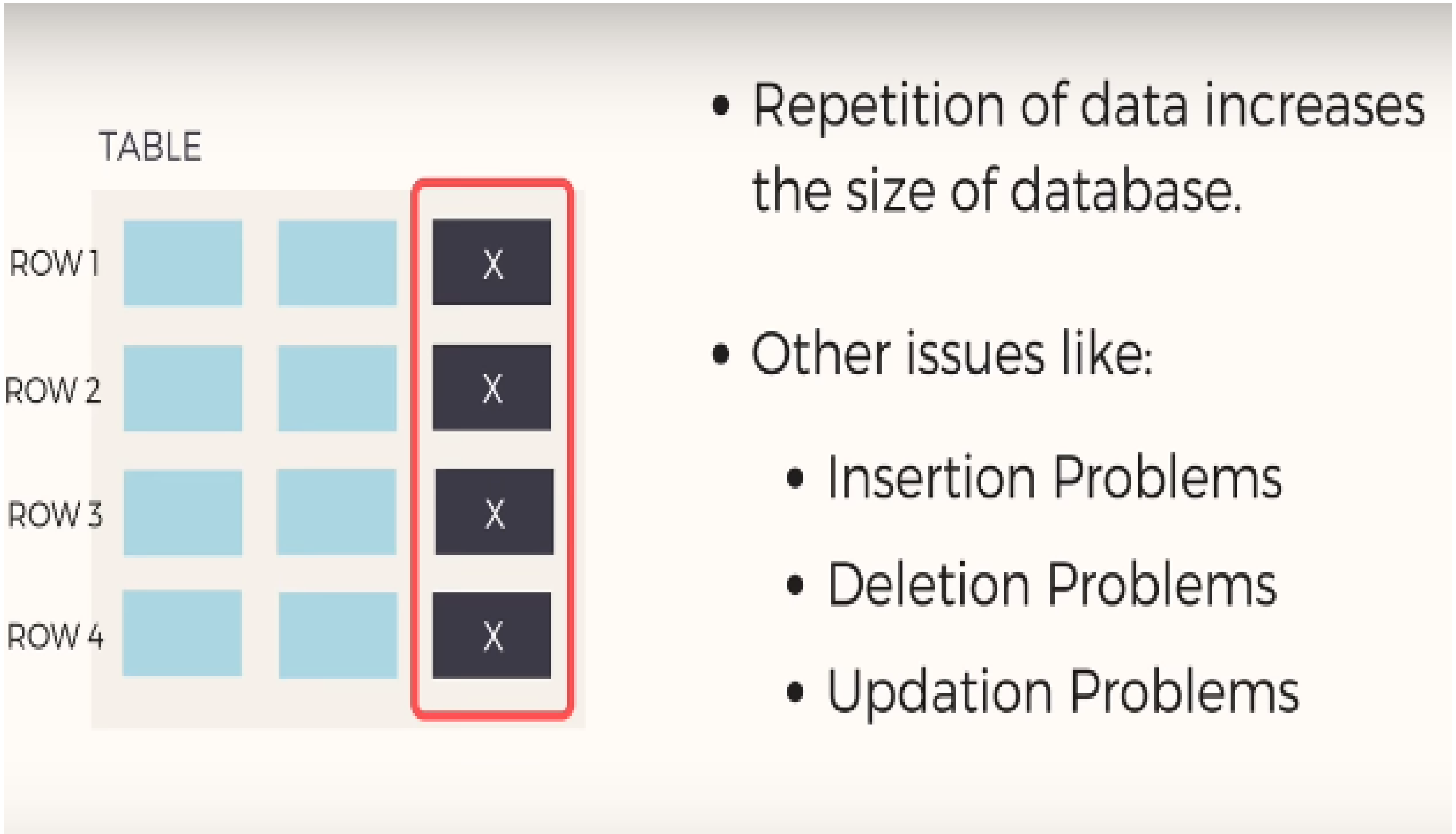
UNIT 3 :NORMALIZATION

Sushma Vankhede
Computer Science and Engineering
Navrachana University

NORMALIZATION

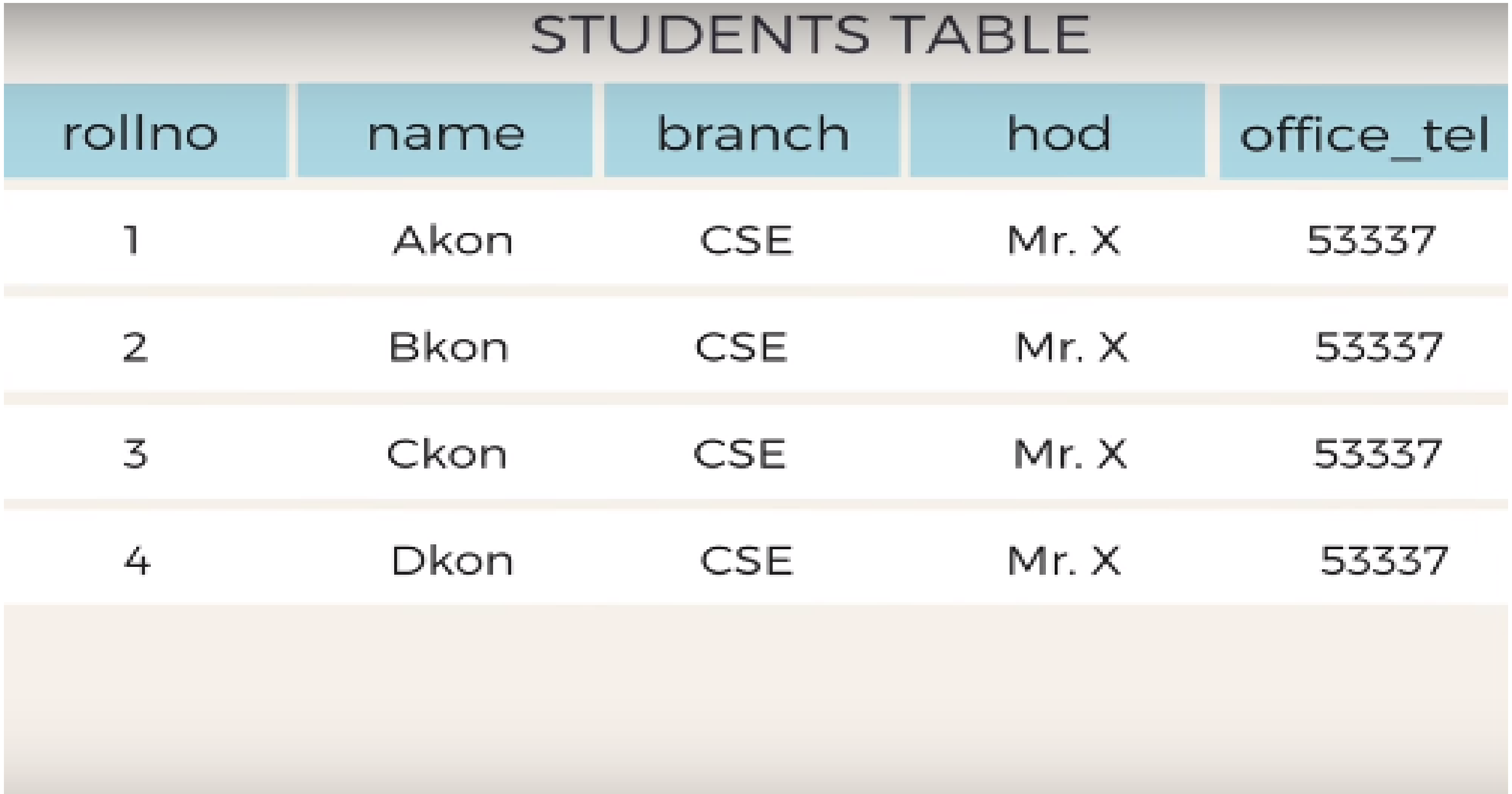
- Normalization is a technique of organizing the data into multiple related tables to minimize *data redundancy*.

WHAT IS DATA REDUNDANCY? WHY TO REDUCE IT?



- Repetition of data increases the size of database.
- Other issues like:
  - Insertion Problems
  - Deletion Problems
  - Updation Problems

EXAMPLE



STUDENTS TABLE

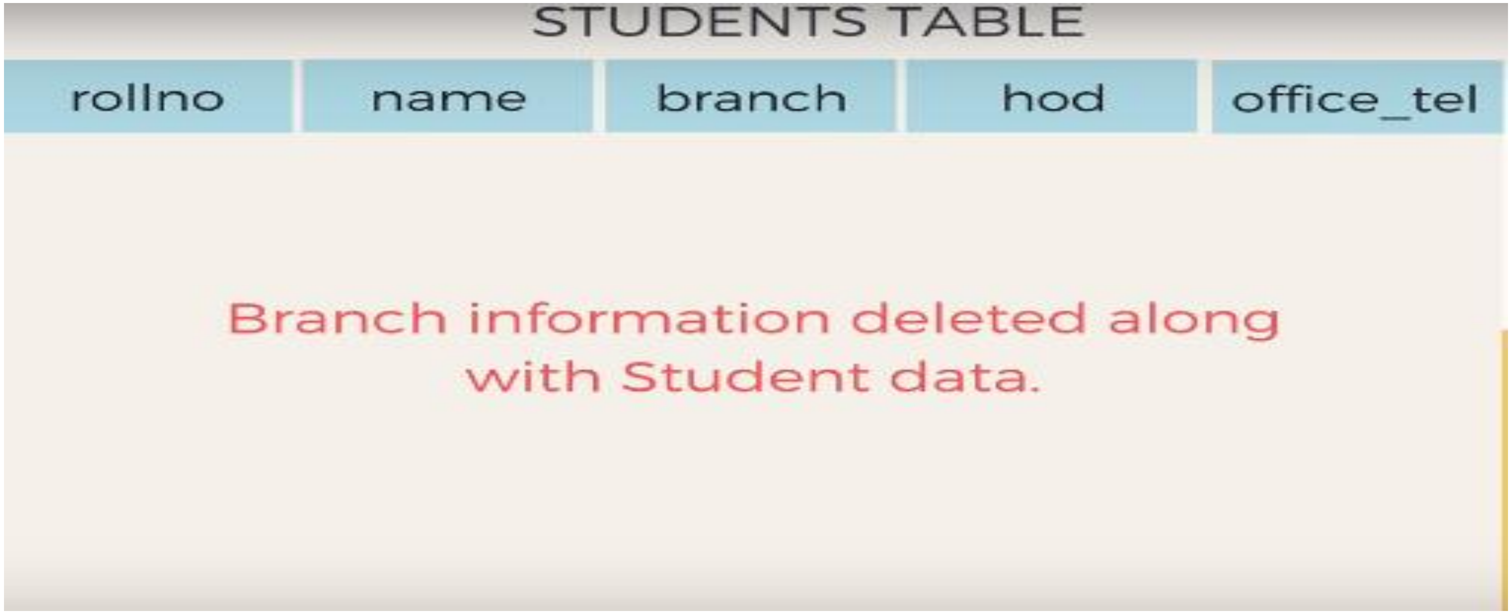
rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

ISSUES OF DATA REDUNDANCY

- Insertion Anomaly
  - To insert data for every new row (of student data in our case) is a data insertion problem or anomaly
- Reason for data repetition
  - To different but related data is stored in the same table.

ISSUES OF DATA REDUNDANCY..CONT

- Deletion anomaly
  - Loss of related dataset when some other dataset is deleted.



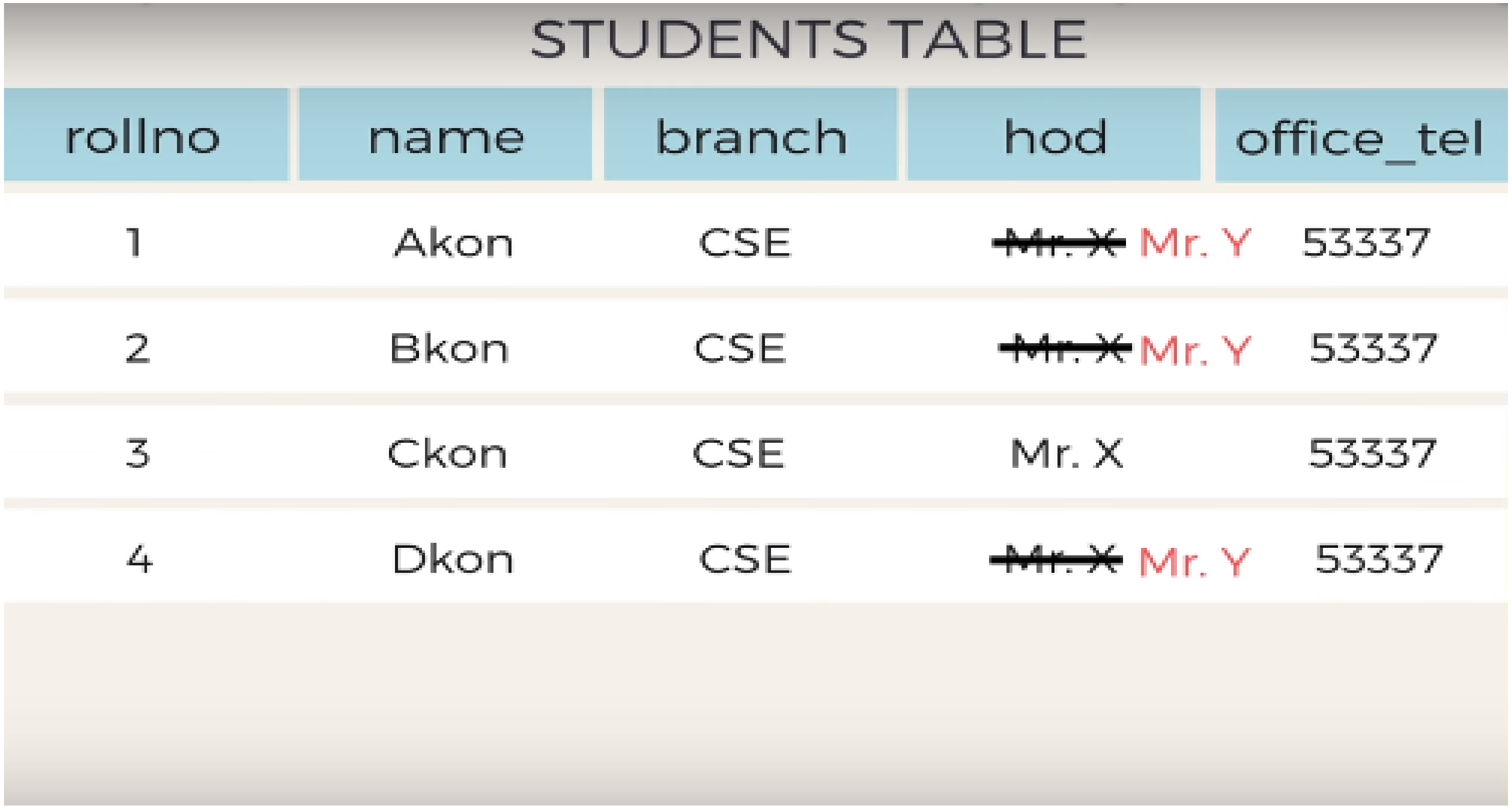
STUDENTS TABLE

rollno	name	branch	hod	office_tel

Branch information deleted along with Student data.

ISSUES OF DATA REDUNDANCY..CONT

- Updation anomaly
  - When Mr. X leave and Mr. Y joins as New HOD



STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

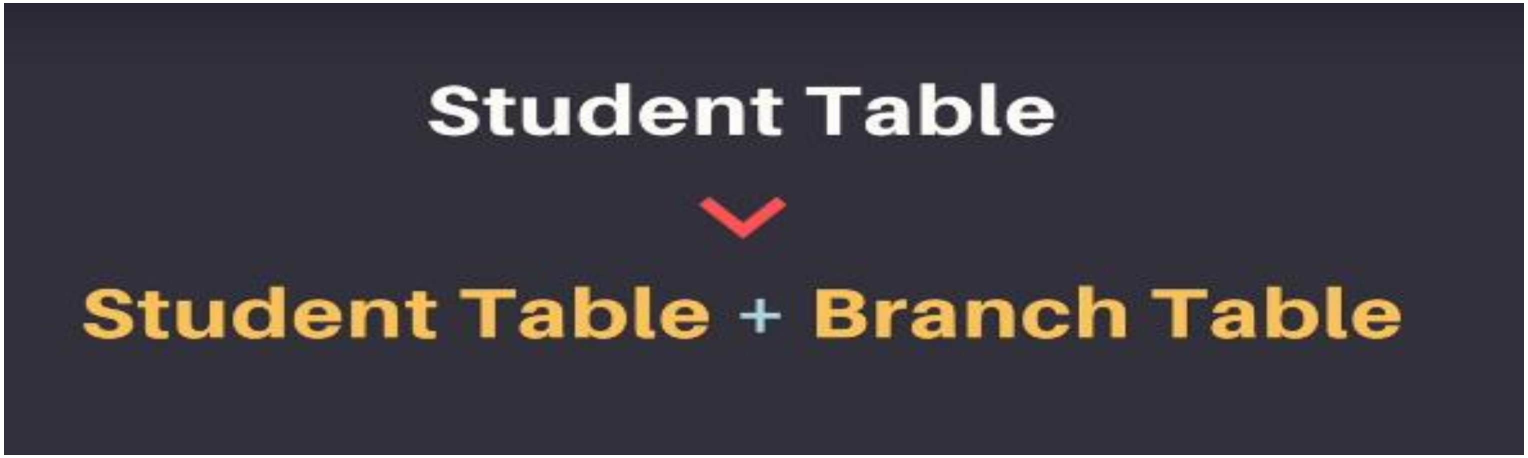
DATA REDUNDANCY

- Repetition of data hence needs extra space.
- Leads to insertion, deletion and Updation issues.

How Normalization will solve all these problems?

NORMALIZATION

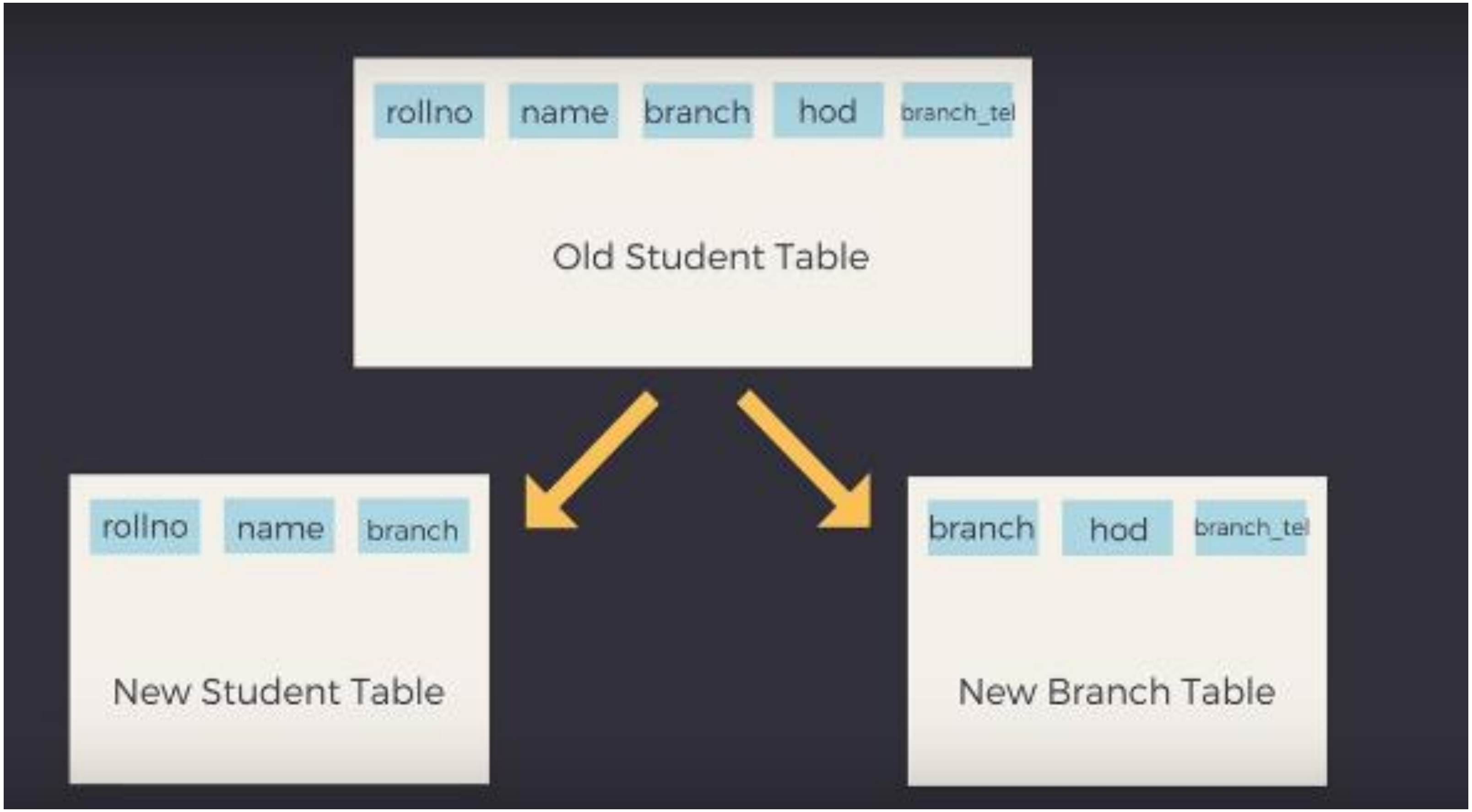
- It will break one table into two tables



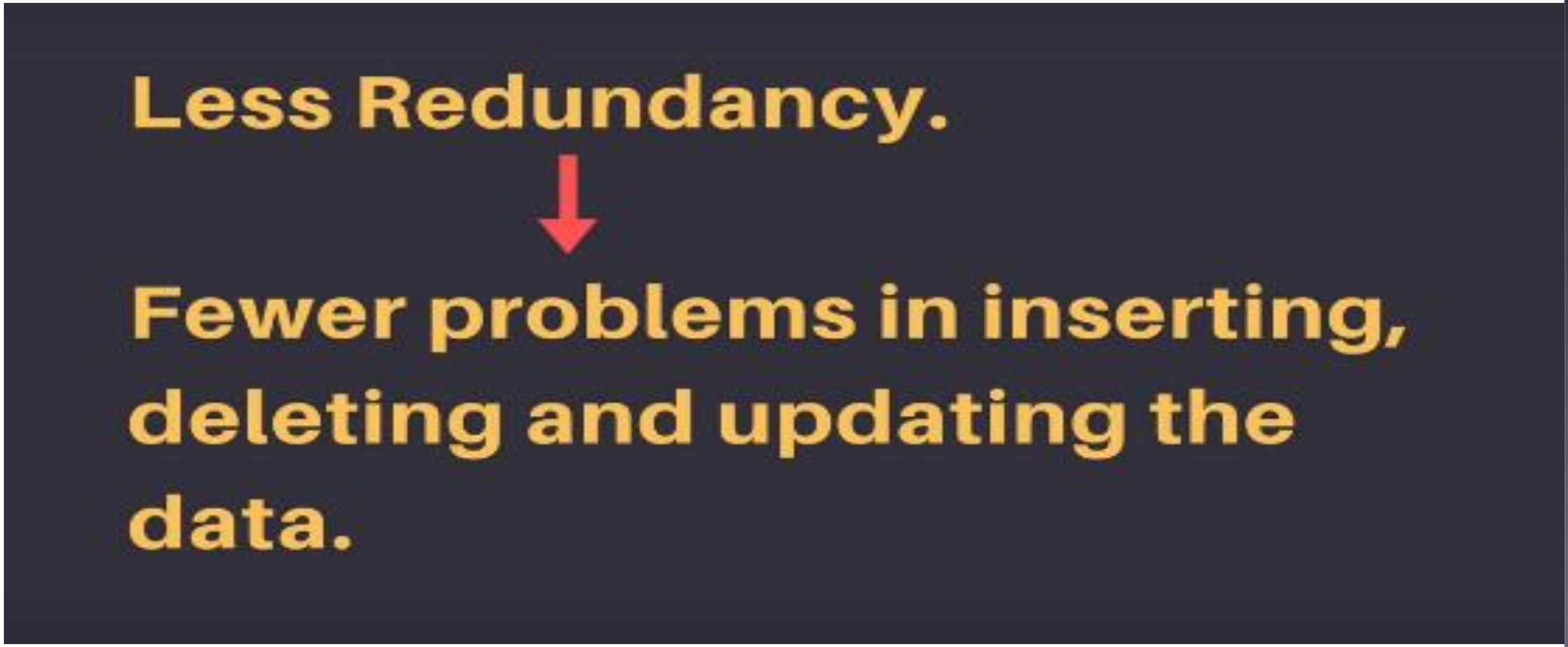
Student Table

Student Table + Branch Table

NORMALIZATION



NORMALIZATION

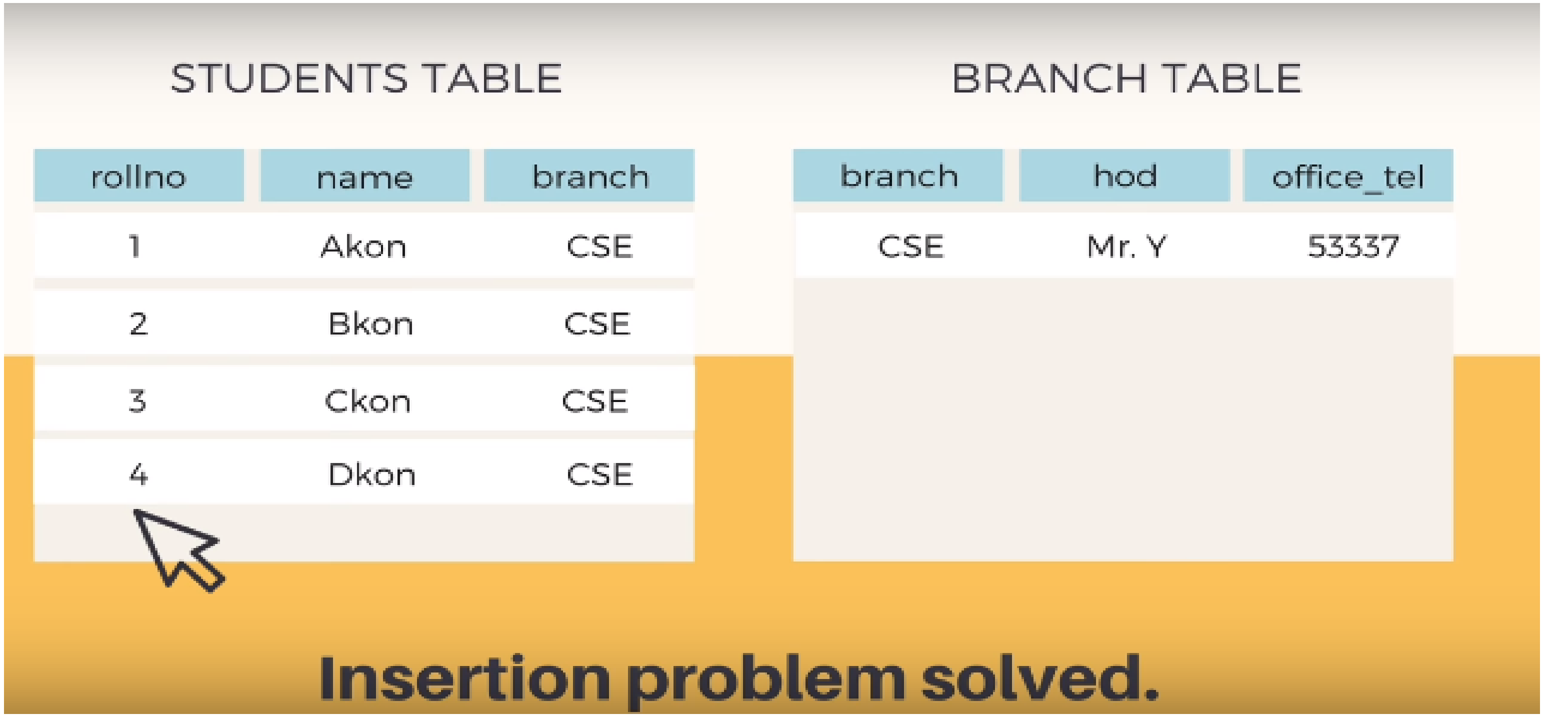


Less Redundancy.

↓

Fewer problems in inserting, deleting and updating the data.

NORMALIZATION



STUDENTS TABLE

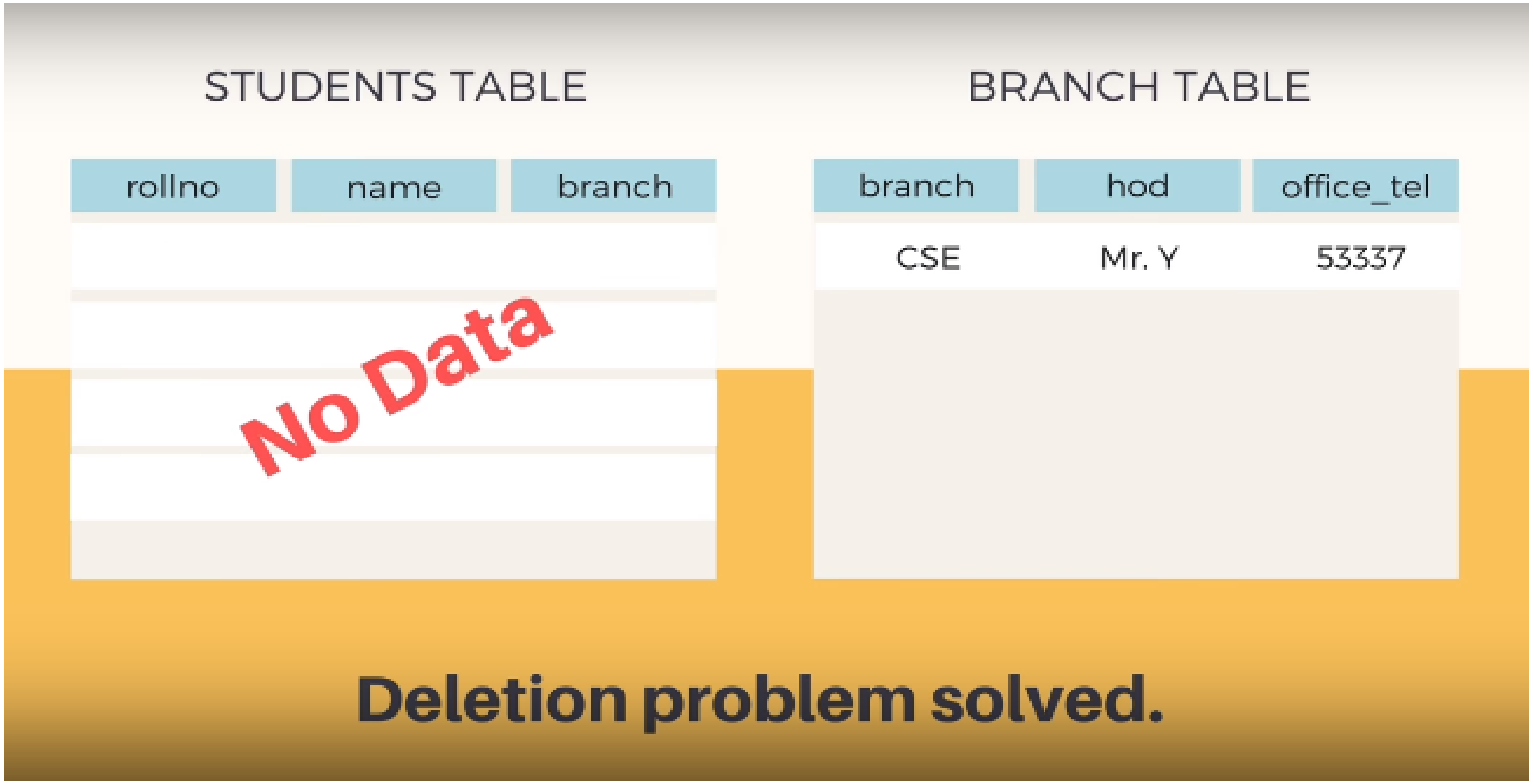
rollno	name	branch
1	Akon	CSE
2	Bkon	CSE
3	Ckon	CSE
4	Dkon	CSE

BRANCH TABLE

branch	hod	office_tel
CSE	Mr. Y	53337

Insertion problem solved.

NORMALIZATION



STUDENTS TABLE

rollno	name	branch

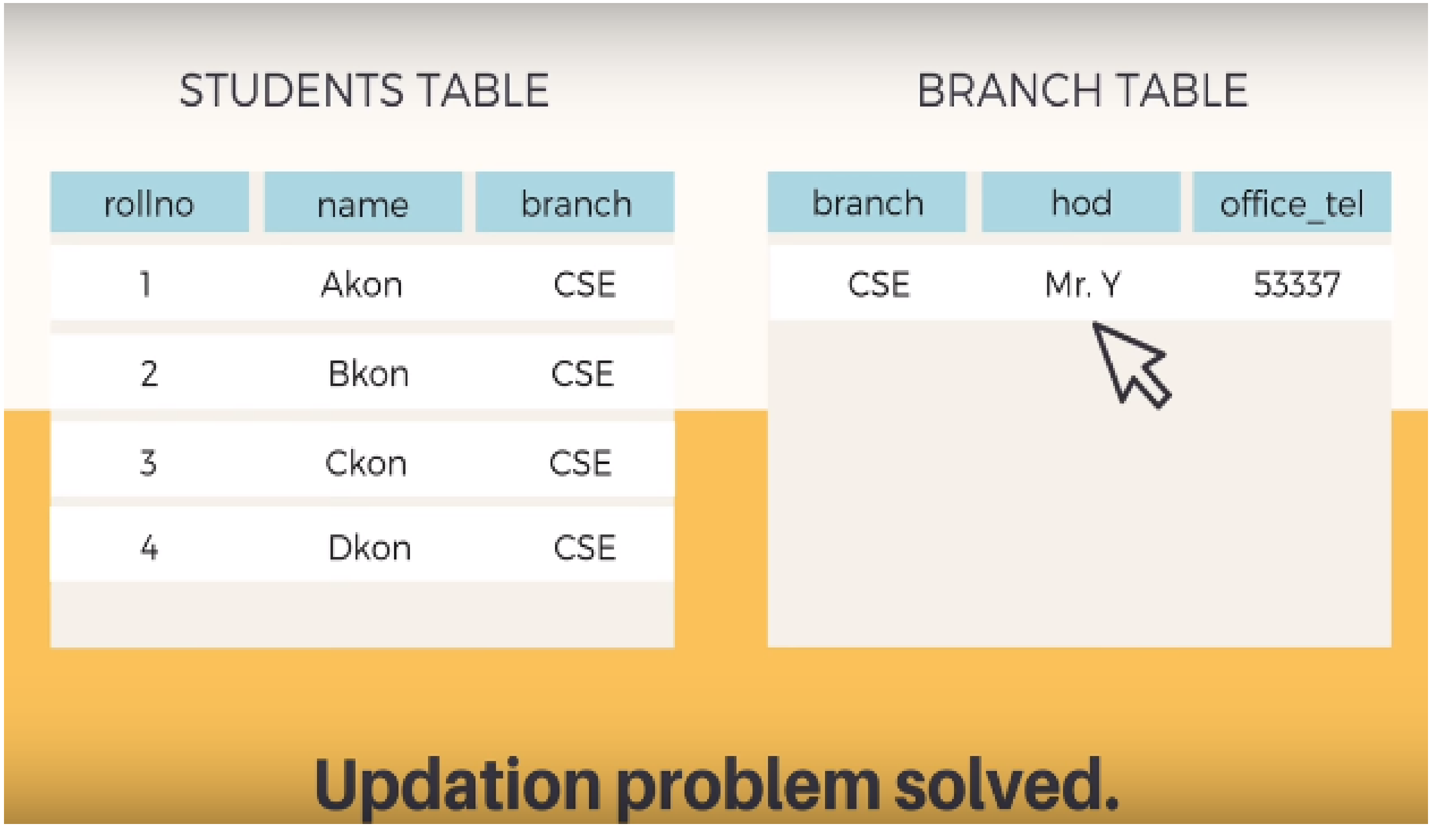
No Data

BRANCH TABLE

branch	hod	office_tel
CSE	Mr. Y	53337

Deletion problem solved.

NORMALIZATION



STUDENTS TABLE

rollno	name	branch
1	Akon	CSE
2	Bkon	CSE
3	Ckon	CSE
4	Dkon	CSE

BRANCH TABLE

branch	hod	office_tel
CSE	Mr. Y	53337

Updation problem solved.

NORMALIZATION

- It Divides data into separate independent logical entities and relating them with common key
- It can be achieved in multiple ways:
- Three basic Normal Form
  - 1NF
  - 2NF
  - 3NF
- Advance are:
  - BCNF (Higher version of 3NF)
  - 4NF
  - 5NF

FIRST NORMAL FORM (1NF)

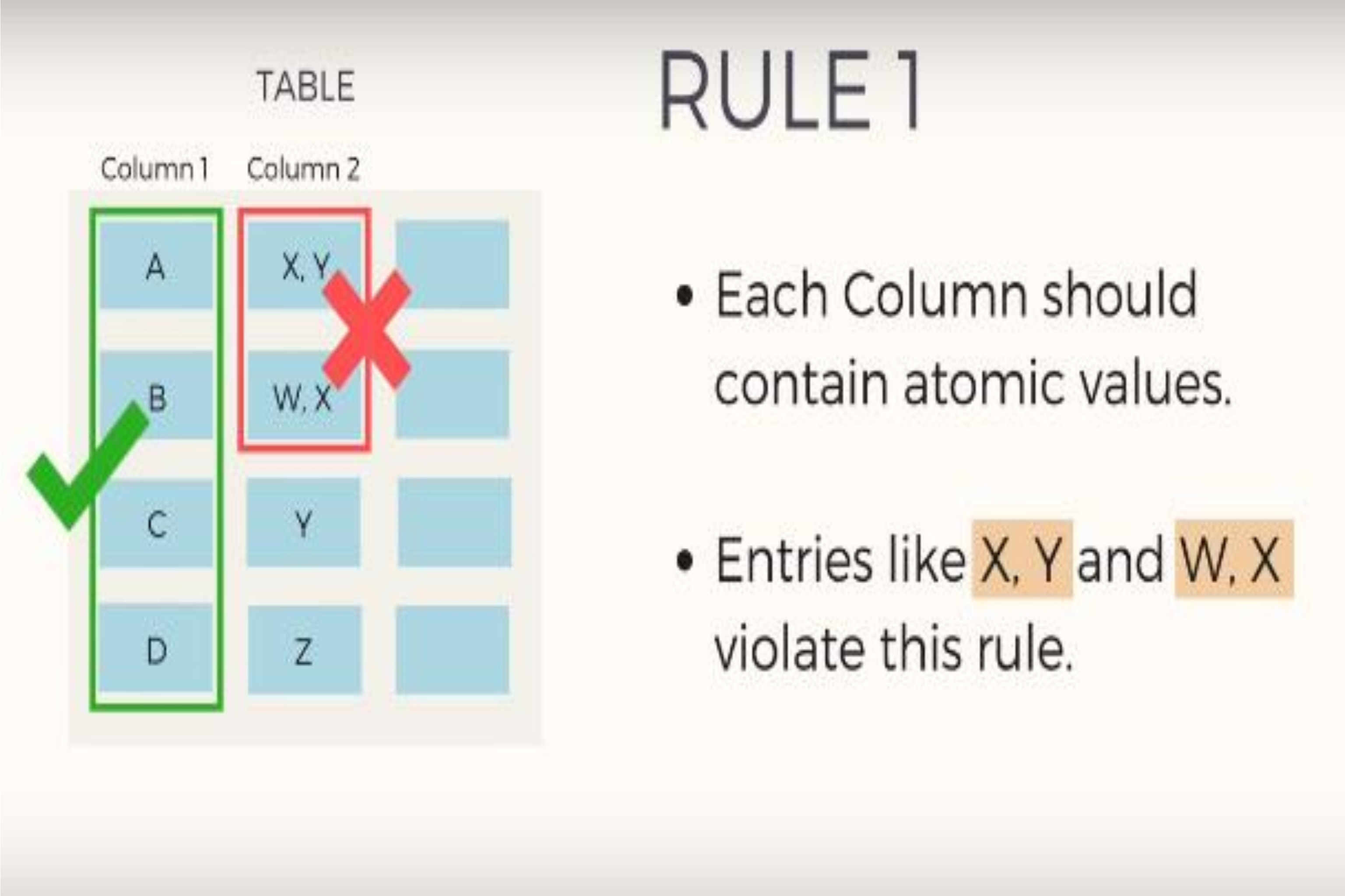
- For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

FIRST NORMAL FORM (1NF)

- Every table in your database should at least be in the 1NF or else it can be considered as BAD database.

FIRST NORMAL FORM (1NF)



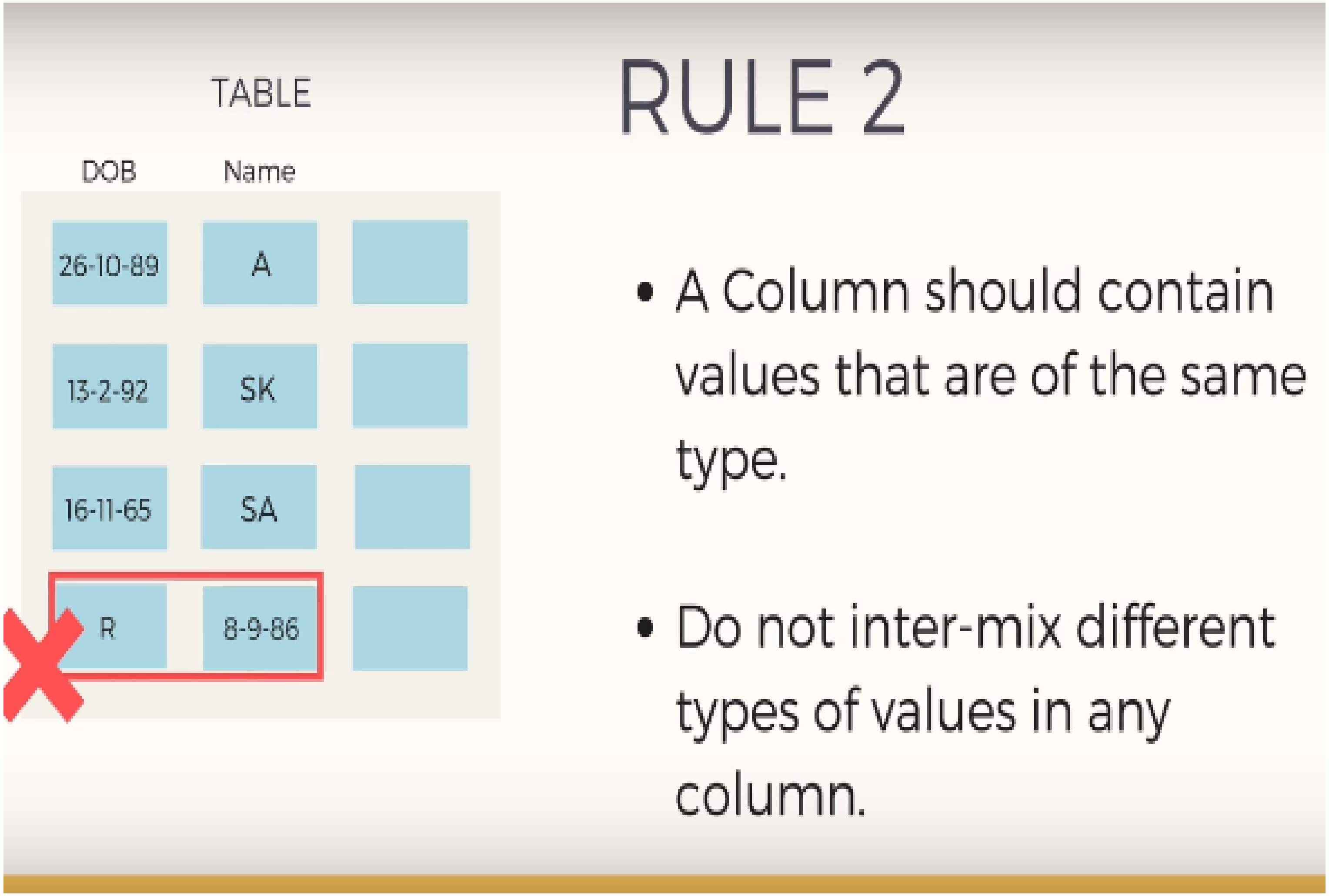
TABLE

Column 1	Column 2	
A	X, Y	
B	W, X	
C	Y	
D	Z	

RULE 1

- Each Column should contain atomic values.
- Entries like X, Y and W, X violate this rule.

FIRST NORMAL FORM (1NF)



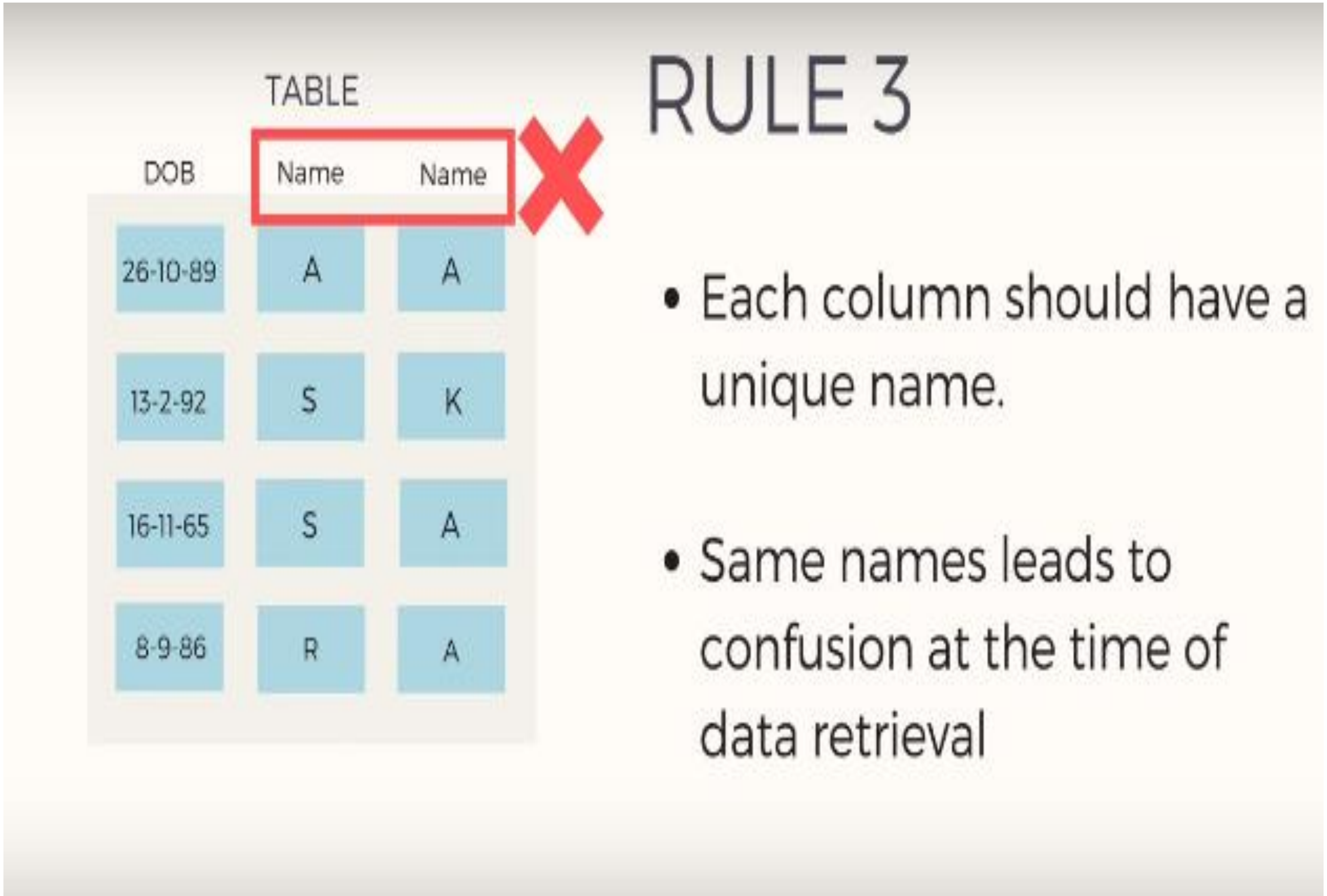
TABLE

DOB	Name	
26-10-89	A	
13-2-92	SK	
16-11-65	SA	
R	8-9-86	

RULE 2

- A Column should contain values that are of the same type.
- Do not inter-mix different types of values in any column.

FIRST NORMAL FORM (1NF)



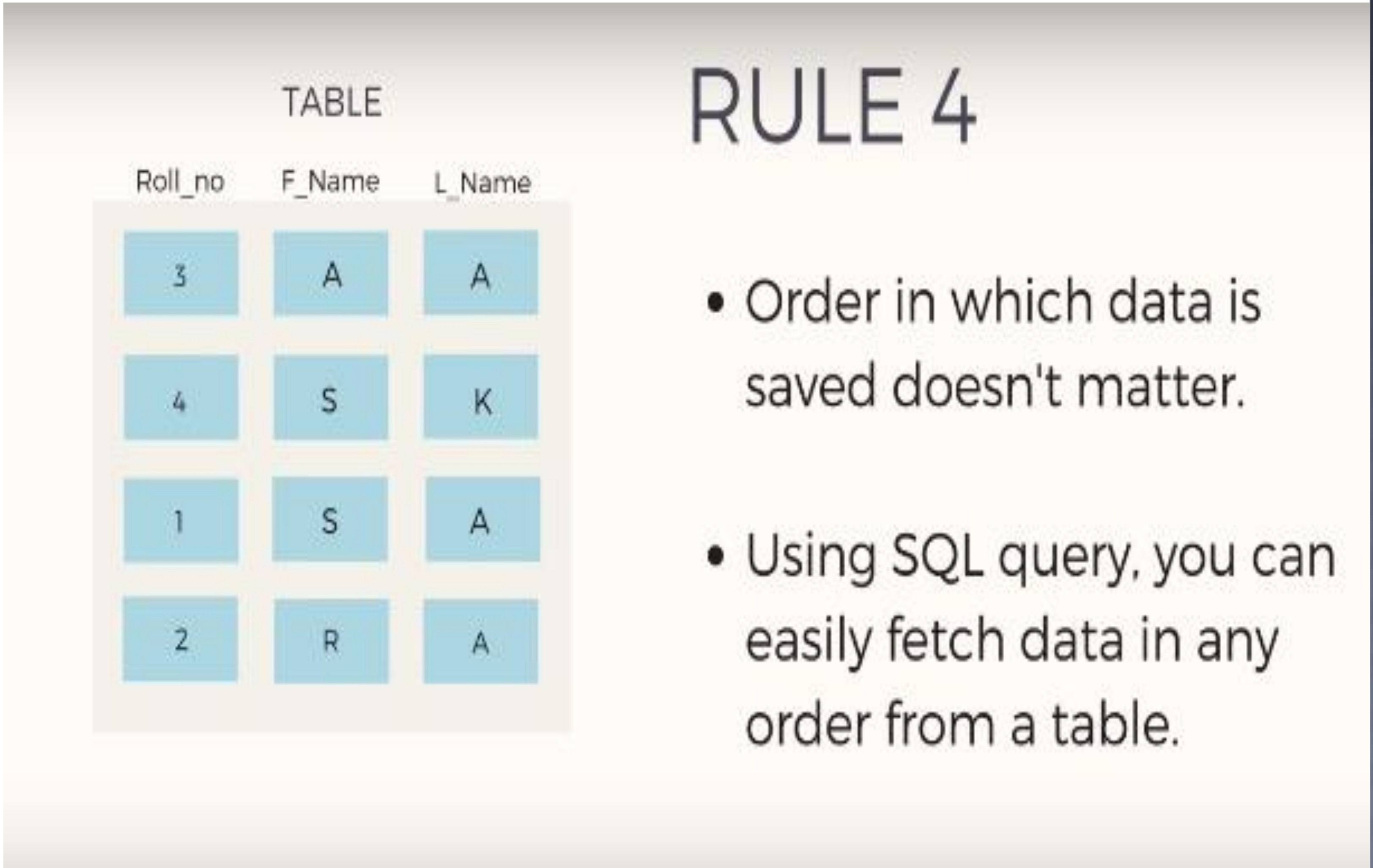
RULE 3

TABLE

DOB	Name	Name
26-10-89	A	A
13-2-92	S	K
16-11-65	S	A
8-9-86	R	A

- Each column should have a unique name.
- Same names leads to confusion at the time of data retrieval

FIRST NORMAL FORM (1NF)



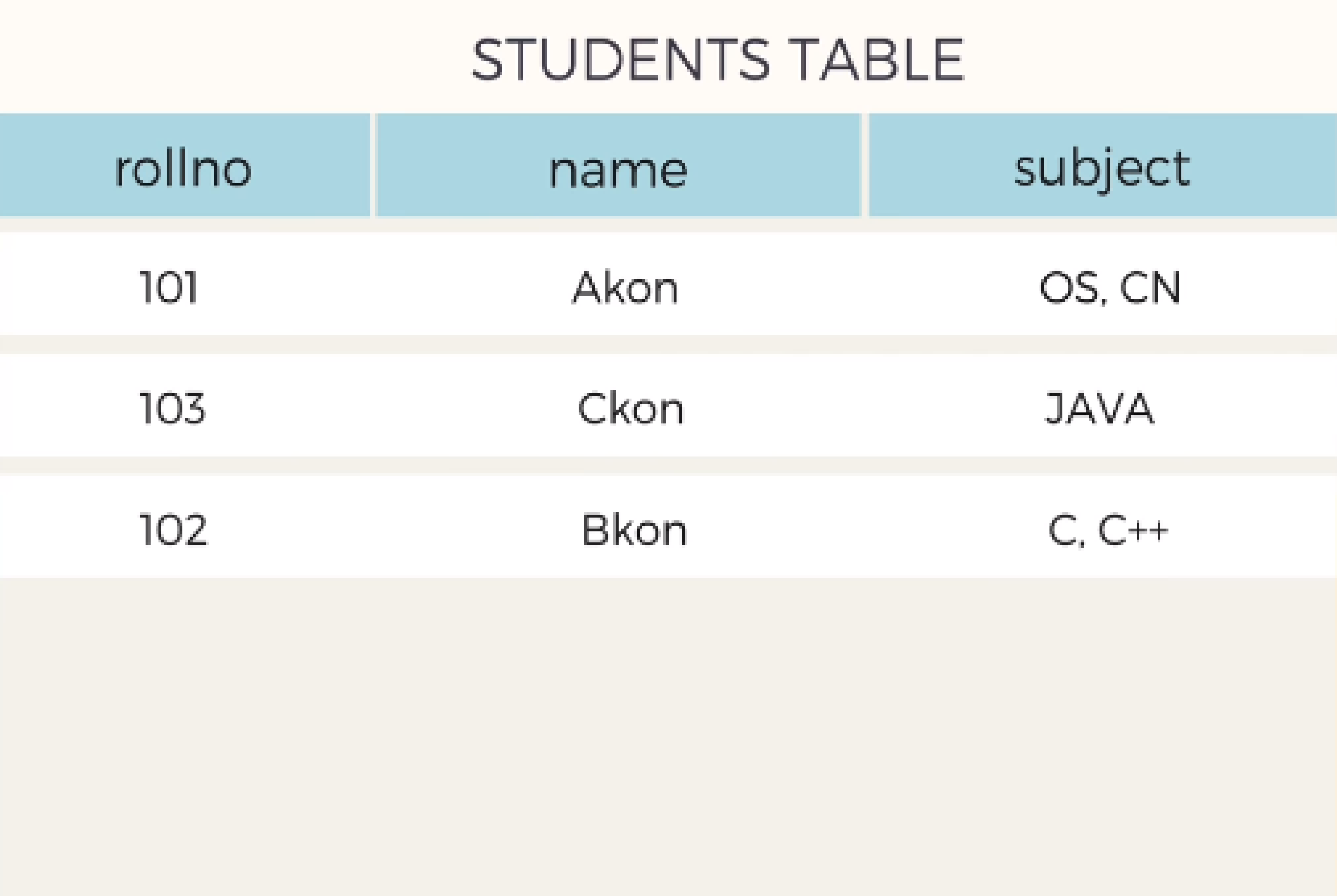
TABLE

Roll_no	F_Name	L_Name
3	A	A
4	S	K
1	S	A
2	R	A

RULE 4

- Order in which data is saved doesn't matter.
- Using SQL query, you can easily fetch data in any order from a table.

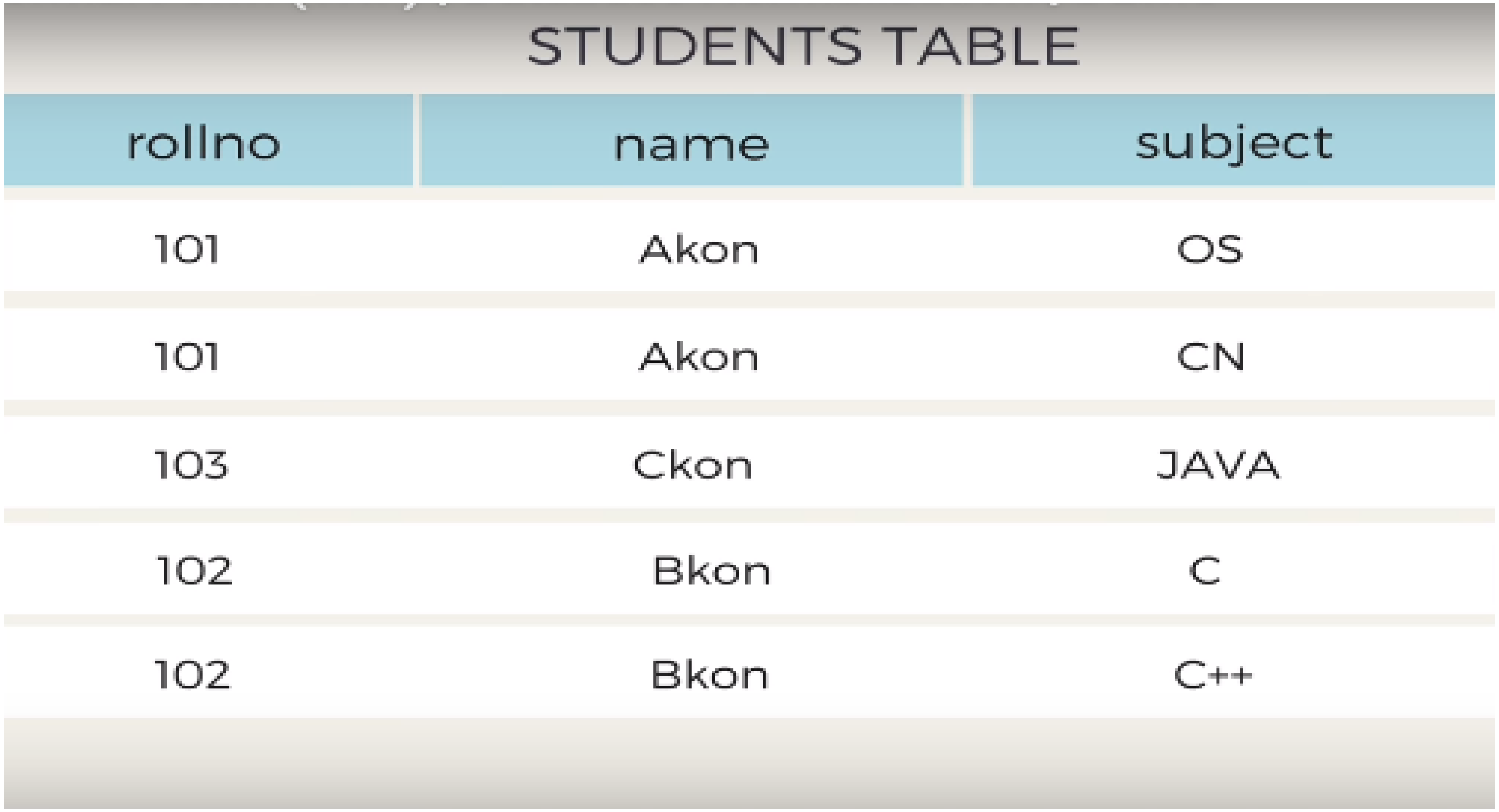
EXAMPLE



STUDENTS TABLE

rollno	name	subject
101	Akon	OS, CN
103	Ckon	JAVA
102	Bkon	C, C++

1 NF



STUDENTS TABLE

rollno	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	JAVA
102	Bkon	C
102	Bkon	C++

SECOND NORMAL FORM (2NF)

- For a table to be in the Second Normal Form, it must satisfy two conditions:
  1. The table should be in the First Normal Form.
  2. There should be no Partial Dependency.

DEPENDENCY

- Functional Dependency
  - We say an attribute, B, has a *functional dependency* on another attribute, A, if for any two records, which have
  - the same value for A, then the values for B in these two records must be the same. We illustrate this as:
    - A → B
- 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.

DEPENDENCY

- Transitive Dependency
  - Consider attributes A, B, and C, and where

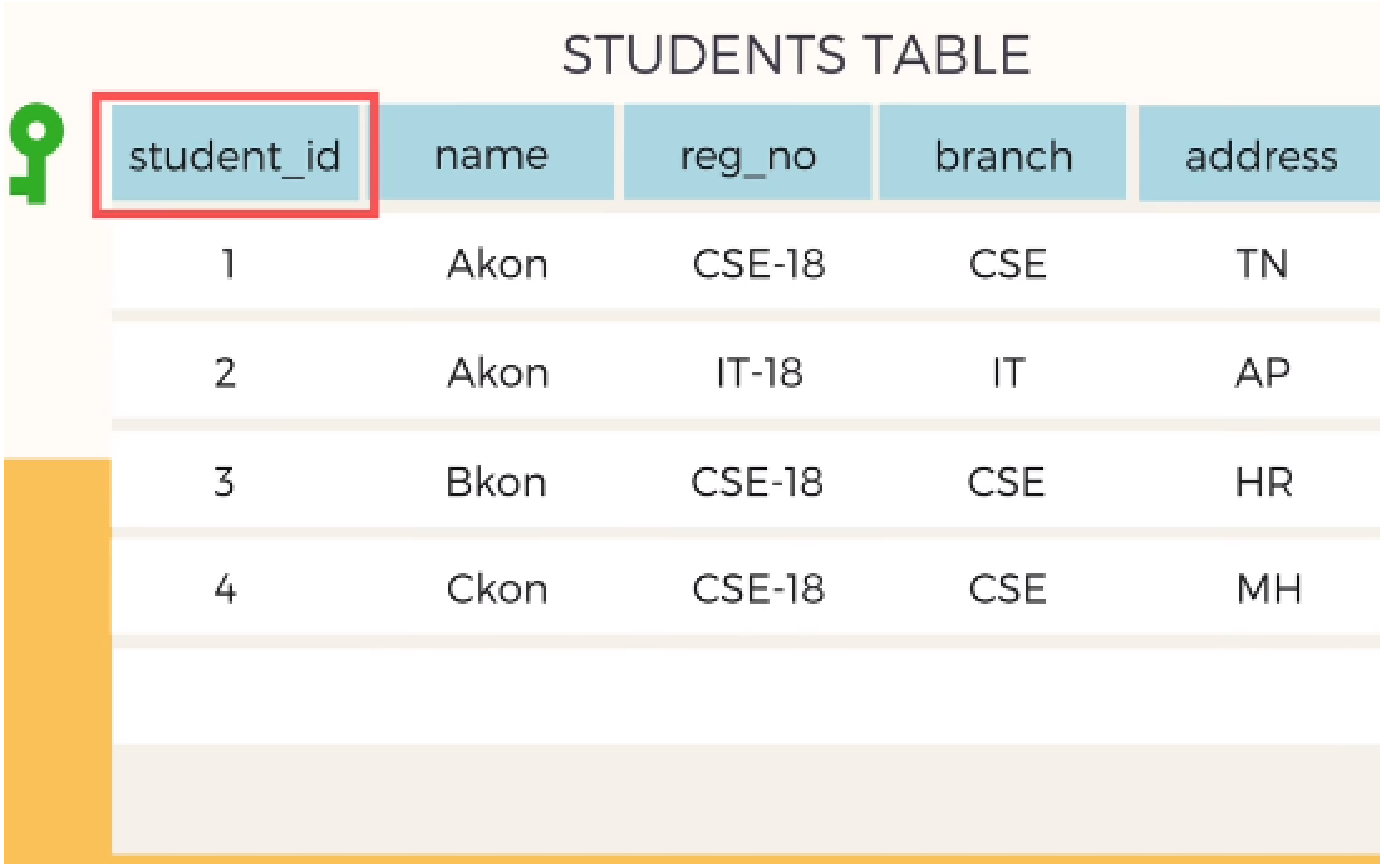
    A → B and B → C.

  - Functional dependencies are transitive, which means that we also have the functional dependency

    A → C

  - We say that C is transitively dependent on A through B.

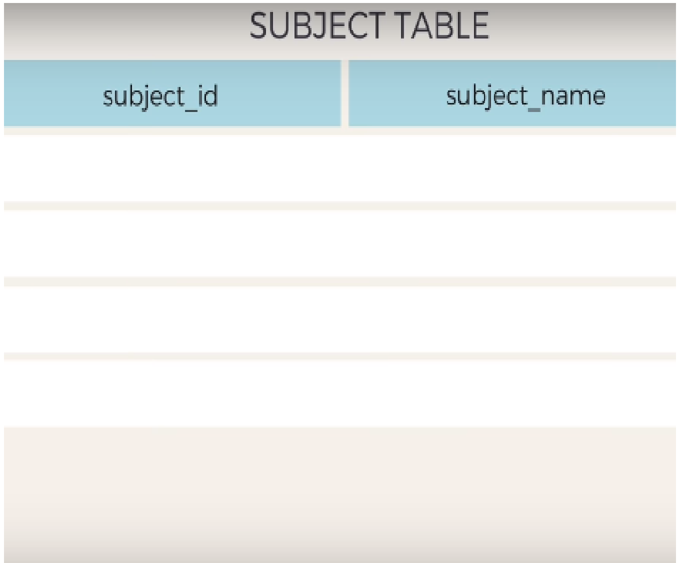
EXAMPLE



STUDENTS TABLE

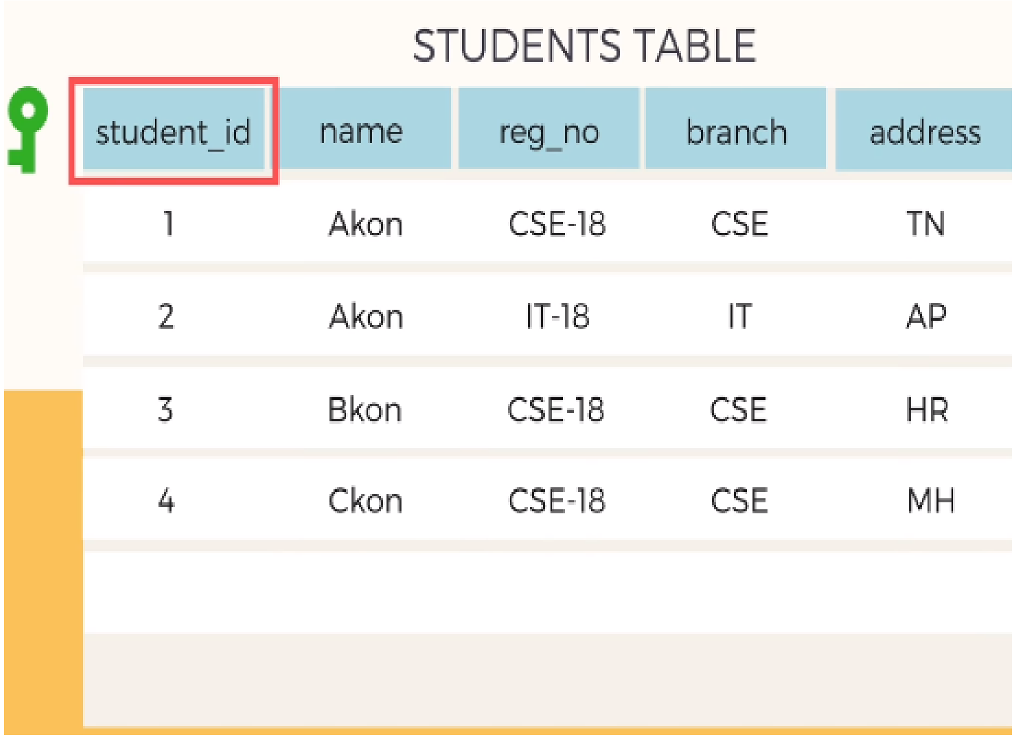
student_id	name	reg_no	branch	address
1	Akon	CSE-18	CSE	TN
2	Akon	IT-18	IT	AP
3	Bkon	CSE-18	CSE	HR
4	Ckon	CSE-18	CSE	MH

Now let's extend our example to see if two or more columns together can act as a primary key.



SUBJECT TABLE

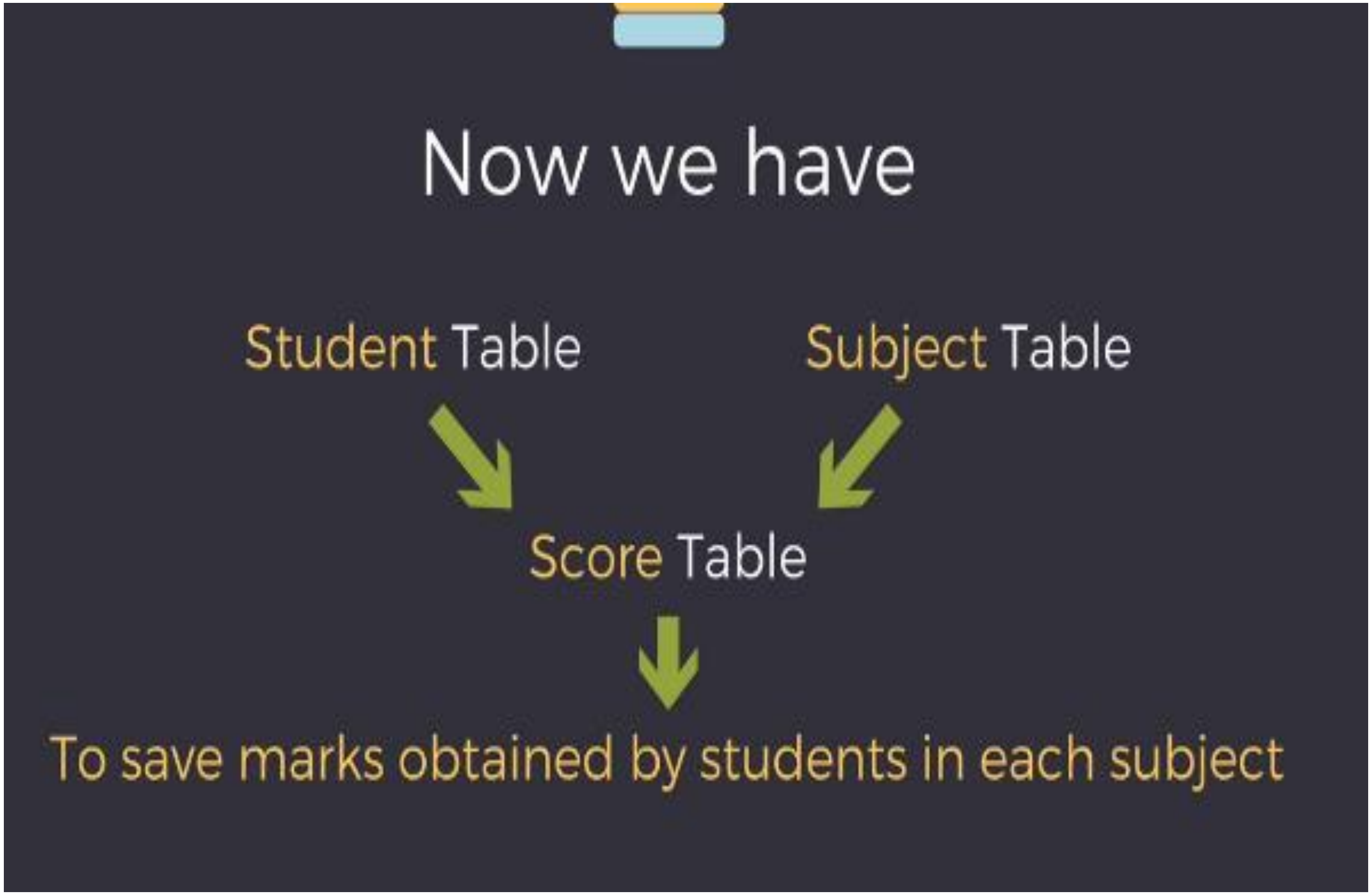
subject_id	subject_name



STUDENTS TABLE

student_id	name	reg_no	branch	address
1	Akon	CSE-18	CSE	TN
2	Akon	IT-18	IT	AP
3	Bkon	CSE-18	CSE	HR
4	Ckon	CSE-18	CSE	MH

EXAMPLE



Now we have

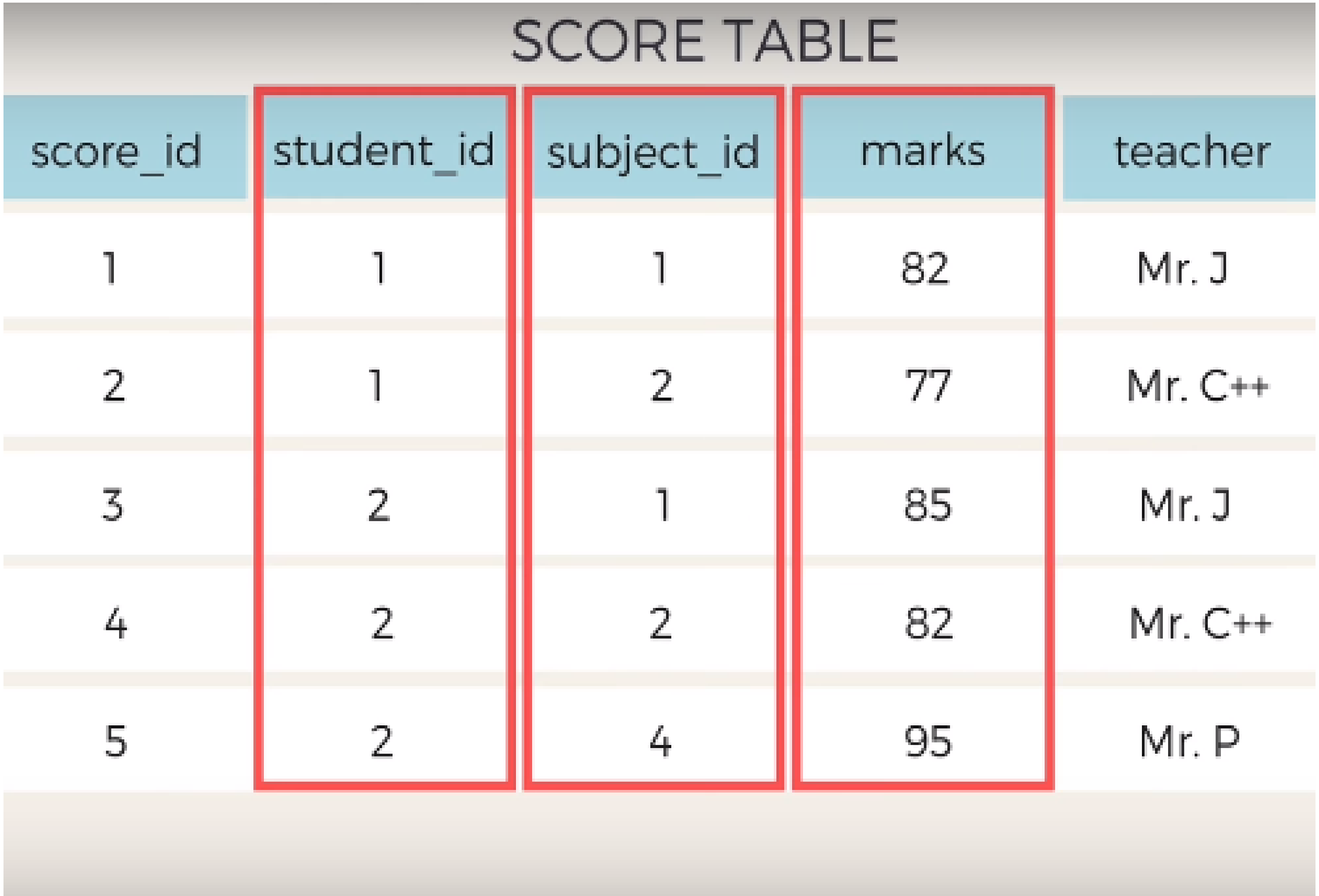
Student Table

Subject Table

Score Table

To save marks obtained by students in each subject

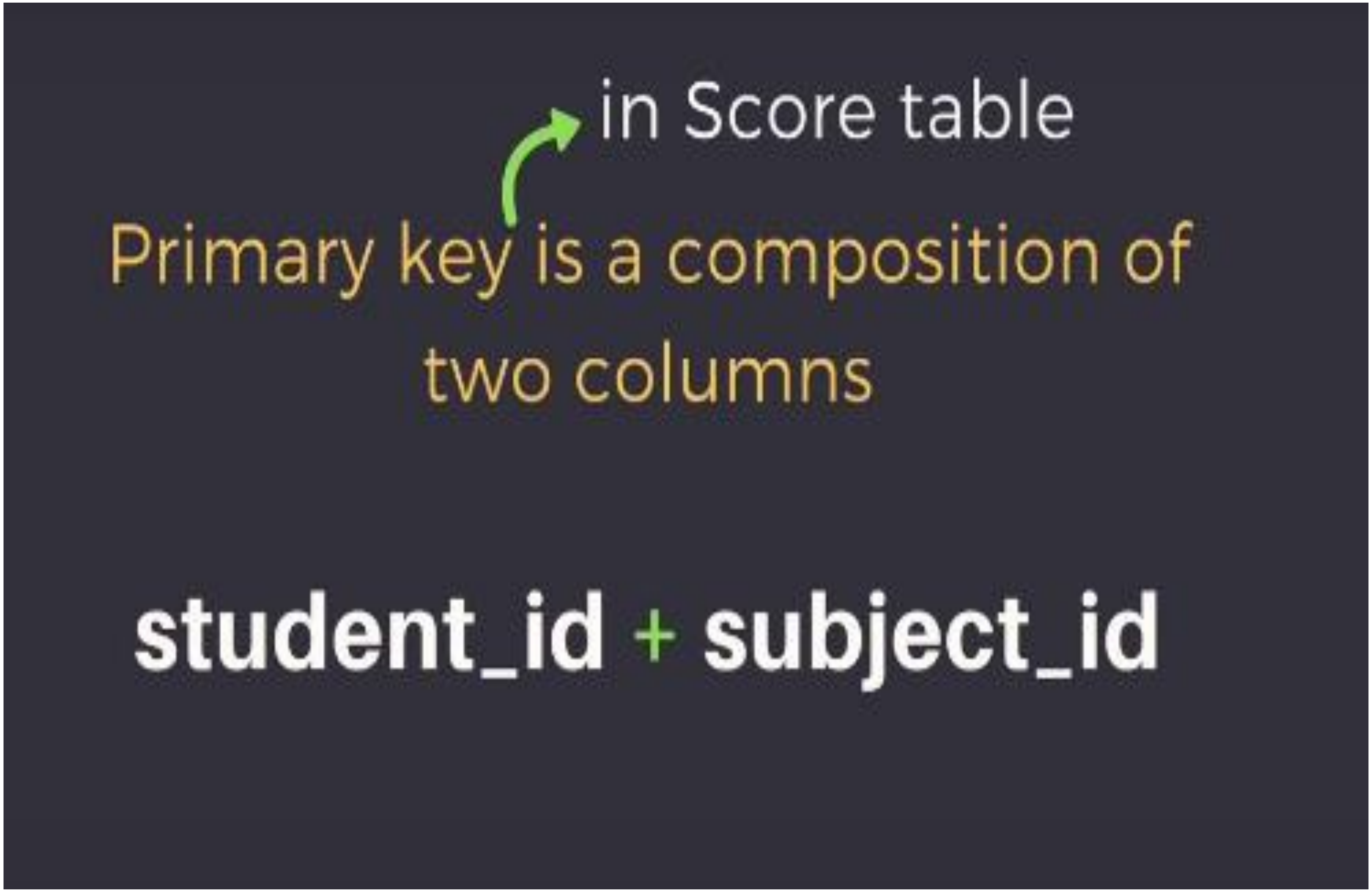
EXAMPLE



SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	1	1	82	Mr. J
2	1	2	77	Mr. C++
3	2	1	85	Mr. J
4	2	2	82	Mr. C++
5	2	4	95	Mr. P

EXAMPLE

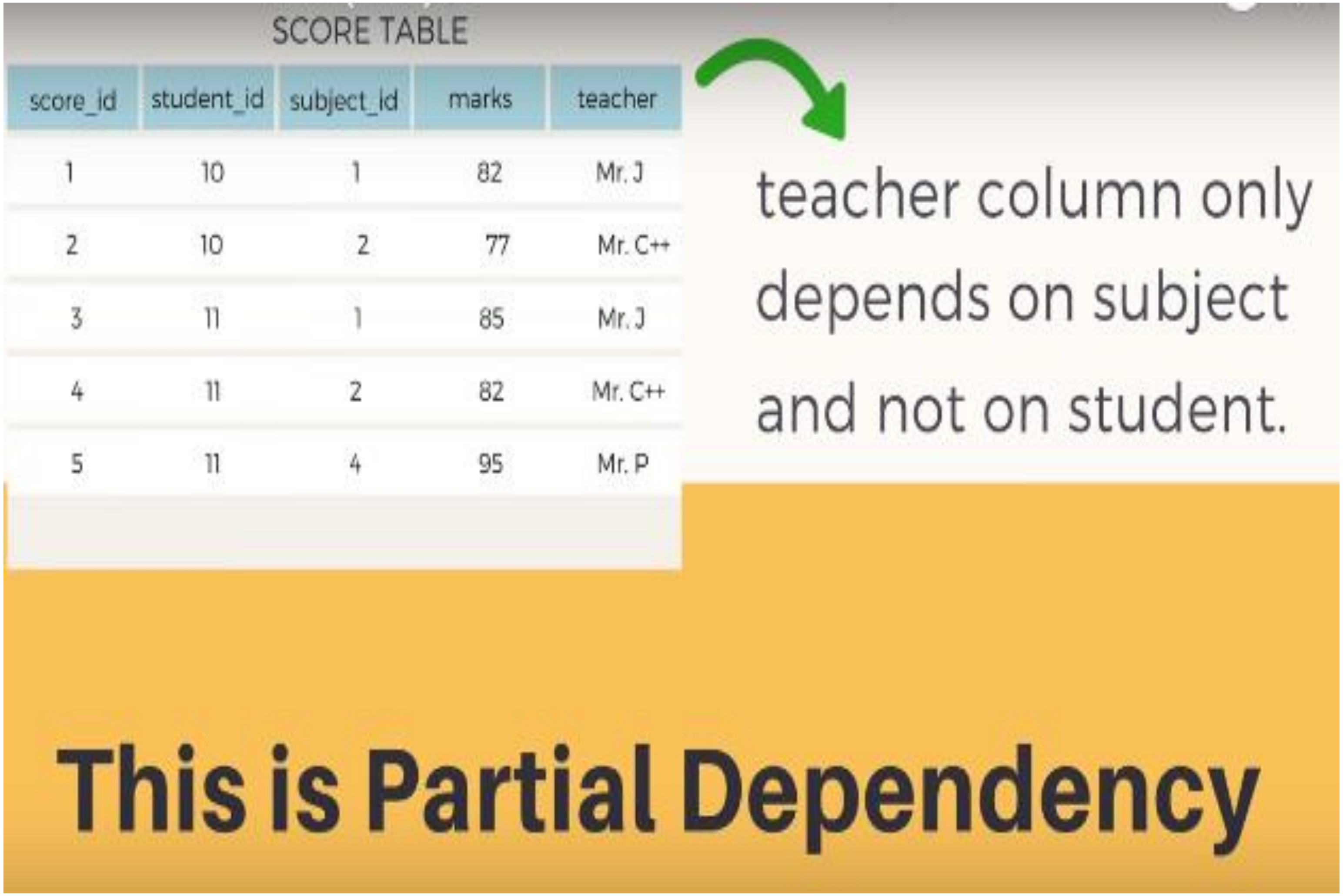


in Score table

Primary key is a composition of two columns

student_id + subject_id

EXAMPLE



SCORE TABLE

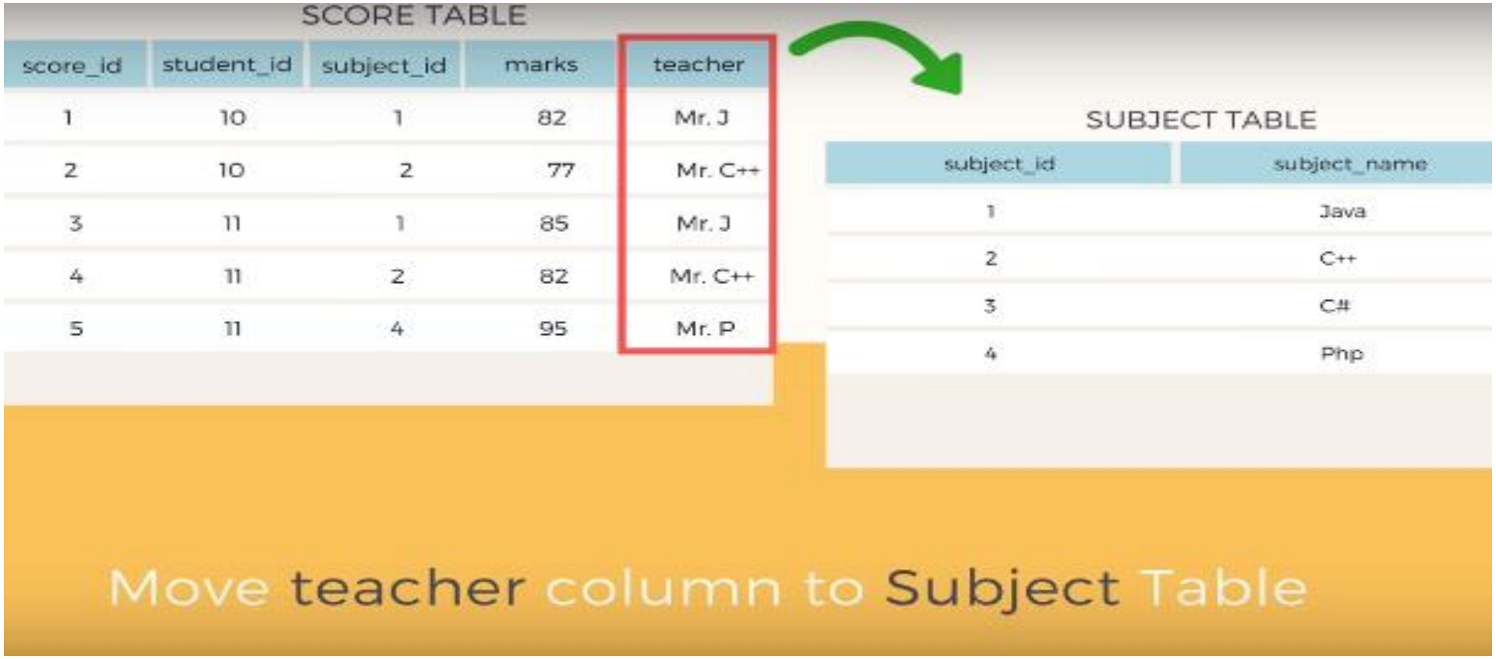
score_id	student_id	subject_id	marks	teacher
1	10	1	82	Mr. J
2	10	2	77	Mr. C++
3	11	1	85	Mr. J
4	11	2	82	Mr. C++
5	11	4	95	Mr. P

teacher column only depends on subject and not on student.

This is Partial Dependency

EXAMPLE

- How to remove partial dependency
- In example we should remove teacher column from score table to remove partial dependency.



SCORE TABLE

score_id	student_id	subject_id	marks	teacher
1	10	1	82	Mr. J
2	10	2	77	Mr. C++
3	11	1	85	Mr. J
4	11	2	82	Mr. C++
5	11	4	95	Mr. P

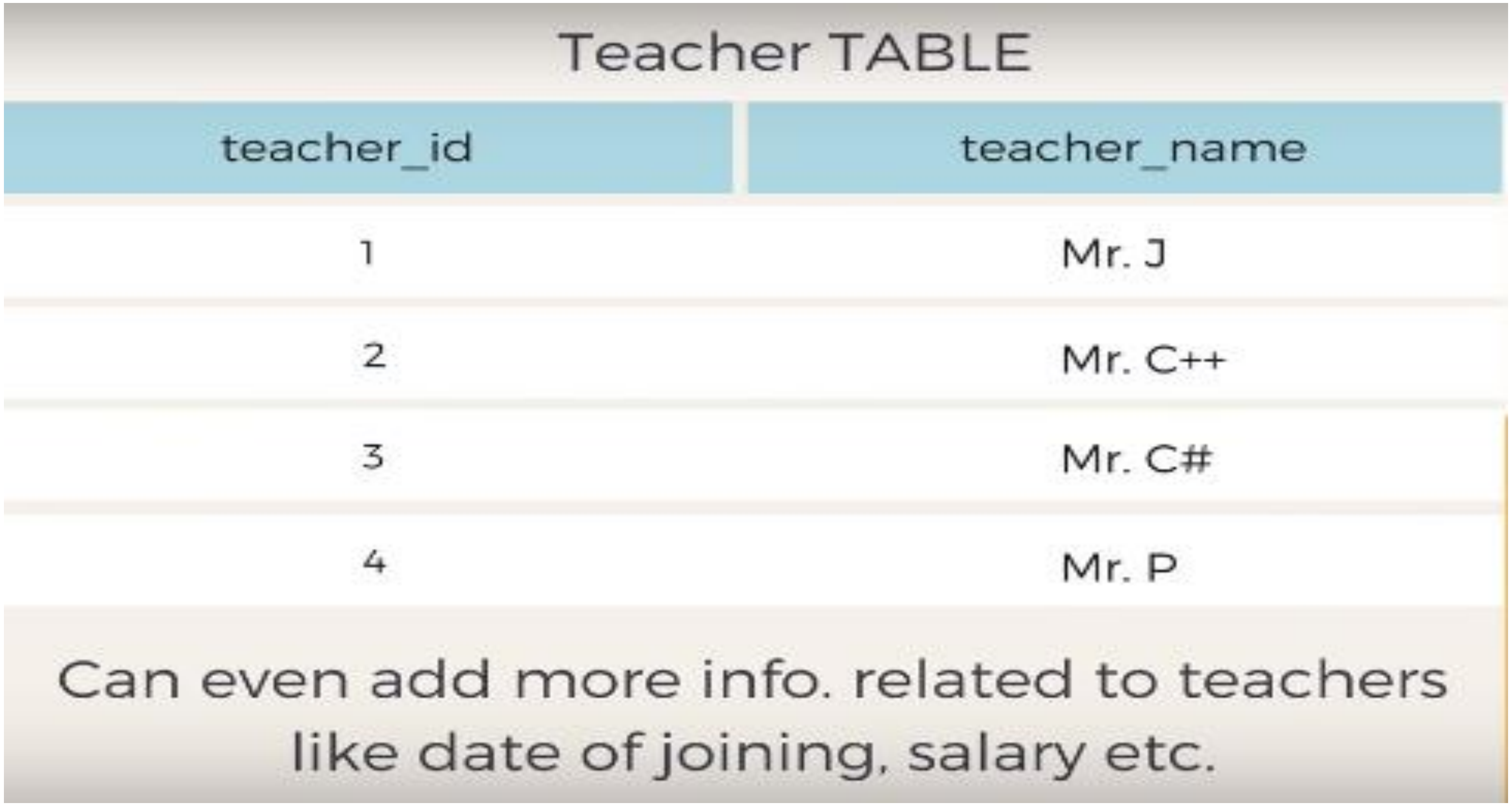
SUBJECT TABLE

subject_id	subject_name
1	Java
2	C++
3	C#
4	Php

Move teacher column to Subject Table

EXAMPLE

- Or we can create new Teacher table and add teachers information here.



Teacher TABLE

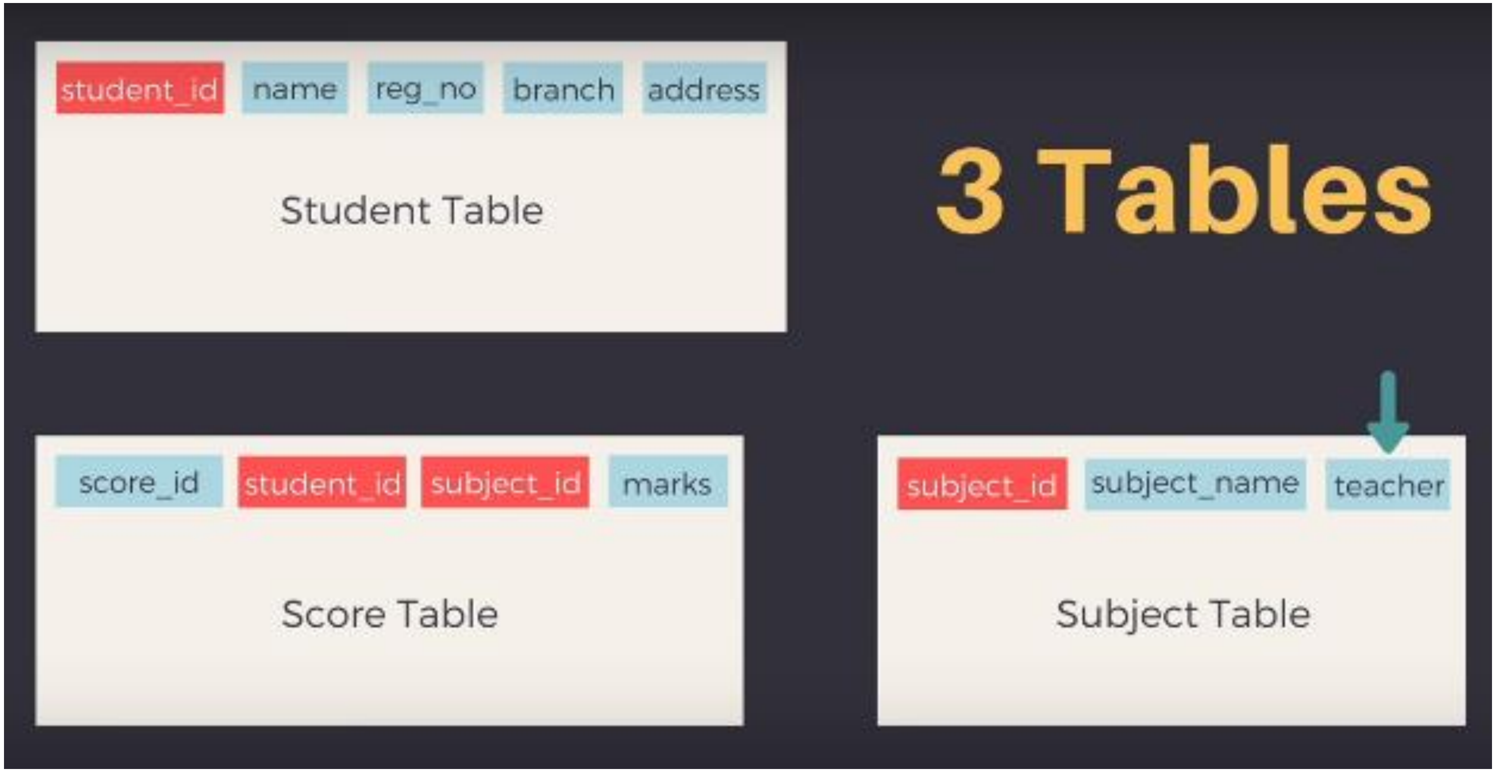
teacher_id	teacher_name
1	Mr. J
2	Mr. C++
3	Mr. C#
4	Mr. P

Can even add more info. related to teachers like date of joining, salary etc.

THIRD NORMAL FORM (3NF)

- It should be in the 2NF.
- And it should not have Transitive Dependency

THIRD NORMAL FORM (3NF)



student_id name reg_no branch address

Student Table

3 Tables

score_id student_id subject_id marks

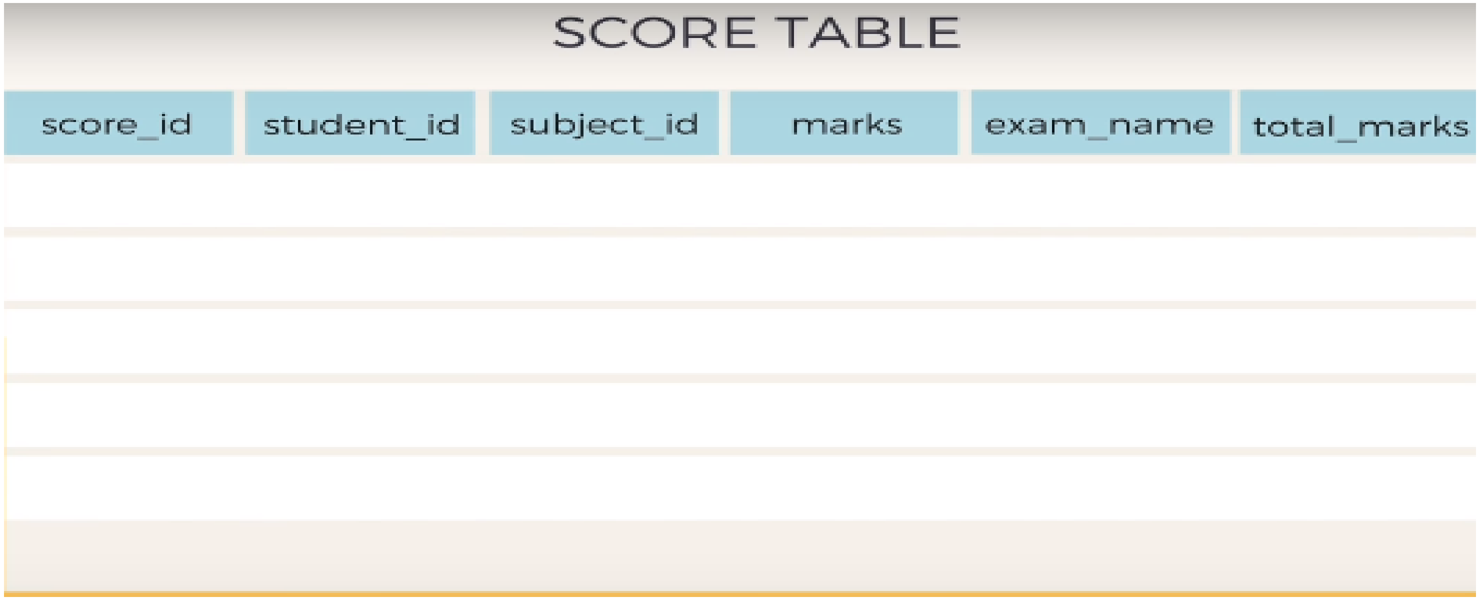
Score Table

subject_id subject_name teacher

Subject Table

EXAMPLE

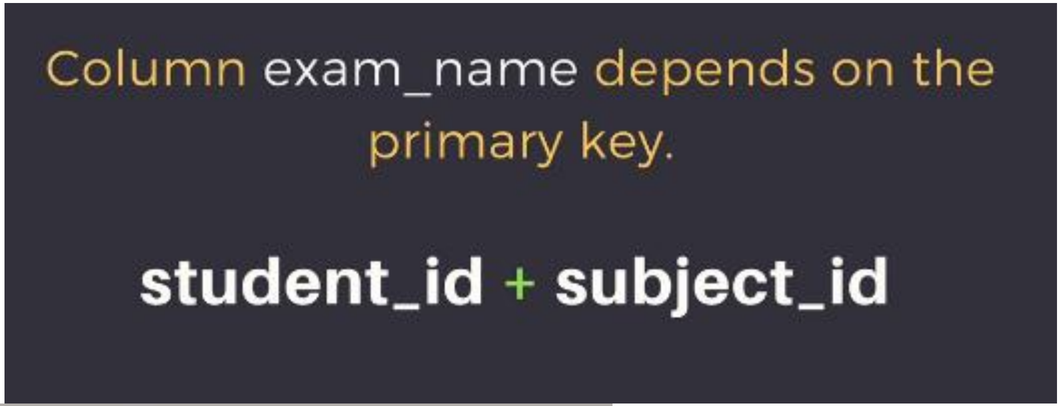
- Score table in 2NF.
- Now we also want to save columns Exam_Name and Total_Marks in score table



SCORE TABLE

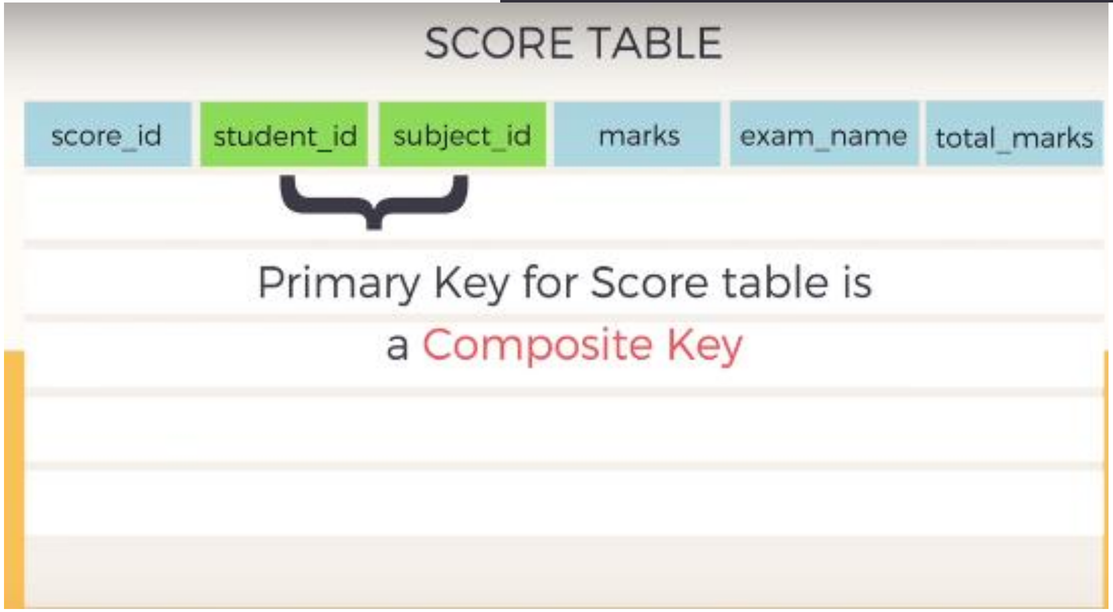
score_id	student_id	subject_id	marks	exam_name	total_marks

EXAMPLE



Column exam_name depends on the primary key.

student_id + subject_id

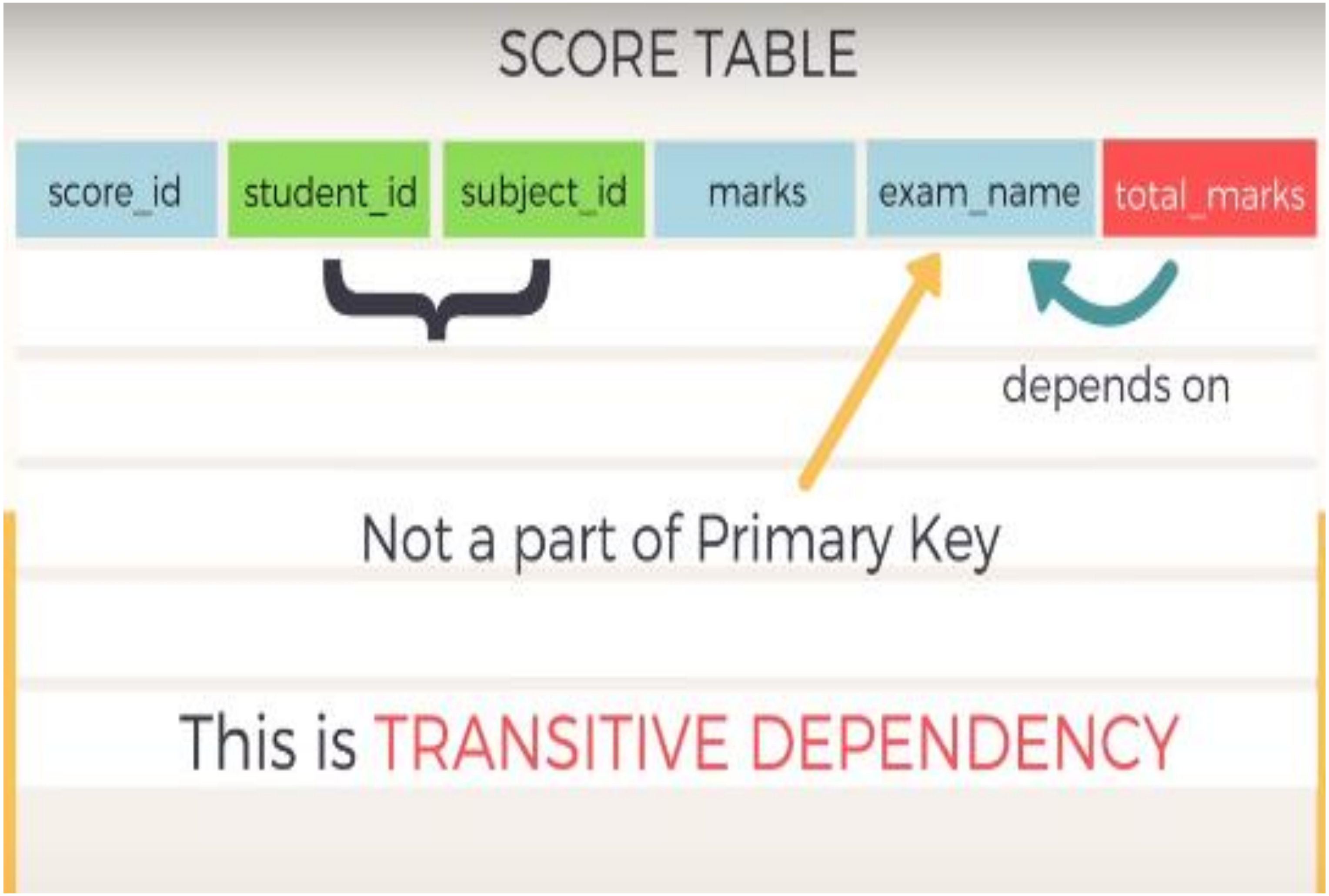


SCORE TABLE

score_id	student_id	subject_id	marks	exam_name	total_marks

Primary Key for Score table is a Composite Key

EXAMPLE



SCORE TABLE

score_id

student_id

subject_id

marks

exam_name

total_marks

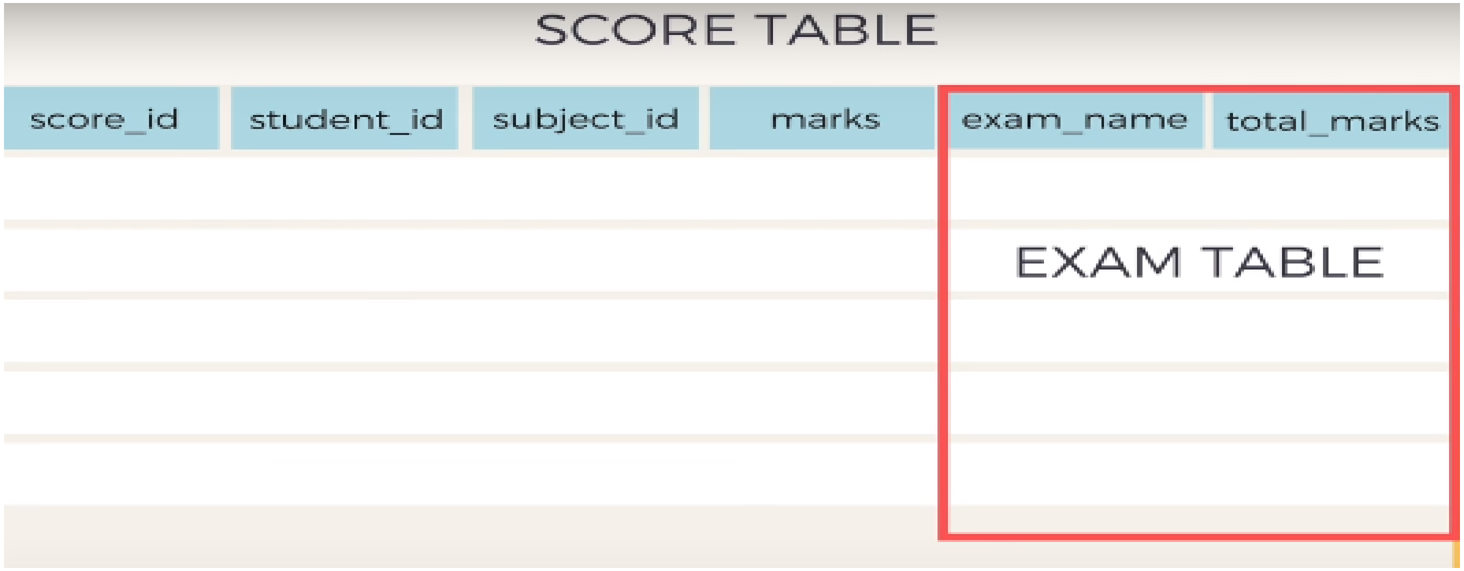
depends on

Not a part of Primary Key

This is TRANSITIVE DEPENDENCY

EXAMPLE

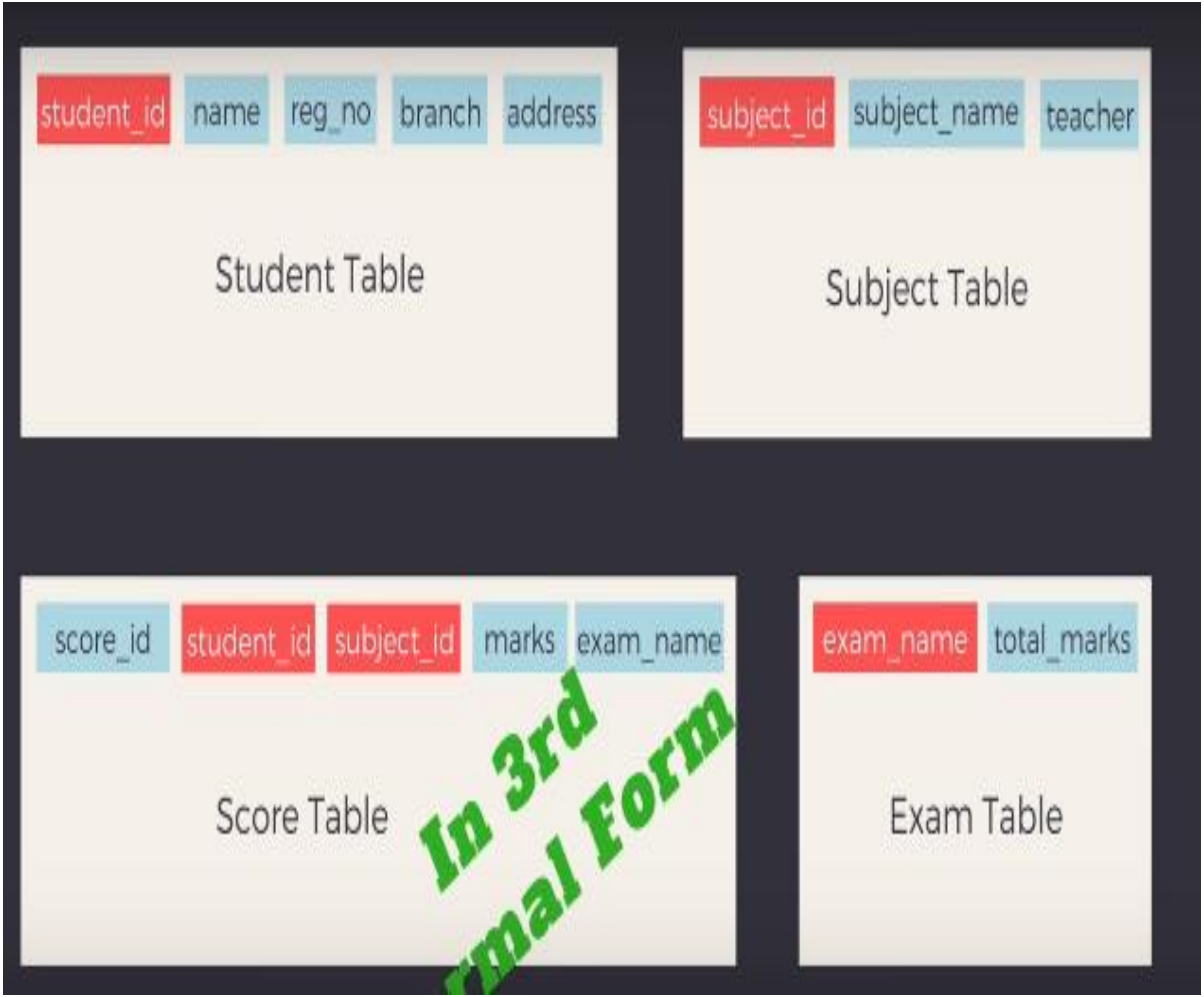
- Solution



SCORE TABLE

score_id	student_id	subject_id	marks	exam_name	total_marks

EXAM TABLE



Student Table

student_id | name | reg_no | branch | address

Subject Table

subject_id | subject_name | teacher

Score Table

score_id | student_id | subject_id | marks | exam_name

Exam Table

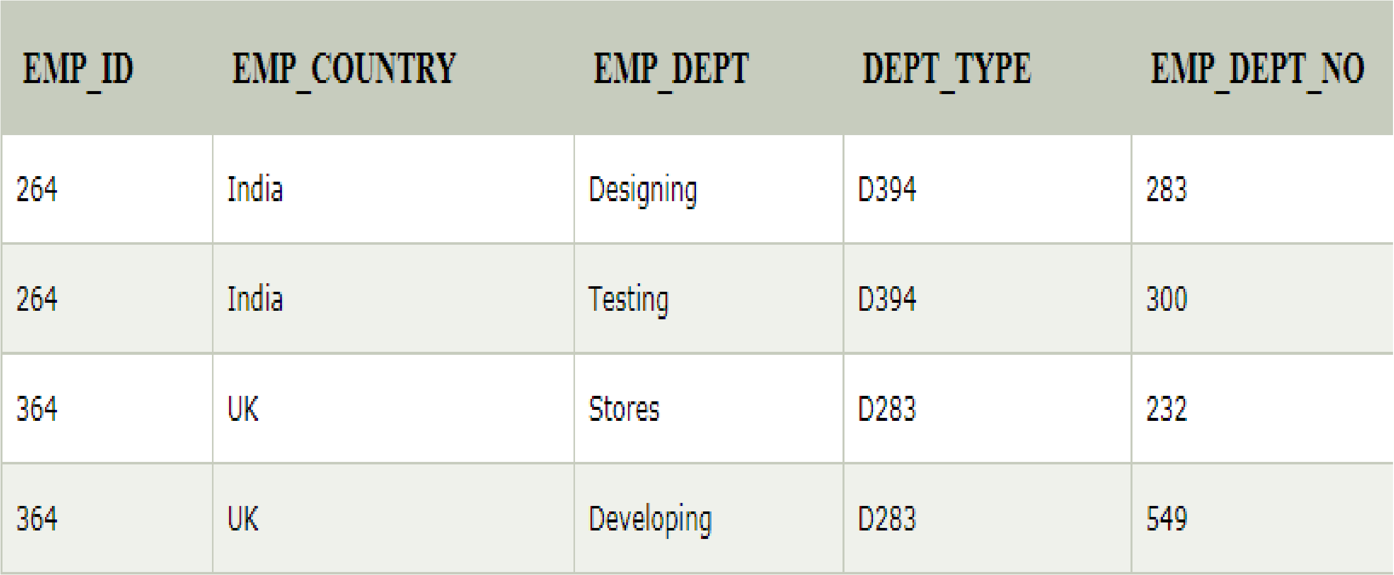
exam_name | total_marks

BOYCE CODD NORMAL FORM (BCNF/3.5NF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- it means, that for a dependency A → B, A cannot be a **non-prime attribute**, if B is a **prime attribute**.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

BCNF

- **Example:** Let's assume there is a company where employees work in more than one department.
- **In the given table Functional dependencies are as follows:**
- EMP_ID → EMP_COUNTRY
- EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

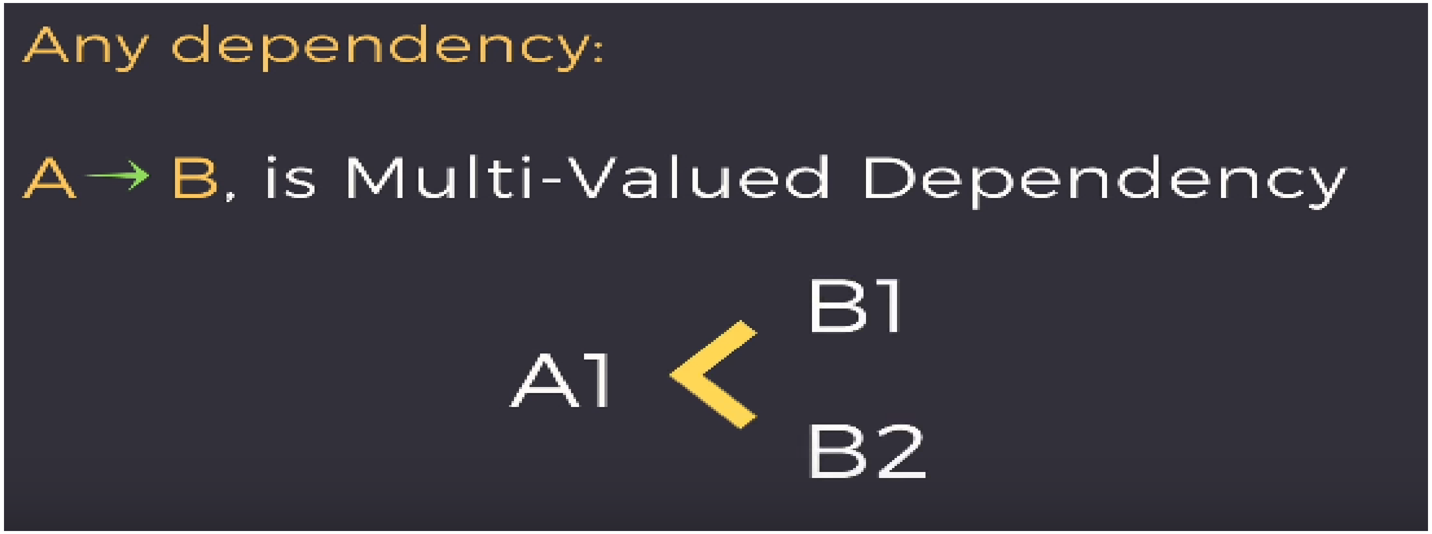


EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

FOURTH NORMAL FORM (4NF)

- A table is said to be in the Fourth Normal Form when,
- It is in the Boyce-Codd Normal Form.
- And, it doesn't have Multi-Valued Dependency.

MULTIVALUED DEPENDENCY

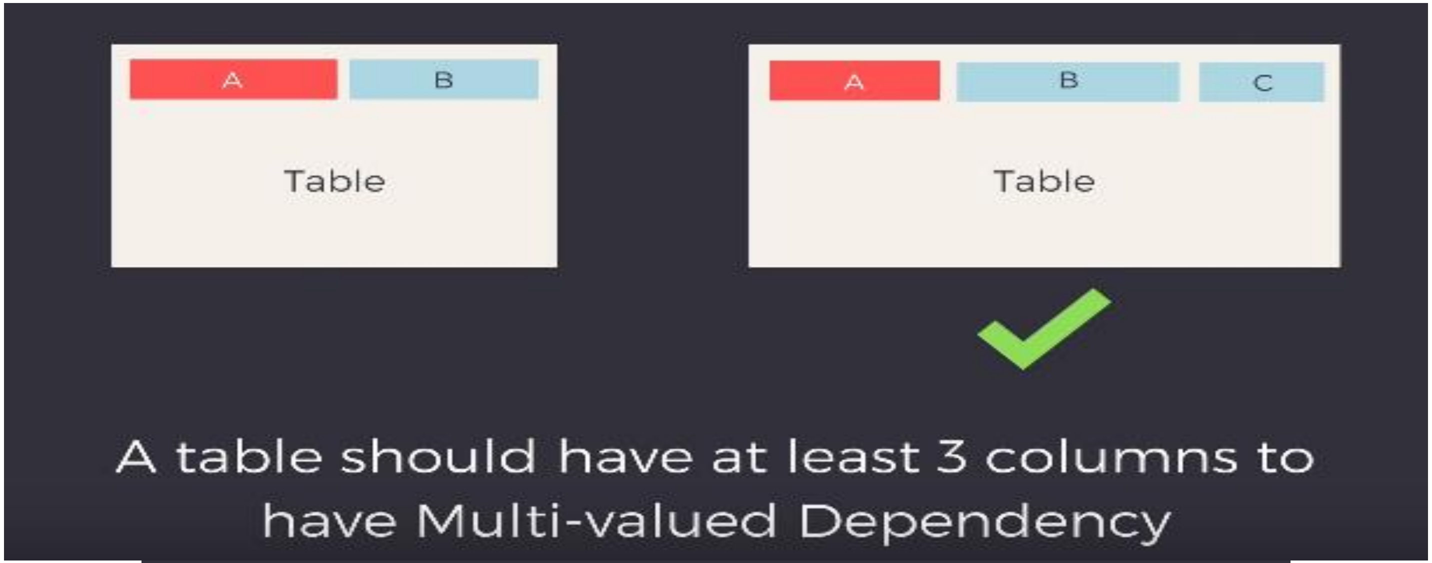


Any dependency:

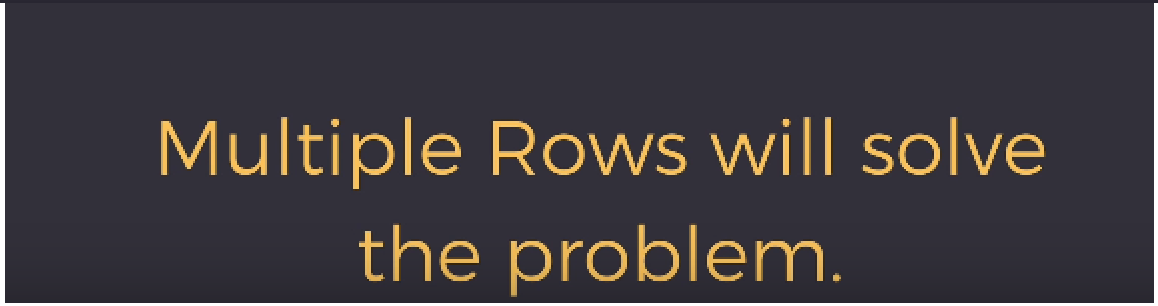
A → B, is Multi-Valued Dependency

A1 < B1 B2

MULTIVALUED DEPENDENCY

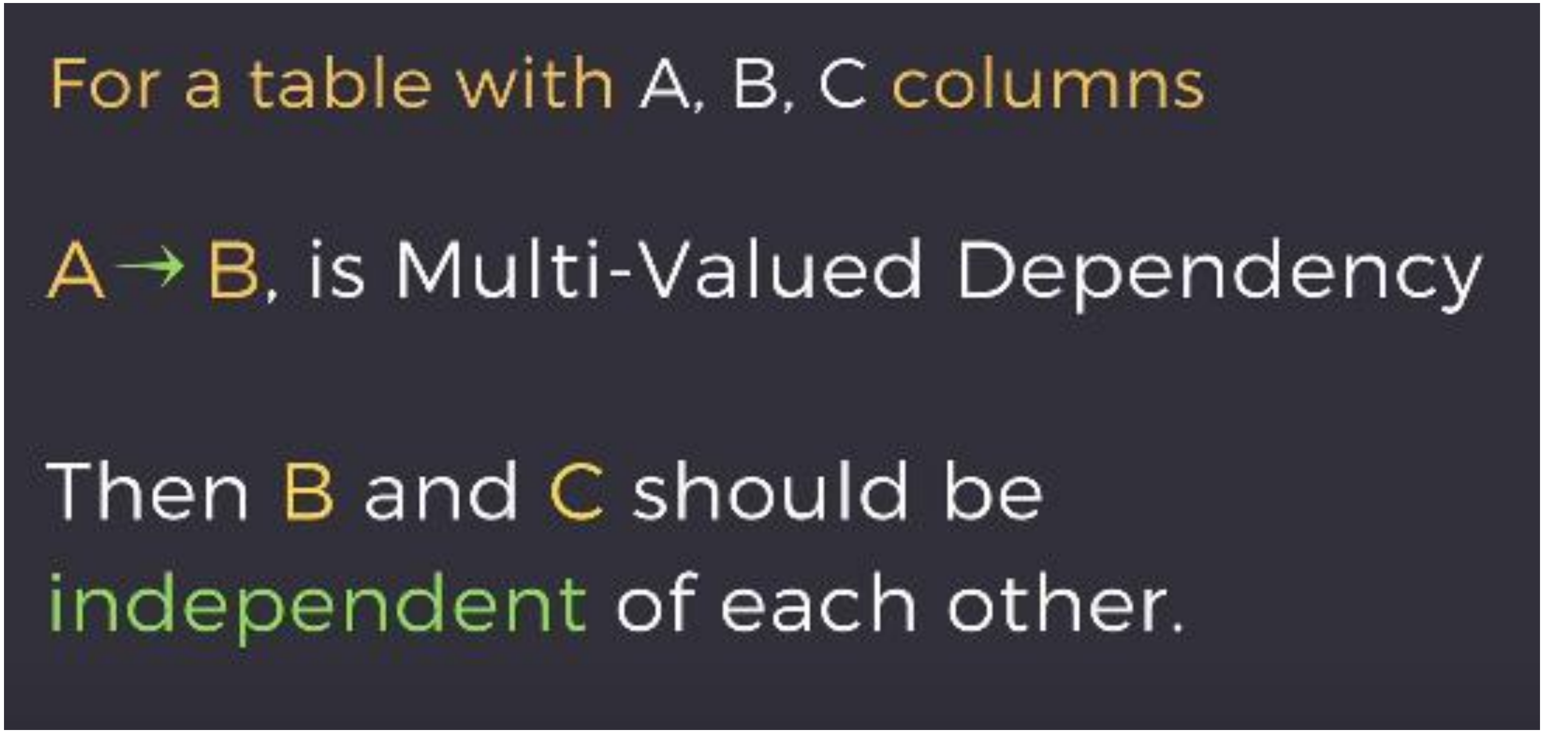


A table should have at least 3 columns to have Multi-valued Dependency



Multiple Rows will solve the problem.

MULTIVALUED DEPENDENCY

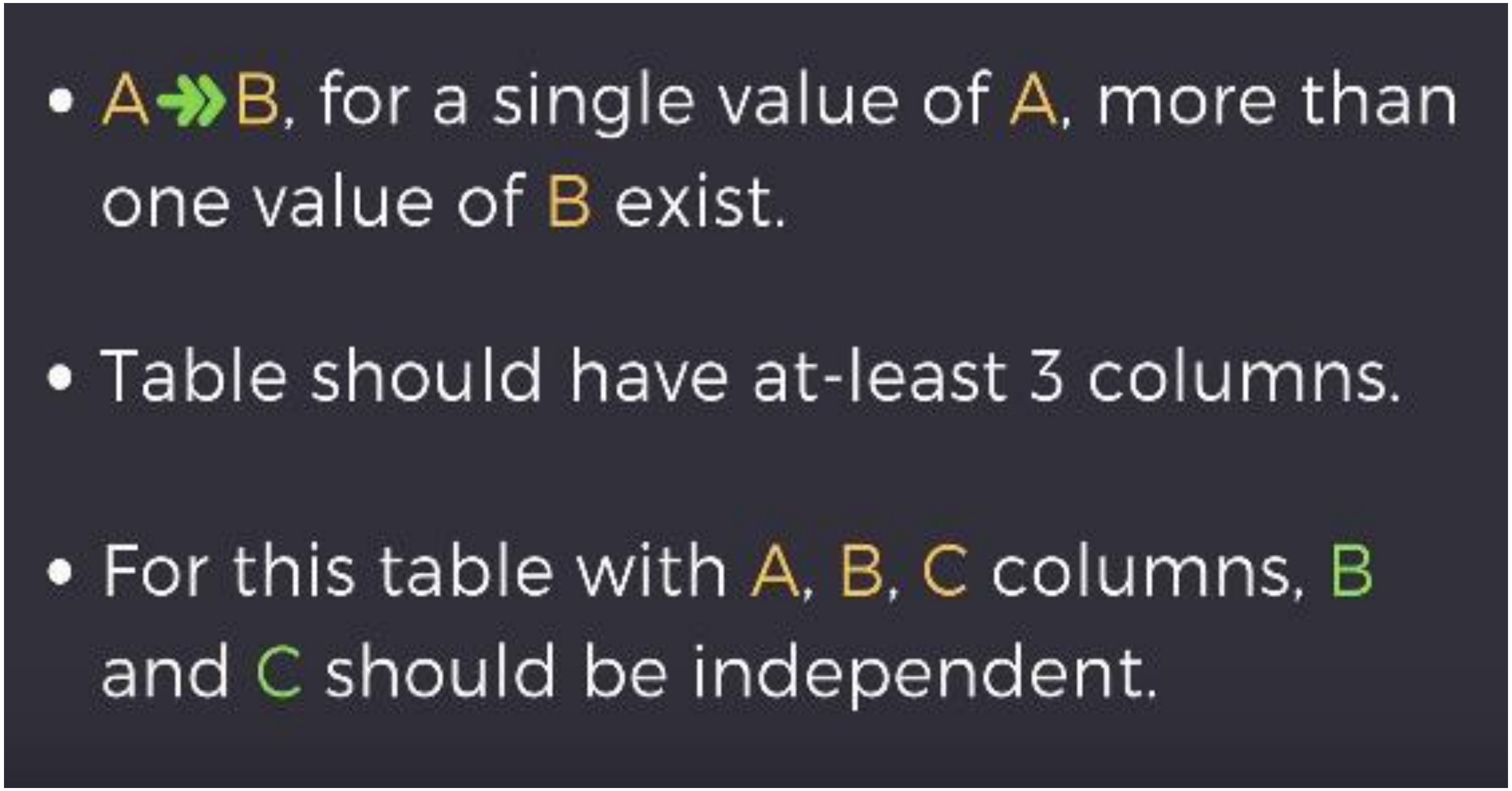


For a table with A, B, C columns

A → B, is Multi-Valued Dependency

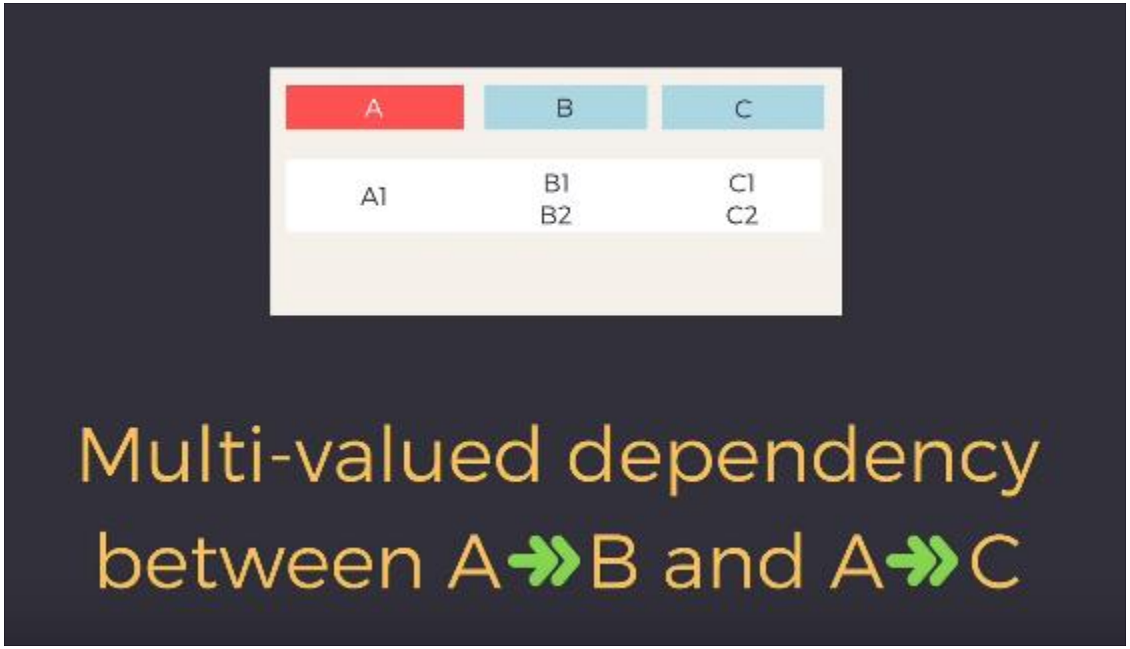
Then B and C should be independent of each other.

MULTIVALUED DEPENDENCY



- $A \twoheadrightarrow B$, for a single value of A, more than one value of B exist.
- Table should have at-least 3 columns.
- For this table with A, B, C columns, B and C should be independent.

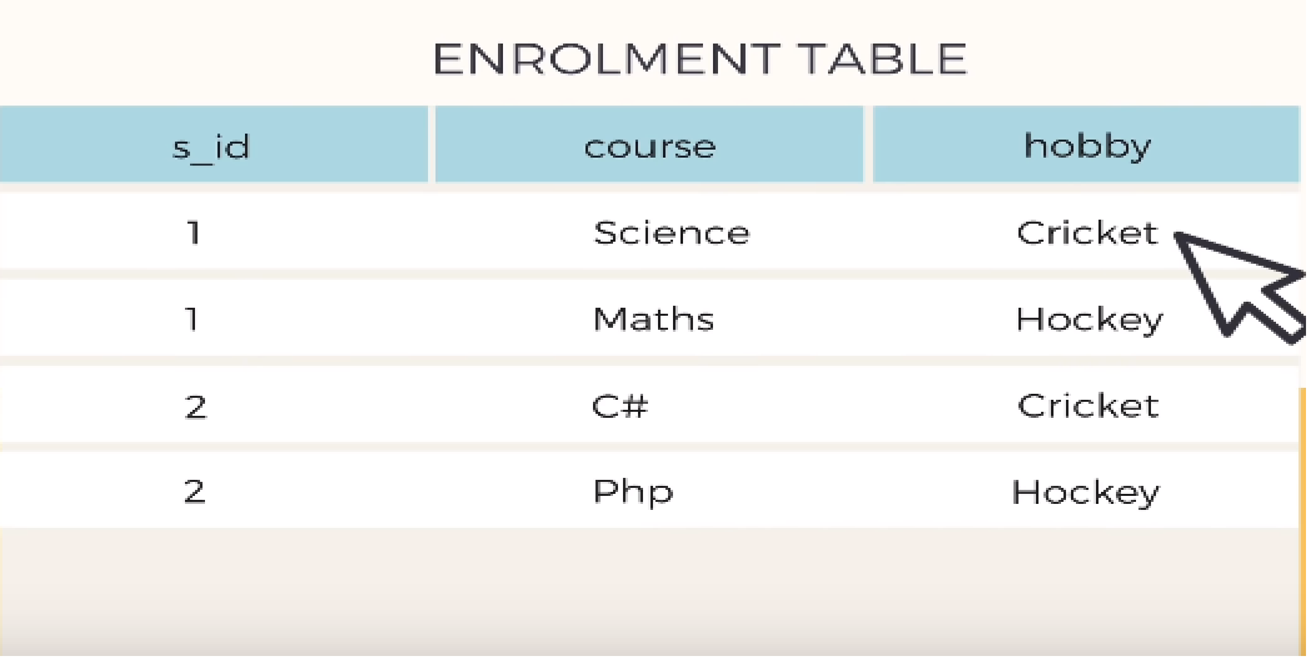
MULTIVALUED DEPENDENCY



A	B	C
A1	B1 B2	C1 C2

Multi-valued dependency between $A \twoheadrightarrow B$ and $A \twoheadrightarrow C$

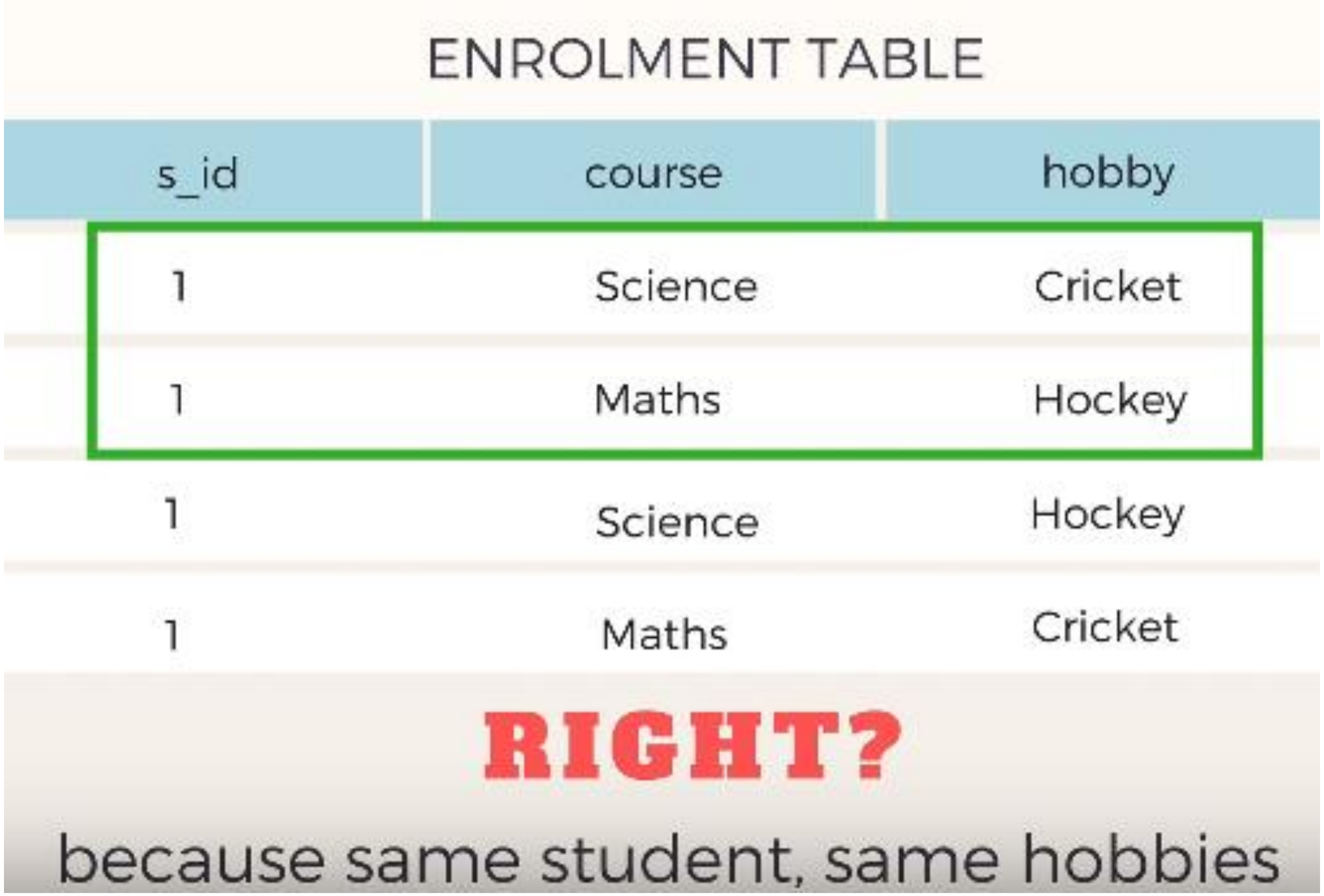
MULTIVALUED DEPENDENCY



ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
2	C#	Cricket
2	Php	Hockey

MULTIVALUED DEPENDENCY



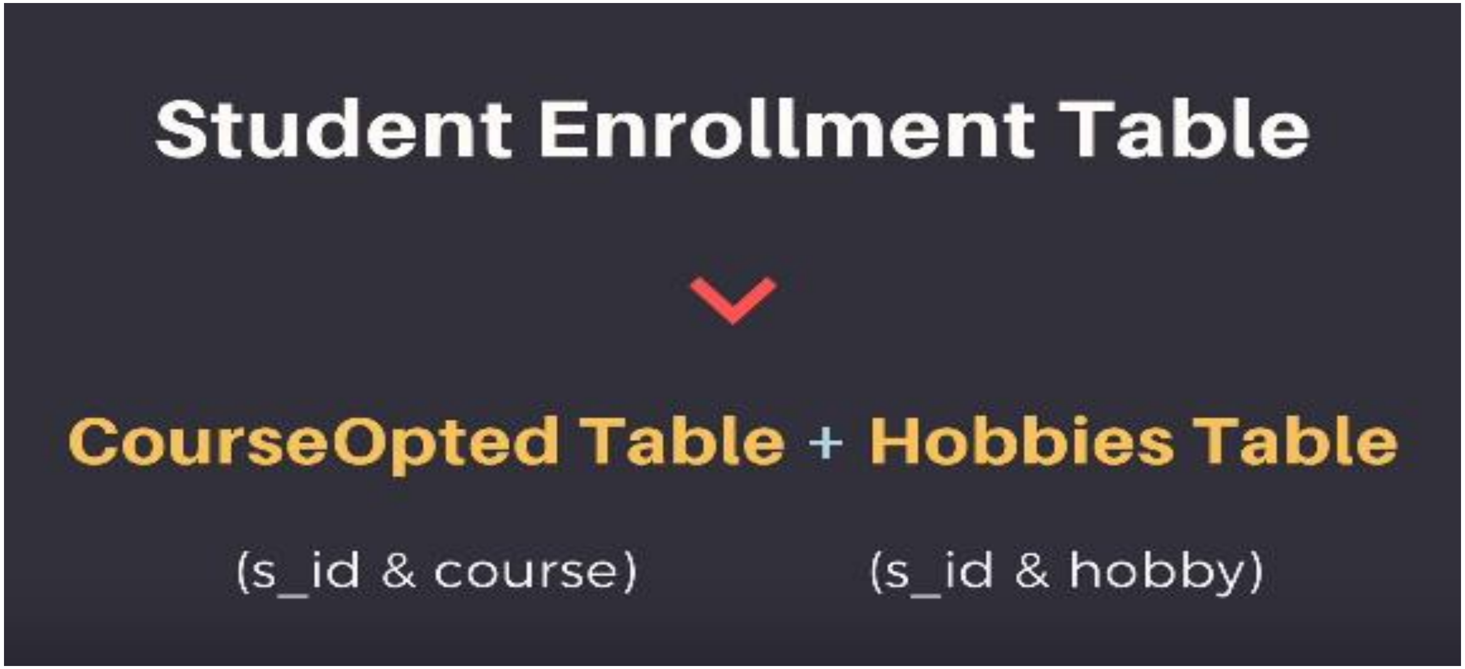
ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

RIGHT?

because same student, same hobbies

MULTIVALUED DEPENDENCY



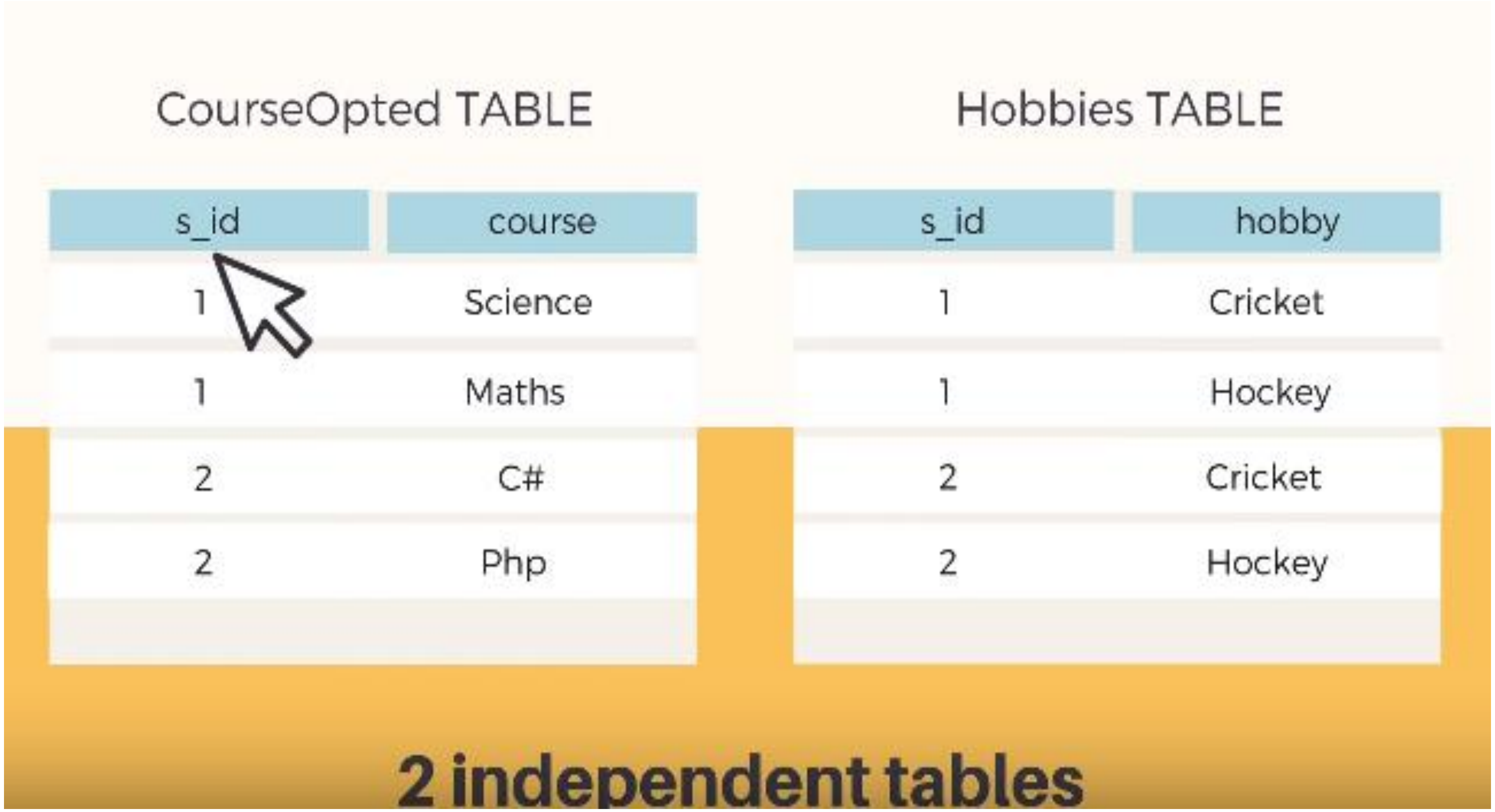
Student Enrollment Table

CourseOpted Table + Hobbies Table

(s_id & course)

(s_id & hobby)

MULTIVALUED DEPENDENCY



CourseOpted TABLE

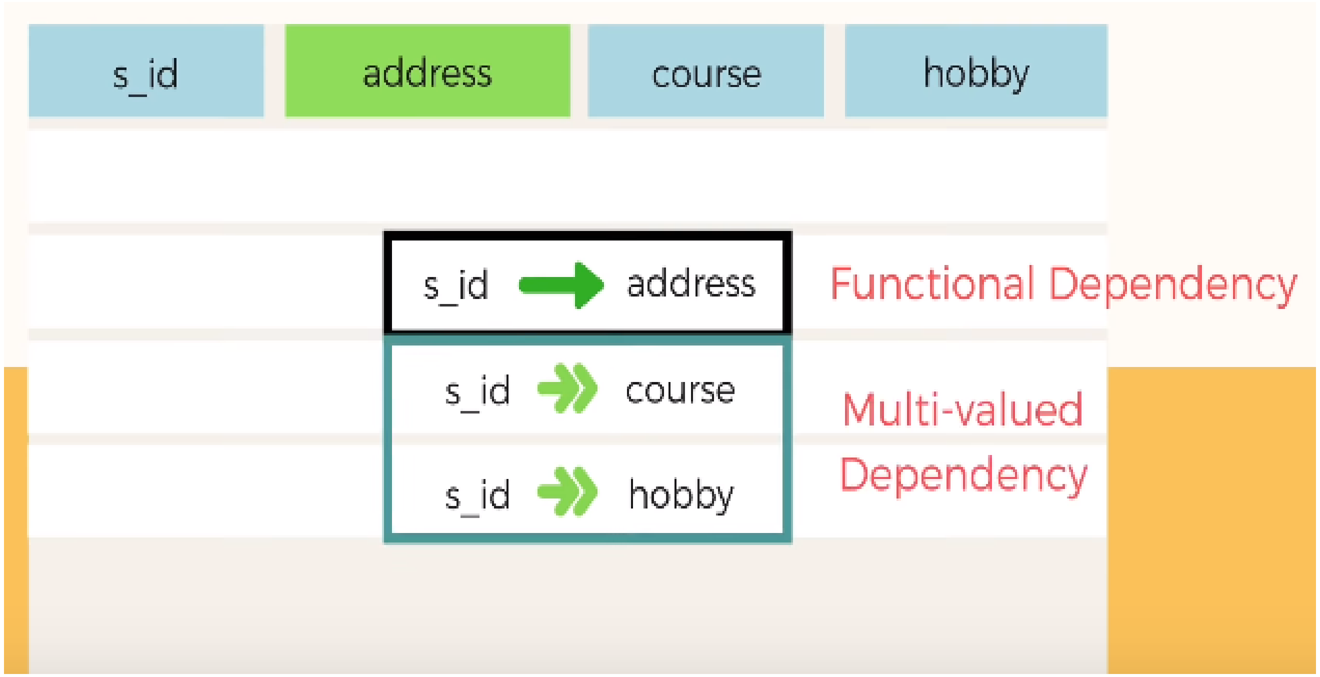
s_id	course
1	Science
1	Maths
2	C#
2	Php

Hobbies TABLE

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

2 independent tables

MULTIVALUED DEPENDENCY



s_id	address	course	hobby

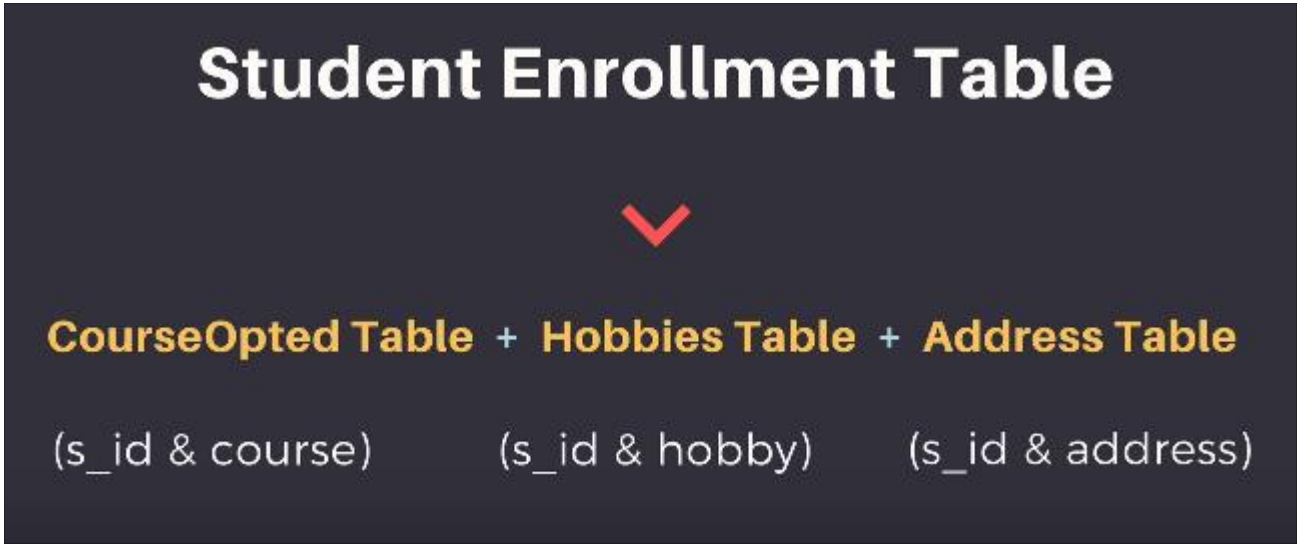
s_id → address — Functional Dependency

s_id $\twoheadrightarrow$ course

s_id $\twoheadrightarrow$ hobby

Multi-valued Dependency

MULTIVALUED DEPENDENCY

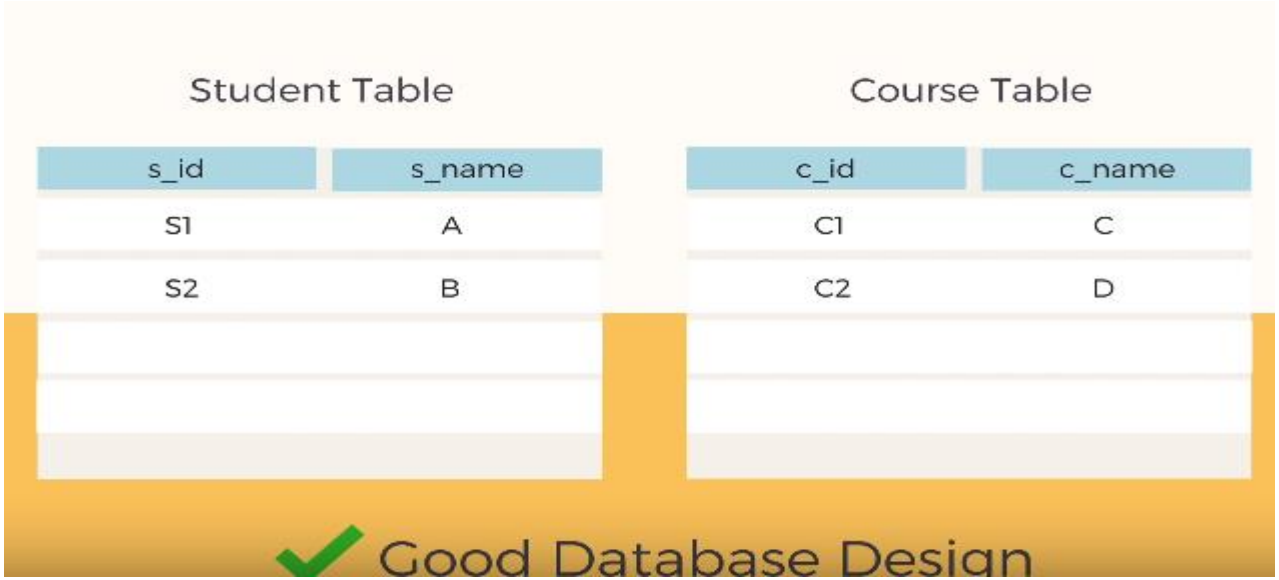


Student Enrollment Table

CourseOpted Table + Hobbies Table + Address Table

(s_id & course) (s_id & hobby) (s_id & address)

4NF



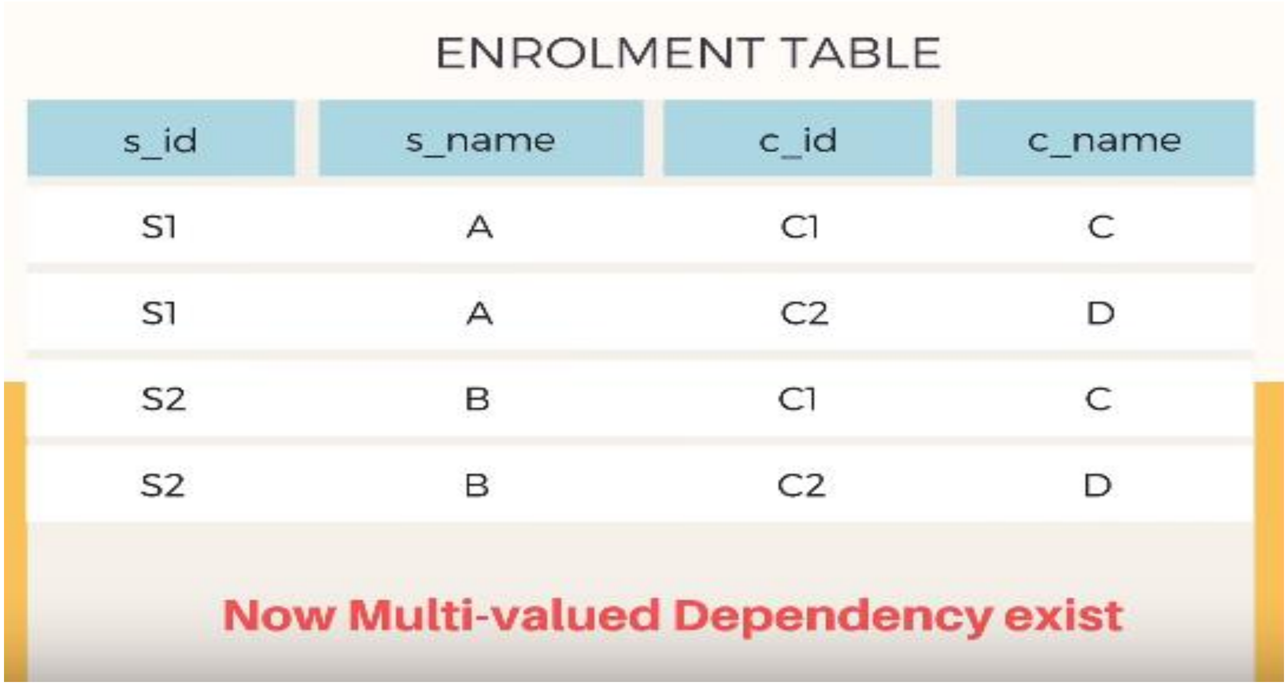
Student Table

s_id	s_name
S1	A
S2	B

Course Table

c_id	c_name
C1	C
C2	D

✓ Good Database Design



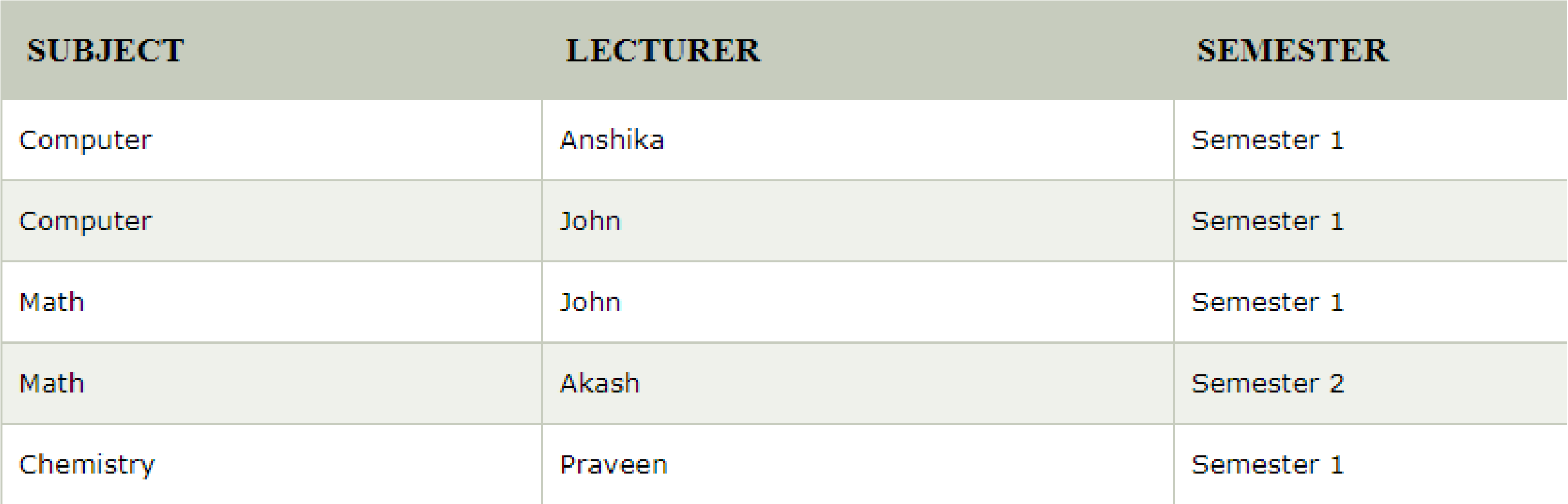
ENROLMENT TABLE

s_id	s_name	c_id	c_name
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

Now Multi-valued Dependency exist

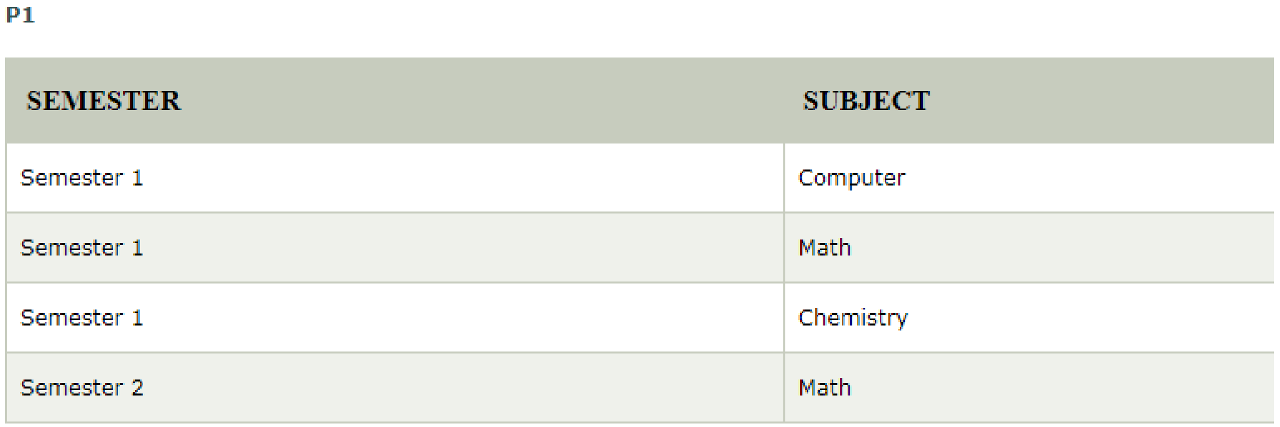
5NF

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



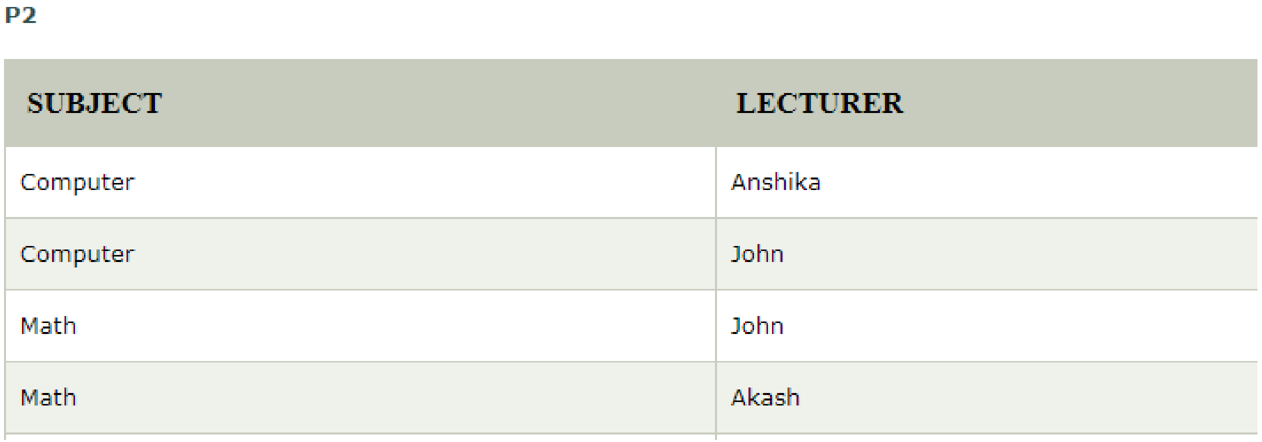
SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

- In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.
- Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.
- So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:



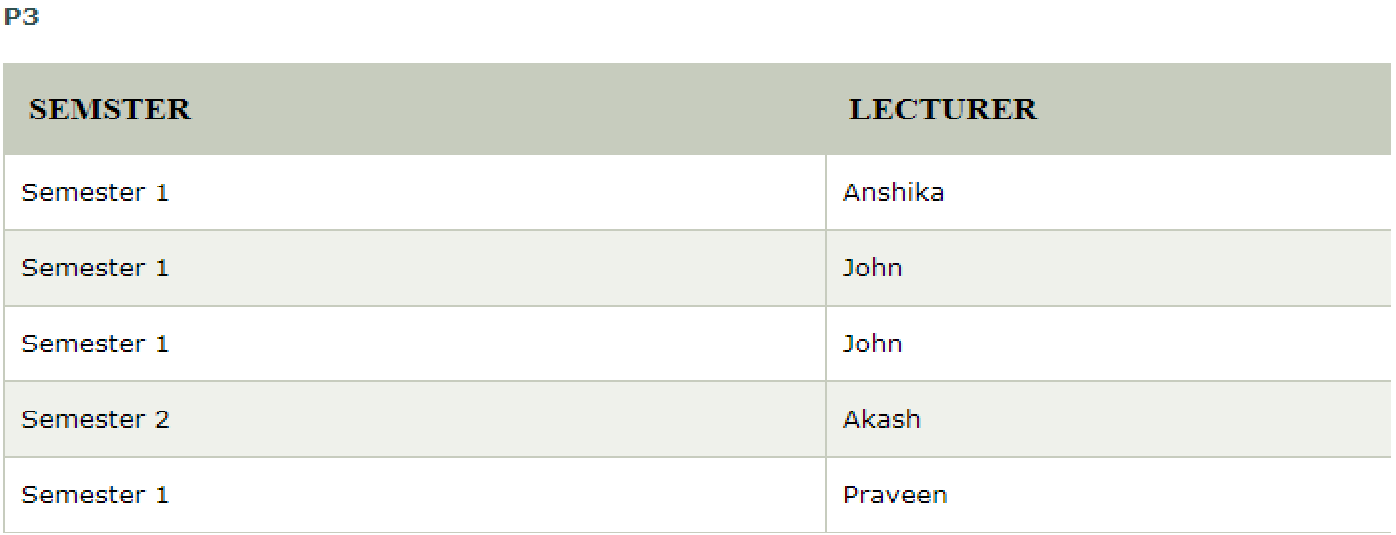
P1

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



P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash



P3

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

JOIN DEPENDENCY

Types of Join dependency

- Lossless Join and
- Dependency Preserving Decomposition

- Decomposition of a relation is done when a relation in relational model is not in appropriate normal form.
- Relation R is decomposed into two or more relations if decomposition is lossless join as well as dependency preserving.

LOSSLESS JOIN DECOMPOSITION

- If we decompose a relation R into relations R1 and R2,
  - Decomposition is lossy if R1 $\bowtie$ R2 $\supset$ R
  - Decomposition is lossless if R1 $\bowtie$ R2 = R
- **To check for lossless join decomposition using FD set, following conditions must hold:**
- Union of Attributes of R1 and R2 must be equal to attribute of R.
  - Each attribute of R must be either in R1 or in R2.

$$\text{Att}(R1)\ U\ \text{Att}(R2) = \text{Att}(R)$$

- Intersection of Attributes of R1 and R2 must not be NULL.

$$\text{Att}(R1) \cap \text{Att}(R2) \neq \Phi$$

- Common attribute must be a key for at least one relation (R1 or R2)

$$\text{Att}(R1) \cap \text{Att}(R2) \text{ -> } \text{Att}(R1) \text{ or } \text{Att}(R1) \cap \text{Att}(R2) \text{ -> } \text{Att}(R2)$$

EXAMPLE

- A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is a lossless join decomposition as:
- First condition holds true as Att(R1) U Att(R2) = (ABC) U (AD) = (ABCD) = Att(R).
- Second condition holds true as Att(R1) ∩ Att(R2) = (ABC) ∩ (AD) ≠ Φ
- Third condition holds true as Att(R1) ∩ Att(R2) = A is a key of R1(ABC) because A->BC is given.

DEPENDENCY PRESERVING DECOMPOSITION

- If we decompose a relation R into relations R1 and R2, All dependencies of R either must be a part of R1 or R2 or must be derivable from combination of FD’s of R1 and R2.

- For Example, A relation R (A, B, C, D) with FD set{A->BC} is decomposed into R1(ABC) and R2(AD) which is dependency preserving because FD A->BC is a part of R1(ABC).

EXAMPLE

- **Consider a schema R(A,B,C,D) and functional dependencies A->B and C->D. Then the decomposition of R into R1(AB) and R2(CD) is ____.**
  A. dependency preserving and lossless join
  B. lossless join but not dependency preserving
  C. dependency preserving but not lossless join
  D. not dependency preserving and not lossless join

Answer:

For lossless join decomposition, these conditions must hold true:

$$\text{Att}(R1) \cup \text{Att}(R2) = ABCD = \text{Att}(R)$$

$$\text{Att}(R1) \cap \text{Att}(R2) = \Phi,$$

which violates the condition of lossless join decomposition. Hence the decomposition is not lossless.

For dependency preserving decomposition,
A->B can be ensured in R1(AB) and C->D can be ensured in R2(CD). Hence it is dependency preserving decomposition.
So, the correct option is C.

END