are an attribute set of a relation, Y is fully functional dependent on X, if Y is functionally dependent on X but not on any proper subset of X.

e.g. In the relation ABC->D, attribute D is fully functionally dependent on ABC and not on any proper subset of ABC. That means that subsets of ABC like AB, BC, A, B, etc cannot determine D.

supplier_id	item_id	price
1	1	540
2	1	545
1	2	200
2	2	201
1	1	540
2	2	201
3	1	542

{ supplier_id , item_id } -> price

https://www.geeksforgeeks.org/differentiate-between-partial-dependency-and-fully-functional-dependency/

Partial dependency

A functional dependency X->Y is a partial dependency if Y is functionally dependent on X and Y can be determined by any proper subset of X.

name	roll_no	course
Ravi	2	DBMS
Tim	3	OS
John	5	Java

{name} → course
{roll_no} → course

https://www.geeksforgeeks.org/differentiate-between-partial-dependency-and-fully-functional-dependency/

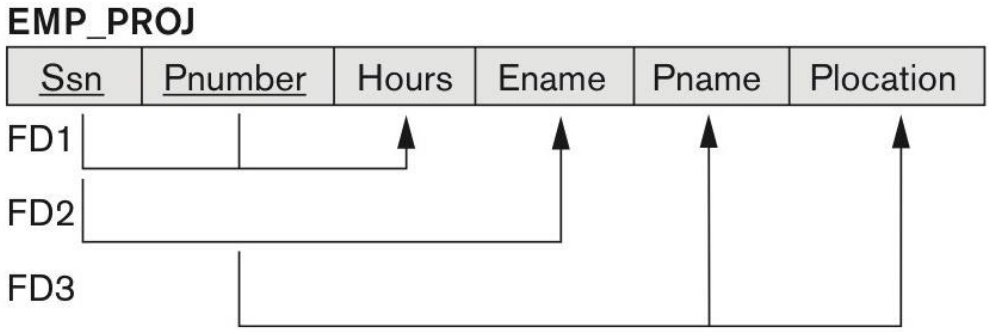
Second Normal Form (2)

A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key

> A **Prime attribute** must be a member of *some* candidate key
>
> A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

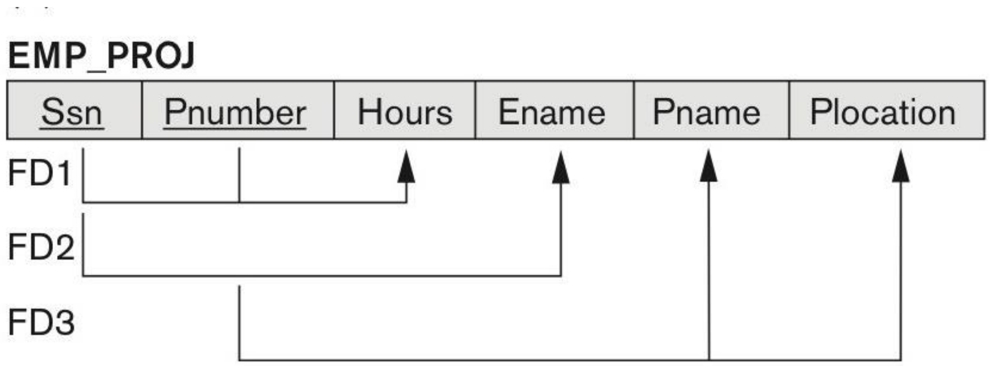
R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

2NF



Any violation?

2NF



Any violation?

FD2, FD3 violates 2NF

Partial dependency: {ssn} → {Ename}, {pnumber} → {pname, plocation} & {ssn, pnumber} are primary keys; not fully functionally dependent

Normalizing into 2NF

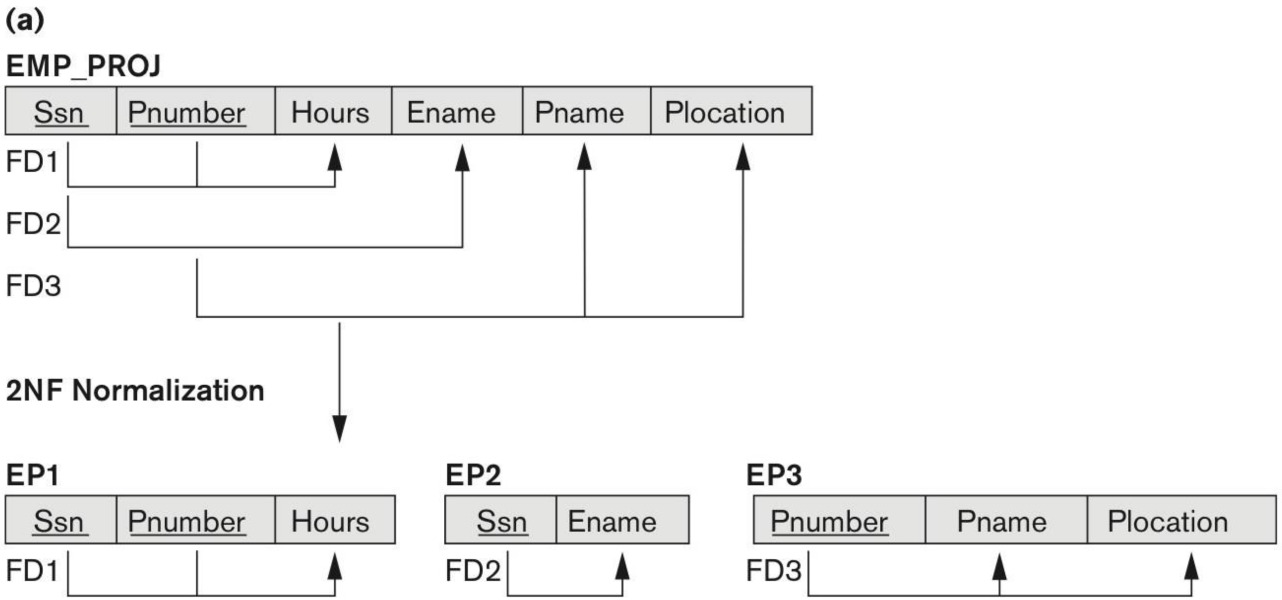


Figure 14.11
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations.

EP1, EP2, EP3 are fully functionally dependent

Summary

Property	Explanation
Partial Dependencies	Yes — `Ssn → Ename` and `Pnumber → Pname, Plocation`
Primary Key	(Ssn, Pnumber)
Why Not 2NF	Non-prime attributes depend on *part* of the composite key
Fix	Decompose into EMPLOYEE, PROJECT, and EMP_PROJ

Third normal form

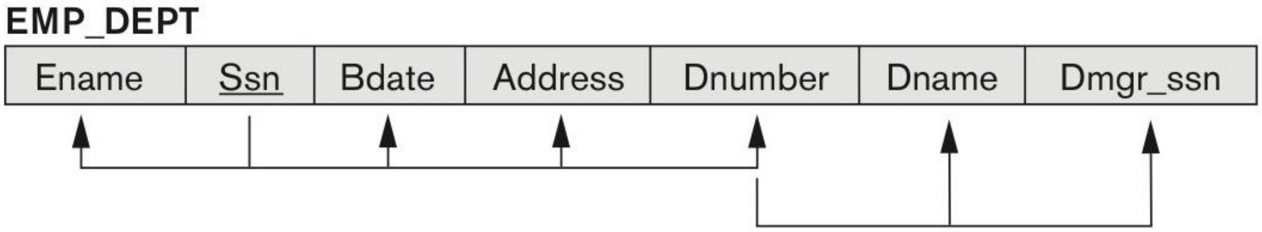
Transitive Dependency

$X \rightarrow Y$ in R is transitive dependency, if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R and both $X \rightarrow Z$ & $Z \rightarrow Y$ hold.

Third normal form

Transitive Dependency

X → Y in R is transitive dependency, if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R and both X → Z & Z → Y hold.

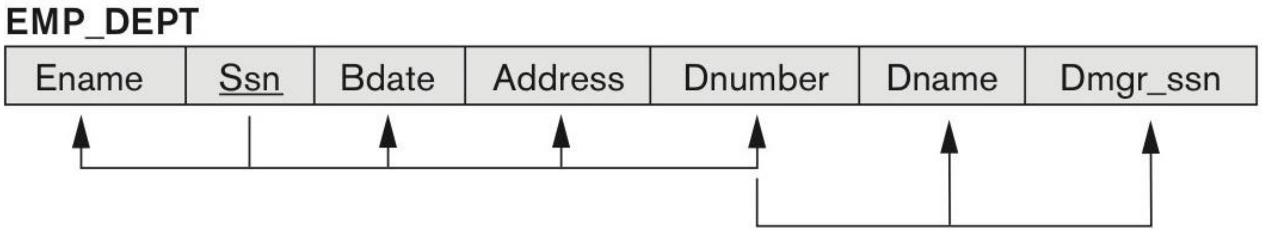


Any transitivity?

Third normal form

Transitive Dependency

X → Y in R is transitive dependency, if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R and both X → Z & Z → Y hold.



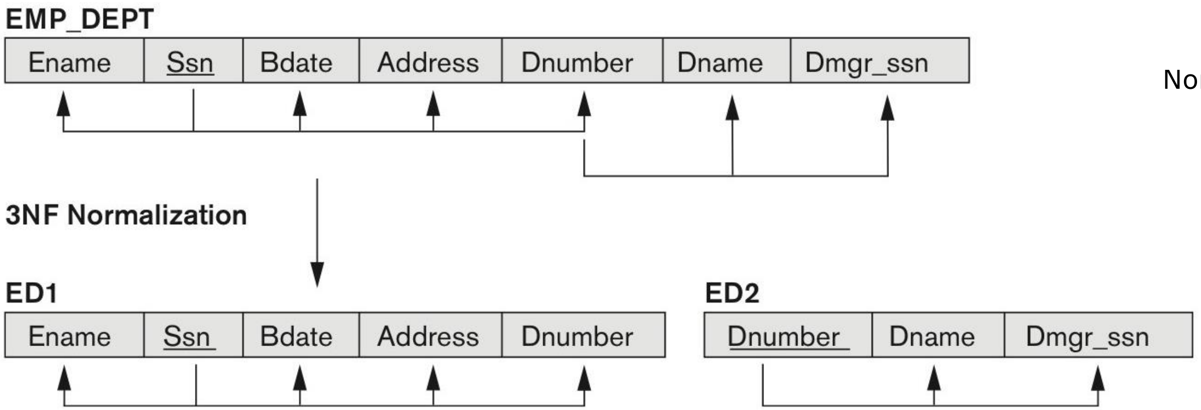
Any transitivity?

Ssn → dmgr_ssn is transitive through dnumber

Both ssn → dnumber & dnumber → dmgr_ssn hold & dnumber is neither a key nor a subset of a key

Third normal form

R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key



Normalizing EMP_DEPT into 3NF relations.

ED1 & ED2 represent independent facts about employees & departments

Another example

```
STUDENT (RollNo, Sname, DeptNo, DeptName, DeptLocation)
```

Functional Dependencies:

1. RollNo → Sname, DeptNo
2. DeptNo → DeptName, DeptLocation

Step 1: Identify the key

- `RollNo` is the **primary key** (uniquely identifies a student).

Step 2: Find indirect dependencies

- `RollNo → DeptNo` (direct)
- `DeptNo → DeptName, DeptLocation`

➡ So, **RollNo → DeptName, DeptLocation (transitively)**

Another example

Why It's a Problem

Attributes like `DeptName` and `DeptLocation` depend **indirectly** on `RollNo` — through `DeptNo`.
That means the department info is **redundantly stored** for every student in that department.

If a department moves to a new location, we must update many rows — this leads to **update anomalies**.

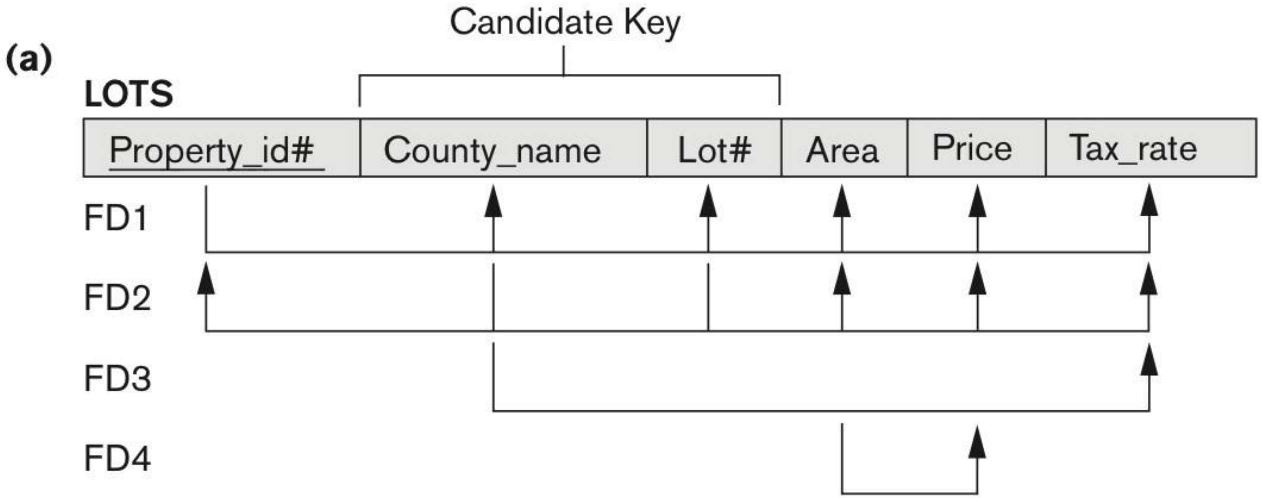
Solution

To eliminate transitive dependencies, we move to **Third Normal Form (3NF)**:

STUDENT (RollNo, Sname, DeptNo)
DEPARTMENT (DeptNo, DeptName, DeptLocation)

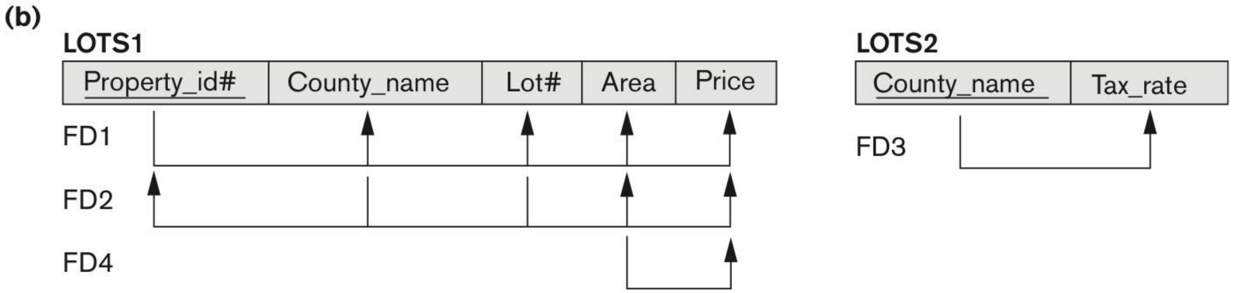
RollNo determines Sname, DeptNo — direct dependency.
DeptNo determines DeptName, DeptLocation — separate table.
No transitive dependencies.

Figure 14.12a Normalization into 2NF and 3NF. The LOTS relation with its functional dependencies FD1 through FD4.



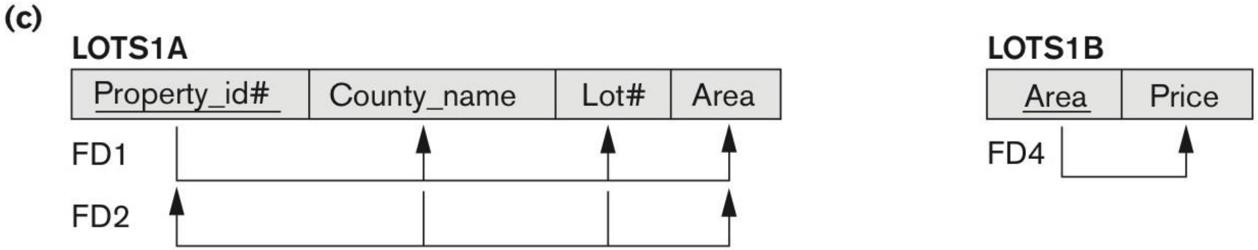
continued on next slide

Figure 14.12b Normalization into 2NF and 3NF. Decomposing into the 2NF relations LOTS1 and LOTS2.



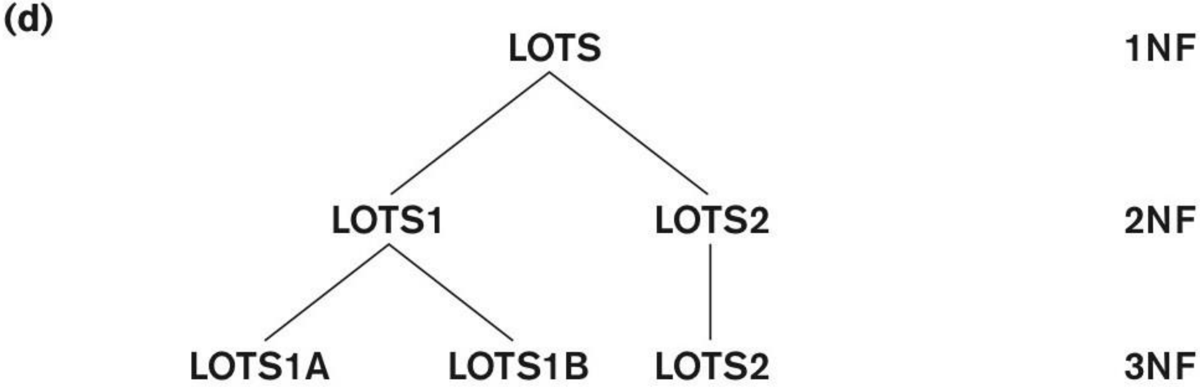
continued on next slide

Figure 14.12c Normalization into 2NF and 3NF. Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B.



continued on next slide

Figure 14.12d Normalization into 2NF and 3NF. Progressive normalization of LOTS into a 3NF design.



5. BCNF (Boyce-Codd Normal Form)

A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X → A** holds in R, then **X is a superkey** of R

Refresher: Roll number: Candidate key; Roll number + Name: Super key

Each normal form is strictly stronger than the previous one

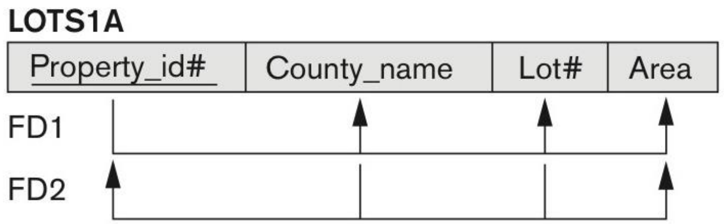
- Every 2NF relation is in 1NF
- Every 3NF relation is in 2NF
- Every BCNF relation is in 3NF

There exist relations that are in 3NF but not in BCNF

Hence BCNF is considered a stronger form of 3NF

The goal is to have each relation in BCNF (or 3NF)

Figure 14.13 Boyce-Codd normal form



Is this in BCNF?

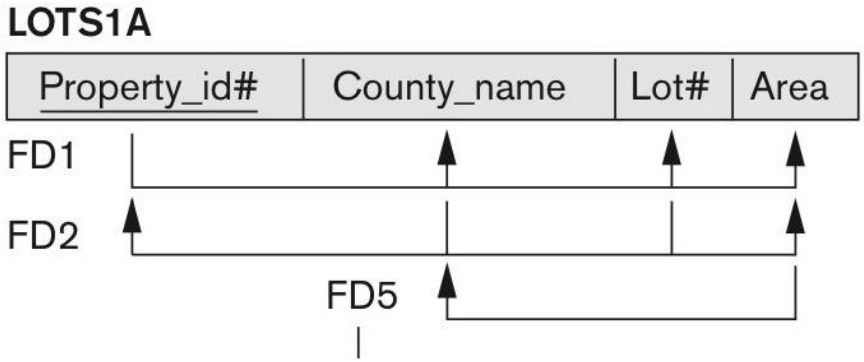
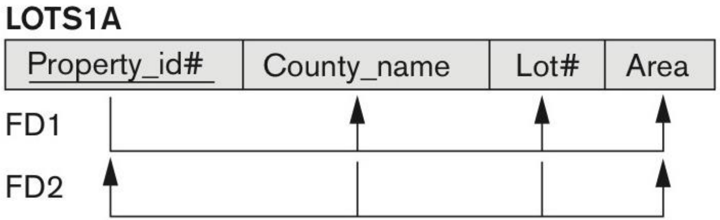
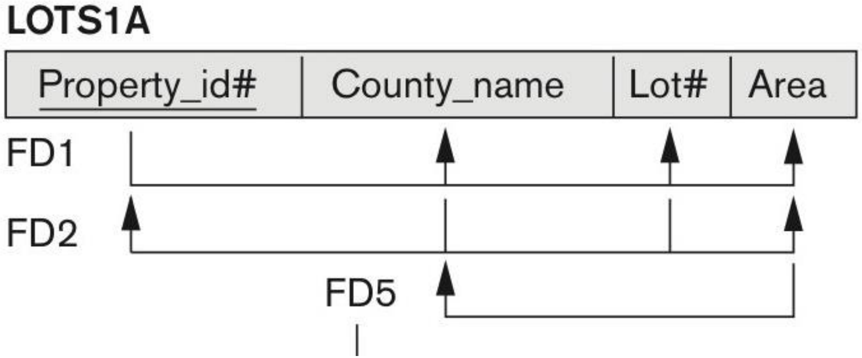


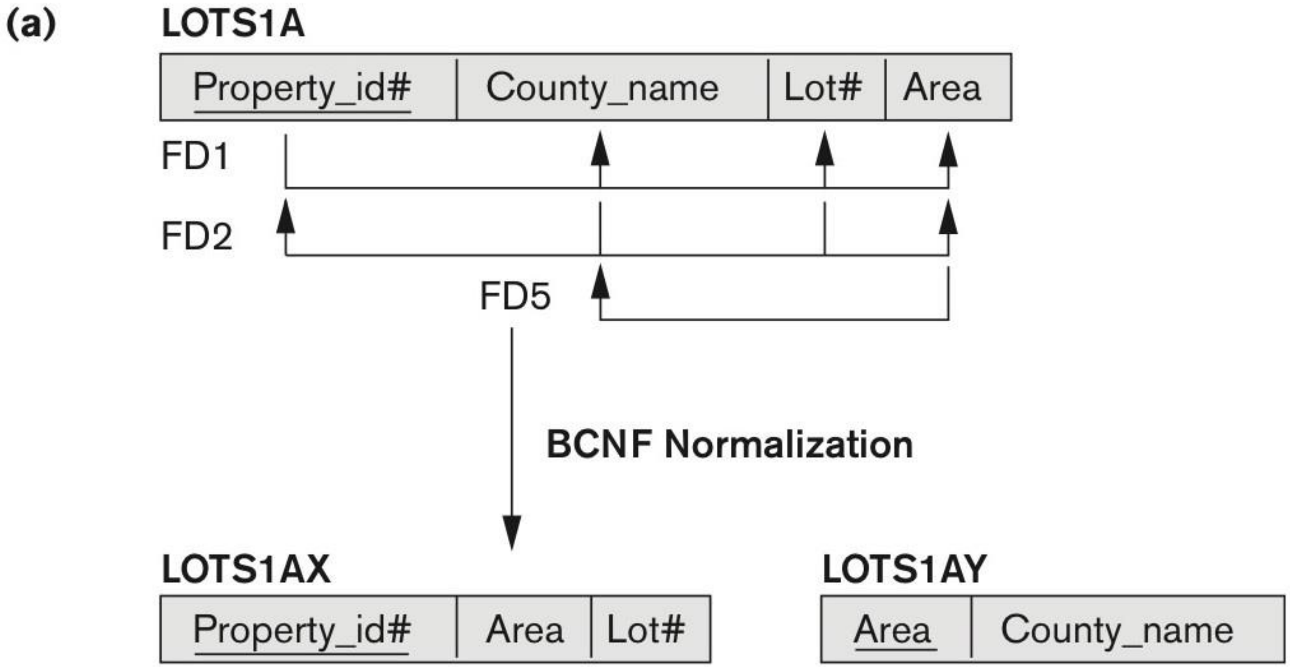
Figure 14.13 Boyce-Codd normal form





Is this in BCNF?
Area is not a superkey of LOTS1A

Figure 14.13 Boyce-Codd normal form



Normalization

https://www.youtube.com/watch?v=ABwD8IYByfk

Search for: Ponnurangam Kumaraguru

https://www.linkedin.com/in/ponguru/

@PK.PROFGIRI

https://twitter.com/ponguru

Interested in working with Precog?

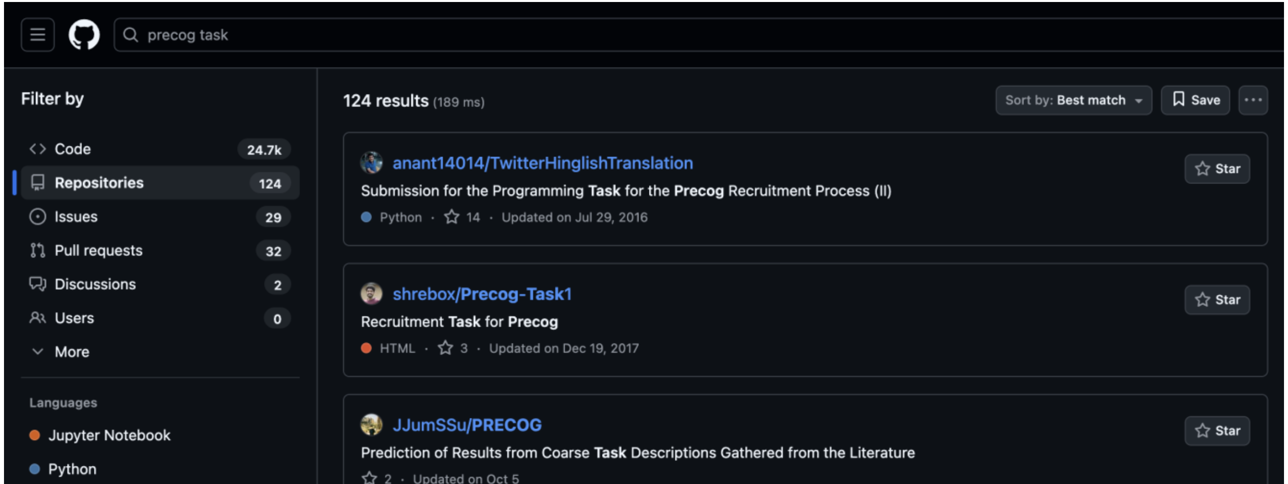
Look for an email in Jan 1st week after start of classes

Areas / Questions: NLP & Responsible AI; Computer Vision; Graphs; Math / Quant

Process: Apply (SOP, CV, etc.) – Task – Technical Interview

What to prepare? Pick an area from above & try out topics

All process done by mid sem



Administrativia

Quiz 3 on 17th Nov

Make up quiz on the day of end-sem exams; if you take the make up quiz it will be counted one of the 3

Same day as end sem at 8 – 9PM

Activity: 13th Nov

(0) Write 2 prime & 2 full dependency attributes in Infinium DB
(1) Write 2 non-prime & 2 partial dependency attributes in Infinium DB

Relation: EMP_PROJECT

| EmpNo | EmpName | ProjNo | ProjName | DeptNo | DeptName | HoursWorked | DeptLocation |

Given information:

1. Each employee works in one department, but a department can have many employees.
2. Each project is controlled by one department.
3. An employee can work on multiple projects, and a project can have multiple employees.
4. `HoursWorked` is the number of hours that an employee works on a particular project.

(2a) Identify all functional dependencies (FDs).

(2b) Find the candidate keys.

(2c) Convert the relation into 1NF, 2NF, 3NF, and BCNF.

(2d) Draw the resulting schemas at each stage.

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you for attending the class!!!