Topics Covered in Todays Class

Unit 3: Database Design Theory and Normalization:
- Informal Design Guidelines for Relation Schemas
- Functional Dependencies
- Normal Forms Based on Primary Keys

What is Normalization ?

- Database normalization is the process of organizing data to minimize data redundancy (data duplication), which in turn ensures data consistency.

Problems of Data Redundancy

1. Disk Space Wastage
2. Data inconsistency
3. DML queries can become slow

What is Normalization ?

- Database Normalization is step by step process.
- There are five normal forms, First Normal Form (1NF) through fifth Normal Form (5NF)
- Most databases are in third normal form (3NF).
- There are certain rules, that each normal form should follow.

Redundant Information in records and Update Anomaly

Example

Student Table

usn	**name**	**dep_ num**
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14IS001	Avinash	20
1BM14IS002	Dinesh	20

Department Table

d_id	**dep_ name**	**HOD**
10	CSE	Dr.Guruprasad
20	ISE	Dr.Gowrishankar

Example

Student-Department Table

<u>usn</u>	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Storing two different independent (student and department) information will cause Update Anomalies

Update Anomalies

- Insertion Anomaly
- Deletion Anomaly
- Modification Anomaly

Insertion Anomaly

Student-Department Table

<u>usn</u>	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Say we want to introduce new department with only department details for which students have not joined this department yet ?

Student-Department Table

<u>usn</u>	name	dep_ num	Dep_name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar
???	???	30	Aerospace	Dr. Satish Jain

Violating Primary key constraint

Insertion Anomaly

Student-Department Table

usn	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Say we want to introduce new department with only department details for which students have not joined this department yet ?

Student-Department Table

usn	name	dep_ num	Dep_name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar
???	???	30	Aerospace	Dr. Satish Jain

Violating Primary key constraint

Insertion anomaly means that that some data can not be inserted in the database.

Insertion Anomaly

Insertion Anomaly will never occur if the table design was as follows:

Student Table

usn	name	dep_num
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14IS001	Avinash	20
1BM14IS002	Dinesh	20

Department Table

d_id	dep_name	HOD
10	CSE	Dr.Guruprasad
20	ISE	Dr.Gowrishankar
30	Aerospace	Dr. Satish Jain

Deletion Anomaly

Student-Department Table

usn	name	dep_ num	Dep_name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar
1BM14AS001	Abhijit	30	Aerospace	Dr. Satish Jain

Say we want to delete Abhijit record from the table, then the information related to Aerospace department will be lost.

Deletion Anomaly

Student-Department Table

usn	name	dep_ num	Dep_name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar
1BM14AS001	Abhijit	30	Aerospace	Dr. Satish Jain

Say we want to delete Abhijit record from the table, then the information related to Aerospace department will be lost.

Deletion anomaly means deleting some data cause other information to be lost.

Deletion Anomaly

Deletion Anomaly would not have occurred
if the table design was as follows. Here student and
department information is stored separately

Student Table

usn	**name**	**dep_ num**
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14IS001	Avinash	20
1BM14IS002	Dinesh	20
~~1BM14AS001~~	~~Abhijit~~	~~30~~

Department Table

d_id	**dep_name**	**HOD**
10	CSE	Dr.Guruprasad
20	ISE	Dr.Gowrishankar
30	Aerospace	Dr. Satish Jain

Modification Anomaly

Say we want to change the name of
HOD for the Department CSE. Then we have
do changes to all the records referring to CSE department

Student-Department Table

usn	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Changing name of
CSE HOD should
be reflected in three
records

Modification Anomaly

Say we want to change the name of HOD for the Department CSE. Then we have do changes to all the records referring to CSE department

Student-Department Table

usn	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Changing name of CSE HOD should be reflected in three records

Update anomaly means we have data redundancy in the database and to make any modification we have to change all copies of the redundant data or else the database will contain incorrect data.

Modification Anomaly

Modification Anomaly will never occur if the table design was as follows:

Student Table

usn	name	dep_num
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14IS001	Avinash	20
1BM14IS002	Dinesh	20

Department Table

d_id	dep_name	HOD
10	CSE	Dr.Guruprasad H S
20	ISE	Dr.Gowrishankar

Changing in only one record

Normalization

Student-Department Table

usn	name	dep_ num	Dep_ name	HOD
1BM14CS001	Avinash	10	CSE	Dr.Guruprasad
1BM14CS002	Balaji	10	CSE	Dr.Guruprasad
1BM14CS003	Chandan	10	CSE	Dr.Guruprasad
1BM14IS001	Avinash	20	ISE	Dr.Gowrishankar
1BM14IS002	Dinesh	20	ISE	Dr.Gowrishankar

Normalized Table Design

Student Table

usn	name	dep_ num
1BM14CS001	Avinash	10
1BM14CS002	Balaji	10
1BM14CS003	Chandan	10
1BM14IS001	Avinash	20
1BM14IS002	Dinesh	20

Department Table

d_id	dep_ name	HOD
10	CSE	Dr.Guruprasad
20	ISE	Dr.Gowrishankar

Informal Design Guidelines for relation Schemas

Four informal measures of quality for relation schema design

1. Semantics of the Relation Attributes
2. Redundant Information in Tuples and Update Anomalies
3. Null Values in Tuples
4. Spurious Tuples

Guideline 1: Try to make user interpretation easy

- 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
  - Entity and relationship attributes should be kept apart as much as possible.
- <u>Bottom Line:</u> *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*
- Example:

EMP(FNAME, LNAME, SSN)
WORKS_ON (SSN, PNO)
PROJECT_LOC(PNO, PLOC)

EMP(FNAME,LNAME,SSN, PNO,PLOC)

Perhaps this schema has too much information to absorb per record ?

Guideline 2: Try to reduce Redundancy and avoid Update Anomalies

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

Guideline 3:Avoid too many NULL values

Avoid too many NULL values

- Space is wasted
- Problems occur when using aggregate functions like count or sum
- NULLs can have different intentions
  - Attribute does not apply
  - Value unknown and will remain unknown
  - Value unknown at present

Guideline 4: Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
  - Split a table into smaller tables (with fewer columns in each), When reconstructing the “original” data, should not introduce spurious tuples

- GUIDELINE 4:
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

What is spurious tuples ?

- A spurious tuple is, basically, a record in a database that gets created when two tables are joined badly. In database, spurious tuples are created when two tables are joined on attributes that are neither primary keys nor foreign keys.
- Example, consider

CAR (ID, Make, Color)

123	Toyota	Blue
456	Audi	Blue
789	Toyota	Red

CAR1 (ID, Color)

123	Blue
456	Blue
789	Red

CAR2 (Color, Make)

Blue	Toyota
Blue	Audi
Red	Toyota

What happens when we join (or combine) CAR1 and CAR2 ?

What is spurious tuples ?

- Example, consider

CAR (ID, Make, Color)

123	Toyota	Blue
456	Audi	Blue
789	Toyota	Red

CAR1 (ID, Color)

123	Blue
456	Blue
789	Red

CAR2 (Color, Make)

Blue	Toyota
Blue	Audi
Red	Toyota

What happens when we join (or combine) CAR1 and CAR2 ?

123	Blue	Toyota
123	Blue	Audi
456	Blue	Toyota
456	Blue	Audi
789	Red	Toyota

What is spurious tuples ?

- Example, consider

CAR (ID, Make, Color)

123	Toyota	Blue
456	Audi	Blue
789	Toyota	Red

CAR1 (ID, Color)

123	Blue
456	Blue
789	Red

CAR2 (Color, Make)

Blue	Toyota
Blue	Audi
Red	Toyota

What happens when we join CAR1 and CAR2 ?

123	Blue	Toyota
123	Blue	Audi
456	Blue	Toyota
456	Blue	Audi
789	Red	Toyota

Spurious Records

So, What we will learn next

A **theory of schema design**

- ☐ Functional dependencies and normalization

☐ Using functional dependencies we define “normal forms” of schema

Functional Dependency

- Functional Dependency **represents relationship among attributes.**
- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.
- Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

Functional Dependency

- Example

Student Table

USN	Name	Grade
1BM14CS001	Avinash	S
1BM14CS002	Balaji	A
1BM14CS003	Chandan	B
1BM14IS001	Avinash	C
1BM14IS002	Balaji	A

What is the Grade of **Avinash** ?

Functional Dependency

- Example

Student Table

USN	Name	Grade
1BM14CS001	Avinash	S
1BM14CS002	Balaji	A
1BM14CS003	Chandan	B
1BM14IS001	Avinash	C
1BM14IS002	Balaji	A

What is the Grade of **Avinash** ?

Which Avinash grade ?
Is it 1BM14CS001 or 1BM14IS001

Functional Dependency

- Example

Student Table

USN	Name	Grade
1BM14CS001	Avinash	S
1BM14CS002	Balaji	A
1BM14CS003	Chandan	B
1BM14IS001	Avinash	C
1BM14IS002	Balaji	A

What is the Grade of **Avinash** ?

Which Avinash grade ?
Is it 1BM14CS001 or 1BM14IS001

Here USN will help to **determine** Grade of the student

USN → Grade   Functional Dependency

USN is Determinant Attribute

Grade is Dependent Attribute

Functional Dependency

- Example

Student Table

USN	Name	Grade
1BM14CS001	Avinash	S
1BM14CS002	Balaji	A
1BM14CS003	Chandan	B
1BM14IS001	Avinash	C
1BM14IS002	Balaji	A

What is the Grade of **Avinash** ?

Which Avinash grade ?
Is it 1BM14CS001 or 1BM14IS001

Here USN will help to **determine** Grade of the student

USN → Grade   Functional Dependency

Note that the attribute to the right of the **arrow** is **functionally dependent** on the attribute in the left of the arrow.

Functional Dependency

- Example

Student Table

USN	Name	Grade
1BM14CS001	Avinash	S
1BM14CS002	Balaji	A
1BM14CS003	Chandan	B
1BM14IS001	Avinash	C
1BM14IS002	Balaji	A

In this table **Name** alone cannot Determine **Grade** of the student
And also **Name** alone does not identify the **entire row** in the table.
So **Name Cannot be PRIMARY KEY.**

What is the Grade of **Avinash** ?

Which Avinash grade ?
Is it 1BM14CS001 or 1BM14IS001

Here USN will help to **determine** Grade of the student

USN → Grade   Functional Dependency

Functional Dependency

- Example

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Grade** of **Avinash** ?

Functional Dependency

- ## Example

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Grade** of **Avinash** ?

Which **USN** Avinash **Garde** and in which **Course** ?

Functional Dependency

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Grade** of **Avinash** ?

Which **USN** Avinash **Grade** and in which **Course** ?

(USN,Course-ID) → Grade

(USN,Course-ID)	Grade
(1BM14CS001,10)	S
(1BM14CS001,20)	A

Here two attributes
(USN,Course-ID) or **(USN,Course-Name)**
Will determine the **Grade**

Functional Dependency

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Mobile** number of **Avinash** ?

Which **USN** Avinash **mobile number ?**

USN ➔ Mobile

USN	Mobile
1BM14CS001	9449189795
1BM14CS002	8441609444

USN ➔ Name

USN	Mobile
1BM14CS001	Avinash
1BM14CS002	Avinash

Functional Dependency

- Example

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Course-Name** whose **Credits** is **6** ?

Functional Dependency

- Example

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

What is the **Course-Name** whose **Credits** is **6** ?

Course-ID ➔ (Course-Name, Credits)

```
Course-ID   (Course-Name, Credits)
---------------------------------------
10      (DBMS,6)
30            (DC,6)
```

Functional Dependency

- Example

Student-Course Table

USN	Name	Mobile	Course -ID	Course -name	Credits	Grade
1BM14CS001	Avinash	9449189795	10	DBMS	6	S
1BM14CS001	Avinash	9449189795	20	Maths	4	A
1BM14CS002	Avinash	8441609444	30	DC	6	A
1BM14CS003	Balaji	7958994491	10	DBMS	6	B
1BM14CS003	Balaji	7958994491	20	Maths	4	C

USN ➔ (Name, Mobile)

(USN,Course-ID) ➔ Grade

Course-ID ➔ (Course-Name, Credits)

Functional dependency

Attribute A	FD	Attribute B
a1	$\longrightarrow$	b3
a2	$\longrightarrow$	b3
a3	$\longrightarrow$	b4

Attribute A	Not FD	Attribute B
a1	$\longrightarrow$	b3
a1	$\longrightarrow$	b2

To DO

- Determine all Functional Dependencies in the following table

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

To Do

A	B	C	D	E
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
1	2	4	3	6
3	2	5	1	8

Find at least *three* FDs which hold on this instance

{ } □ { }
{ } □ { }
{ } □ { }

Functional Dependency

- Given that X, Y, and Z are sets of attributes in a relation R, one can derive several properties of functional dependencies. Among the most important are Armstrong's axioms, which are used in database normalization:
- Subset Property (Axiom of Reflexivity): If Y is a subset of X, then X ? Y
- Augmentation (Axiom of Augmentation): If X -> Y, then XZ -> YZ
- Transitivity (Axiom of Transitivity): If X -> Y and Y -> Z, then X -> Z

- Union: If X -> Y and X -> Z, then X -> YZ
- Decomposition: If X -> YZ, then X -> Y and X -> Z
- Pseudotransitivity: If X -> Y and YZ -> W, then XZ -> W

Unit 3

Super key, Candidate key and Primary key

Recall: Functional Dependency

Functional Dependencies provide a formal mechanism to express Constraints between attributes.

It is a mean of identifying how values of certain attributes are Determined by values of other attributes.

A functional dependency (FD) generalizes the concept of a key.

Book (acc_no, yr_pub, title)

Acc_no is Primary Key

Formal representation of Constraints

acc_no $\longrightarrow$ yr_pub

acc_no $\longrightarrow$ title

Normalization

- Normalization is a process of removing redundancy using functional dependencies.
- To reduce redundancy it is necessary to decompose a relation into a number of smaller relations.
- There are several normal forms
  - First Normal Form (1 NF)
  - Second Normal Form (2 NF)
  - Third Normal Form (3 NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form
  - Fifth Normal Form

To Understand “First Normal form”

To understand “First Normal Form”, we will look at the definitions of Super key, Candidate key and Primary key.

Database Table Keys

- A **Key is a single** or combination of multiple fields in a table.
- Its is used to **fetch or retrieve records**/data-rows from data table according to the condition/requirement.
- Keys are also used to **create relationship among different database tables** or views.

What is Super Key, Candidate Key, Primary Key ?

- **Super Key:** Super key is a set of one or more than one keys that can be used to identify a record uniquely in a table.

  In super key redundant attributes can exist.

- **Candidate Key:** A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple Candidate Keys in one table. Each Candidate Key can work as Primary Key.

  In candidate key no redundant attributes.

- **Primary Key:** Primary key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It can not accept null, duplicate values. Only one Candidate Key can be Primary Key.

  Primary key is one of the instance of Candidate Key

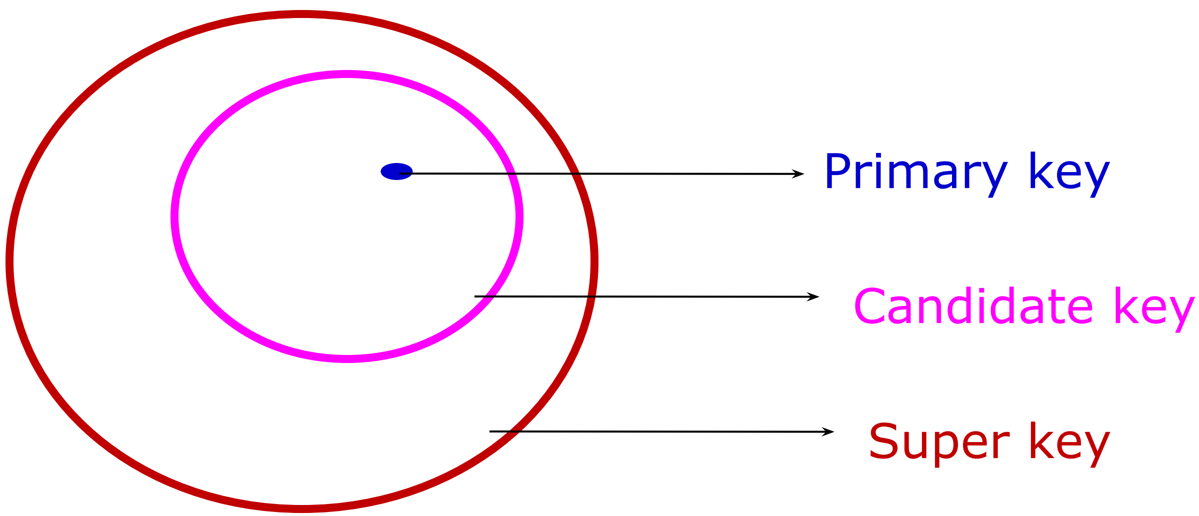
What is Super Key, Candidate Key, Primary Key ?

- **Super Key:** Super key is a set of one or more than one keys that can be used to identify a record uniquely in a table.

  In super key redundant attributes can exist.
- **Candidate Key:** A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple Candidate Keys in one table. Each Candidate Key can work as Primary Key.

  In candidate key no redundant attributes.
- **Primary Key:** Primary key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It can not accept null, duplicate values. Only one Candidate Key can be Primary Key.

  Primary key is one of the instance of Candidate Key



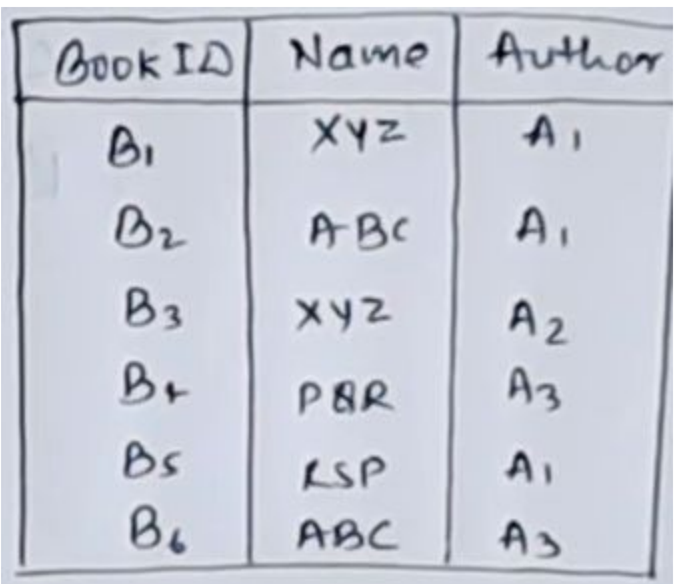
What is Super Key, Candidate Key, Primary Key ?

- Super Key: Specifies that **no two distinct records** have the **same value** for Super Key.
- Super key *may have redundant attributes*.
- Example

Book ID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	LSP	A_1
B_6	ABC	A_3

What is Super Key, Candidate Key, Primary Key ?

- Super Key: Specifies that **no two distinct records** have the **same value** for Super Key.
- Super key *may have redundant attributes*.
- Example



Book ID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	KSP	A_1
B_6	ABC	A_3

Super Keys

{BookID}
{BookID, Name}
{BookID, Author}
{Name, Author}
{BookID, Name,Author}

Super Key

- Super Key: Specifies that no two distinct records have the same value for Super Key.
- Super key may have redundant attributes.
- Example

BookID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	RSP	A_1
B_6	ABC	A_3

Super Keys

{BookID}
{BookID, Name}
{BookID, Author}
{Name, Author}
{BookID, Name,Author}

	Super Key With **redundant Attributes**
{BookID, Name, Author}	Name and Author

Candidate Key

- Candidate Key is a super key without redundancy
- Not reducible
- More than one possible candidate keys

BookID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	RSP	A_1
B_6	ABC	A_3

Candidate Keys

{BookID}
{Name, Author}

Primary Key

- Primary is one of the instance of Candidate Key.
- Primary key chosen by Database designer, which will **not have null** values

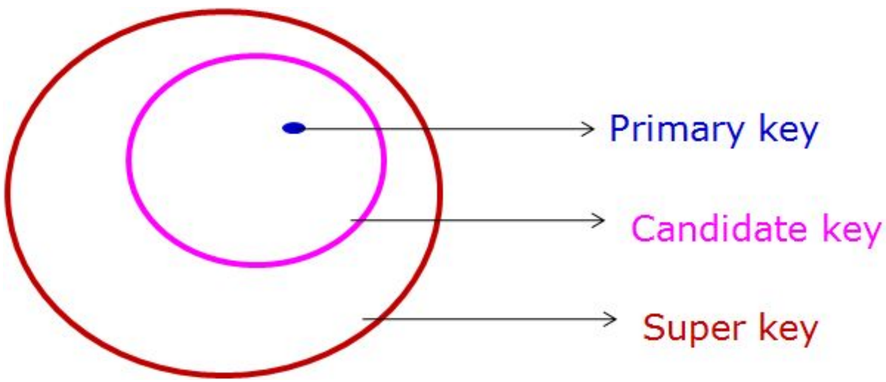
Book ID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	KSP	A_1
B_6	ABC	A_3

Primary Key

{BookID}

Relation between Super, Candidate and Primary key

BookID	Name	Author
B_1	XYZ	A_1
B_2	ABC	A_1
B_3	XYZ	A_2
B_4	PQR	A_3
B_5	KSP	A_1
B_6	ABC	A_3



Super Keys

{BookID}
{BookID, Name}
{BookID, Author}
{Name, Author}
{BookID, Name,Author}

Candidate Keys

{BookID}
{Name, Author}

Primary Key

{BookID}

Summarizing

- Candidate Key: are **individual columns** in a table that qualifies for uniqueness of **each row**.
- Primary Key: is the **column** you choose to maintain uniqueness in a table at **row level**.
- Super Key: If you **add any other Column** to a **Primary Key** then it **become a Super Key**

We will next learn

How to **find candidate keys** from the given functional dependencies

For this we have to understand **Closure of Attributes**

Finding Closure of Attributes

- Consider following functional dependencies

R(A,B,C,D)

$A \rightarrow B$

$B \rightarrow D$

$C \rightarrow B$

Closure of Attribute **A** is written as A^+

Closure of attribute is set
Of attributes which can be
determined by A

Closure of A^+ is {A,B,D}

Finding Closure of Attributes

- Consider following functional dependencies

R(A, B, C, D)

$A \rightarrow B$

$B \rightarrow D$

$C \rightarrow B$

Closure of Attribute **A** is written as A^+

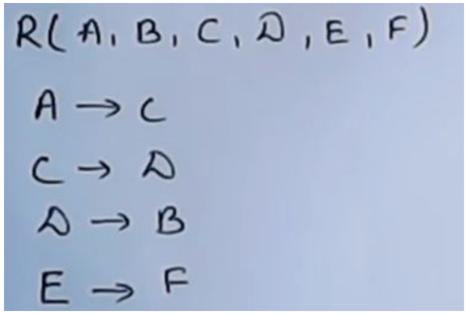
Closure of attribute is set
Of attributes which can be
determined by A

Closure of A^+ is {A,B,D}

What is closure of B and C ?

Finding Candidate Keys

- Find candidate key using following Functional Dependencies



To find a Candidate key, **First try** this approach
Step1: Find the attributes that are neither on the left and right side
Step 2: Find attributes that are only on the right side
Step 3: Find attributes that are only on the left side
Step 4: Combine the attributes on step 1 and 3
Step 5: Test if the closures of attributes on step 4 can determine all the attributes

To find all possible candidate key, **Second try** this approach
Step 1:
(a) Find all Single Attributes closures
A+: B+: C+: D+: E+: F+:
(b) Find all Two Attribute closures
AB+: AC+: AD+: AE+: AF+: BC+: BD+: BE+: BF+: CD+: CE+: CF+: EF+:
(c) Find all Three attribute closures
ABC+: ABD+ ……………………………………………..
(d) Find all Four attribute closures
ABCD+: BCDE+:
(e)Find all Five attribute closures
ABCDE+………………..
Step 2:
Candidate Keys will be: From the above closures above identify which will determine all attributes of the relation and they are minimal i.e. subsets of the attributes is not a key already.

Finding Candidate Keys

Compute following closures and determine which all are candidate keys.
Note: Candidate Keys will be the closures which will determine all attributes of the relation and they are minimal i.e. subsets of the attributes is not a key already.

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

A → C
C → D
D → B
E → F

A+=ACDBF
B+=
C+=
D+=
E+=
F+=
AB+=
AC+=
AD+=
AE+=
AF+=
BC+=
BD+=
BE+=
BF+=
CD+=
CE+=
CF+=
EF+=

ABC+=
ABD+ =
ABE+=
ABF+=
ACD+=
ACE+=
ACF+=
ADE+=
ADF+=
BCD+=
BCE+=
BCF+=
CDE+=
CDF+=
DEF+=

ABCD+=
ABCE+=
ABCF+=
BCDE+=
BCDF+=
CDEF+=
ABCDEF+=ABCDEF

Finding Candidate Keys

Solve the given problems to identify candidate key

1. R{A,B,C,D,E,F,G}
FD's AB->F , AD-> E, F->G

Answer: One possible CK is {ABCD}

2. R{F,O,R,UM}
FD's FO->U , ORU->M

Answer: One possible CK is {FOR}

Trivial Functional Dependency

If Y ⊂ X , Y is **subset** of X
then we have trivial functional dependency
X -> Y

Example:
{BC} ⊂ {ABC} So the trivial functional dependency ABC ->BC
Y ⊂ X X ->Y

Trivial Functional Dependency

Some functional dependencies are said to be trivial because they are satisfied by all relations. Functional dependency of the form A−>B is trivial if $B \subset A$ or A trivial Functional Dependency is the one where **RHS is a subset of LHS**.

Example,

A-->A is satisfied by all relations involving attribute A.

SSN-->SSN

PNUMBER-->PNUMBER

SSN PNUMBER -->PNUMBER

SSN PNUMBER --> SSN PNUMBER

Non Trivial Functional Dependency

If a functional dependency X->Y holds true where Y is **not a subset** of X then this dependency is called non trivial Functional dependency.

- **For example**:
  An employee table with three attributes: emp_id, emp_name, emp_address.
  The following functional dependencies are **non-trivial**:
  emp_id -> emp_name (emp_name is not a subset of emp_id)
  emp_id -> emp_address (emp_address is not a subset of emp_id)

- On the other hand, the following dependencies are **trivial**:
  {emp_id, emp_name} -> emp_name [emp_name is a subset of {emp_id, emp_name}]

Completely non trivial FD:
If a FD X->Y holds true where X intersection Y is null then this dependency is said to be completely non trivial function dependency.

Example: SSN --> Ename, PNUMBER --> PNAME, PNUMBER--> BDATE

Unit 3

Normalization:
First Normal Form
Second Normal Form

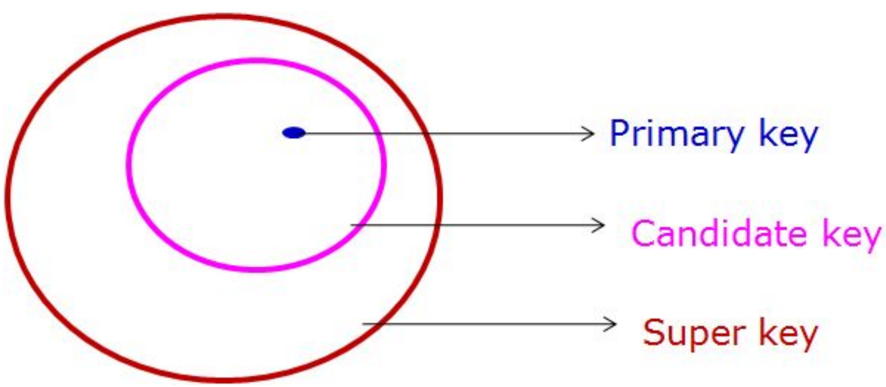
Doubt Clarification on Super and Candidate key

Book ID	Name	Author
B1	xyz	A1
B2	ABC	A1
B3	xyz	A2
B4	PQR	A3
B5	KSP	A1
B6	ABC	A3

Candidate Key: Is Super key without Redundancy and Not reducible

A super key K={A1,A2…….An}
is Candidate Key iff
K-{Ai} is not already a key
for any Ai, 1<=i<=n

Identifying which of the Super keys Can become candidate keys



Super Keys

{BookID}
{BookID, Name}
{BookID, Author}
{Name, Author}
{BookID, Name,Author}

Candidate Keys

{BookID}
{Name, Author}

Various Database table Normal Forms

- First Normal Form (1 NF)
- Second Normal Form (2 NF)
- Third Normal Form (3 NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form
- Fifth Normal Form

Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization:** the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

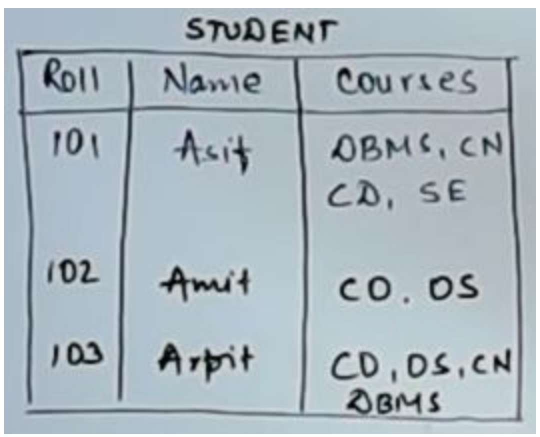
- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- **All entries** in any **column** must be of the **same kind**
- Each **column** must have **a unique name**
- **No two rows** are **identical**

1NF: Disallows composite attributes, multivalued attributes, and **nested relations**; attributes whose values *for an individual record* are non-atomic.

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- **All entries** in any **column** must be of the same **same kind**
- Each **column** must have **a unique name**
- **No two rows** are **identical**


STUDENT

Roll	Name	Courses
101	Asif	DBMS, CN CD, SE
102	Amit	CO. OS
103	Arpit	CD, OS, CN DBMS

Whether this table is in First Normal form ?

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- **All entries** in any **column** must be of the same **same kind**
- Each **column** must have **a unique name**
- **No two rows** are **identical**

STUDENT

Roll	Name	Courses
101	Asif	DBMS, CN CD, SE
102	Amit	CO. OS
103	Arpit	CD, OS, CN DBMS

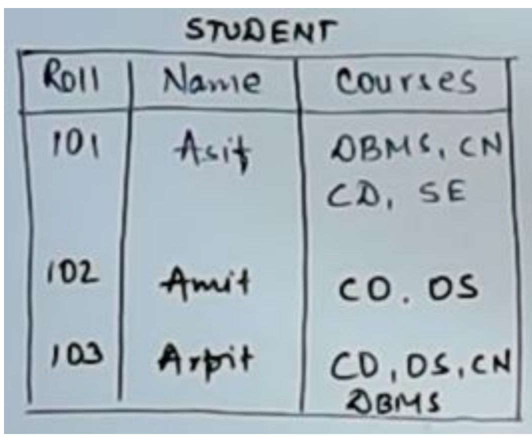
Not atomic values

Whether this table is in First Normal form ?

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- All entries in any column must be of the same kind
- Each column must have a unique name
- **No two rows are identical**


STUDENT

Roll	Name	Courses
101	Asif	DBMS, CN CD, SE
102	Amit	CO. OS
103	Arpit	CD, DS, CN DBMS

1 NF

Roll	Name	Courses
101	Asif	DBMS
101	Asif	CN
101	Asif	CD
101	Asif	SE
102	Amit	CO
102	Amit	OS
103	Arpit	CD
103	Arpit	DS
103	Arpit	CN
103	Arpit	DBMS

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- All entries in any column must be of

the same kind

- Each column must have a unique name
- No two rows are identical

Convert following table to 1 NF

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

1 NF

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- All entries in any column must be of the same kind
- Each column must have a unique name
- No two rows are identical

Convert following table to 1 NF

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

1 NF →

Solution 1

Product ID	Color	Price
1	red	15.99
1	green	15.99
2	yellow	23.99
3	green	17.50
4	yellow	9.99
4	blue	9.99
5	red	29.99

First Normal Form (1 NF)

A database table is said to be in **1 NF** if

- Values of each attribute is **atomic**. An atomic value is a value that **cannot be divided**.
- All entries in any column must be of the same kind
- Each column must have a unique name
- No two rows are identical

Given ProductID is Primary Key

Solution 2

Convert following table to 1 NF

TABLE_PRODUCT

Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

1 NF →

TABLE_PRODUCT_PRICE

Product ID	Price
1	15.99
2	23.99
3	17.50
4	9.99
5	29.99

TABLE_PRODUCT_COLOR

Product ID	Color
1	red
1	green
2	yellow
3	green
4	yellow
4	blue
5	red

We will next learn Second Normal Form

For this we have to understand
Prime and non-prime attribute

Transitive Functional Dependency

Fully functional dependent attribute
Partial functional dependent attribute

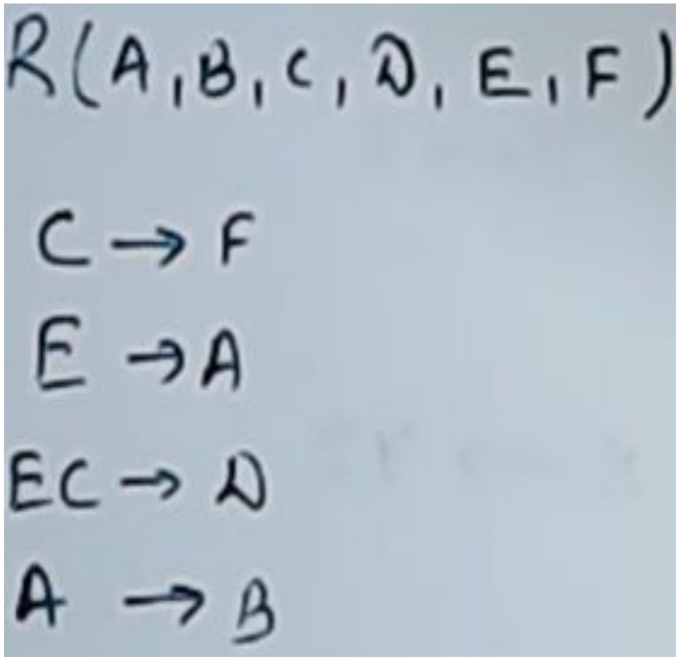
Prime And Non-prime attribute

- Prime attribute: Attribute present in candidate key
- Non-prime attribute: Attribute not present in candidate key

Prime attribute
If attribute is a part of candidate key then
it will be prime attribute

Non-prime attribute,
An attribute which is never included in any candidate key.

Identify prime and non-prime attribute for given functional dependencies



First find Candidate Key

Candidate Key is {CE}

So
Prime attributes are {C, E}

Non-Prime attributes are {A,B,D,F}

Transitive Functional Dependency

$X \rightarrow Y$

$Y \rightarrow Z$

$\Rightarrow$ $X \rightarrow Z$

X transitively determines Z

Fully Functional Dependent Attribute

X → Y

Y is fully functional dependent on X

Only if Y cannot determined by **any proper subset** of X, i.e., removal of any attribute A from X means that the dependency does not hold

Example

We say for the Functional Dependency **ABC □ D**,
D is fully functional dependent on **ABC**

Only if we **do not have** the FD's
{A □ D,
B□D,
C □ D,
AB □ D
BC □ D,
AC □ D,}

Partial Functional Dependent Attribute

X → Y   Y is partially functional dependent on X
Only if Y can be determined by **any proper subset** of X

Note: A functional dependency X -> Y is partial dependency if some attribute A ϵ X can be removed from X and the dependency still holds; that is, for some A ϵ X,(X-{A})-> Y holds

Partial Functional Dependent Attribute

X ➔ Y   Y is partially functional dependent on X
Only if Y can be determined by **any proper subset** of X

Example
We say for the Functional Dependency **AC □ P**,
P attribute is partially functional dependent on **AC**

Only if we **have** the FD's
{A □ P }
or
{C □ P }

Announcement

DBMS Extra Class on
Wednesday, 2-3-2016, 8:55am, Room number 5002

Normal Forms Based on a Primary Key

1 NF: is independent of Primary Key

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

2 NF: A relation schema R is in second normal form (2NF) if it is in **1 NF** and if every **non-prime attribute A in R is fully functionally dependent on the primary key**

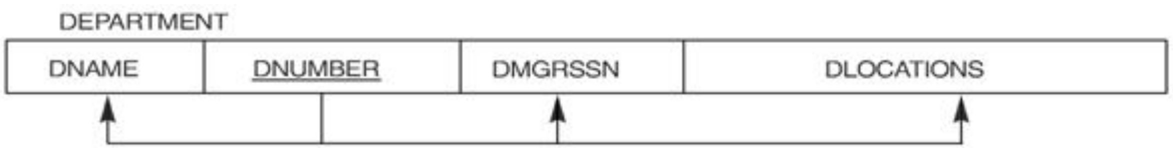
3 NF: A relation schema R is in third normal form (3NF) if it is in **2NF** *and* **no non-prime attribute A in R is transitively dependent on the primary key**

Example: Normalization into 1NF

1 NF: is independent of Primary Key
Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values for an **individual tuple are non-atomic**

1NF: Removes repeating groups

Table Department with
Attributes DNAME, DNUMBER, DLOCATIONS
Primary Key DNUMBER
Functional Dependency DNUMBER -> (DNAME,DMGRSSN,DLOCATIONS)



DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Not atomic values

Not in 1NF

Example: Normalization into 1NF

Approach 1

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Conversion to 1NF

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Disadvantage: Introducing redundancy in the table.

Example: Normalization into 1NF

Approach 2

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Conversion to 1NF

If given DNUMBER is primary Key

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DNUMBER	DLOCATIONS
5	Bellaire
5	Sugarland
5	Houston
4	Stafford
1	Houston

Propagating Primary Key

Advantage: This approach does not suffer from redundancy.

Example: Normalization into 1NF

Approach 3

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Conversion to 1NF

If given that at most maximum
Three department locations can exist

DNAME	DNUMBER	DMGRSSN	**DLOCATIONS 1**	**DLOCATIONS2**	**DLOCATIONS 3**
Research	5	333445555	Bellaire	Sugarland	Houston
Administration	4	987654321	Stafford		
Headquarters	1	888665555	Houston		

Disadvantage: This approach introduces NULL values for most departments having fewer than three locations.

Example: Normalization to 1NF

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, AliciaJ.	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

EMP_PROJ

Ssn	Ename	Pnumber	Hours

Ssn in primary key

To 1NF

EMP_PROJ1

<u>Ssn</u>	Ename

EMP_PROJ2

<u>Ssn</u>	<u>Pnumber</u>	Hours

Example: Normalizing to 2 NF based on Primary Key

2 NF: A relation schema R is in second normal form (2NF)
if it is in 1 NF
and
if every non-prime attribute A in R is fully functionally dependent on the primary key

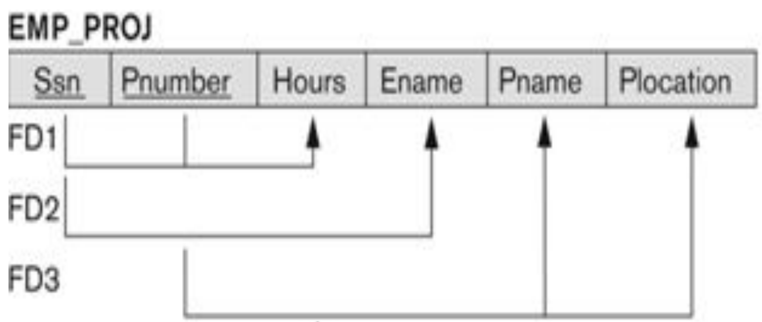
2NF: Removes Partial dependencies

Note: For tables where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of primary key.

Example: Normalizing to 2 NF based on Primary Key

2 NF: A relation schema R is in second normal form (2NF)
if it is in 1 NF
and
if every non-prime attribute A in R is fully functionally dependent on the primary key

Question: Is the following table **EMP_PROJ** satisfies 2 NF



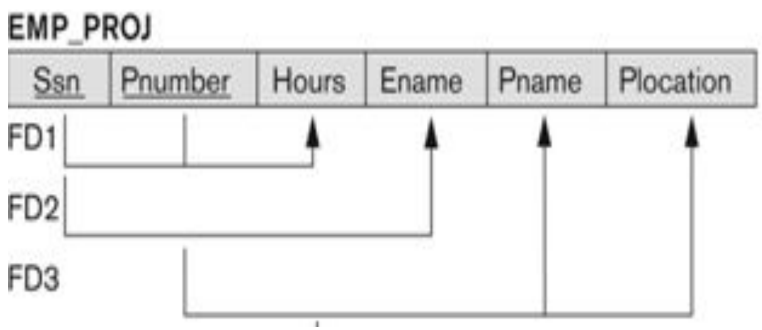
Given
Primary key (Ssn, Pnumber)
FD's
FD 1: (Ssn,Pnumber) ->Hours
FD 2: (Ssn) -> Ename
FD 3: (Pnumber) -> (Pname,Plocation)

Example: Normalizing to 2 NF based on Primary Key

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

Examples of FULL and PARTIAL
{SSN, PNUMBER} -> HOURS is a full FD since neither
SSN -> HOURS nor PNUMBER -> HOURS holds

{SSN, PNUMBER} -> ENAME is *not* a full FD , it is called a *partial dependency*
because SSN -> ENAME also holds

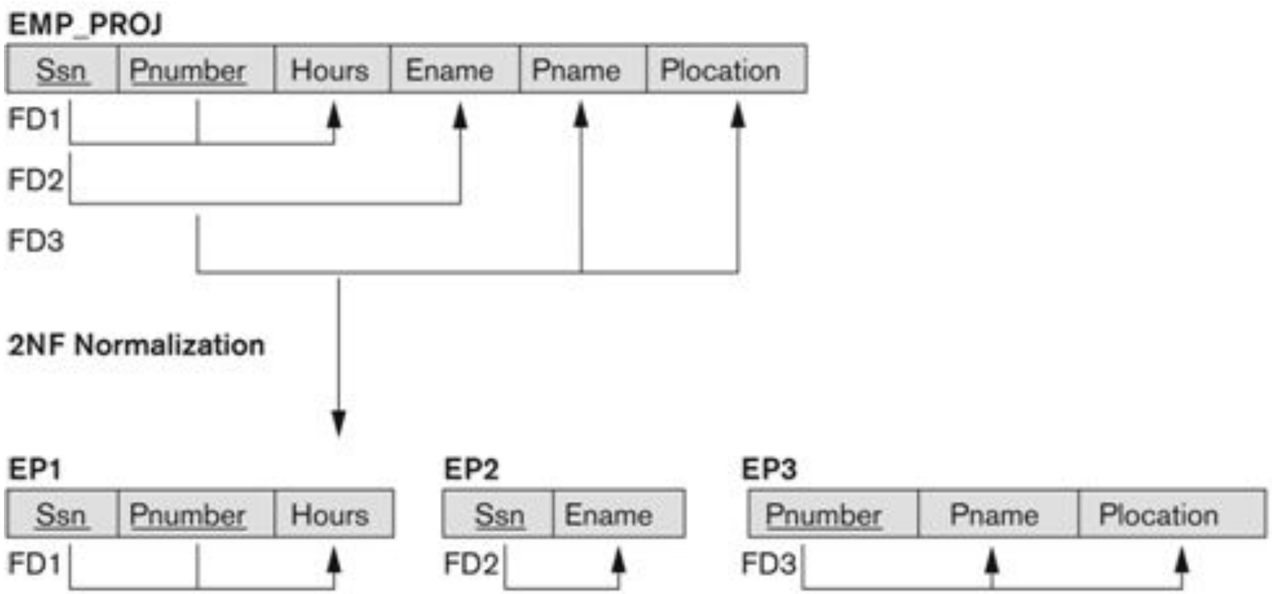


Given
Primary key (Ssn, Pnumber)
FD's
FD 1: (Ssn,Pnumber) ->Hours
FD 2: (Ssn) -> Ename
FD 3: (Pnumber) -> (Pname,Plocation)

Relation EMP_PROJ(Ssn,Pnumber,Hours,Ename,Pname,Plocation) is not in 2 NF
Because in FD 2: (Ssn) -> Ename, Ssn alone can determine Ename
Ssn is part of primary key (Ssn,Pnumber) i.e Ssn proper subset of (Ssn,Ename)
Similarly in FD 3 : (Pnumber) -> (Pname,Plocation), Pnumber is part of primary key (Ssn,Ename)

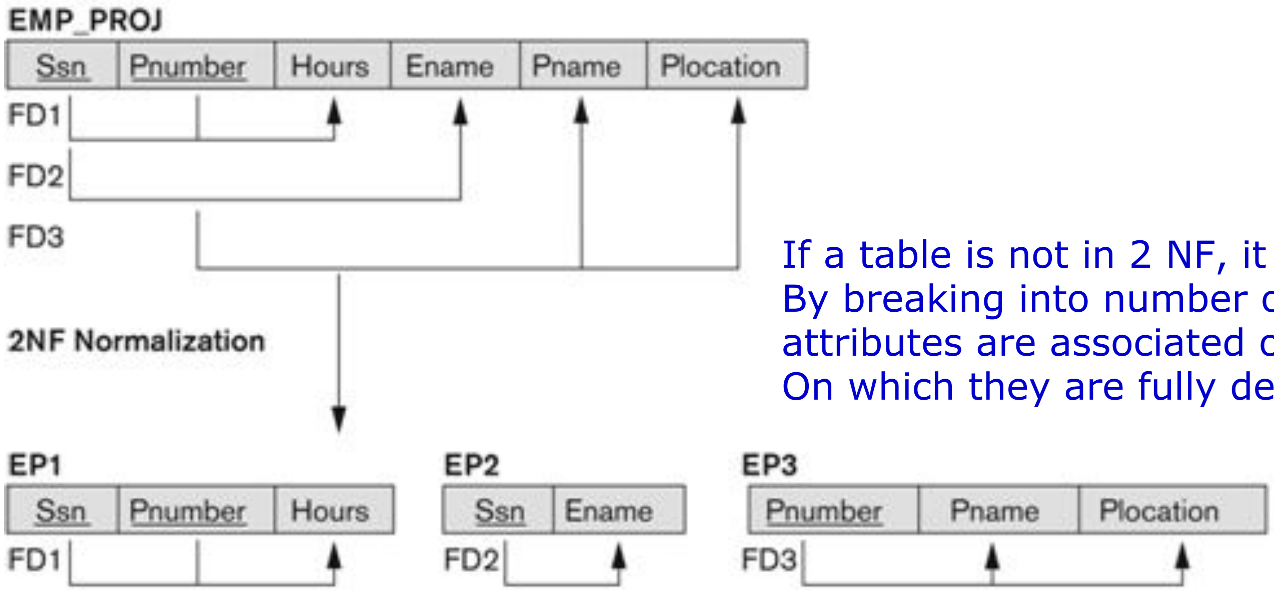
Example: Normalizing to 2 NF based on Primary Key

2 NF: A relation schema R is in **second normal form** (**2NF**) if it is in 1 NF and if every non-prime attribute A in R is fully functionally dependent on the primary key



Example: Normalizing to 2 NF based on Primary Key

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



If a table is not in 2 NF, it can be normalized to 2 NF
By breaking into number of 2 NF tables in which nonprime
attributes are associated only with part of primary key
On which they are fully dependent.

Note:
English words meaning **Table** or **Relation** are one and the same w.r.t. database

Normalizing to 3 NF based on Primary Key

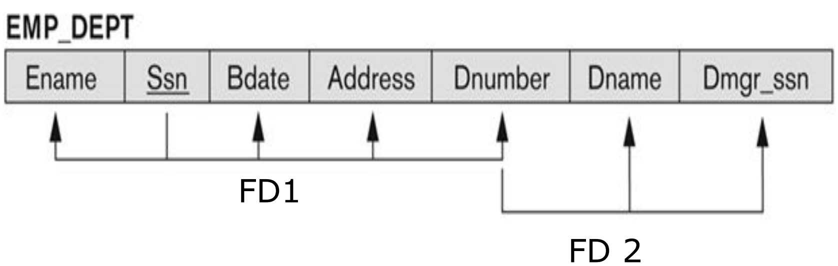
3 NF: A relation schema R is in third normal form (3NF)
if it is in 2NF
and
no non-prime attribute A in R is transitively dependent on the primary key.

3NF: Removes transitive dependencies

Normalizing to 3 NF based on Primary Key

3 NF: A relation schema R is in third normal form (3NF)
if it is in 2NF
and
no non-prime attribute A in R is transitively dependent on the primary key.

Question: In the following table **EMP_PROJ** , what is the transitive dependency that exist on primary key

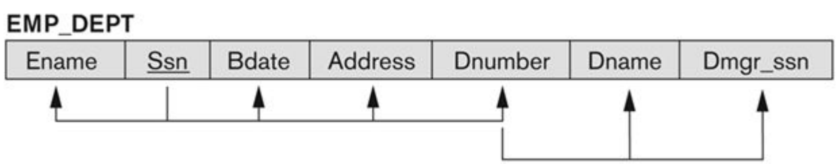


Given
Primary key {Ssn}
FD 1: Ssn -> {Ename,Bdate,Address, Dnumber}
FD 2: Dnumber -> {Dname,Dmgr_ssn}

Normalizing to 3 NF based on Primary Key

3 NF: A relation schema R is in third normal form (3NF)
if it is in 2NF
and
no non-prime attribute A in R is transitively dependent on the primary key.

Question: Is the following table **EMP_PROJ** satisfies 3 NF

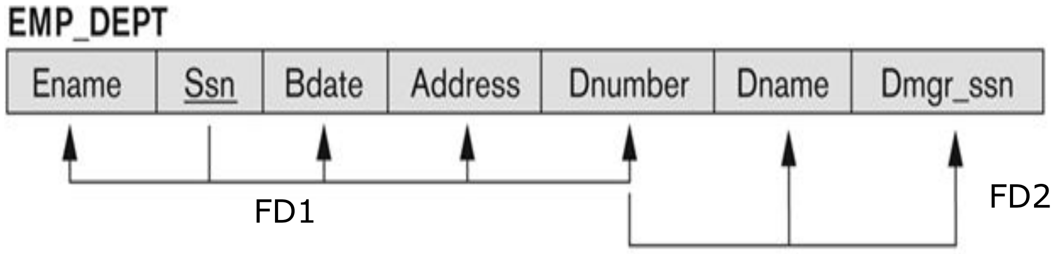


Given
Primary key {Ssn}
FD 1: Ssn -> {Ename,Bdate,Address, Dnumber}
FD 2: Dnumber -> {Dname,Dmgr_ssn}

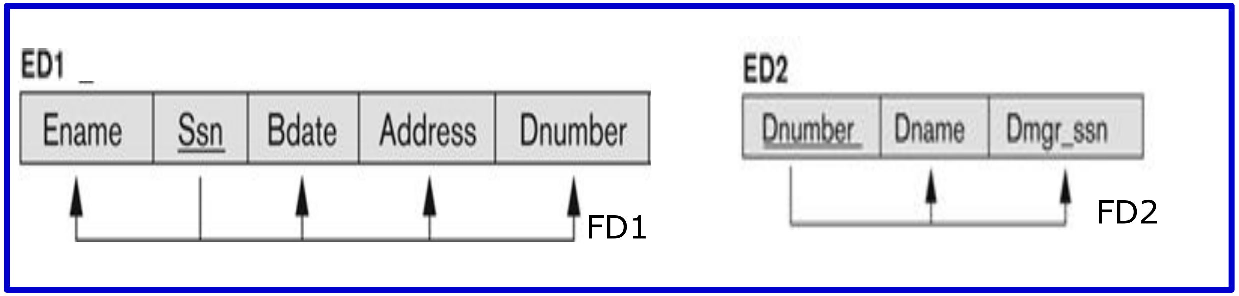
- SSN -> DMGRSSN is a **transitive** FD
  - Since SSN -> DNUMBER and DNUMBER -> DMGRSSN hold
- SSN -> ENAME is **non-transitive**
  - Since there is no set of attributes X where SSN -> X and X -> ENAME

Example: Normalizing to 3 NF based on Primary Key

Given
Primary key {Ssn}
FD 1: Ssn -> {Ename,Bdate,Address, Dnumber}
FD 2: Dnumber -> {Dname,Dmgr_ssn}



Converting to 3 NF



Until now What we have learned

First Normal Form.

Second and **Third** Normal form based on PRIMARY KEY.

Next we will learn **Second** and **Third** Normal form based on ALL CANDIDATE KEYs in a given relation.

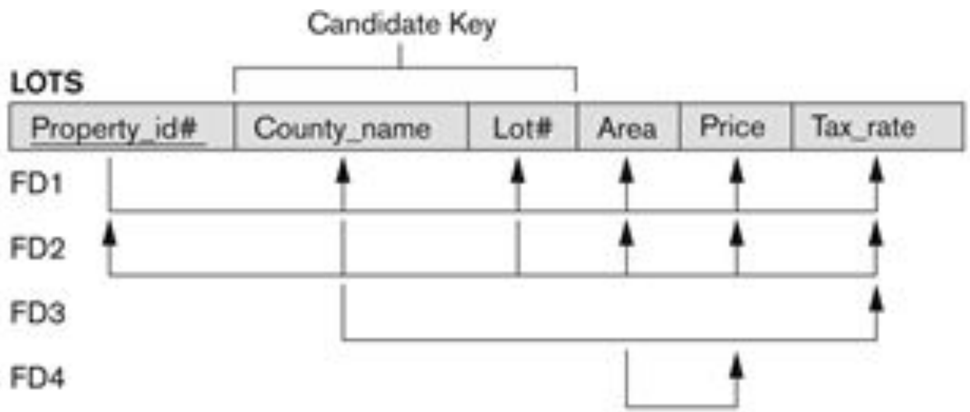
General definition of Second Normal Form

- General definitions of 2NF take into account relations with multiple candidate keys.

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

Example: Normalizing into 2 NF

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



Given
Primary Key {Property_id#}
Candidate key {County_name,Lot#}

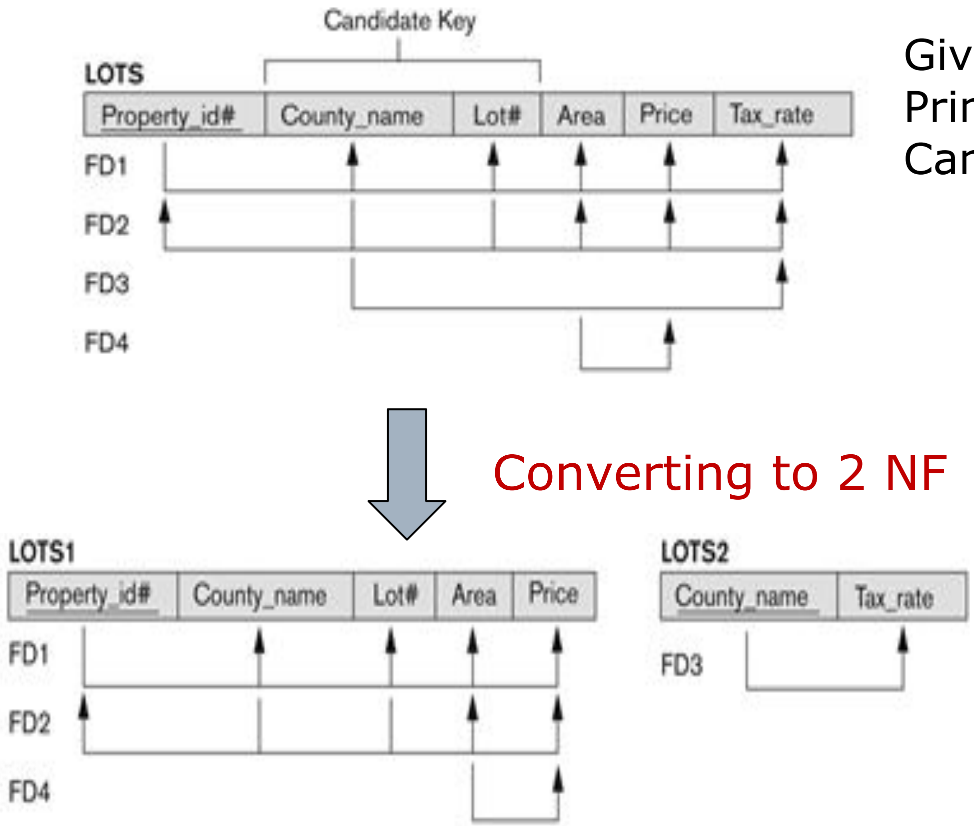
Question

Which attributes are prime, which are nonprime?
Are any nonprime attributes partially dependent on prime attributes?

Example: Normalizing into 2 NF

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

Given
Primary Key {Property_id#}
Candidate key {County_name,Lot#}



General definition of Third Normal Form

- General definition of 3 NF take into account relations with multiple candidate keys.

3 NF: A relation schema R is in 3 NF if **every nonprime attribute** of R meets both of the following conditions:

(a) It is fully functionally dependent on every key of R

(b) It is **non-transitively** dependent on **every key** of R

Note: There should not be any Transitive Functional Dependency, i.e., **there should not be any functional dependencies like a non-key (non-prime) attribute depends on another non-key (non-prime) attributes**. Simply, we need all the non-key attributes must depend on the key only

General definition of Third Normal Form

- General definition of 3 NF take into account relations with multiple candidate keys.

3 NF: A relation schema R is in 3 NF if **every nonprime attribute** of R meets both of the following conditions:

(a) It is fully functionally dependent on every key of R

(b) It is **non-transitively** dependent on **every key** of R

Note: There should not be any Transitive Functional Dependency, i.e., there should not be any functional dependencies like a non-key (non-prime) attribute depends on another non-key (non-prime) attributes. Simply, we need all the non-key attributes must depend on the key only

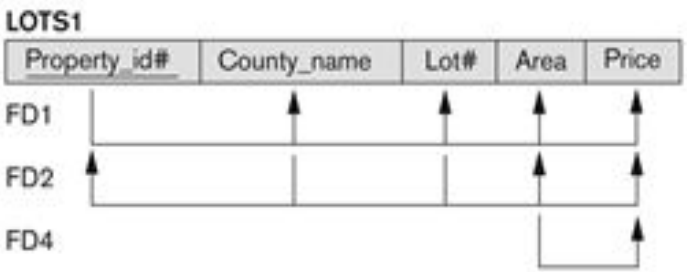
Example: R(A,B, C, D, E). Consider candidate keys as {A}, {BC} with FD's {A->BCDE, BC->ADE, D->E}

Then Non-prime attributes are {D,E}. The functional dependency {D->E} violates the condition of 3NF because D and E are non-prime attributes

Example: Normalizing into 3 NF

3 NF: A relation schema R is in 3 NF if **every nonprime attribute** of R meets both of the following conditions:
(a) It is fully functionally dependent on every key of R
(b) It is **non-transitively** dependent on **every key** of R

Note: There should not be any Transitive Functional Dependency, i.e., there should not be any functional dependencies like a non-key (non-prime) attribute depends on another non-key (non-prime) attributes. Simply, we need all the non-key attributes must depend on the key only



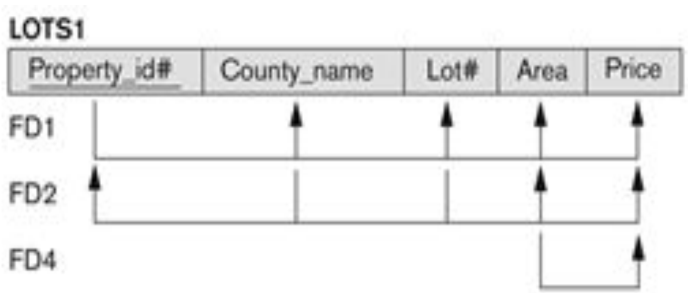
Given
Primary Key {Property_id#}
Candidate key {County_name,Lot#}

Question
Are any nonprime attribute dependent on nonprime attribute?

Example: Normalizing into 3 NF

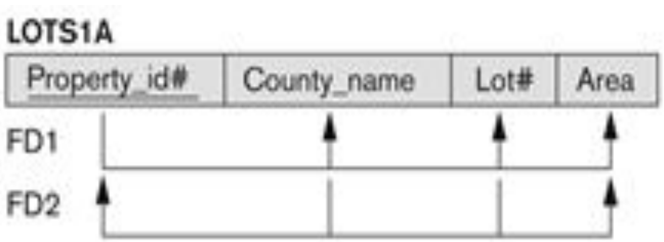
3 NF: A relation schema R is in 3 NF if **every nonprime attribute** of R meets both of the following conditions:
(a) It is fully functionally dependent on every key of R
(b) It is **non-transitively** dependent on **every key** of R

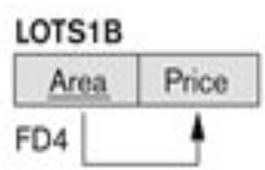
Note: There should not be any Transitive Functional Dependency, i.e., there should not be any functional dependencies like a non-key (non-prime) attribute depends on another non-key (non-prime) attributes. Simply, we need all the non-key attributes must depend on the key only



Given
Primary Key {Property_id#}
Candidate key {County_name,Lot#}

Converting to 3 NF





Summarizing 1NF, 2NF, 3 NF

1NF: Removes repeating groups

1NF
Column values should atomic

First Restriction

↓

2NF: Removes Partial dependencies

2NF
Non-Prime attributes should be Fully Dependent on each candidate key

Second Restriction

↓

3NF: Removes transitive dependencies

3NF
Non-Prime attributes should not Determined by a non-prime attribute

Third Restriction

Summarizing 1NF, 2NF, 3 NF

1NF: No Multivalued attributes or All columns should be atomic

2NF: It should be in **1 NF** and All non-prime attributes should be fully functionally dependent on primary key or each candidate key

3NF:

For relation R to be in 3 NF check

(a) R should be in **2 NF**

(b) No non-prime attribute should be transitively dependent on Candidate key or There should not be the case that a non-prime attribute is determined by another non-prime attribute.

Problem to Solve: Normalizing to 2 NF and 3 NF

Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies

F = {{A, B} → {C},
{A} → {D, E},
{B} → {F},
{F} → {G, H},
{D} → {I, J}}.

FD1	{A, B} → {C}
FD2	{A} → {D, E}
FD3	{B} → {F}
FD4	{F} → {G, H}
FD5	{D} → {I, J}

Given the **Key of R** as {**AB**}

(a) Decompose R into 2NF.

(b) Decompose R into 3NF.

Problem to Solve: Normalizing to 2 NF and 3 NF

Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B} → {C}, {A} → {D, E}, {B} → {F}, {F} → {G, H}, {D} → {I, J}}. Given the Key of R as {AB}

(a) Decompose R into 2NF.

(b) Decompose R into 3NF.

Splitting out attributes based on relations only partially dependent on the key gives:

$$
\begin{array}{lcll}
R_1 & = & \{\underline{A}, D, E, I, J\} & \text{preserves the functional dependencies} \\
 & & & \{\{A\} \rightarrow \{D, E\}, \{D\} \rightarrow \{I, J\}\} \\
R_2 & = & \{\underline{B}, F, G, H\} & \text{preserves } \{\{B\} \rightarrow \{F\}, \{F\} \rightarrow \{G, H\}\} \\
R_3 & = & \{\underline{A}, \underline{B}, C\} & \text{preserves } \{\{A, B\} \rightarrow \{C\}\}
\end{array}
$$

The primary keys of these subrelations are underlined.

Decompose R into 2NF

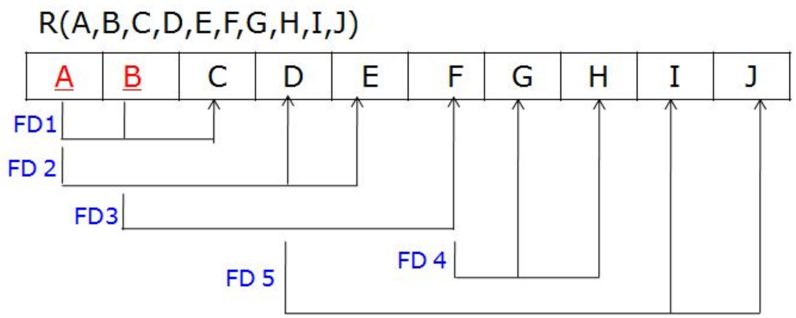
Further splitting attributes with transitive dependencies on their keys gives:

$$
\begin{array}{lcl}
R_{1a} & = & \{\underline{A}, D, E\} \\
R_{1b} & = & \{\underline{D}, I, J\} \\
R_{2a} & = & \{\underline{B}, F\} \\
R_{2b} & = & \{\underline{F}, G, H\} \\
R_3 & = & \{\underline{A}, \underline{B}, C\}
\end{array}
$$

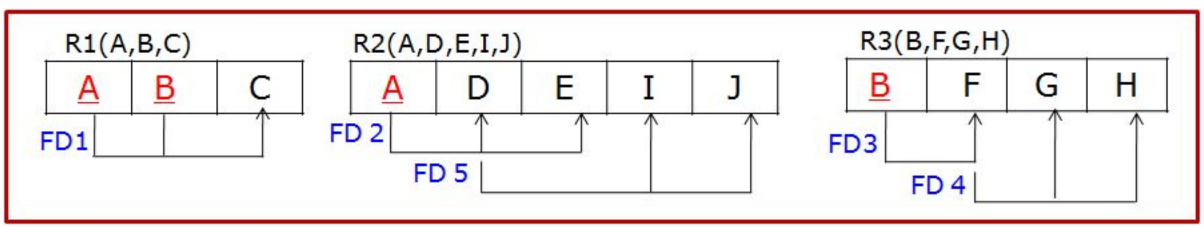
Decompose R into 3NF

Problem to Solve: Normalizing to 2 NF and 3 NF

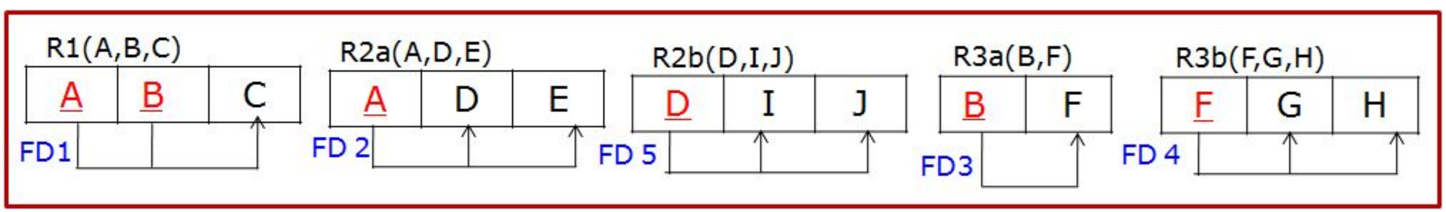
Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B} → {C}, {A} → {D, E}, {B} → {F}, {F} → {G, H}, {D} → {I, J}}. Given the Key of R as {AB} (a) Decompose R into 2NF. (b) Decompose R into 3NF.



Converting to 2 NF



Converting to 3 NF



Problem to Solve: Normalizing to 2 NF and 3 NF

Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies

F = {{A, B} → {C},
{B,D} → {E, F},
{A,D} → {G,H},
{A} → {I},
{H} → {J}}.

Given the **Key of R** as {**ABD**}

(a) Decompose R into 2NF.

(b) Decompose R into 3NF.

Problem to Solve

Given relation R(A,B,C,D,E)

with dependencies

AB -> C

CD -> E

DE -> B

Is AB a candidate key of this relation?

If not, is ABD? Explain your answer.

Problem to Solve

Given relation R(A,B,C,D,E)
with dependencies AB -> C CD -> E DE -> B
is AB a candidate key of this relation?
If not, is ABD? Explain your answer.

Answers:
Closure of AB+ = {A,B,C}
NO, AB is NOT KEY
Because AB cannot determine all attributes, Closure of AB+ is {A,B,C} which is proper subset of {A,B,C,D,E}

Problem to Solve

Given relation R(A,B,C,D,E) with dependencies AB -> C, CD -> E, DE -> B
is AB a candidate key of this relation? If not, is ABD? Explain your answer.

Answers: Yes, ABD+ = {A,B,C,D,E}. Yes, **ABD** is a candidate key. No subset of its attributes is a key.

A -> A
B -> B
C -> C
D -> D
E -> E
AB -> ABC
AC -> AC
AD -> AD
AE -> AE
BC -> BC
BD -> BD
BE -> BE
CD -> BCDE
CE -> CE
DE -> BDE
ABD -> ABCDE

Problem to Solve

Consider the following relation:

A	B	C	TUPLE#
10	b1	c1	#1
10	b2	c2	#2
11	b4	c1	#3
12	b3	c4	#4
13	b1	c1	#5
14	b3	c4	#6

(a) Given the above extension (state), which of the following dependencies may hold in the above relation? If the dependency cannot hold, explain why by specifying the tuples that cause the violation.

i. A -> B, ii. B -> C, iii. C -> B, iv. B -> A, v. C -> A

Problem to Solve

A	B	C	TUPLE#
10	b1	c1	#1
10	b2	c2	#2
11	b4	c1	#3
12	b3	c4	#4
13	b1	c1	#5
14	b3	c4	#6

- **1.**
  - **A→B NO**
    For A=10, B=b1,b2 ***Ex: tuple 1,2***
  - **A→C NO**
    For A=10, C=c1,c2 ***Ex: tuple 1,2***
  - **B→C Yes**
    For B=b4, C=c1 ; B=b1, C=c1 ***Ex: tuple 3,5 (or Ex: tuple 4,6)***
  - **C→B NO**
    For C=C1 , B=b1,b4 ***Ex: tuple 1,3***
  - **B→A NO**
    For B=b3 , A=12,14 ***Ex: tuple 4,6***
  - **C→A NO**
    For C=c1, A=10,11,13 ***Ex: tuple 1,3,5***

  - ***AB*** **→C YES**
  - ***AC*** **→B YES**
  - **BC →A NO For (b1,c1)->10 ; (b1,c1) -> 13 Tuple 1,5**

Problem to Solve

Consider a relation with schema R(A, B, C, D)

and

FD = $\{AB \rightarrow C,$

$C \rightarrow D,$

$D \rightarrow A\}$

(a) What are all the non-trivial FDs that follow from the given FD's?

(b) What are all the candidate keys of R?

(c) What are all the superkeys of R that are not candidate keys?

Non Trivial Functional Dependency

If a functional dependency X->Y holds true where Y is **not a subset** of X then this dependency is called non trivial Functional dependency.

- **For example**:
  An employee table with three attributes: emp_id, emp_name, emp_address.
  The following functional dependencies are **non-trivia**l:
  emp_id -> emp_name (emp_name is not a subset of emp_id)
  emp_id -> emp_address (emp_address is not a subset of emp_id)

- On the other hand, the following dependencies are **trivial**:
  {emp_id, emp_name} -> emp_name [emp_name is a subset of {emp_id, emp_name}]

Completely non trivial FD:
If a FD X->Y holds true where **X intersection Y is null** then this dependency is said to be completely non trivial function dependency.

Example: SSN --> Ename, PNUMBER --> PNAME, PNUMBER--> BDATE

Answers

Consider a relation with schema R(A, B, C, D) and FD = {AB → C, C → D, D → A}

(a) What are all the non-trivial FDs that follow from the given FD's?

We need to compute the closures of all 15 nonempty sets of attributes.

Single Attributes:
A+ = A
B+ = B
C+ = ACD (New dependency: C → A)
D+ = AD

Pairs of Attributes:
AB+ = ABCD (New dependency: AB → D)
AC+ = ACD (New dependency: AC → **D**)
AD+ = AD
BC+ = ABCD (New dependencies: BC → A and BC→D)
BD+ = ABCD (New dependencies: BD → A and BD→C)
CD+ = ACD (New dependencies: CD → A)

Triples of Attributes:
ABC+ = ABCD (New dependencies: ABC → D)
ACD+ = ACD
BCD+ = ABCD (New dependencies: BCD → A)
ABD+ = ABCD (New dependencies: ABD → C)

All the Attributes:
ABCD+ = ABCD

So, we get a total of 11 non-trivial FDs.

Answers

(b) What are all the candidate keys of R?

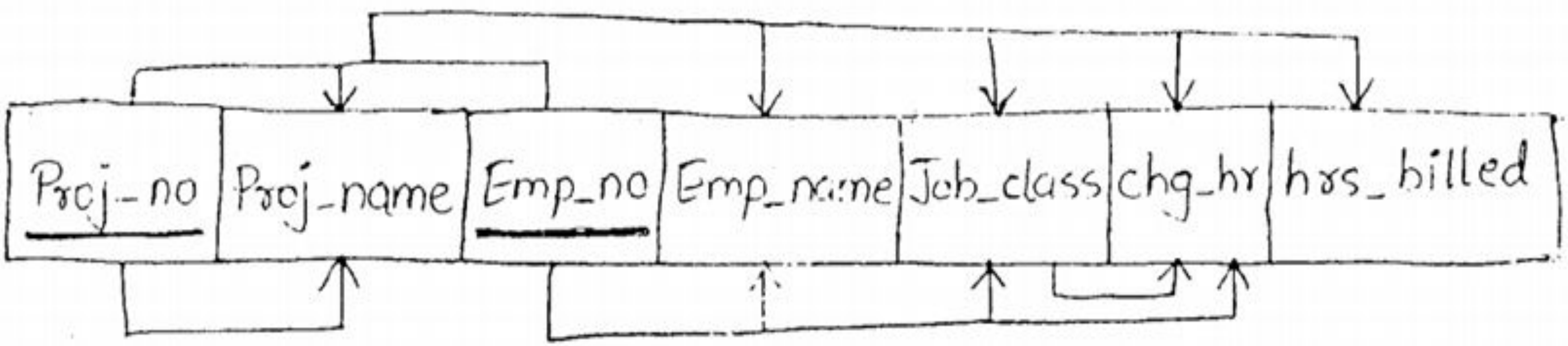
From the closures above, we find that **AB, BC, and BD** are keys, because they have ABCD as the closure, and they are minimal.

(c) What are all the superkeys of R that are not candidate keys?

The superkeys are all those that contain one of those three candidate keys. Thus, the **superkeys** are **ABC, ABD, BCD, and ABCD.**

Problem to Solve

Consider a dependency diagram of relation R and normalize it up to third normal form.



Why we remove Partial Functional Dependency in second Normal form ?

Because **Partial Functional Dependency** is "BAD"

Why we remove Partial functional Dependency in second Normal form ?

Student(USN,Subject,Grade,Name,DOB)

USN	Subject	Grade	Name	DOB
1BM14CS001	DBMS	S	Avinash	14/11/1997
1BM14CS001	OS	B	Avinash	14/11/1997
1BM14CS002	DBMS	A	Balaji	11/03/1998
1BM14CS002	OS	C	Balaji	11/03/1998
1BM14CS003	DBMS	B	Mohan	21/08/1996

Primary Key (USN,Subject)
Full Functional Dependency (USN,Subject) -> (Grade)
Partial Functional Dependency (USN) -> (Name,DOB)

Why we remove Partial functional Dependency in second Normal form ?

Student(USN,Subject,Grade,Name,DOB)

<u>USN</u>	<u>Subject</u>	Grade	Name	DOB
1BM14CS001	DBMS	S	Avinash	14/11/1997
1BM14CS001	OS	B	Avinash	14/11/1997
1BM14CS002	DBMS	A	Balaji	11/03/1998
1BM14CS002	OS	C	Balaji	11/03/1998
1BM14CS003	DBMS	B	Mohan	21/08/1996

Primary Key (USN,Subject)
Full Functional Dependency (USN,Subject) -> (Grade)
Partial Functional Dependency (USN) -> (Name,DOB)

Student(<u>USN</u>,<u>Subject</u>,Grade,Name,DOB)

Converting To 2NF

Student_1(<u>USN</u>,Name,DOB)

Student_2(<u>USN</u>,<u>Subject</u>,Grade)

Why we remove Partial functional Dependency in second Normal form ?

Student(USN,Subject,Grade,Name,DOB)

<u>USN</u>	<u>Subject</u>	Grade	Name	DOB
1BM14CS001	DBMS	S	Avinash	14/11/1997
1BM14CS001	OS	B	Avinash	14/11/1997
1BM14CS002	DBMS	A	Balaji	11/03/1998
1BM14CS002	OS	C	Balaji	11/03/1998
1BM14CS003	DBMS	B	Mohan	21/08/1996

Redundant Name and DOB

Converting To 2NF

<u>USN</u>	Name	DOB
1BM14CS001	Avinash	14/11/1997
1BM14CS002	Balaji	11/03/1998
1BM14CS003	Mohan	21/08/1996

<u>USN</u>	<u>Subject</u>	Grade
1BM14CS001	DBMS	S
1BM14CS001	OS	B
1BM14CS002	DBMS	A
1BM14CS002	OS	C
1BM14CS003	DBMS	B

Why we remove Transitive Functional Dependency in Third Normal form ?

Because **Transitive Functional Dependency** is “BAD”

Why we remove Transitive Functional Dependency in Third Normal form ?

Project(Project-Num, Manager, Mgr_City)

Project-Num	**Manager**	**Mgr_City**
P1	Avinash	Bangalore
P2	Avinash	Bangalore
P3	Balaji	Chennai
P4	Balaji	Chennai
P5	Mohan	Mumbai

Primary Key (Project-Num)

Functional Dependency (Project-Num) -> (Manager, Mgr_City)
Functional Dependency (Manager) -> (Mgr_City)   non-prime attribute determining non-prime

Why we remove Transitive Functional Dependency in Third Normal form ?

Project(Project-Num, Manager, Mgr_City)

Project-Num	**Manager**	**Mgr_City**
P1	Avinash	Bangalore
P2	Avinash	Bangalore
P3	Balaji	Chennai
P4	Balaji	Chennai
P5	Mohan	Mumbai

Primary Key (Project-Num)

Functional Dependency (Project-Num) -> (Manager, Mgr_City)
Functional Dependency (Manager) -> (Mgr_City) non-prime attribute determining non-prime

Project(Project-Num, Manager, Mgr_City)

Converting To 3NF

Project(Manager, Mgr_City)

Project(Project-Num, Manager)

Why we remove Transitive Functional Dependency in Third Normal form ?

Project(Project-Num, Manager, Mgr_City)

<u>Project-Num</u>	**Manager**	**Mgr_City**
P1	Avinash	Bangalore
P2	Avinash	Bangalore
P3	Balaji	Chennai
P4	Balaji	Chennai
P5	Mohan	Mumbai

Redundant Mgr_City

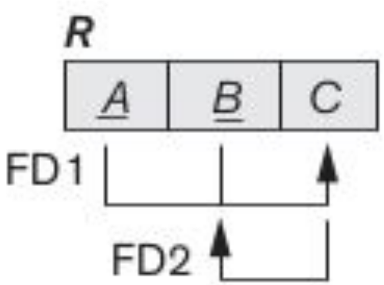
Converting To 3NF

Manager	**Mgr_City**
Avinash	Bangalore
Balaji	Chennai
Mohan	Mumbai

<u>Project-Num</u>	**Manager**
P1	Avinash
P2	Avinash
P3	Balaji
P4	Balaji
P5	Mohan

Boyce-Codd Normal Form

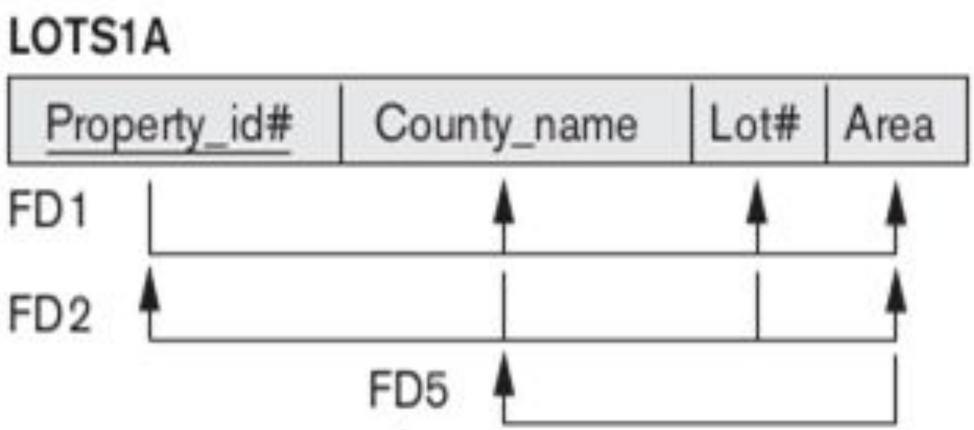
A relation schema R is in *BCNF* if whenever a *nontrivial* functional dependency X→A holds in R, then X is a superkey/candiate key of R.



Relation R(A,B,C)
With key (AB) is in 3 NF but not in BCNF
Because in FD, C->B , C is not a super key

Boyce-Codd Normal Form

A relation schema R is in *BCNF* if whenever a *nontrivial* functional dependency X→A holds in R, then X is a superkey/candiate key of R.

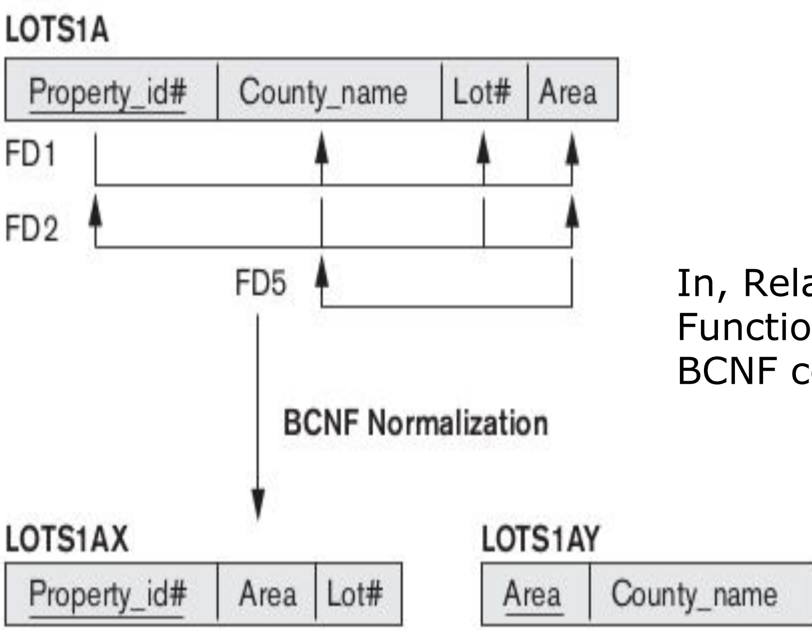


Given
Primary Key {Property_id#}
Candidate key {County_name,Lot#}

Question
Which Function Dependency Violates BCNF in the relation LOTS1A ?

Boyce-Codd Normal Form

A relation schema R is in *BCNF* if whenever a *nontrivial* functional dependency X→A holds in R, then X is a superkey/candiate key of R.



In, Relation LOTS1A(Property_id#,County_name, Lot#,Area) Functional Dependency, Area -> County_name, violates BCNF condition because Area is not Super Key

Problem To Solve:
Check whether the given FD's satisfies BCNF

Consider a relation with schema R(A,B,C,D) and FDs {AB -> CD,

D -> A}.

Candidate keys are: AB, BC, and BD

List out among the give FD's which violates BCNF for R

D->A, because D is not Super key

Why in BCNF we remove functional dependency X-> Y, if X is not a Key ?

Problems BCNF overcomes

BCNF: Every determinant is a candidate key FD: X-> Y, X is determinant

StudMajor(StudNo,Major,Advisor)

StudNo	Major	Advisor
123	PHYSICS	Eshwar
123	MUSIC	Mohan
456	BIOLOGY	Dinesh
789	PHYSICS	Balaji
999	PHYSICS	Eshwar

Candidate Key (StudNo,Major)

FD's
(StudNo,Major) -> Advisor
(Advisor) -> Major

Problems BCNF overcomes

BCNF: Every determinant is a candidate key FD: X-> Y, X is determinant

StudMajor(StudNo,Major,Advisor)

StudNo	Major	Advisor
123	PHYSICS	Eshwar
123	MUSIC	Mohan
456	BIOLOGY	Dinesh
789	PHYSICS	Balaji
999	PHYSICS	Eshwar

Candidate Key (StudNo,Major)

FD's
(StudNo,Major) -> Advisor
(Advisor) -> Major

ANOMALIES
Deleting student deletes advisor info
Insert a new advisor – need a student
Update – inconsistencies

Problems BCNF overcomes

BCNF: Every determinant is a candidate key   FD: X-> Y, X is determinant

StudMajor(StudNo,Major,Advisor)

StudNo	**Major**	**Advisor**
123	PHYSICS	Eshwar
123	MUSIC	Mohan
456	BIOLOGY	Dinesh
789	PHYSICS	Balaji
999	PHYSICS	Eshwar

Candidate Key (StudNo,Major)

FD's
(StudNo,Major) -> Advisor
(Advisor) -> Major

If the record for student 456 is deleted we lose not only information on student 456 but also the fact that Dinesh advises in BIOLOGY.
We cannot record the fact that Vignesh can advise on COMPUTING until we have a student majoring in COMPUTING to whom we can assign Vignesh as an advisor.

Problems BCNF overcomes

StudNo	Major	Advisor
123	PHYSICS	Eshwar
123	MUSIC	Mohan
456	BIOLOGY	Dinesh
789	PHYSICS	Balaji
999	PHYSICS	Eshwar

Candidate Key (StudNo,Major)

FD's
(StudNo,Major) -> Advisor
(Advisor) -> Major

In BCNF we have two tables:

StudNo	Advisor
123	Eshwar
123	Mohan
456	Dinesh
789	Balaji
999	Eshwar

Advisor	Major
Eshwar	PHYSICS
Mohan	MUSIC
Dinesh	BIOLOGY
Balaji	PHYSICS

Next we will learn

Fourth Normal Form based on Multivalued Dependencies

Fourth Normal Form

- It should be in BCNF
- There should not be any **Multivalued Dependency**.

What is **Multivalued Dependency ?**

Multivalued Dependency is a kind of dependency which comes when their is two 1:N relationship in single table

Name	**Skill**	**Laguage**
Smitha	{Cooking, Typing}	{Kannada, Hindi, English

1 NF

Name	**Skill**	**Language**
Smitha	Cooking	Kannada
Smitha	Cooking	Hindi
Smitha	Cooking	English
Smitha	Typing	Kannada
Smitha	Typing	Hindi
Smitha	Typing	English

Multivalued Dependency (MVD)

Name ->-> Skill

Name ->-> Language

What is **Multivalued Dependency ?**

Multivalued Dependency is a kind of dependency which comes when their is two 1:N relationship in single table

Name	**Skill**	**Laguage**
Smitha	{Cooking, Typing}	{Kannada, Hindi, English

1 NF

Name	**Skill**	**Language**
Smitha	Cooking	Kannada
Smitha	Cooking	Hindi
Smitha	Cooking	English
Smitha	Typing	Kannada
Smitha	Typing	Hindi
Smitha	Typing	English

Multivalued Dependency (MVD)
Name ->-> Skill
Name ->-> Language

Problem
w.r.t Insertion or Deletion of Skill or Language for Smitha

Normalizing to Fourth Normal Form

Name	Skill	Language
Smitha	Cooking	Kannada
Smitha	Cooking	Hindi
Smitha	Cooking	English
Smitha	Typing	Kannada
Smitha	Typing	Hindi
Smitha	Typing	English

Multivalued Dependency (MVD)

Name ->-> Skill

Name ->-> Language

4 NF

Name	Skill
Smitha	Cooking
Smitha	Typing

Name	Language
Smitha	Kannada
Smitha	Hindi
Smitha	English

Next we will learn **Fifth Normal Form**

To understand Fifth Normal Form, We should know **JOIN DEPENDENCY**

Decompositions in General

$$R(A_1,\ldots,A_n,B_1,\ldots,B_m,C_1,\ldots,C_p)$$

$$R_1(A_1,\ldots,A_n,B_1,\ldots,B_m) \qquad R_2(A_1,\ldots,A_n,C_1,\ldots,C_p)$$

R_1 = the *projection* of R on $A_1, \ldots, A_n, B_1, \ldots, B_m$

R_2 = the *projection* of R on $A_1, \ldots, A_n, C_1, \ldots, C_p$

Theory of Decomposition

Product

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Sometimes a decomposition is "correct"

i.e. it is a **<u>Lossless decomposition</u>**

Product 1

Name	Price
Gizmo	19.99
OneClick	24.99
Gizmo	19.99

Product 2

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossy Decomposition

However sometimes it isn't

What's wrong here?

Product

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Product 1

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Product 2

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Question

Marks

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

SGM

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

AM

Assignment	Mark
A1	80
A2	60
A1	60

Joining two tables SGM and AM Produces Spurious tuples or not ?

Answer

SGM

Student	Group	Mark
Ann	G1	80
Ann	G3	60
Bob	G2	60

AM

Assignment	Mark
A1	80
A2	60
A1	60

Joining on marks

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Ann	A1	G3	60
Bob	A2	G2	60
Bob	A1	G2	60

ORGINAL Table

Marks

Student	Assignment	Group	Mark
Ann	A1	G1	80
Ann	A2	G3	60
Bob	A1	G2	60

What is Join Dependency ?

- Join Dependency: Denoted by JD(R1, R2, ...Rn) specified on relation schema R specifies a constraint on the states r of R.
  - Every legal state r of R should have non additive join decomposition in to R

Example Join Dependency

R

supp	parts	proj
S_1	P_1	r_1
S_1	P_2	r_2
S_2	P_1	r_1
S_2	P_1	r_2

Supplier S1 supplies part P1 to project r1 and part P2 Project r2

Decomposing

R_1

supp	parts
S_1	P_1
S_1	P_2
S_2	P_1

R_2

supp	proj
S_1	r_1
S_1	r_2
S_2	r_1
S_2	r_2

R_3

parts	proj
P_1	r_1
P_1	r_2
P_2	r_2

Example Join Dependency

R1

supp	parts
S1	P1
S1	P2
S2	P1

R2

supp	proj
S1	r1
S1	r2
S2	r1
S2	r2

R3

parts	proj
P1	r1
P1	r2
P2	r2

Joining any two R1 & R3 tables

Original Table

R

supp	parts	proj
S1	P1	r1
S1	P2	r2
S2	P1	r1
S2	P1	r2

Table after joining

supp	parts	pro.
S1	P1	r1
S1	P1	r2
S1	P2	r2

ADDITIVE JOIN
Because it Generates spurious tuples

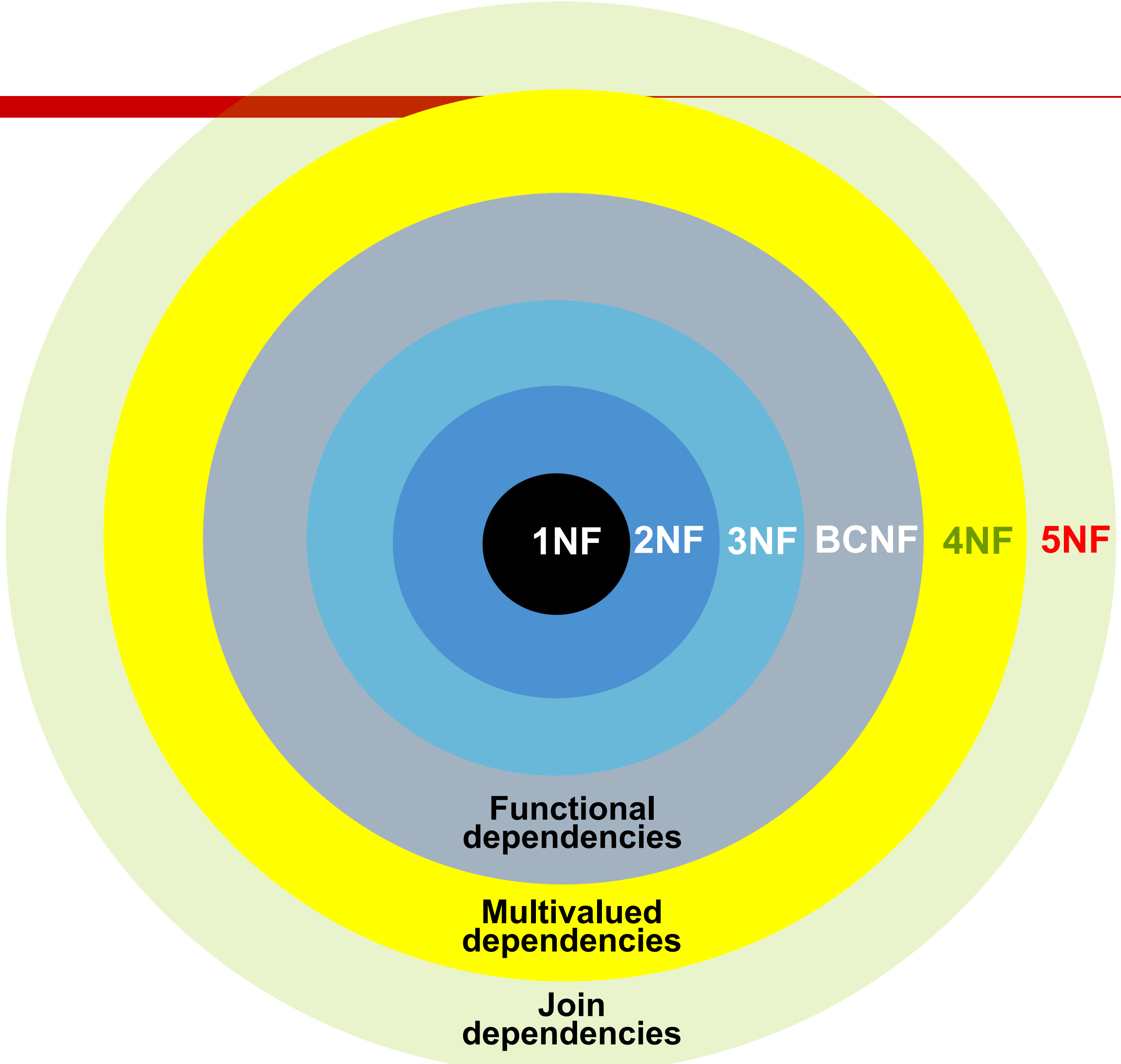
Fifth Normal Form

- 5NF requires that there are no non-trivial join dependencies that not follow from the key constraints.
- A table is said to be in the 5NF if and only if it is in 4NF and every join dependency in it is implied by the candidate keys.

- It should be in fourth normal form
- If Join dependency does not exist then it will be in 5NF
- Else if Join Dependency Exist
  - If Only trivial Join dependency then it is in 5NF
  - Else
    - If (All Ri is superkey R=(R1,R2,…..Rn) then it is in fifth normal form
    - Else not in 5NF

Trivial JD: When R is broken into R1,R2,R3 and one of the Ri is R it self

Normal Forms



Summarizing 1NF, 2NF, 3 NF,BCNF, 4NF, 5NF

Removes	Condition	Restriction
1NF: Removes repeating groups	**1NF**Column values should be atomic	First Restriction
2NF: Removes Partial dependencies	**2NF** Non-Prime attributes should be Fully Dependent on each candidate key	Second Restriction
3NF: Removes transitive dependencies	**3NF** Non-Prime attributes should not be Determined by a non-prime attribute	Third Restriction
BCNF: Make Sure every Determinant is a candidate key	**BCNF** Determinant should be key	Fourth Restriction
4NF: Removes Multivalued dependencies	**4NF** No more than two multi valued dependency in a single table	Fifth Restriction
5NF: Removes join dependencies	**5NF:** A table is said to be in the 5NF if and only if it is in 4NF and every join dependency in it is implied by the candidate keys.	Sixth Restriction

A brief history of normal forms:

- First,Second,Third Normal Forms (1NF,2NF,3NF) (Codd 1972)
- Boyce-Codd Normal Form (BCNF) (1974)
- Fourth Normal Form (4NF) (Zaniolo 1976, Fagin 1977)
- Fifth Normal Form (5NF) (Fagin 1979)

- NF hierarchy: 5NF $\Rightarrow$4NF $\Rightarrow$BCNF $\Rightarrow$3NF $\Rightarrow$2NF $\Rightarrow$1NF

- 1NF allows most redundancy; 5NF allows least redundancy.

Normal Forms – Basic Rules and Solutions

First Normal Form (1NF)

- A relation schema R is in 1NF if and only if,
- All the attributes of the relation are atomic (indivisible into meaningful sub parts),
- Every attribute contains single value (per record).

How to convert un-normalized table into 1NF normalized table?

- Expand the table by duplicating records for every value of multi-valued attributes (Flatten the table).

Normal Forms – Basic Rules and Solutions

Second Normal Form (2NF)

A relation schema R is in 2NF if and only if,

- At the first place the table is in 1NF,
- All the non-key attributes of the table are fully functionally dependent on the Primary key of the table.

How to convert un-normalized or 1NF table into 2NF normalized table?

- It can be done using decomposition. That is, by breaking Non-2NF relations into smaller tables using the Functional Dependencies derived.

Normal Forms – Basic Rules and Solutions

Third Normal Form (3NF)

A relation schema R is in 3NF if and only if,

- The table is in 2NF,
- There is no Functional Dependency such that both Left Hand Side and Right Hand Side attributes of the FD are non-key attributes. In other words, no transitive dependency is allowed.

How to convert un-normalized or 2NF table into 3NF normalized table?

- It can be done using decomposition. That is, by breaking Non-3NF relations into smaller tables using the Functional Dependencies derived. Especially, by creating a separate table for non-key attributes dependencies.

Normal Forms – Basic Rules and Solutions

Boyce-Codd Normal Form (BCNF)

A relation schema R is in BCNF if an only if,
The table is in 3NF

- For all the non-trivial FDs held on R, the left hand side of those non-trivial FDs must be Super Keys.

Thanks for Listening