Normalization

By :
Dr. Rinkle Rani
Associate Professor, CSED
TIET, Patiala

Normalization of Database

Database Normalization is a technique of organizing the data in the database.

Normalization is a systematic approach of decomposing tables to eliminate data redundancy (repetition) and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process.

Normalization is used for mainly two purposes,

ÅEliminating redundant data.

ÅEnsuring data dependencies make sense i.e data is logically stored.

Problems Without Normalization

If a table is not properly normalized and have data redundancy then it will not only take extra memory space but will also make it difficult to handle and update the database, without facing data loss.

Insertion, Updation and Deletion Anamolies are very frequent if database is not normalized.

To understand these anomalies let us take an example of a **Student** table.

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

In the table above, we have data of 4 Computer Sci. students. As we can see, data for the fields branch, hod (Head of Department) and office_tel is repeated for the students who are in the same branch in the college, this is Data Redundancy.

Insertion Anomaly

Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL.

Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students.

These scenarios are nothing but Insertion anomalies.

Updation Anomaly

What if Mr. X leaves the college? or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updation anomaly.

Deletion Anomaly

In our Student table, two different information are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

Normalization Rules

Normalization rules are divided into the following normal forms:

Å First normal form(1NF)
Å Second normal form (2NF)
Å Third normal form (3NF)
Å Boyce & Codd normal form (BCNF)

First Normal Form (1NF)

A relation is in 1NF if

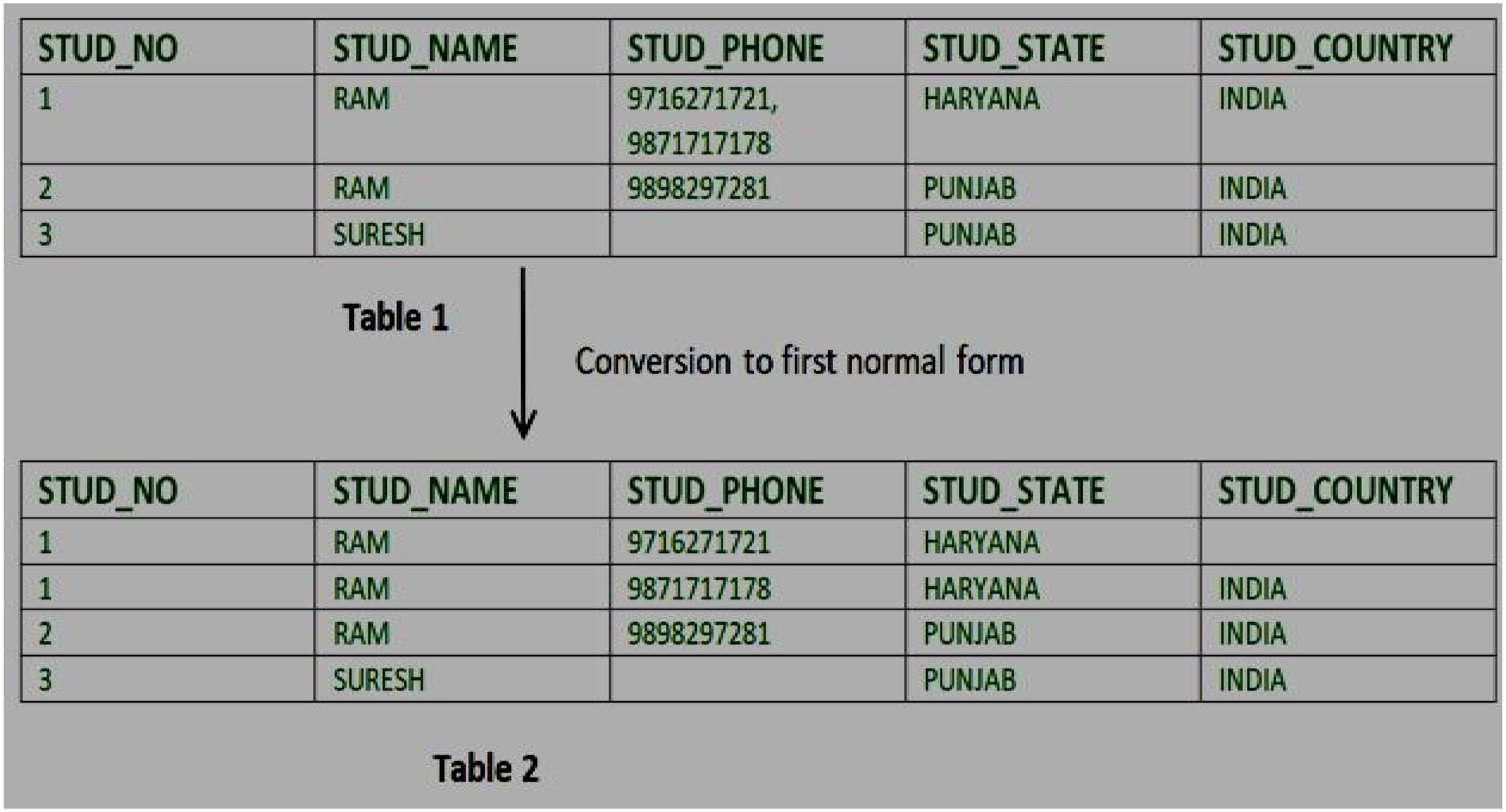
a) it contains atomic values only
b) Values stored in a column should be of the same domain.

First Normal Form (1NF) Disallows

composite attributes

multivalued attributes

Example 1 – Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE.



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

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

Å Table is in 1NF (First normal form)

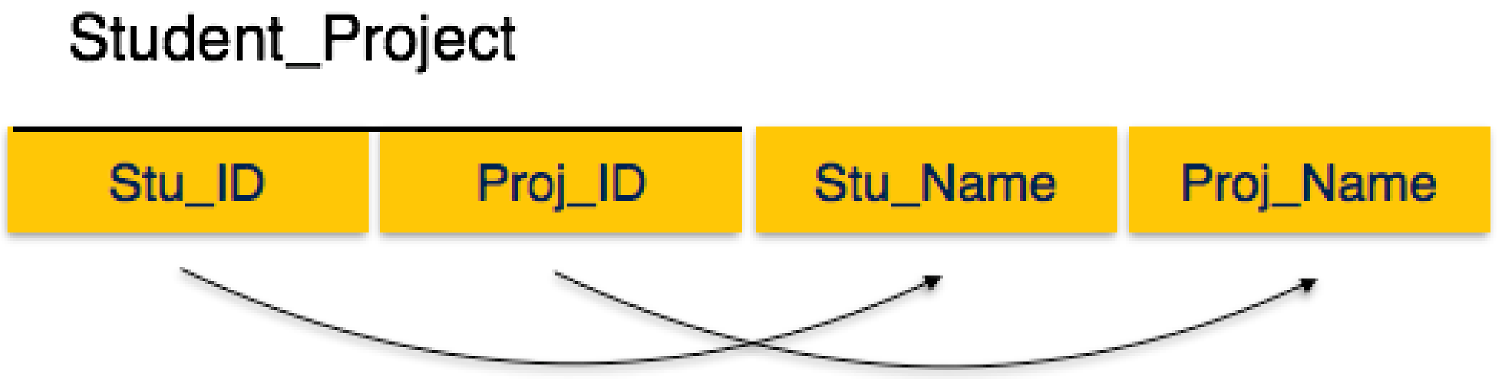
Å All non-key attributes are fully functional dependent on the primary key. No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

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

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

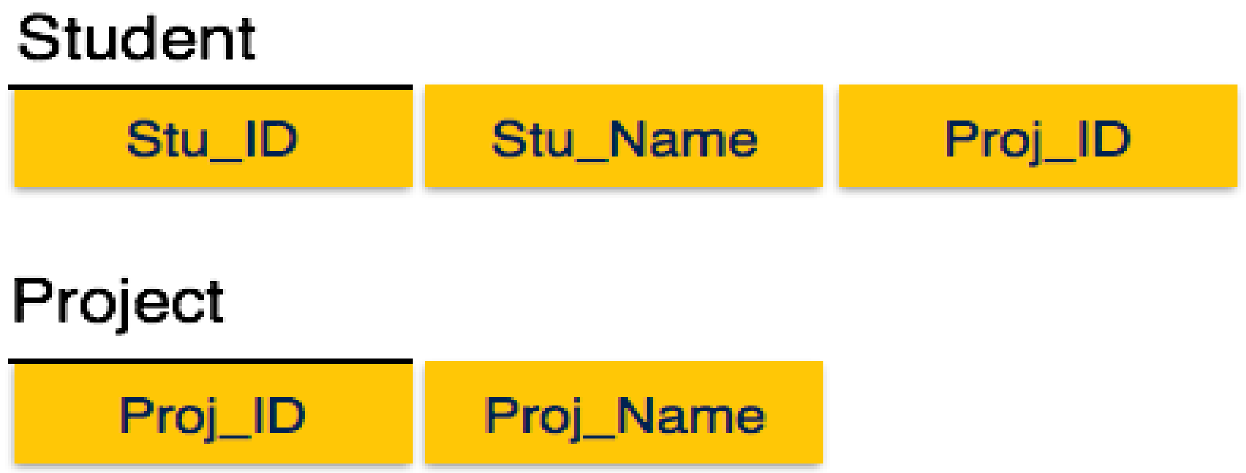
If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute.

That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually.

But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently. This is called partial dependency, which is not allowed in Second Normal Form.



We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

What is a transitive dependency?

Within a relation if we see

A ->B [B depends on A]

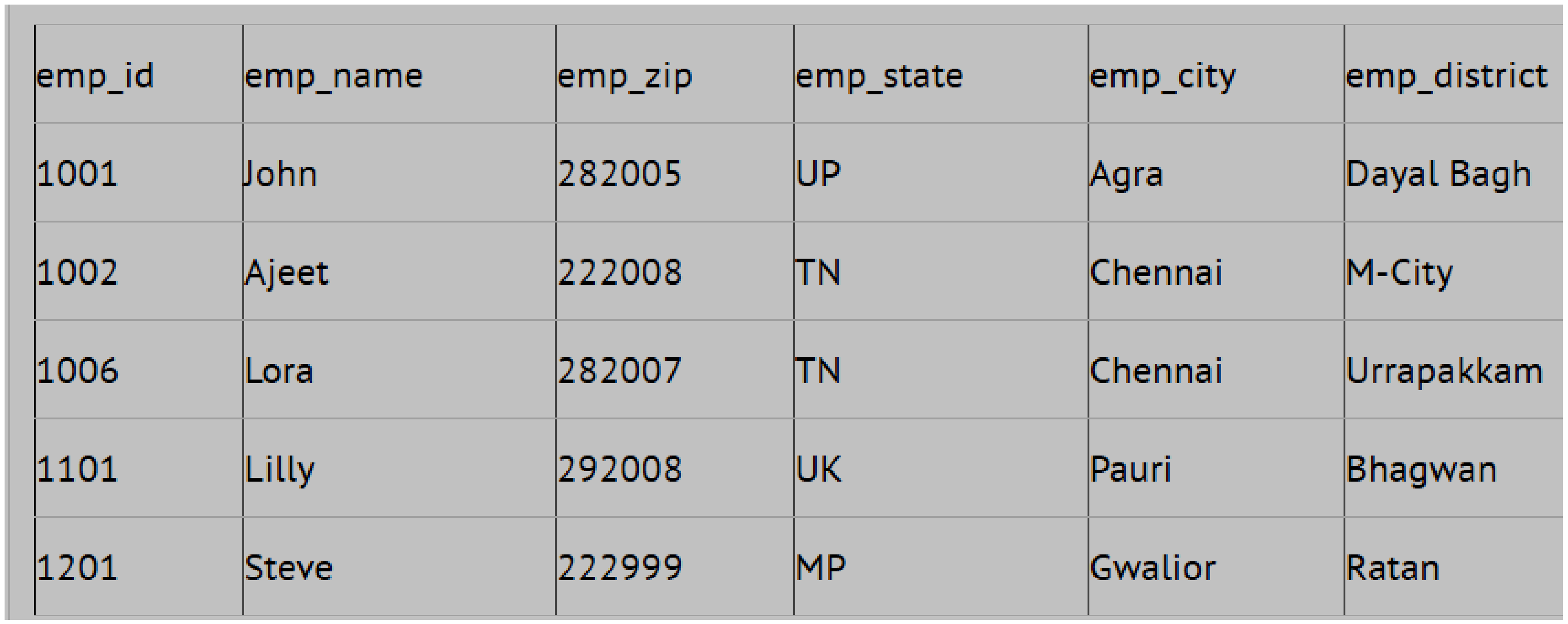
And

B -> C [C depends on B]

Then we may derive

A -> C [C depends on A]

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:



emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam
1101	Lilly	292008	UK	Pauri	Bhagwan
1201	Steve	222999	MP	Gwalior	Ratan

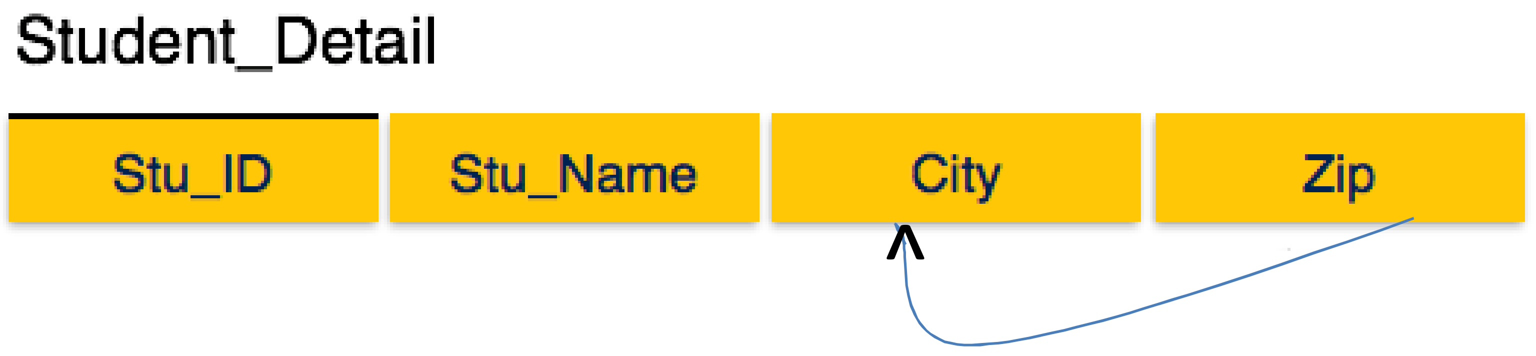
Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}
Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

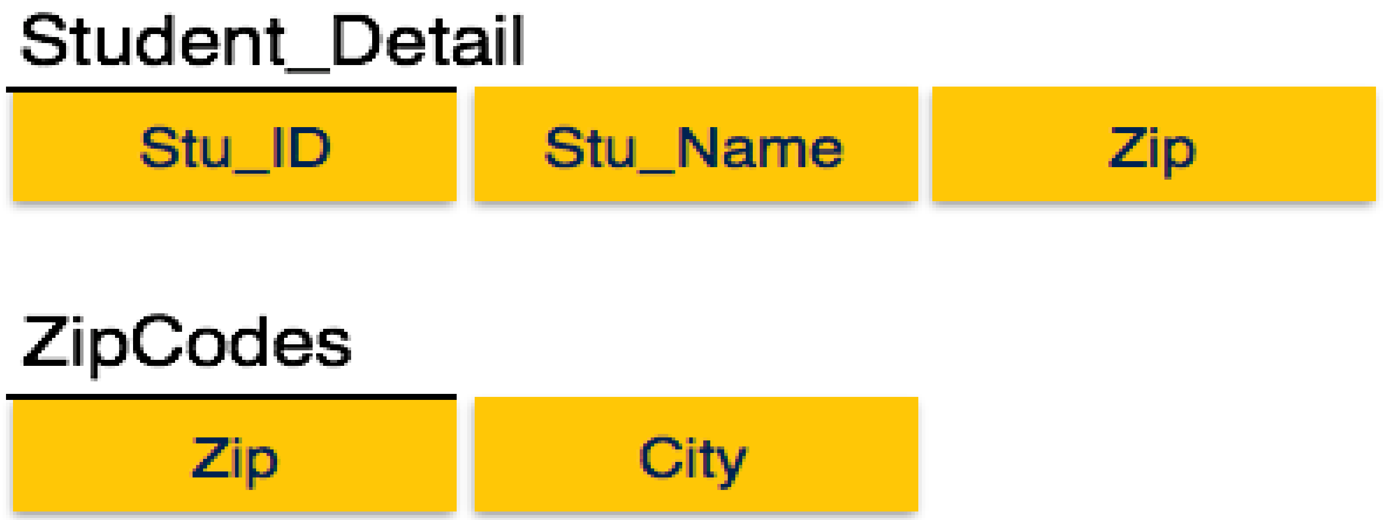


We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute.

We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a super key nor is City a prime attribute.

Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the rela�on into two rela�ons as follows –



BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X -> A** holds in R, then **X is a candidate key** of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

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.**

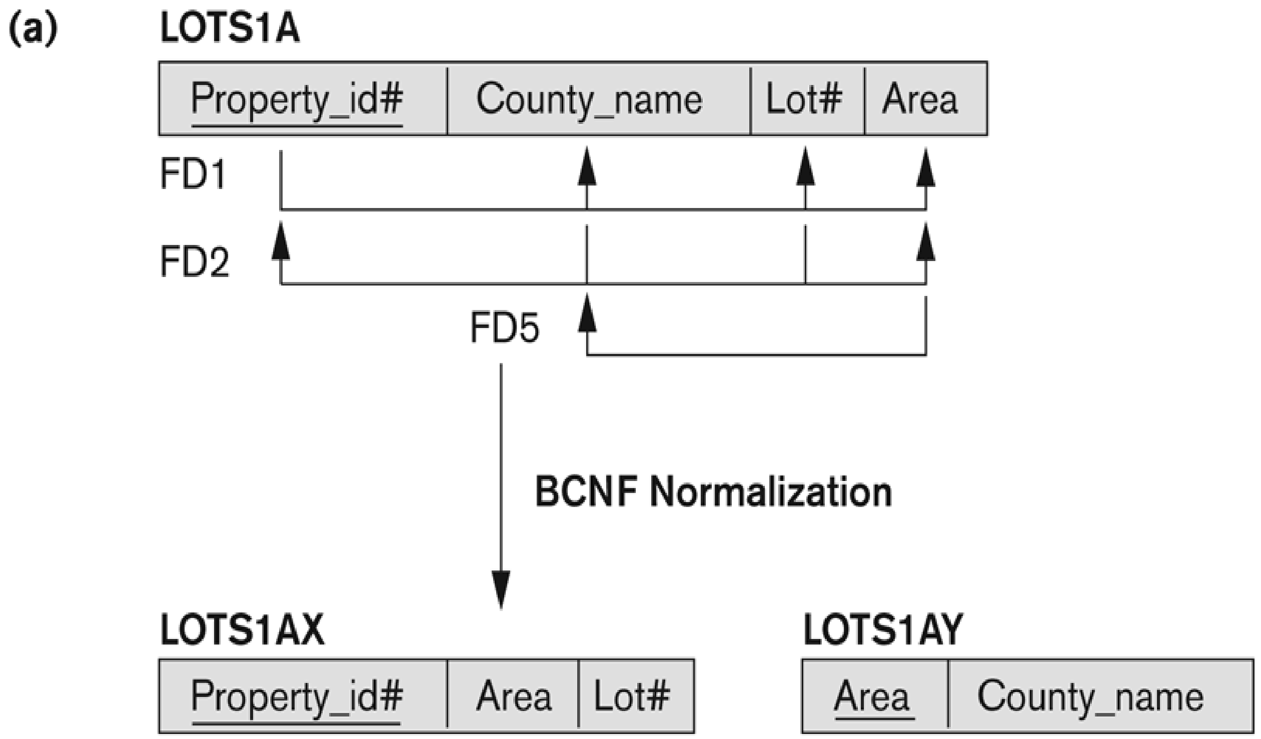
Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

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

$$\begin{tikzcd}
 & \text{R(A, B, C, D)} \arrow[dl, no head] \arrow[dr, no head] & \\
\text{R1 (A, D, C)} & & \text{R2(D, B)}
\end{tikzcd}$$

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

Boyce-Codd normal form



Difference between 3NF and BCNF

Å Assume the following things;

Å A and B are set of attributes.

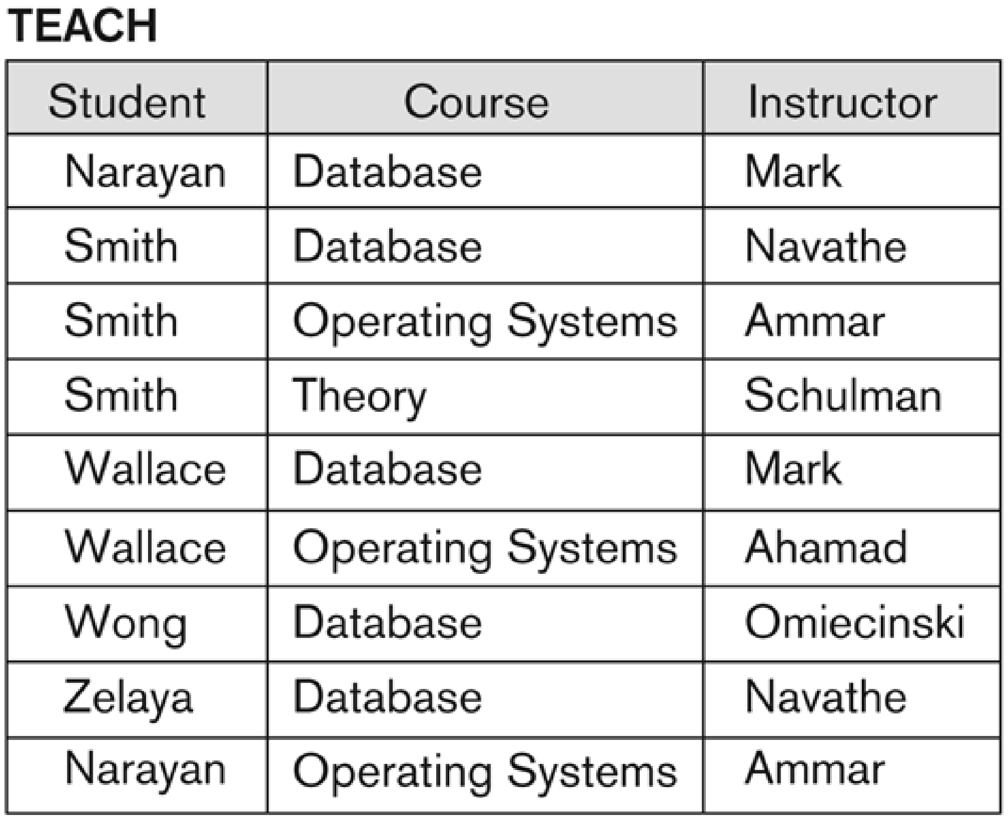
Å A is non-key attribute and B is the primary key.

Å FD – { A → B. }

Å The above Functional Dependency is about the dependency of primary key on a non-key attribute. This functional dependency is permitted in Third Normal Form (3NF).

Å This Functional Dependency is not permitted in Boyce-Codd Normal Form (BCNF), because BCNF expects the determiner should be a candidate key. In our example, A is not a candidate key. This is why BCNF is termed as strict 3NF.

Exercise : A relation TEACH that is in 3NF but not in BCNF ???



TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

Two FDs exist in the relation TEACH:

- fd1: { student, course} -> instructor
- fd2: instructor -> course

{student, course} is a candidate key

- So this relation is in 3NF *but not in* BCNF

3NF – BCNF Comparisons

Properties	**3NF**	**BCNF**
Achievability	Always achievable	Not always achievable
Quality of the tables	Less	More
Non-key Determinants	Can have non-key attributes as determinants	Cannot have.
Proposed by	Edgar F. Codd	Raymond F.Boyce and Edgar F.Codd jointly proposed
Decomposition	Loss-less join decomposition can be achieved	Sometimes Loss-less join decomposition cannot be achieved