

Normalization



Functional dependency (FD), Closure of FD, Closure of Attributes, Cover, Equivalence of FD, Canonical cover, Key generation, **Normalization, Desirable properties of decomposition**

Normalization

Normalization in DBMS is a technique using which you can **organize the data in the database** tables so that:

- There is **less repetition of data**,
- A **large set of data is structured** into a **bunch of smaller tables**,
- and the tables have a **proper relationship between** them.
- **Database normalization** can essentially be defined as the practice of **optimizing table structures**. Optimization is accomplished as a result of a thorough **investigation of the various pieces of data** that will be stored within the database, in particular **concentrating upon how this data is interrelated**.

Why we need Normalization

Normalization is required for,

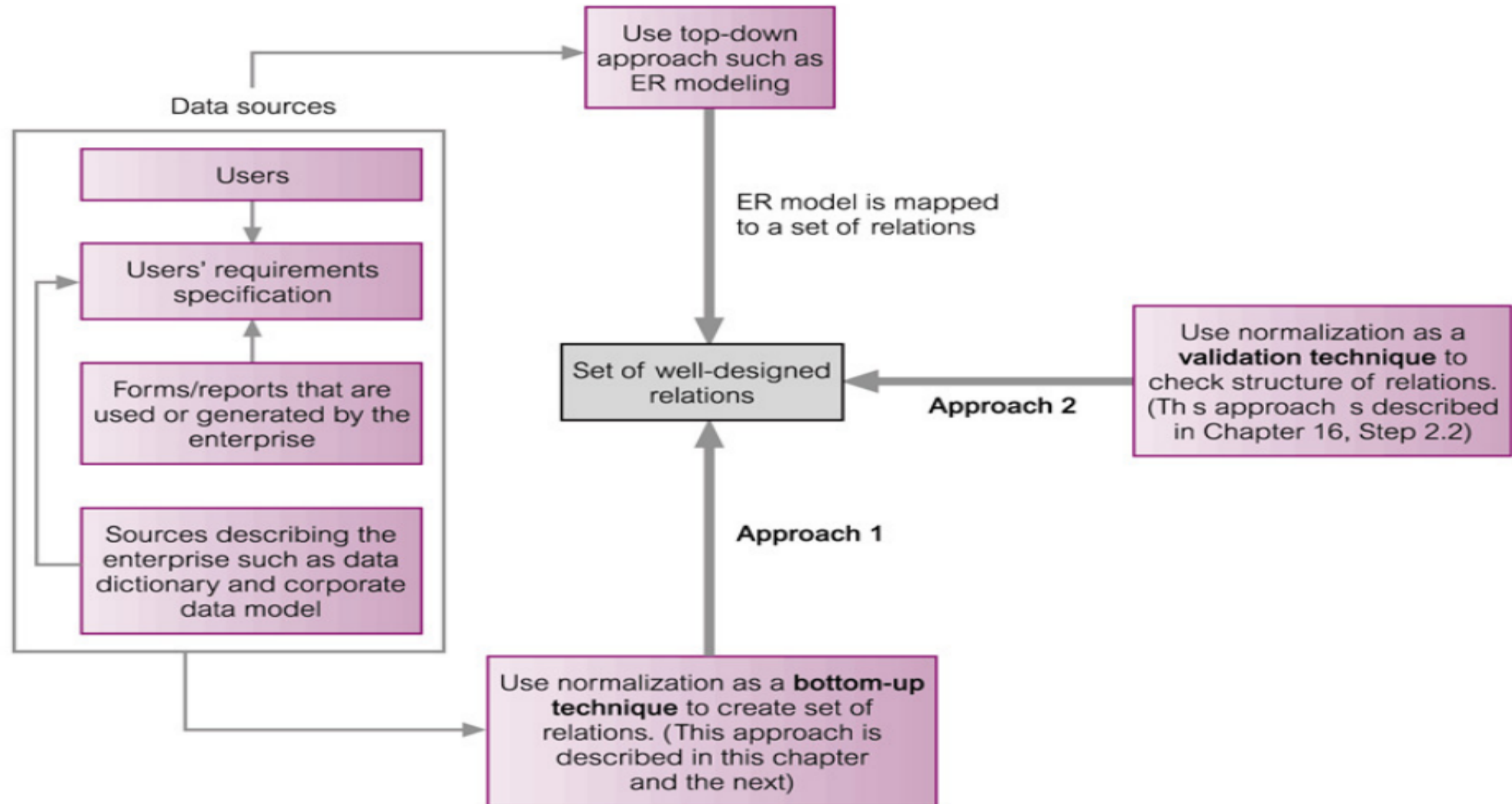
- Eliminating redundant(useless) data, therefore handling data integrity, because if data is repeated it increases the chances of inconsistent data.
- Normalization helps in keeping data consistent by storing the data in one table and referencing it everywhere else.
- Storage optimization although that is not an issue these days because Database storage is cheap.
- Breaking down large tables into smaller tables with relationships, so it makes the database structure more scalable and adaptable.
- Ensuring data dependencies make sense i.e. data is logically stored.

Why we need Normalization

Normalization Avoids

- **Duplication of Data**- The same data is listed in multiple lines of the database
- **Insert Anomaly**- A record about an entity cannot be inserted into the table without first inserting information about another entity - Cannot enter a customer without a sales order
- **Delete Anomaly**- A record cannot be deleted without deleting a record about a related entity. Cannot delete a sales order without deleting all of the customer's information.
- **Update Anomaly**- Cannot update information without changing information in many places. To update customer information, it must be updated for each sales order the customer has placed

How Normalization Supports Database Design



Problems without Normalization

For example

Rollno	Name	Branch	Hod	Office_tel
401	Akon	CSE	Mr. X	53337
402	Bkon	CSE	Mr. X	53337
403	Ckon	CSE	Mr. X	53337
404	Dkon	CSE	Mr. X	53337

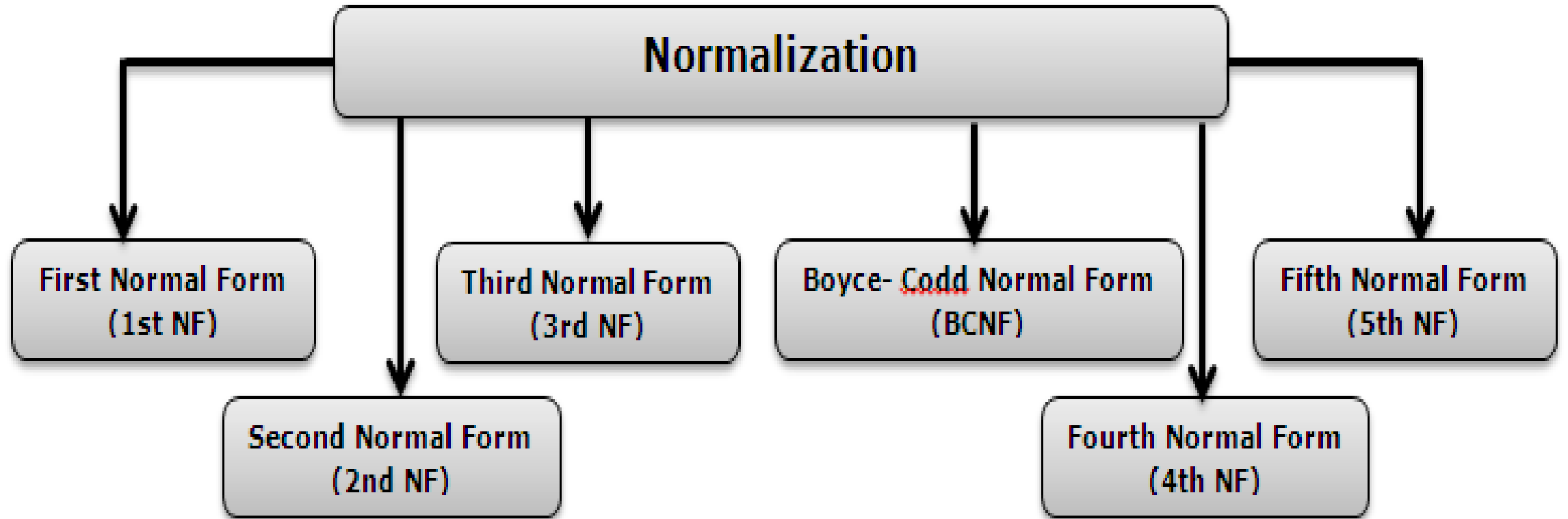
- **Data Redundancy..**
- **eat up extra memory**

Insertion Anomaly

Updation Anomaly

Deletion Anomaly

Types of Normal Forms



First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 5 rules:

- Value of each attribute should be **Atomic/single value**.
- No composite values and **Multi-value** Attributes.
- Each column must have a **unique name** and all entries in any column must be of the same kind
- **Identify** each set of **related data** with a **primary key**
- No **two rows are identical**.
- If the relation does not have a **primary key**, create a composite key using a combination of attributes that uniquely identify each row.
- If no combination of attributes can uniquely identify rows, introduce a surrogate

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 5 rules:

An entity is in the **first normal form** if it contains **no repeating groups**. In relational terms, a table is in the first normal form if **it contains no repeating columns**. Repeating columns make your data **less flexible, waste disk space, and makes it more difficult to search for data**.

First Normal Form (1NF)

Example

Employee

emp_id	emp_name	emp_mobile	emp_skills
1	John Tick	9999957773	Python, JavaScript
2	Darth Trader	8888853337	HTML, CSS, JavaScript
3	Rony Shark	7777720008	Java, Linux, C++

The above table has 4 columns:

- All the columns have **different names**.
- All the columns hold values of the same type like emp_name has all the names, emp_mobile has all the contact numbers, etc.
- The order in which we save data doesn't matter
- **But the emp_skills column holds multiple comma-separated values**, while as per the **First Normal form**, each column should **have a single value**.

Hence the above table fails to pass the First Normal form.

First Normal Form (1NF)

Example

Employee

emp_id	emp_name	emp_mobile	emp_skills
1	John Tick	9999957773	Python, JavaScript
2	Darth Trader	8888853337	HTML, CSS, JavaScript
3	Rony Shark	7777720008	Java, Linux, C++

So how do you fix the above table? There are two ways to do this:

- 1.Remove the **emp_skills** column from the **Employee** table and keep it in some other table.
- 2.Or add **multiple rows** for the employee and **each row is linked** with one skill.

Employee

emp_id	emp_name	emp_mobile
--------	----------	------------

Employee_skill

emp_id	emp_skill
--------	-----------

First Normal Form (1NF)

Example

Employee

emp_id	emp_name	emp_mobile	emp_skills
1	John Tick	9999957773	Python, JavaScript
2	Darth Trader	8888853337	HTML, CSS, JavaScript
3	Rony Shark	7777720008	Java, Linux, C++

Add Multiple rows for Multiple skills:

Emp_id	emp_name	emp_mobile	emp_skill
1	John Tick	9999957773	Python
1	John Tick	9999957773	JavaScript
2	Darth Trader	8888853337	HTML
2	Darth Trader	8888853337	CSS
2	Darth Trader	8888853337	JavaScript
3	Rony Shark	7777720008	Java
3	Rony Shark	7777720008	Linux
3	Rony Shark	7777720008	C++

- Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique

First Normal Form (1NF)

Example

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 1



Conversion to first normal form

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Table 2


First Normal Form (1_{st}NF)

Example

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations



(b)

DEPARTMENT

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

(c)

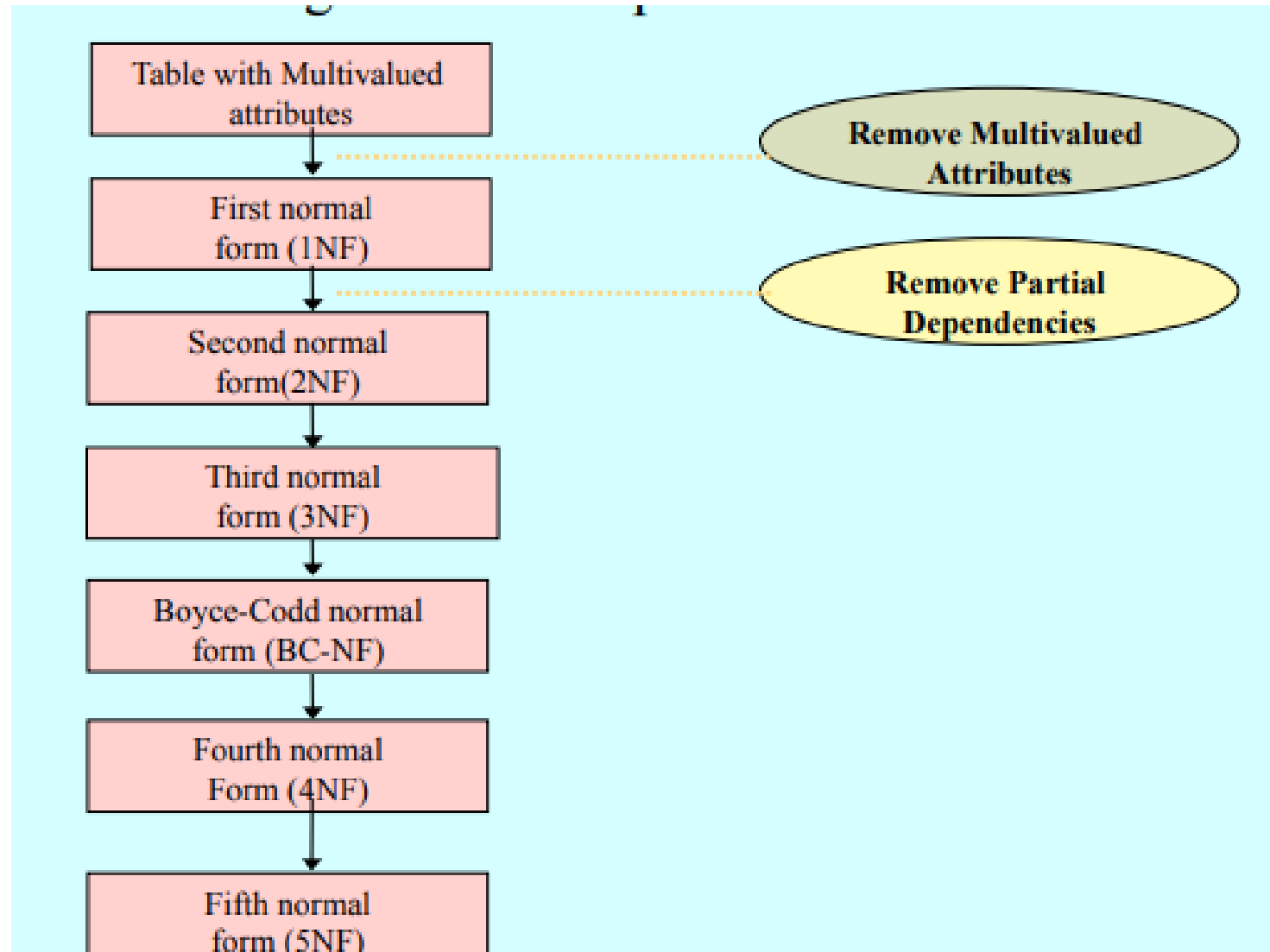
DEPARTMENT

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

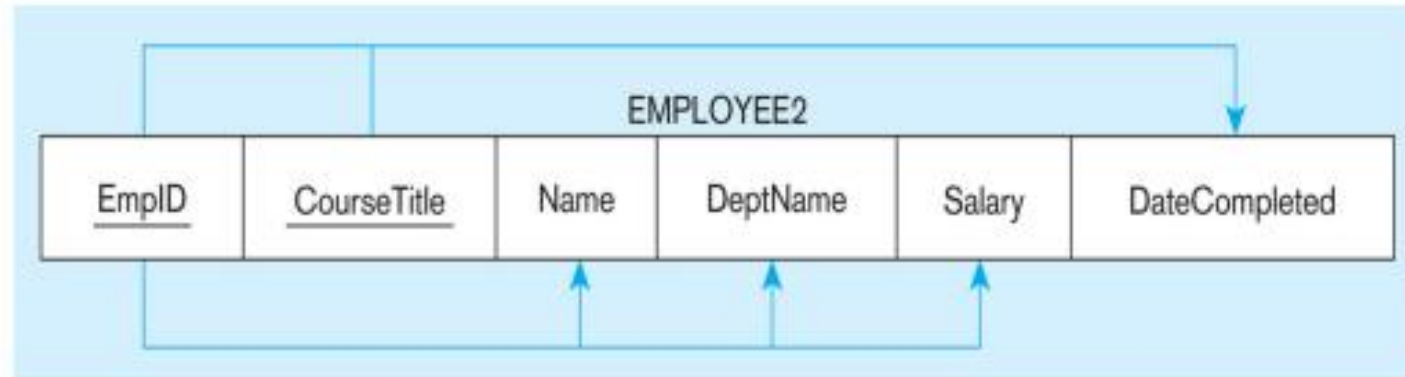
Figure 14.9

Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

First Normal Form (1_{st}NF)



Representing Functional Dependencies



EmpID →

EmpID, CourseTitle →

<u>EmpID</u>	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/201X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/201X

Second Normal Form(2_{nd}NF)

The Second Normal Form, it must satisfy following conditions:

1.Remove Partial Dependencies.

2.Functional Dependency: The value of one attribute in a table is determined entirely by the value of another.

3.Partial Dependency: A type of functional dependency where an attribute is functionally dependent on only part of the primary key (the primary key must be a composite key).

4.Create a separate table with the functionally dependent data and the part of the key on which it depends. The tables created at this step will usually contain descriptions of resources.

A relation is in 2NF if it is in 1NF and every non-key attribute is fully dependent on each candidate key of the relation.

Second Normal Form(2_{nd}NF)

Partial Dependency

- Let us assume a relation **R** with attributes **A, B, C, and D**. Also, assume that the set of functional dependencies **F** that hold on **R** as follows;

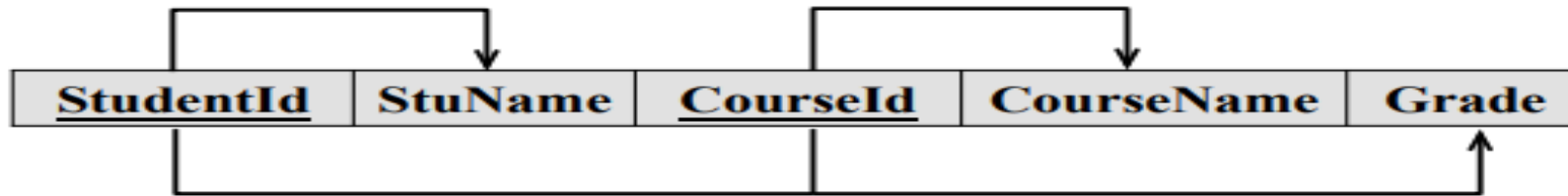
$$F = \{A \rightarrow B, D \rightarrow C\}.$$

- From a set of attributes **F**, we can derive the primary key.
- For **R**, the key can be (**A,D**), a **composite primary key**.
- That means, **AD → BC**, **AD** can uniquely identify **B** and **C**.
- To identify **B**, attribute **A** is enough.
- Likewise, to identify **C**, attribute **D** is enough.
- The functional dependencies **AD → B** or **AD → C** are called as **Partial functional dependencies**

Second Normal Form(2_{nd}NF)

Partial Dependency

Functional Dependencies in Student



Can represent FDs with arrows as above, or

- $\text{StudentId} \rightarrow \text{StuName}$,
- $\text{CourseId} \rightarrow \text{CourseName}$
- $\text{StudentId}, \text{CourseId} \rightarrow \text{Grade}$ (and $\text{StuName}, \text{CourseName}$)

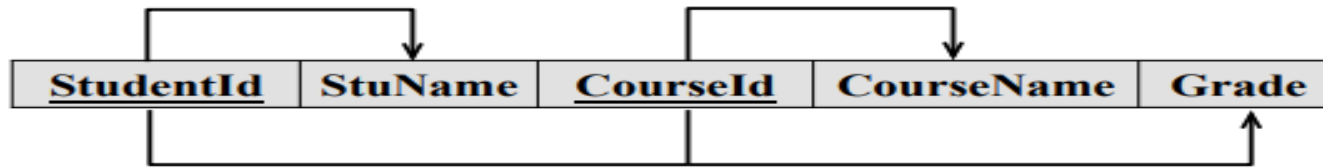
Any partial FDs ?

Second Normal Form(2_{nd}NF)

Partial Dependency

- Partial dependency occurs when one primary key determines some other attribute/attributes.
- Partial dependency is a situation in which a non-key attribute of a table depends on only a part of the primary key.

Student



The prime key attributes are **StudentID** and **CourseId**

StudentID = Unique ID of the student , **StudentName** = Name of the student

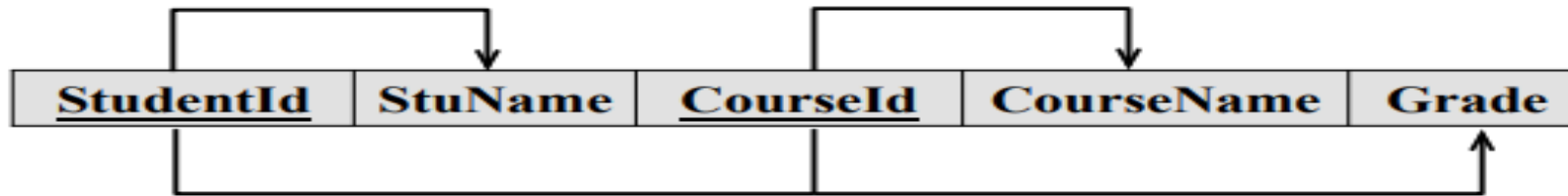
CourseId = Unique ID of the Course, **CourseName** = Name of the Course

As stated, the non-prime attributes i.e. **StudentName** and **CourseName** should be functionally dependent on part of a **candidate key**, to be **partially dependent**.

Second Normal Form(2_{nd}NF)

Partial Dependency

Functional Dependencies in Student



Can represent FDs with arrows as above, or

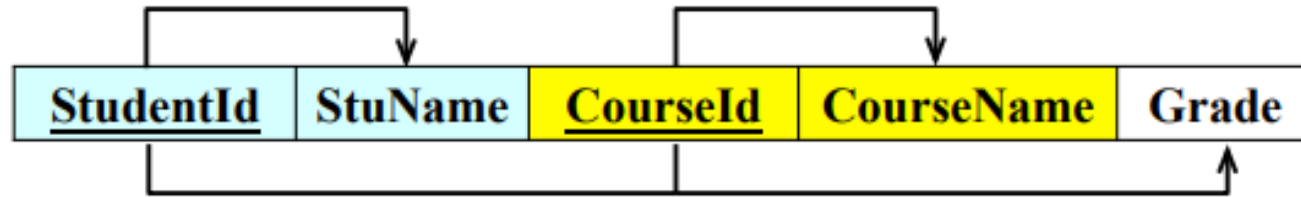
- $\text{StudentId} \rightarrow \text{StuName}$,
- $\text{CourseId} \rightarrow \text{CourseName}$
- $\text{StudentId}, \text{CourseId} \rightarrow \text{Grade}$ (and $\text{StuName}, \text{CourseName}$)

Therefore, NOT in 2nd Normal Form!!

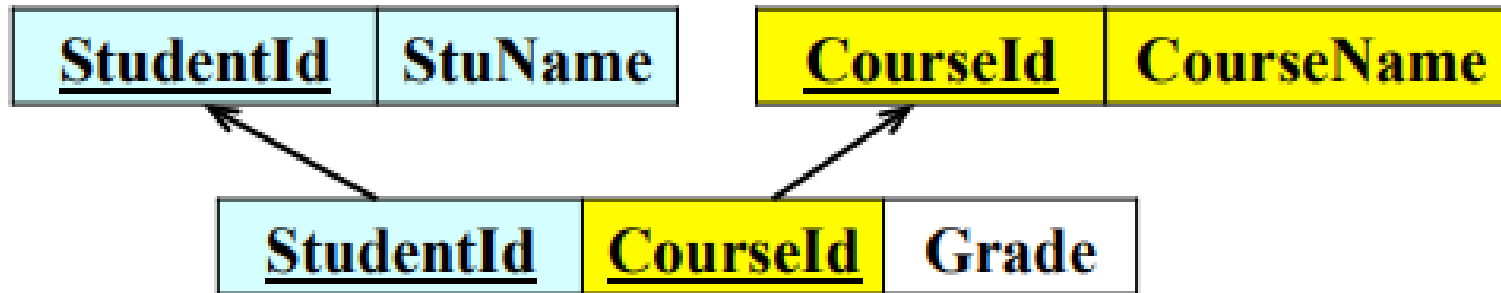
Second Normal Form(2_{nd}NF)

2NF: Normalizing

- How do we **convert the partial dependencies** into **normal ones**? By **breaking into more tables**.



Becomes ... (notice above arrows mean functional dependency, below they mean **FK constraints**)

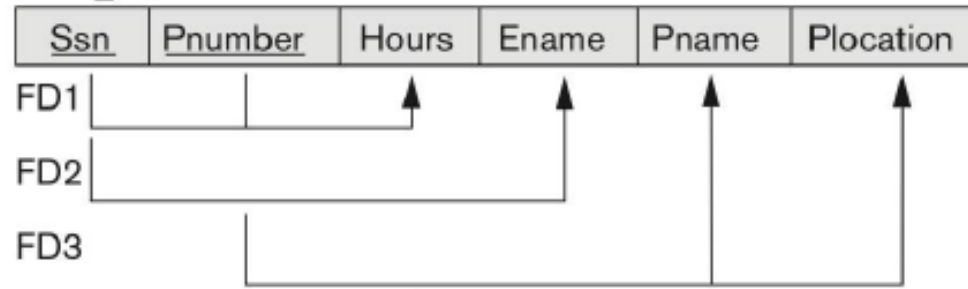


Second Normal Form(2_{nd}NF)

2NF: Normalizing

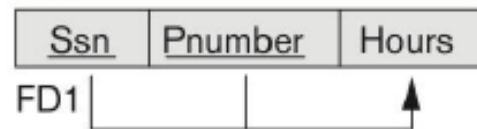
(a)

EMP_PROJ

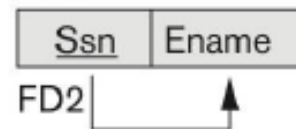


2NF Normalization

EP1



EP2



EP3

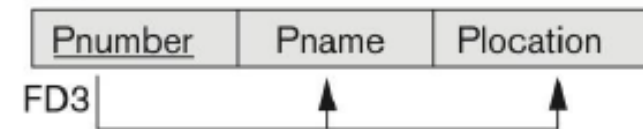


Figure 14.11

Normalizing into 2NF and 3NF.
(a) Normalizing EMP PROJ into 2NF relations.

Second Normal Form(2_{nd}NF)

2NF: Normalizing

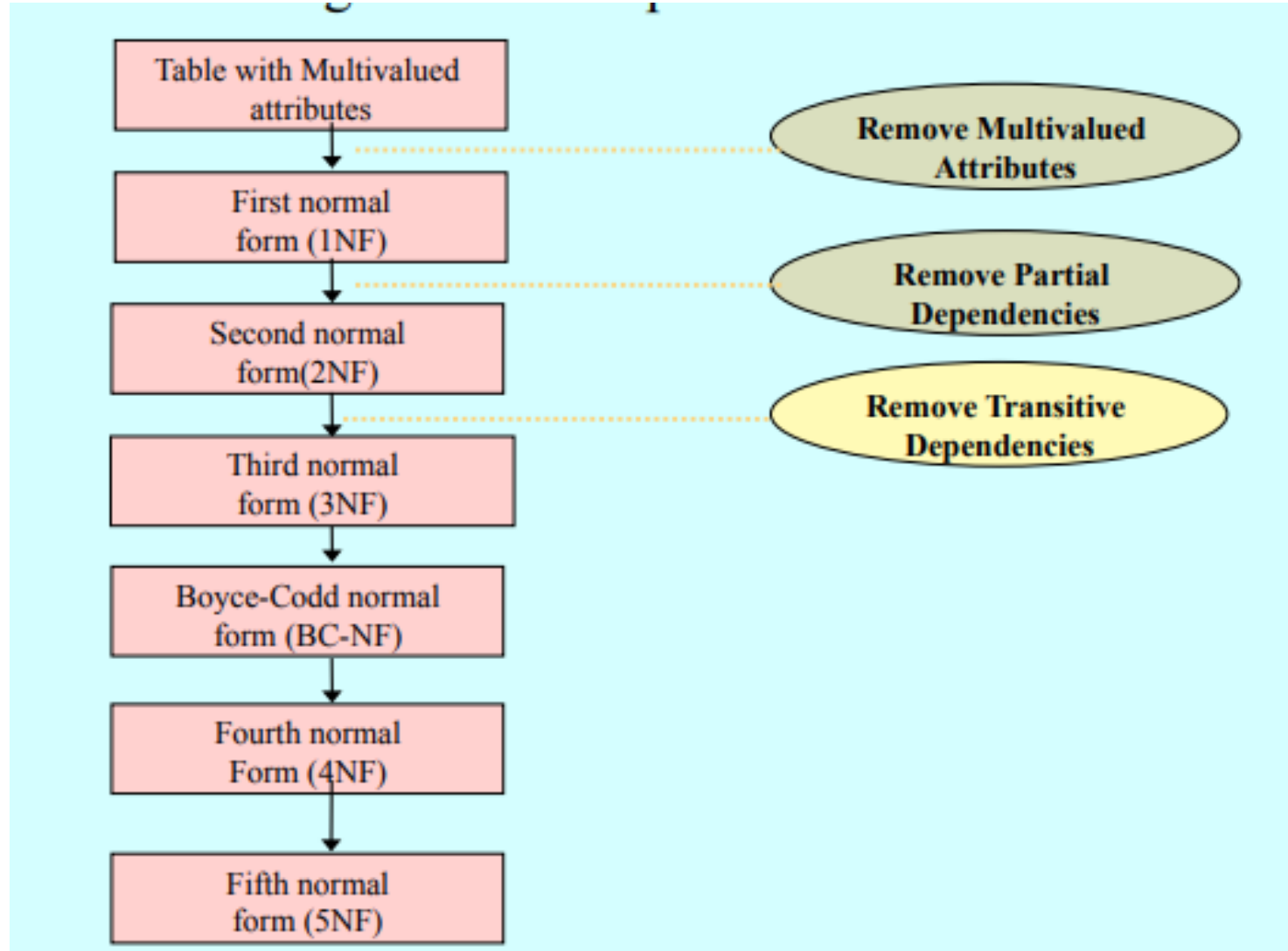
You Try

<u>SeriesId</u>	<u>EpisodeId</u>	SeriesTitle	EpisodeTitle	AiringDate
-----------------	------------------	-------------	--------------	------------

Summary

- For a table to be in the Second Normal form, it should be in the First Normal form and it should not have Partial Dependency.
- Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.
- To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

Second Normal Form(2_{nd}NF)



Third Normal Form (3_{rd} NF)

- The third Normal Form is an upgrade to the Second Normal Form. When a table is in the Second Normal Form and has no transitive dependency, then it is in the Third Normal Form.

3_{rd} NF = 2NF and no transitive dependencies

- A transitive dependency is when a non-key attribute depends on another non-key attribute
- Note: This is called transitive because the primary key is a determinant for another attribute, which in turn is a determinant for a third attribute.

Transitive functional dependency: a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$

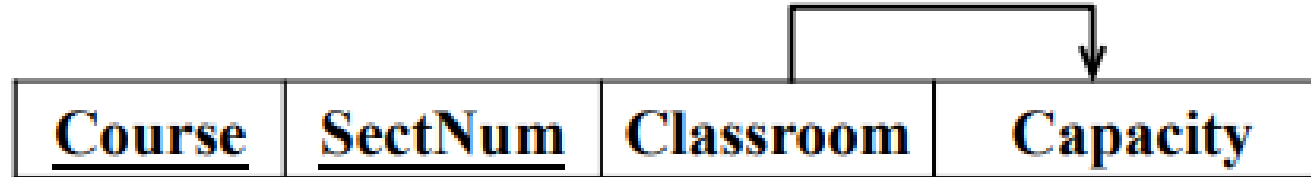
Third Normal Form (3_{rd} NF)

The Third Normal Form, it must satisfy following conditions:

- If a relation is in 2NF AND does not have any transitive partial dependencies, it is said to be in 3NF.
- The amount of duplicate data present in a database is decreased with 3NF. Additionally, it is employed to ensure data integrity.
- A relation is in 3NF if the non-prime attributes are not transitively dependant on any candidate keys.
- A relation can be considered to be in 3NF if at least any one of the following constraints for every non-trivial dependency given by $A \rightarrow B$.
 - * A is a super key
 - * B is part of some candidate key, i.e. B is a prime attribute.

Third Normal Form (3_{rd}NF)

3NF Example

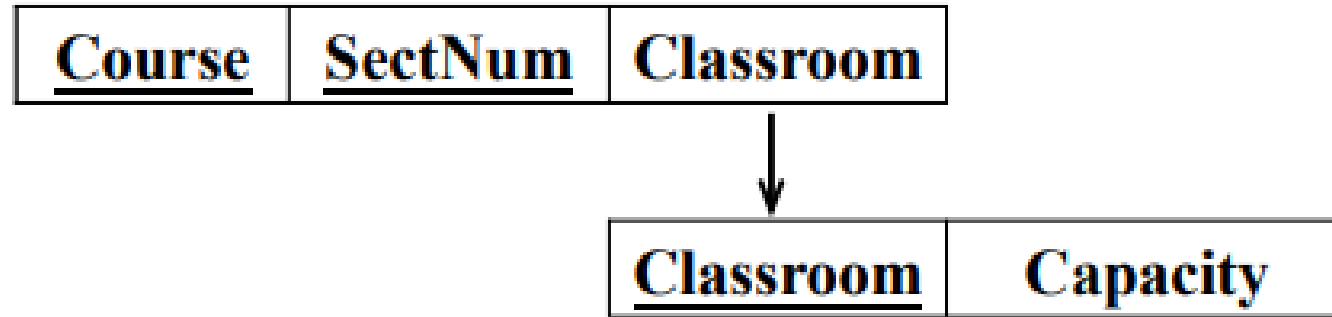


Classroom → Capacity TRANSITIVE

- Any partial FDs? NO
- Any transitive FDs? YES !
- How do we eliminate it?
- By breaking into its own table

Third Normal Form (3_{rd}NF)

3NF Normalization



3NF Normalization example

student_id	student_name	student_state	student_city	student_pincode
101	Ananya	Karnataka	Mysore	570010
102	Anitha	Karnataka	Bengaluru	560010
103	Ankitha	Tamilnadu	Chennai	600010
104	Aryan	Andhra	Tirupati	517502
105	Brijesh	Tamilnadu	Coimbatore	641001

- {student_id} is the **Candidate Key** (an attribute that identifies the tuple uniquely). All the other attributes are **non-prime attributes**.
- student_state and student_city are **dependant** on student_pincode which is inturn **dependant** on student_id

Third Normal Form (3_{rd}NF)

3NF Normalization example

Student_Table

student_id	student_name	student_pincode
101	Ananya	570010
102	Anitha	560010
103	Ankitha	600010
104	Aryan	517502
105	Brijesh	641001

Student_Address Table

student_pincode	student_state	student_city
570010	Karnataka	Mysore
560010	Karnataka	Bengaluru
600010	Tamilnadu	Chennai
517502	Andhra	Tirupati
641001	Tamilnadu	Coimbatore

Third Normal Form (3_{rd}NF)

3NF Normalization

Exam

StudentId	Name	ExamType	MaxMarks
201	Saghir	Viva	20
202	Harris	Theroy	100
203	Maxwell	Practical	50
204	Andy	Practical	50
205	Simon	Viva	20
206	Sam	Theroy	100
207	Jim	Theroy	100
208	Taylor	Practical	50

ExamType	MaxMarks
Viva	20
Theroy	100
Practical	50

Primary
key

Third Normal Form (3_{rd}NF)

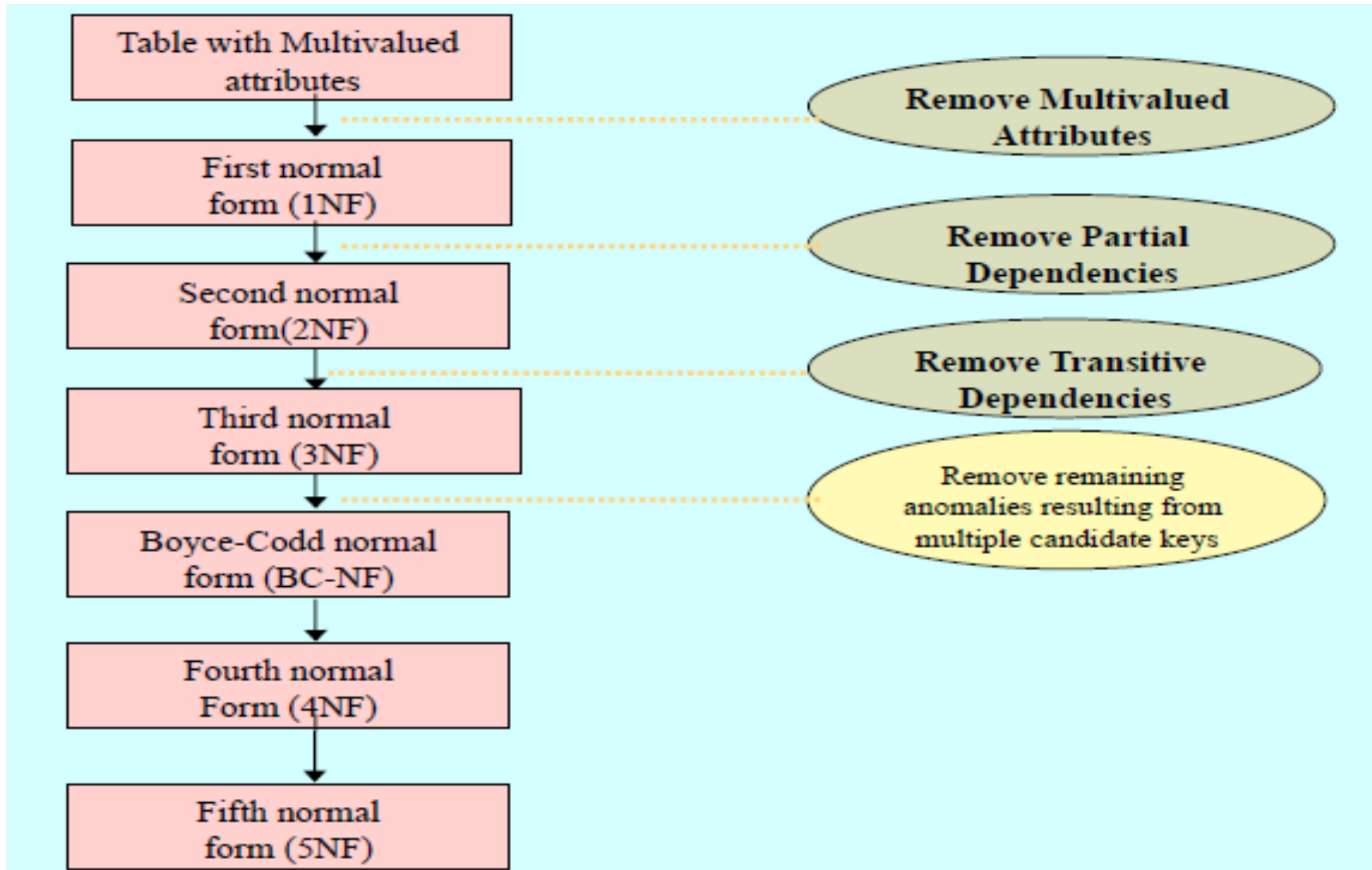
You Try ...

<u>StudentId</u>	ProgramId	StudentName	ProgramName
------------------	-----------	-------------	-------------

Partial FDs? Eliminate, if any.

Transitive FDs? Eliminate, if any.

Boyce-Codd Normal Form (BCNF)



Boyce-Codd Normal Form (BCNF)

- BCNF stands for **Boyce-Codd normal form**, which is a **special case of 3NF** and is also known as **3.5 NF**.
- **BCNF** is a normal form used in the normalization of databases and has **more strict rules as compared to 3NF**.

BCNF rules

To check if the table satisfies the conditions of BCNF, the following two conditions should exist:

- The table must be in 3NF form.
 - For any dependency $X \rightarrow Y$, X must be a candidate key or super key. In other words, for dependency $X \rightarrow Y$, if Y is a prime attribute, X cannot be a non-prime attribute.
-
- 3NF states that **the transitive dependency must not exist**. Transitive dependency is that the LHS (left-hand side) of the functional dependency must **consist of a super key/candidate key** or the RHS (right-hand side) must have a **prime attribute**. BCNF adds more restrictions by stating that LHS of functional dependency must have a super key and removes the RHS condition.

Boyce-Codd Normal Form (BCNF)

Example of BCNF

Student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- One student can enroll for multiple subjects. For example, the student with student_id 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.

What do you think should be the **Primary Key**?

- Well, in the table above student_id, subject together form the primary key, because using student_id and subject, we can find all the columns of the table.

Boyce-Codd Normal Form (BCNF)

Example of BCNF

Student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- One more important point to note here is, **one professor teaches only one subject**, but one **subject** may have two different professors.
- Hence, there is a **dependency between the subject and professor** here, where the **subject depends on the professor's name**.
- This **table satisfies the 1st Normal form** because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.
- This table also satisfies the **2nd Normal Form** as there is no Partial Dependency.
- And, there is **no Transitive Dependency**, hence the table also satisfies the 3rd Normal Form.
- But this table is **not in Boyce-Codd Normal Form**.

Boyce-Codd Normal Form (BCNF)

Student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

Example of BCNF

Why this table is not in BCNF?

In the table above, `student_id`, `subject` form primary key, which means `subject` column is a prime attribute.

But, there is one more dependency, `professor` \rightarrow `subject`.

And while the `subject` is a prime attribute, the `professor` is a non-prime attribute, which is not allowed by BCNF.

Boyce-Codd Normal Form (BCNF)

Example of BCNF

Student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

How to satisfy BCNF?

- To make this relation(table) **satisfy BCNF**, we will decompose this table into two tables, **student table** and **professor table**.

Student Table

student_id	p_id
101	1
101	2

Professor Table

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++

Boyce-Codd Normal Form (BCNF)

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

and following dependencies :

A -> BCD

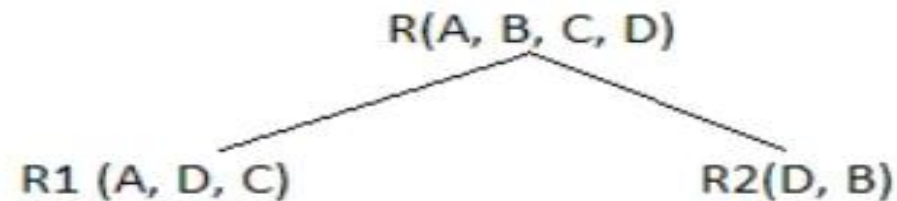
BC -> AD

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.

Boyce-Codd Normal Form (BCNF)

The difference between 3NF and BCNF

- functional dependency $A \rightarrow B$, 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key, whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.

Boyce-Codd Normal Form (BCNF)

Example 1:

Let's take another general example to understand the concept of decomposition in detail:

We have a relation $R(A, B, C, D)$ that is already in 3NF.

Candidate Keys: $\{A, BC\}$ Prime Attributes: $\{A, B, C\}$ Non-Prime Attributes: $\{D\}$

Functional dependencies are as follows: $\{A \rightarrow BCD, BC \rightarrow AD, D \rightarrow B\}$

The above relation is not in BCNF because $\{D \rightarrow B\}$ is not in BCNF as $\{D\}$ is neither a candidate key nor a prime attribute. Hence, we will decompose the relation R into $R_1\{A, D, C\}$ and $R_2\{D, B\}$.

Boyce-Codd Normal Form (BCNF)

Example 1:

In this example, we have to find the highest normalization form, and for that, we are given a relation

$R(A, B, C, D, E)$ with functional dependencies as follows: $\{BC \rightarrow D, AC \rightarrow BE, B \rightarrow E\}$

- As we can see, $(AC)^+ = \{A, C, B, E, D\}$ and also, none of its subsets can determine all the attributes of the relation. There is another point to be noted that A or C can't be derived from any other attribute of the relation, and therefore, there is only one candidate key, $\{AC\}$.
- Prime attributes in DBMS are always part of the candidate keys, and for this relation R , prime attributes are: $\{A, C\}$ while non-prime attributes are: $\{B, E, D\}$.
- Clearly, there is no multi-valued attribute in the relation R , and hence, it is at least in 1NF.
- $BC \rightarrow D$ is in 2NF because BC is not a proper subset of the candidate key AC . $AC \rightarrow BE$ is also in 2NF because AC itself is a candidate key, and lastly, $B \rightarrow E$ is again in 2NF. For 2NF, there must not be any partial dependency present in the table, and hence, relation R here is in 2NF.
- The relation R is not in 3NF because $BC \rightarrow D$ at the start is not in 3NF (BC is not a candidate key, and also, D is not a prime attribute). Hence, the relation R has 2NF as the highest normalization form.

Example 4: Give $R(A, B, C, D)$ and Set of Functional Dependency $FD = \{ AB \rightarrow CD, C \rightarrow A, D \rightarrow B \}$. The question is to calculate the candidate key and no. of candidate key in above relation R using a given set of FDs