Dayananda Sagar College of Engineering

Kumara Swamy Layout,Bangalore-560078

Department of Artificial Intelligence & Machine Learning

Unit -4

Normalization:

Database Design Theory – Introduction to Normalization using Functional and Multivalued Dependencies: Informal design guidelines for relation schema, Functional Dependencies, Normal Forms based on Primary Keys, Second and Third Normal Forms, Boyce-Codd Normal Form, Multivalued Dependency and Fourth Normal Form, Join Dependencies and Fifth Normal Form.

Text Book:

1. Database systems Models, Languages, Design and Application Programming, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.

2. Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

Prepared by,
Dr.Aruna M G
Associate Professor
Department of AI&ML
DSCE
Bangalore

Informal Design guidelines for Relation Schema

1. Imparting clear semantics of the attributes
2. Reducing the redundant information in tuples.
3. Reducing Null values in tuples.
4. Disallowing the possibility of generating spurious tuples.

1. Imparting clear semantics of the attributes

• **Semantics of a relation :**

Interpretation of attribute values in a tuple.

If semantics of a relation are easier to understand, relation schema design will be better.

Example : Simplified COMPANY schema

• **Guidelines**

- Design a relation schema so that it is easy to explain its meaning.
- Do not combine attributes from multiple entity types and relationship types into single relation.

• **Example :**

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION

- A tuple in emp_dept represents a single employee but includes additional information like Dname, DMGSSN
- A tuple in emp_proj has additional information like Ename, Pname, Plocation.
- This is considered as poor designs because they violate Guideline1 by mixing attributes from distinct real world entities.
- EMP_DEPT mixes attributes of employees and departments.
- EMP_PROJ mixes attributes of employees, projects and works_on

2. Reducing the redundant information in tuples

a) Insertion anomaly
b) Modification anomaly
c) Delete anomaly

Employee

Ename	SSN	Bdate	Address	Dnumber
Amit	10	—	—	5
Kalyan	11	—	—	5
Manvi	12	—	—	4
Nithya	13	—	—	1
Bhuvan	14	—	—	4

Department

Dname	Dnumber	Dmgrssn
Research	5	11
Admin	4	12
Head Quarters	1	13

Project

Pname	Pnumber	Plocation	Dnum
X	1	Bangalore	5
Y	2	Delhi	5
Z	3	Pune	5
P	10	Mumbai	4
Q	20	Pune	1
R	30	Mumbai	4

a) Insertion anomaly

- To insert new values into EMP_DEPT, we must include attribute values for either department that the employee works for or NULL.
- To insert new department that has no employees as yet. This is not possible because SSN is a primary key.

b) Modification anomaly

If we change value of manager of department of 5, we must update all those tuples of employees who work in that department, else inconsistency occurs.

c) Deletion anomaly

Trying to delete employee tuple where only one record is left out, the information about that department is lost from the data base.

- **Guideline**

Design the base relation schemas so that no insertion, deletion or modification anomalies are present in the relations.

3. Reducing Null values in tuples

Relations should have as few NULL values as possible because

- Wastage of memory at the storage level
- Difficult with aggregation functions like count, avg.
- Difficult with join operations because the results will be unpredictable.

4. Spurious tuples

Extra tuples generated when tables are joined.

(a)

EMP_LOCS

ENAME	PLOCATION

p.k.

EMP_PROJ1

SSN	PNUMBER	HOURS	PNAME	PLOCATION

p.k.

EMP_LOCS * EMP_PROJ1 would result in

	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.

let R be a relational schema, let X & Y are subsets of relation R
the FD from X to Y denoted by X->Y, holds in R if and if only for every instance r, of R , for any two tuples t1 & t2 in r that have t1[x]=t2[x],they must also have t1[Y] =t2[Y].

• Functional Dependency

A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples $t1$ and $t2$ in r that have $t1[X] = t2[X]$, they must also have $t1[Y] = t2[Y]$.

- In a given relation R, X and Y are attributes. Attribute Y is functionally dependent on X, if each value of X determines EXACTLY ONE value of Y.

• **Example :**

SSN → Ename

Attribute SSN functionally determines attribute Ename if the value of SSN determines a unique value for Ename.

- X→Y holds - whenever 2 tuples have same value for X, they must have the same value for Y.

• **Example :**

If SSN→ Ename

12345→ Ramesh

Whenever SSN is used, it searches for Ramesh.

- For any 2 tuples t1 and t2 in a relation R, if t1[x]=t2[x] then t1[y]=t2[y]

• **Example :**

	x	**y**
tuple 1	a	1
tuple 2	a	1
tuple 3	b	1
tuple 4	c	2

X→Y in R specifies a constraint on all relational instances.

• **Question :**

Consider the following relation state. Which of the following functional dependencies may hold in the above relation?

i) A → B
ii) B → C
iii) C → B
iv) B → A
v) C → A

A	**B**	**C**
10	b1	c1
10	b2	c2
11	b4	c1
12	b3	c4
13	b1	c1
14	b3	c4

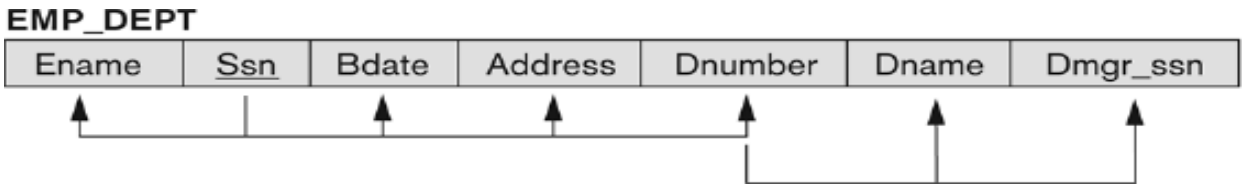
Solution :
i) A → B : ✗
ii) B → C : ✓
iii) C → B : ✗
iv) B → A : ✗
v) C → A : ✗

Explain how functional dependencies contribute to the normalization process.

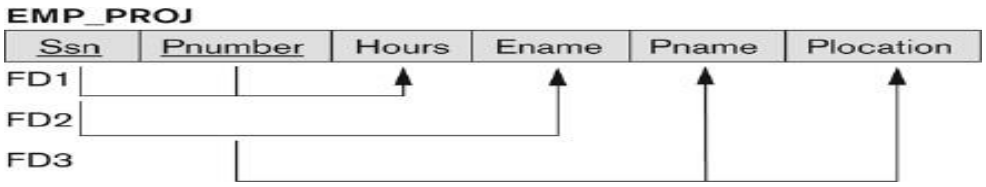
--Normalization is the process of organizing a database in a way that reduces redundancy and dependency. Redundancy means having duplicate data in the database, which can lead to inconsistency and waste of storage space. Dependency means having attributes that depend on other attributes, which can lead to update anomalies and loss of information.

--Functional dependencies help us to identify the keys and dependencies in a relation, and then decompose the relation into smaller relations that satisfy certain normal forms. Normal forms are rules that define the desirable properties of a relation, such as having no partial dependencies, no transitive dependencies, or no multivalued dependencies. By applying normalization, we can improve the design and performance of the database.

• Graphical representation of Functional Dependency



SSN → {Ename, Bdate, Address, Dnumber}
Dnumber → {Dname, Dmgr_ssn}



{SSN, Pnumber} → Hours
SSN → Ename
Pnumber → {Pname, Plocation}

• Normalization

- Database which is designed based on ER model may have some amount of inconsistency, ambiguity (uncertainty) and redundancy.
- To resolve these issues, refinement is required. This refinement process is called as normalization.
- It use a approach normal form as the set of rules. These rules and regulations are known as Normalization.
- Basically, normalization eliminates duplicate data and makes insert, update and delete operations more efficient in terms of performance and space requirement – to store the data.
- Database normalization is data design and organization process applied to data structures based on their functional dependencies and primary keys that help build relational databases.

Normalization Helps:

- Minimizing data redundancy.
- Minimizing the insertion, deletion and update anomalies.
- Reduces input and output delays
- Reducing memory usage.
- Supports a single consistent version of the truth.
- It is an industry best method of tables or entity design.

Uses:

Database normalization is a useful tool for requirements analysis and data modeling process of software development. Thus,
The normalization is the process to reduce the all undesirable problems by using the functional dependencies and keys.

DECOMPOSITION

The decomposition of a relation schemaR = *{A I,A2, .. An}* is its replacement by a set of relation schemes *{R1,R2, ... ,Rm}*

such that ***R1*** **≤** ***R,*** **1 ≤** ***i*** **≤** ***m***

and **R1U R2U R3 U……..Rm =R**

A relation scheme *R* can be decomposed into a collection of relation schemes *{R1, R2, R3, ... , Rm}* to eliminate some of the anomalies contained in the original relation *R.*

The problems in the relation scheme student can be resolved, if we decompose it with the following relation schemes : such as :

Student-INFO (Name, Phone no. Major)

Transcript (Name, Course, Grade)

Teacher (Course Professor)

These decomposition are bad for the following reasons :

(i)" Redundancy and update anomaly, because the data for the attributes phone no. and major are repeated.

(ii) Loss of information, because we lose the fact that a student has a given grade in a particular course.

Terms used in Normalization

1. **Determinant**

   Attribute X is defined as determinant if it uniquely defines the attribute value Y in an entity.
   X→Y means X decides Y
   Here, X is a determinant and Y is dependent
   Example : Result_Details
   Here, marks attribute decides Grade attribute.
   Marks→Grade
   Note : Marks may not be key attribute

2. **Functional Dependency**

   The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

   X → Y

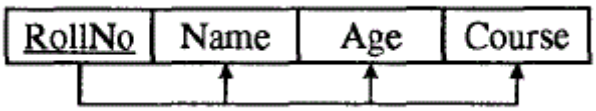
   The left side of FD is known as a determinant, the right side of the production is known as a dependent.

   Consider the following relation.
   Report (<u>student#, course#,</u> coursename, Iname, room#, marks, grade)
   Here

Student#course# (composite key) defines exactly one value for marks.

Student#course# → Marks

Hence, Marks is functionally dependent on student#course#



Armstrong's axioms/properties of functional dependencies:

6 Well-known inference rules for functional dependencies:

IR1 (reflexive rule): If $X \supseteq Y$, then $X \rightarrow Y$.

IR2 (augmentation rule): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.

IR3 (transitive rule): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.

IR4 (decomposition, or projective, rule): $\{X \rightarrow YZ\} \models X \rightarrow Y$.

IR5 (union, or additive, rule): $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$.

IR6 (pseudotransitive rule): $\{X \rightarrow Y, WY \rightarrow Z\} \models WX \rightarrow Z$.

1. **Reflexivity:** If Y is a subset of X, then X→Y holds by reflexivity rule
   Example, {roll_no, name} → name is valid.
2. **Augmentation:** If X → Y is a valid dependency, then XZ → YZ is also valid by the augmentation rule.
   Example, {roll_no, name} → dept_building is valid, hence {roll_no, name, dept_name} → {dept_building, dept_name} is also valid.
3. **Transitivity**: If X → Y and Y → Z are both valid dependencies, then X→Z is also valid by the Transitivity rule.
   Example, roll_no → dept_name & dept_name → dept_building, then roll_no → dept_building is also valid.

Types of Functional Dependency

1. **Trivial FD :**
   A functional dependency of the form $X \rightarrow Y$ is trivial if $Y \subseteq X$.
   Or

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **{roll_no, name} → name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}.**

Similarly, **roll_no → roll_no** is also an example of trivial functional dependency.

Example 2:

Consider a table with two columns Employee_Id and Employee_Name.

{Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as Employee_Id is a subset of {Employee_Id, Employee_Name}.

Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependency.

2. Non-trivial Functional Dependency

In **Non-trivial functional dependency**, the dependent is strictly not a subset of the determinant. i.e. If **X → Y** and **Y is not a subset of X**, then it is called Non-trivial functional dependency.

Or

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **roll_no → name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no.**

Similarly, **{roll_no, name}** → **age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no, name}**

Example2:

1. ID → Name,
2. Name → DOB

3. Full functional dependency

In a given relation R, X and Y are attributes.

Y is fully functionally dependent on X if and only if it is NOT functionally dependent on subset of X.

X may be composite in nature.

Or

A FD $X \rightarrow$ Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more. That is,

***Example Full FD 1* :**

for any attribute $A \in X$, $(X - \{A\})$ does not functionally determine Y.

$(X - \{A\}) \rightarrow$Y is called full functional dependency.

• **Example 2:**

Marks is fully functionally dependent on student#course# and not on subset of student#course#.

Marks cannot be determined by student# or course#

It can be determined only by using student# and course#

4. Partial dependency

In a given relation R, X and Y are attributes.

Y is partially dependent on X if and only if it is functionally dependent on subset of X.

X may be composite in nature.

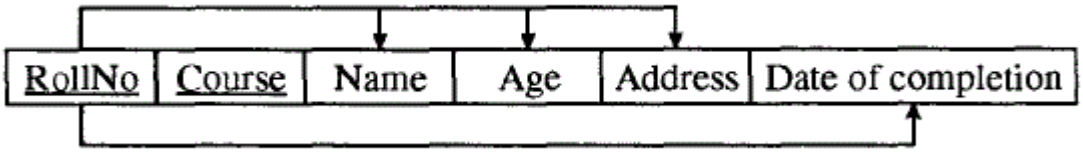
Or

A FD $X \rightarrow$ Y is a partial functional dependency if some attribute $A \in X$ can be removed from X and then the dependency still hold.

That is if for some A ∈ X, (X - {A}) → Y, then it is called partial dependency.

• **Example :**

Coursename, Iname, Room# are partially dependent on student#course#.



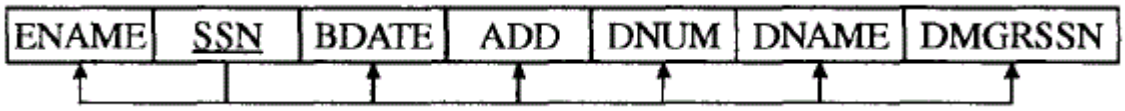
5. **Transitive dependency**

   Room# depends upon Iname and in turn, Iname depends upon course#.

   Hence room# transitively depends upon course#.

   Or

   A functional dependency $X \rightarrow$ Y in a relation scheme R is a transitive dependence if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X $\rightarrow$ Z and Z $\rightarrow$ Y hold.



6. **Key attributes**

   In a given relation R, if the attribute X uniquely defines all other attributes, X is the key attribute.

• **Example :**

   Student#course#

7. **Non-key attributes**

   In a given relation R, the attribute which are not key attributes are non-key attributes.

• Types of Normal Forms

1. First Normal Form (1NF)
2. Second Normal Form (2NF)
3. Third Normal Form (3NF)
4. Boyce Codd Normal Form (BCNF)
5. Fourth Normal Form (4NF)
6. Fifth Normal Form (5NF)

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A table is in BCNF if every functional dependency X → Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

• First Normal Form (1NF)

- A relation will be 1NF if each attribute of a table has atomic (single) value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- 1NF disallows the multi-valued attribute, composite attribute, and their combinations.

• Example 1 :

Before 1NF
Table : Employee

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	Kunal	7272826385, 9064738238	UP
20	Shivani	8574783832	Bihar
12	Gurveer	7390372389, 8589830302	Punjab

After 1NF
Table : Employee

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	Kunal	7272826385	UP
14	Kunal	9064738238	UP
20	Shivani	8574783832	Bihar
12	Gurveer	7390372389	Punjab
12	Gurveer	8589830302	Punjab

Example 2 :

Before 1NF
Table : Student

SID	SNAME	C_NAME
1	Aishwarya	C++/Java
2	Namitha	Java
3	Sunil	DBMS/Python

After 1NF
Table : Student
PK : {Sid, C_Name}

SID	SNAME	C_NAME
1	Aishwarya	C++
1	Aishwarya	Java
2	Namitha	Java
3	Sunil	DBMS
3	Sunil	Python

After 1NF
Table : Student
PK : {SID}

SID	SNAME	C1_NAME	C2_NAME
1	Aishwarya	C++	Java
2	Namitha	Java	
3	Sunil	DBMS	Python

After 1NF
Table : Student
PK : {SID}

SID	SNAME
1	Aishwarya
2	Namitha
3	Sunil

After 1NF
Table : Course
FK : {SID}

SID	C_NAME
1	C++
1	Java
2	Java
3	DBMS
3	Python

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- All the non-prime attributes should be *fully functional dependent* on candidate key.(primary key) (No partial dependencies)

Prime attribute − An attribute, which is a part of the candidate-key, is known as a prime attribute.

Non-prime attribute − An attribute, which is not a part (not member) of the prime-key/ candidate key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.

- **Example 1 :**

Before 2NF
Table : Customer

CUST_ID	STORE_ID	LOCATION
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Candidate Key: {Cust_Id, Store_Id}
Non prime attribute: Location
Location is determined by Store_Id alone
It is determined by a part of the candidate key.

After 2NF
Table : Customer_Detail

CUST_ID	STORE_ID
1	1
1	3
2	1
3	2
4	3

Primary Key :
{Cust_Id, Store_Id}

After 2NF
Table : Store_Detail

STORE_ID	LOCATION
1	Delhi
2	Bangalore
3	Mumbai

Primary Key :
{Store_Id}

Before 2NF
Table : Teacher

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

After 2NF
Table : Teacher_Subject

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Primary Key :
{Teacher_Id, Subject}

After 2NF
Table : Teacher_Detail

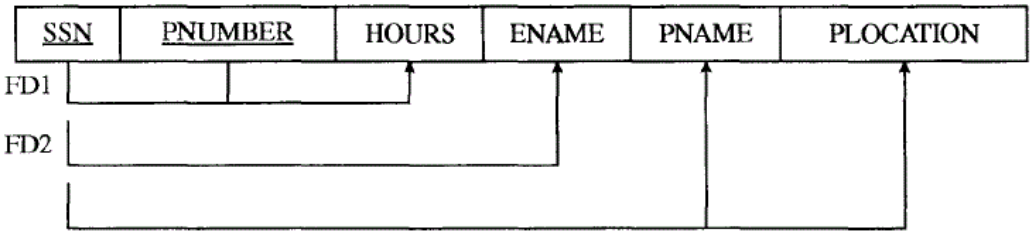
TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

Primary Key :
{Teacher_Id}

The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".

Example3 :

- If the table has a single field as the primary key, it is automatically in 2NF.



FD1: {SSN,PNUMBER} -> HOURS
FD2: {SSN} ->ENAME
FD3: {SSN} ->{PNAME,PLOCATION}

Example4 :

TEACHER :

Course	Prof	Room	Room–Cap	Enrol–Unt
353	Vijay	A532	45	40
351	Vijay	C320	100	60
355	Santosh	H940	50	45
456	Santosh	B278	50	45
459	Gopal	D110	300	200

TEACHER Relation in Second Normal Form

Course	Prof	Enrol–Unt
353	Vijay	40
351	Vijay	60
355	Santosh	45
456	Santosh	45
459	Gopal	200

(a)

Course	Room
353	A532
351	C320
355	H940
456	B278
459	D110

(b)

Room	Room–Cap
A532	45
C320	100
H940	50
B278	50
D110	300

(c)

• Third Normal Form (3NF)

- A relation is in 3NF if it is in 2NF and no non key attribute **is *transitively dependent*** on the primary key.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

• Example 1 :

Before 3NF Student				
SID	SNAME	STATE	COUNTRY	AGE
1	Suresh	Karnataka	India	21
2	Anil	Punjab	India	19
3	Suresh	Bihar	India	20
4	Mithun	Karnataka	India	19

Transitive dependency

If A→B and B→C are two FDs then A→C is called transitive dependency.

FD set:

{ SID → SNAME
SID → STATE
STATE → COUNTRY
SID → AGE
}

✓ SID → COUNTRY
✓ Violates 3NF

Candidate Key:

{SID}

After 3NF Student			
SID	SNAME	STATE	AGE
1	Suresh	Karnataka	21
2	Anil	Punjab	19
3	Suresh	Bihar	20
4	Mithun	Karnataka	19

After 3NF State_Country	
STATE	COUNTRY
Karnataka	India
Punjab	India
Bihar	India
Karnataka	India

Vendor

ID	Name	Account_No	Bank_Code_No	Bank

Startsituation

Result after normalisation

Vendor

ID	Name	Account_No	Bank_Code_No

Bank

Bank_Code_No	Bank

NF	Description	Status
1NF	All attributes are single valued	Exists
2NF	1. Name, Account_No, Bank_Code_No are functionally dependent on ID **(ID → Name, Account_No, Bank_Code_No)** 2. Bank is functionally dependent on Bank_Code_No **(Bank_Code_No → Bank)**	Exists

3NF	Transitive dependency exists between Bank_Code_No and Bank, because Bank_Code_No is not the primary key of this relation. To get 3NF, we have to put the bank name in a separate table together with the clearing number to identify it.	Does not exist

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form or BCNF is an extension to 3NF, and is also known as 3.5 Normal Form.
It is stricter than 3NF.

- **Rules for BCNF**

A table is in BCNF if

- Every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Before BCNF Teacher PK : {SID, Subject}		
SID	**Subject**	**Professor**
101	Java	Chandrakanth
101	C++	Bhuvan
102	Java	Pallavi
103	C#	Praveen
104	Java	Jagan

- One student can enroll for multiple subjects. For example, student with **SID** 101, has opted for subjects - Java & C++
- For each subject, a professor is assigned to the student and, there can be multiple professors teaching one subject like we have for Java.

- **FD :**

  Professor → Subject

NF	Status
1NF	Yes, All values are single valued.
2NF	Yes, No partial dependencies
3NF	Yes, No transitive dependency
BCNF	No {SID, Subject} - **prime attribute**. Consider Professor → Subject And while subject is a prime attribute, professor is a **non-prime attribute**, which is not allowed by BCNF.

Before BCNF Teacher PK : {SID, Subject}		
SID	Subject	Professor
101	Java	Chandrakanth
101	C++	Bhuvan
102	Java	Pallavi
103	C#	Praveen
104	Java	Jagan

After BCNF Student_Detail PK : {SID, PID}	
SID	PID
101	1
101	2
102	3
103	4
104	5

After BCNF Professor_Detail PK : {PID}		
PID	Professor	Subject
1	Chandrakanth	Java
2	Bhuvan	C++
3	Pallavi	Java
4	Praveen	C#
5	Jagan	Java

- **Example 2 :**

EMP				
EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

Candidate key : {EMP-ID, EMP-DEPT}

FD :

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables.

EMP_COUNTRY	
EMP_ID	EMP_COUNTRY
264	India
264	India
364	UK
364	UK

EMP_DEPT		
EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING	
EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

- **Functional dependencies:**

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

- **Candidate keys:**

For the first table: EMP_ID

For the second table: EMP_DEPT

For the third table: {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

• Fourth Normal Form (4NF)

- A relation will be in 4NF if it is in BCNF and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Student		
SID	**COURSE**	**HOBBY**
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.
In the STUDENT relation, a student with SID 21 contains two courses, Computer and Math and two hobbies, Dancing and Singing.
So there is a Multi-valued dependency on SID, which leads to unnecessary repetition of data.

Before 4NF Student		
SID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

After 4NF Student_Course	
SID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

After 4NF Student_Hobby	
SID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

• Fifth Normal Form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Student		
SUBJECT	**LECTURER**	**SEMESTER**
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

Before 5NF Student

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

After 5NF P1

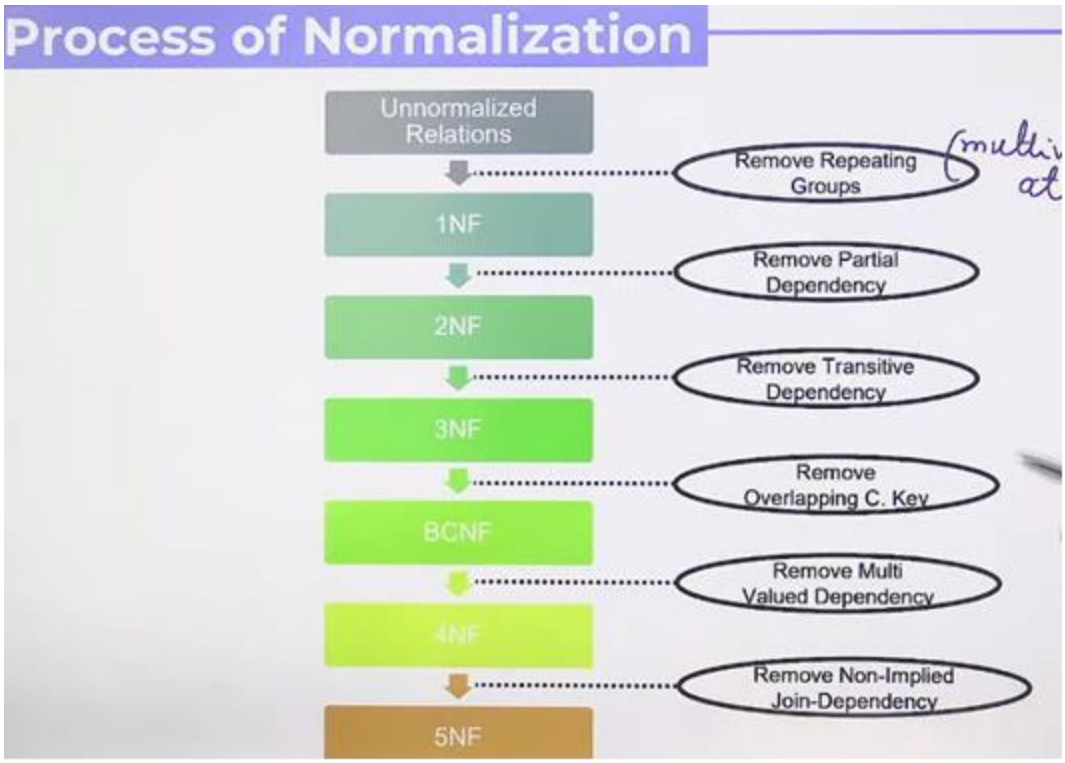
SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

After 5NF P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash

After 5NF P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash

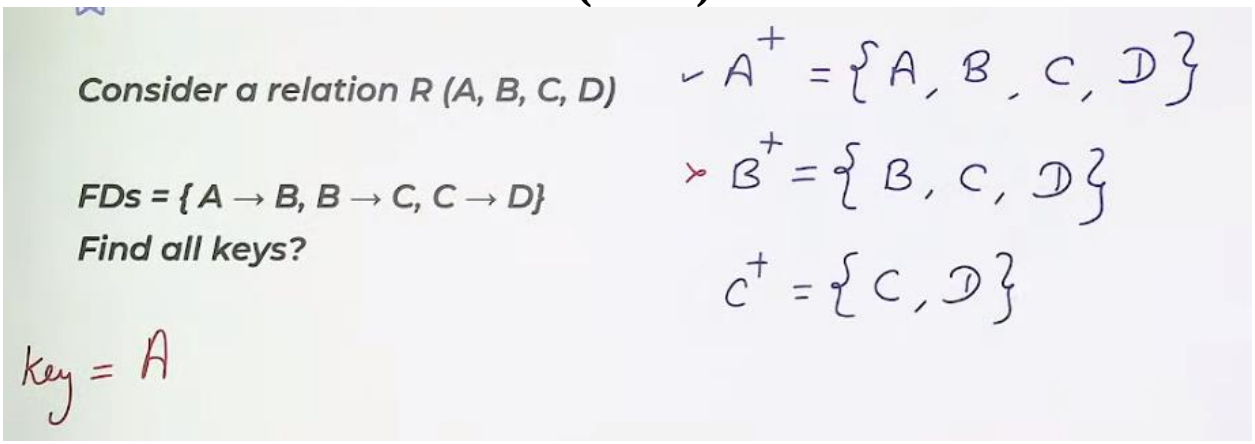


NOTE:
Functional Dependency

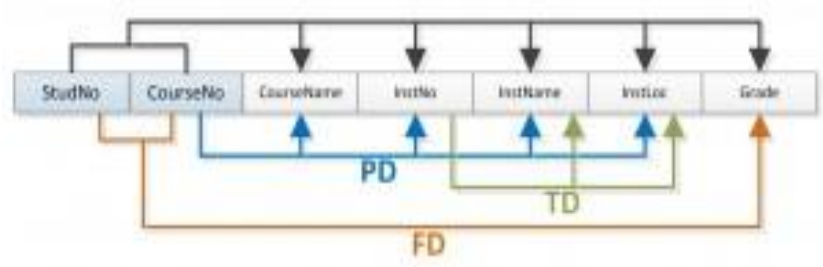
E	A	B	C	D
e_1	a_1	b_1	c_1	d_1
e_2	a_1	b_2	c_1	d_2
e_3	a_2	b_2	c_2	d_2
e_4	a_2	b_2	c_2	d_3
e_5	a_3	b_3	c_2	d_4

Try to find an attribute or set of attributes which can derive all the attributes.

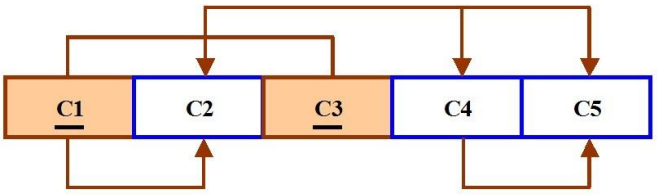
(KEY)



Example #1
Take a look at these functional dependencies in the relation A (P, Q, R, S, T)
Here,
P -> QR,
RS -> T,
Q -> S,
T -> P
In the relation given above, all the possible candidate keys would be {P, T, RS, QR}. In this case, the attributes that exist on the right sides of all the functional dependencies are prime.
Example #2



Example #3 Identify and discuss each of the indicated dependencies in the dependency diagram shown in below Figure.



Example 4: **Click the link : solution for below problem**
https://opentextbc.ca/dbdesign01/chapter/chapter-12-normalization/

1. To keep track of students and courses, a new college uses the table structure in Figure a. Draw the dependency diagram for this table.

Attribute Name	Sample Value	Sample Value	Sample Value
StudentID	1	2	3
StudentName	John Smith	Sandy Law	Sue Rogers
CourseID	2	2	3
CourseName	Programming Level 1	Programming Level 1	Business
Grade	75%	61%	81%
CourseDate	Jan 5th, 2014	Jan 5th, 2014	Jan 7th, 2014

Figure a For question 1, by A. Watt.

2. Using the dependency diagram you just drew, show the tables (in their third normal form) you would create to fix the problems you encountered. Draw the dependency diagram for the fixed table.
3. An agency called Instant Cover supplies part-time/temporary staff to hotels in Scotland. Figure b lists the time spent by agency staff working at various hotels. The national insurance number (NIN) is unique for every member of staff. Use Figure 12.4 to answer questions (a) and (b).

NIN	ContractNo	Hours	eName	hNo	hLoc
1135	C1024	16	Smith J.	H25	East Killbride
1057	C1024	24	Hocine D.	H25	East Killbride
1068	C1025	28	White T.	H4	Glasgow
1135	C1025	15	Smith J.	H4	Glasgow

Figure b: For question 8, by A. Watt.

1. This table is susceptible to update anomalies. Provide examples of insertion, deletion and update anomalies.
2. Normalize this table to third normal form. State any assumptions.

1. **Boyce-Codd Normal Form (BCNF)**

   When a table has more than one candidate key, anomalies may result even though the relation is in 3NF. *Boyce-Codd normal form* is a special case of 3NF. A relation is in BCNF if, and only if, every determinant is a candidate key.

2. BCNF Example 1

   Consider the following table (**St_Maj_Adv**).

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

The *semantic rules* (business rules applied to the database) for this table are:

1. Each Student may major in several subjects.
2. For each Major, a given Student has only one Advisor.
3. Each Major has several Advisors.
4. Each Advisor advises only one Major.
5. Each Advisor advises several Students in one Major.

The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.

1. Student_id, Major ——> Advisor
2. Advisor ——> Major

Anomalies for this table include:

1. Delete – student deletes advisor info
2. Insert – a new advisor needs a student
3. Update – inconsistencies

Note: No single attribute is a candidate key.

PK can be Student_id, Major or Student_id, Advisor**.**

To reduce the **St_Maj_Adv** relation to BCNF, you create two new tables:

1. **St_Adv** (Student_id, Advisor)
2. **Adv_Maj** (Advisor, Major)

St_Adv table

Student_id	Advisor
111	Smith
111	Chan

320	Dobbs
671	White
803	Smith

Adv_Maj table

Advisor	**Major**
Smith	Physics
Chan	Music
Dobbs	Math
White	Physics

3. BCNF Example 2

Consider the following table (**Client_Interview)**.

ClientNo	**InterviewDate**	**InterviewTime**	**StaffNo**	**RoomNo**
CR76	13-May-02	10.30	SG5	G101
CR56	13-May-02	12.00	SG5	G101
CR74	13-May-02	12.00	SG37	G102
CR56	1-July-02	10.30	SG5	G102

FD1 – ClientNo, InterviewDate –> InterviewTime, StaffNo, RoomNo (PK)
FD2 – staffNo, interviewDate, interviewTime –> clientNO (candidate key: CK)
FD3 – roomNo, interviewDate, interviewTime –> staffNo, clientNo (CK)
FD4 – staffNo, interviewDate –> roomNo

A relation is in BCNF if, and only if, every determinant is a candidate key. We need to create a table that incorporates the first three FDs (**Client_Interview2** table) and another table (**StaffRoom** table) for the fourth FD.

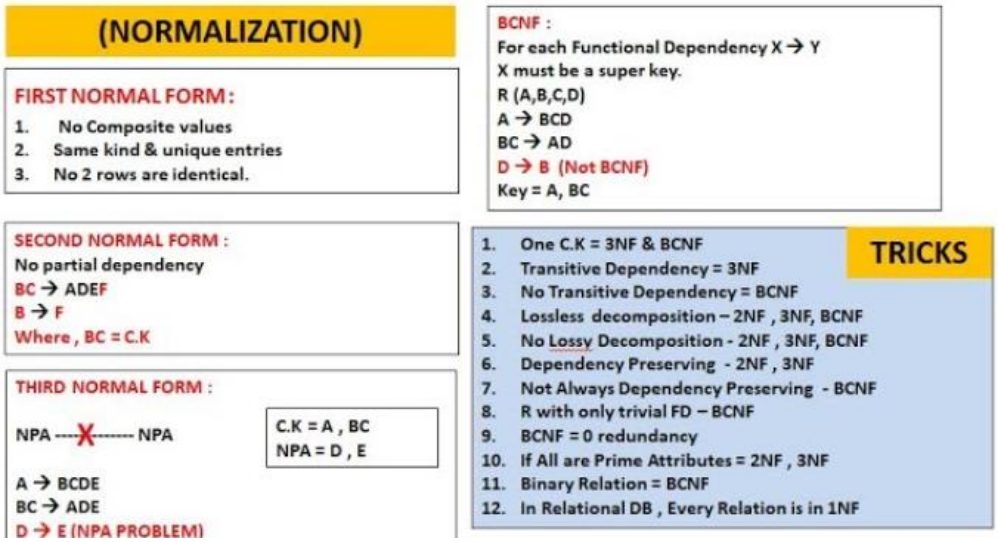
Client_Interview2 table

ClientNo	**InterviewDate**	**InterViewTime**	**StaffNo**
CR76	13-May-02	10.30	SG5
CR56	13-May-02	12.00	SG5
CR74	13-May-02	12.00	SG37

CR56	1-July-02	10.30	SG5

StaffRoom table

StaffNo	InterviewDate	RoomNo
SG5	13-May-02	G101
SG37	13-May-02	G102
SG5	1-July-02	G102



All non prime attributes are fully functional dependent an any key of R.

Non Prime Attributes –
A attribute which is not a part of candidate key is known as non prime attribute.

Fully Functional Dependency –

X → Y
Y is fully FD on X
If X determines Y , Y cannot be determined by any proper subset of X.

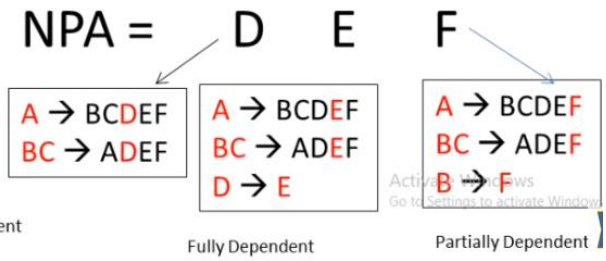
F = { A → BCDEF
BC → ADEF
B → F
D → E }

2ND NF ?

1. Determine Non Prime Attributes

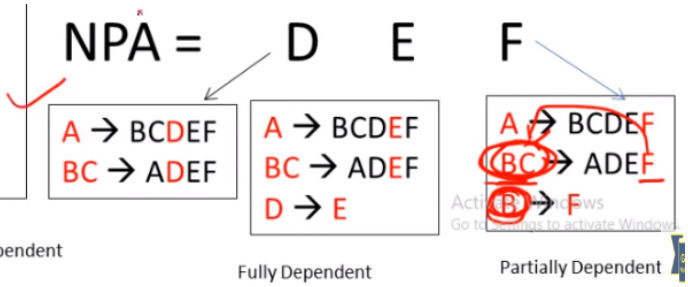
A+ = ABCDEF
BC+ = ABCDEF
C.K = A , BC
NPA = DEF

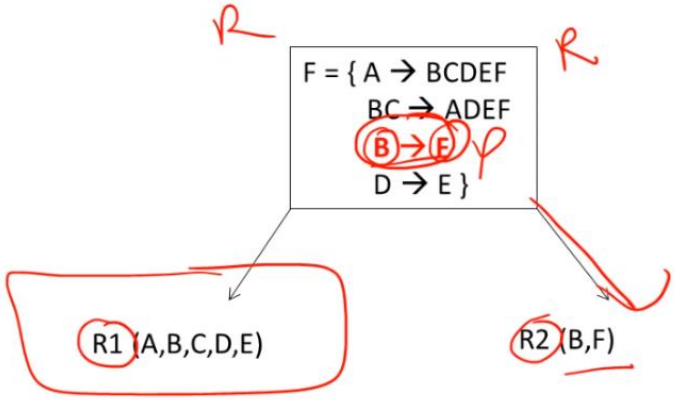
Since F is partially dependent on B Not 2nd NF



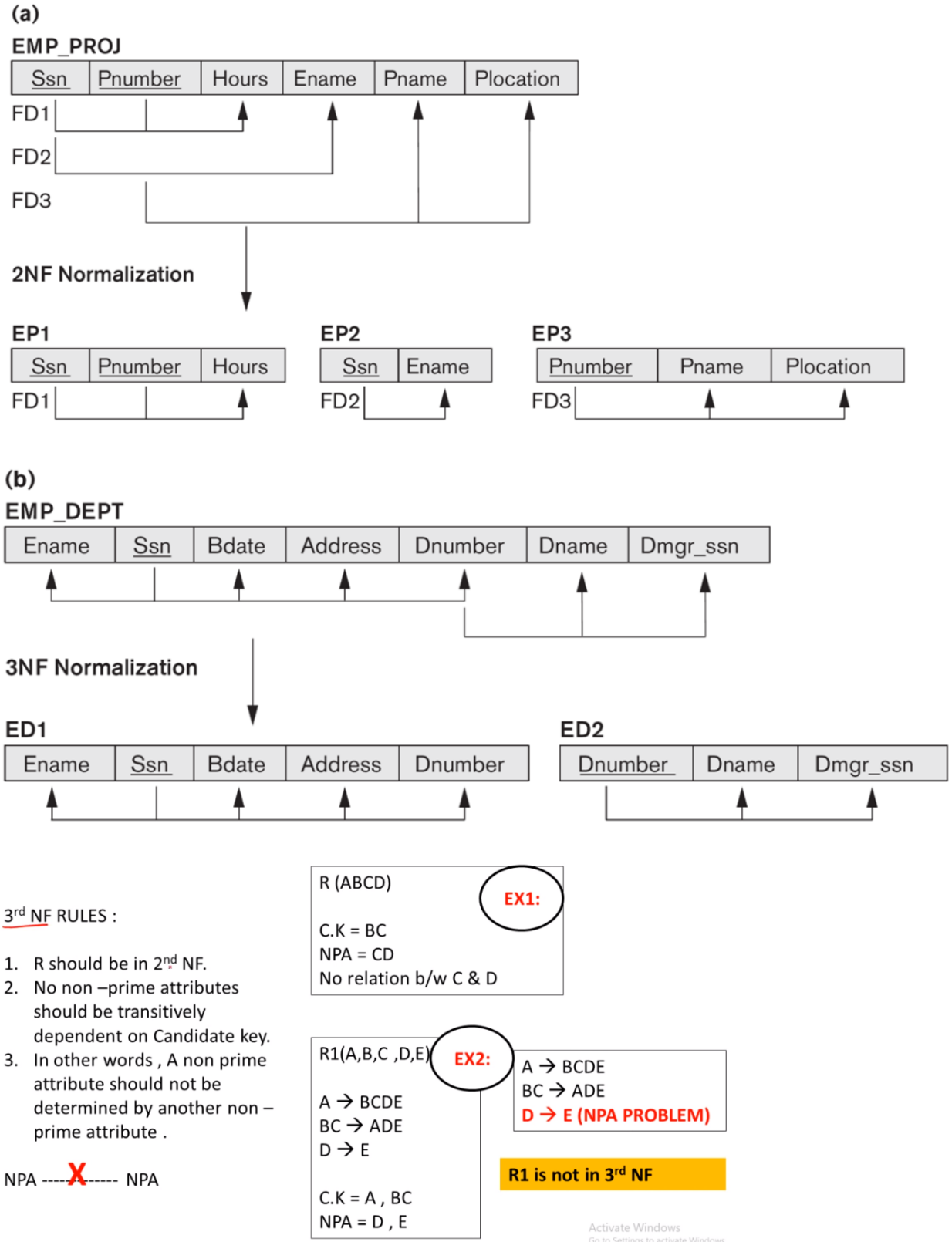
Solution:

X → Y
Y is fully FD on X
If X determines Y , Y cannot be determined by any proper subset of X.





R1 (A,B,C,D,E) Now, it satisfies 2nd NF



3rd NF RULES :

1. R should be in 2nd NF.
2. No non –prime attributes should be transitively dependent on Candidate key.
3. In other words , A non prime attribute should not be determined by another non – prime attribute .

NPA ----X----- NPA

R (ABCD)

EX1:

C.K = BC
NPA = CD
No relation b/w C & D

R1(A,B,C ,D,E)

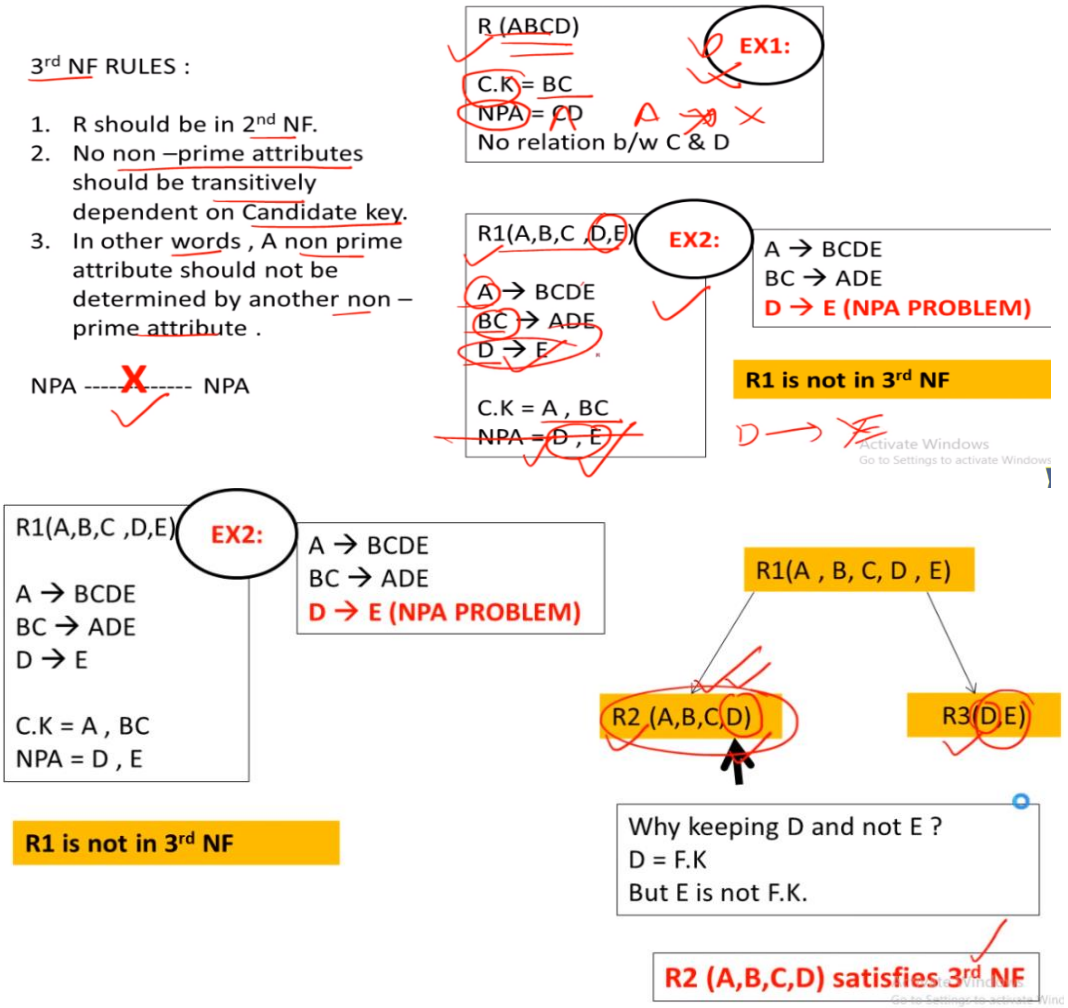
EX2:

A → BCDE
BC → ADE
D → E

C.K = A , BC
NPA = D , E

A → BCDE
BC → ADE
D → E (NPA PROBLEM)

R1 is not in 3rd NF



Consider the following relationship : **R (A,B,C,D)**

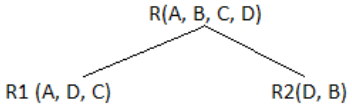
and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC**.

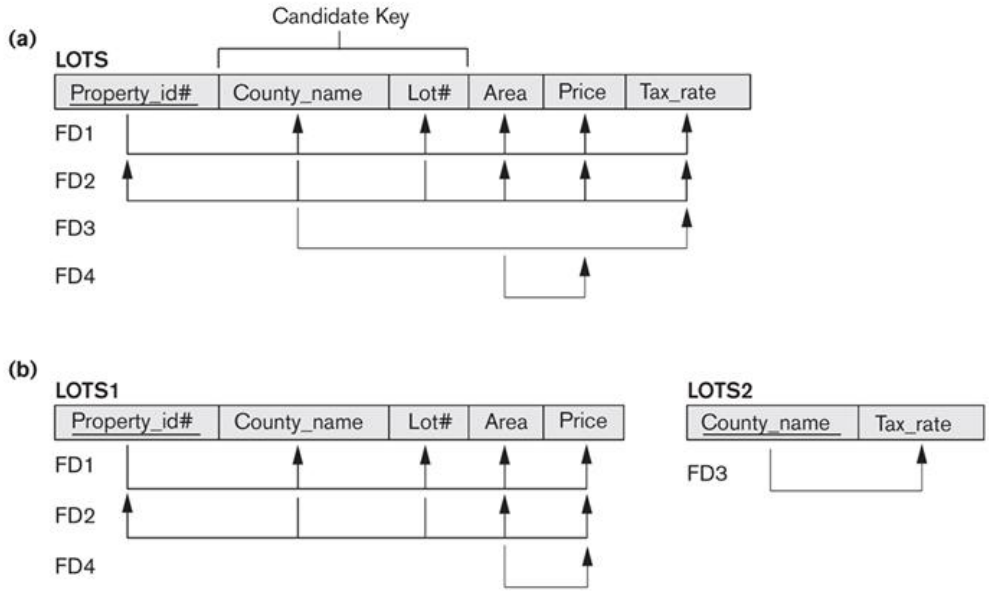
Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.



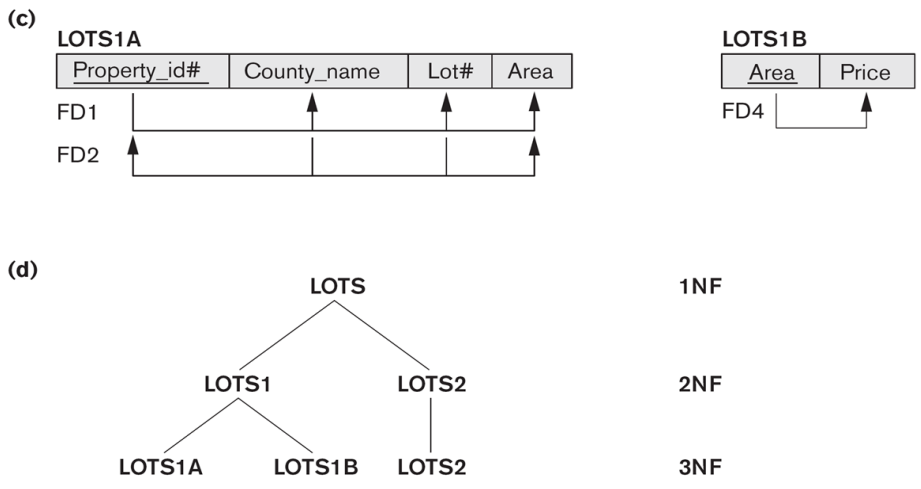
Breaking, table into two tables, one with A, D and C while the other with D and B.

Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.



Can we factor this out?

The LOTS1 relation above (EN fig above) is not 3NF, because of Area ⟶ Price. So we factor on Area ⟶ Price, dividing into LOTS1A(property_ID, county,lot_num,area) and LOTS1B(area,price). Another approach would be to drop price entirely, if it is in fact *proportional* to area, and simply treat it as a computed attribute.



BCNF :

For each Functional Dependency X → Y
X must be a super key.

R (A,B,C,D)
A → BCD
BC → AD
D → B (Not BCNF)
Key = A, BC

BCNF Rules :

1. Relation must be in 3rd NF.
2. For each Functional Dependency X → Y

X must be a super key.

R (A,B,C,D)

A → BCD
BC → AD
D → B (Not BCNF)

Key = A, BC

R (A,B,C,D)

A → BCD
BC → AD
D → B

Key = A, BC

Now, this is not BCNF

example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200
1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate key: {emp_id, emp_dept}
The table is not in BCNF as neither emp_id nor emp_dept alone are keys.
To make the table comply with BCNF we can break the table in three tables like this:
emp_nationality table:

emp_id	emp_nationality

1001	Austrian
1002	American

emp_dept table:

emp_dept	**dept_type**	**dept_no_of_emp**
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	**emp_dept**
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}
Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}
This is now in BCNF as in both the functional dependencies left side part is a key.

NORMALIZATION GATE PROBLEMS

GATE 2012

Which of the following is TRUE?
(A) Every relation in 3NF is also in BCNF
(B) A relation R is in 3NF if every non-prime attribute of R is fully functionally dependent on every key of R
(C) Every relation in BCNF is also in 3NF
(D) No relation can be in both BCNF and 3NF

GATE 2013

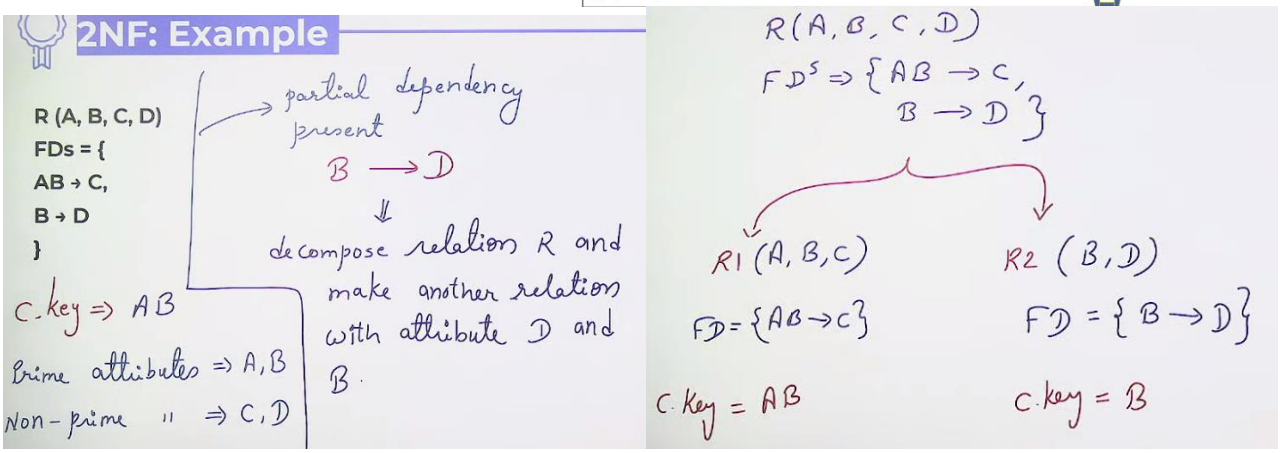
Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values.
The relation R is
F = {CH -> G, A -> BC, B -> CFH, E -> A, F -> EG}
(A) in 1NF, but not in 2NF. (B) in 2NF, but not in 3NF.
(C) in 3NF, but not in BCNF. (D) in BCNF

GATE 2013

Relation R has eight attributes ABCDEFGH. Fields of R contain only atomic values.
F = {CH -> G, A -> BC, B -> CFH, E -> A, F -> EG} is a set of functional dependencies (FDs) so that F+ is exactly the set of FDs that hold for R.
How many candidate keys does the relation R have?

GATE 2005

Which one of the following statements about normal forms is FALSE?
(A) BCNF is stricter than 3NF
(B) Lossless, dependency-preserving decomposition into 3NF is always possible
(C) Lossless, dependency-preserving decomposition into BCNF is always possible
(D) Any relation with two attributes is in BCNF



Consider a relation R (A, B, C, D) with candidate key AB. Select the FD which will make the relation not in 2NF?

(A) AB → C
(B) AB → D
(C) A → D
(D) B → C
}