Normal Forms

1. Candidate Key:

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD_NO in STUDENT relation.

- It is a minimal super key.
- It is a super key with no repeated data is called a candidate key.
- The minimal set of attributes that can uniquely identify a record.
- It must contain unique values.
- It can contain NULL values.
- Every table must have at least a single candidate key.
- A table can have multiple candidate keys but only one primary key (the primary key cannot have a NULL value, so the candidate key with a NULL value can't be the primary key).
- The value of the Candidate Key is unique and may be null for a tuple.
- There can be more than one candidate key in a relationship.

- STUD_NO is the candidate key for relation STUDENT.

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

- The candidate key can be simple (having only one attribute) or composite as well.

Example:

- {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

Table STUDENT_COURSE

STUD_NO	**TEACHER_NO**	**COURSE_NO**
1	001	C001
2	056	C005

- Note: In SQL Server a unique constraint that has a nullable column, allows the value ‘null‘ in that column only once.
- That’s why the STUD_PHONE attribute is a candidate here, but can not be a ‘null’ value in the primary key attribute.

2. Primary Key:

- There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- For Example, STUD_NO, as well as STUD_PHONE, are candidate keys for relation STUDENT but STUD_NO can be chosen as the primary key (only one out of many candidate keys).

- It is a unique key.
- It can identify only one tuple (a record) at a time.
- It has no duplicate values, it has unique values.
- It cannot be NULL.
- Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

Example:

- STUDENT table -> Student(STUD_NO, SNAME, ADDRESS, PHONE) , STUD_NO is a primary key

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

3. Super Key:

- The set of attributes that can uniquely identify a tuple is known as Super Key.
- For Example, STUD_NO, (STUD_NO, STUD_NAME), etc.
- A super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

- Adding zero or more attributes to the candidate key generates the super key.
- A candidate key is a super key but vice versa is not true.

Example:

- Consider the table shown below.
- STUD_NO+PHONE is a super key.

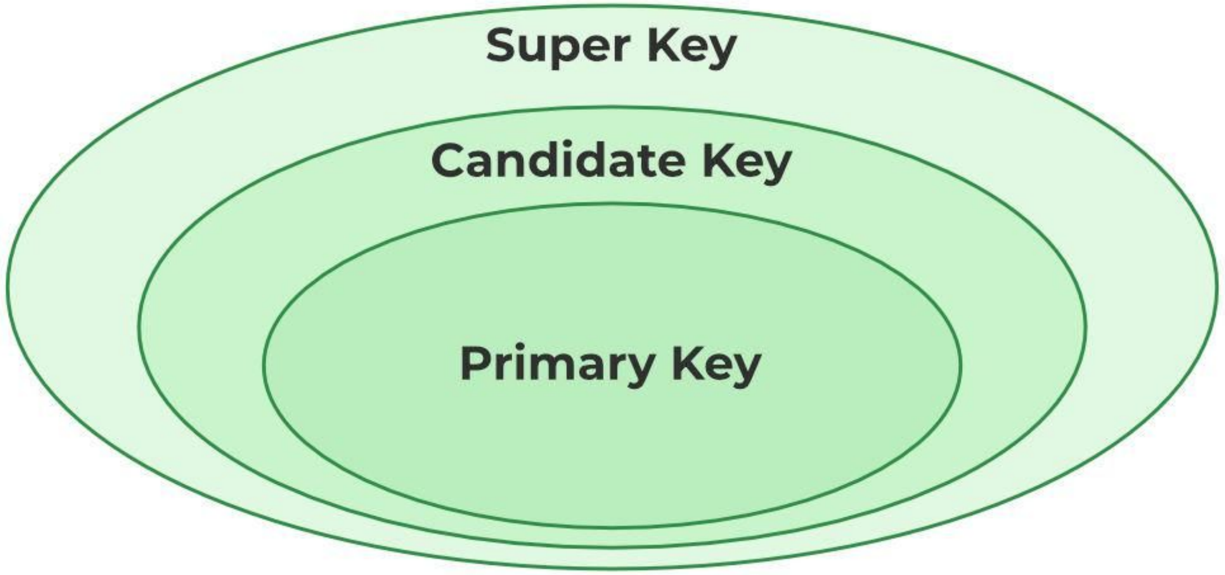


Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965

4. Alternate Key:

- The candidate key other than the primary key is called an alternate key.

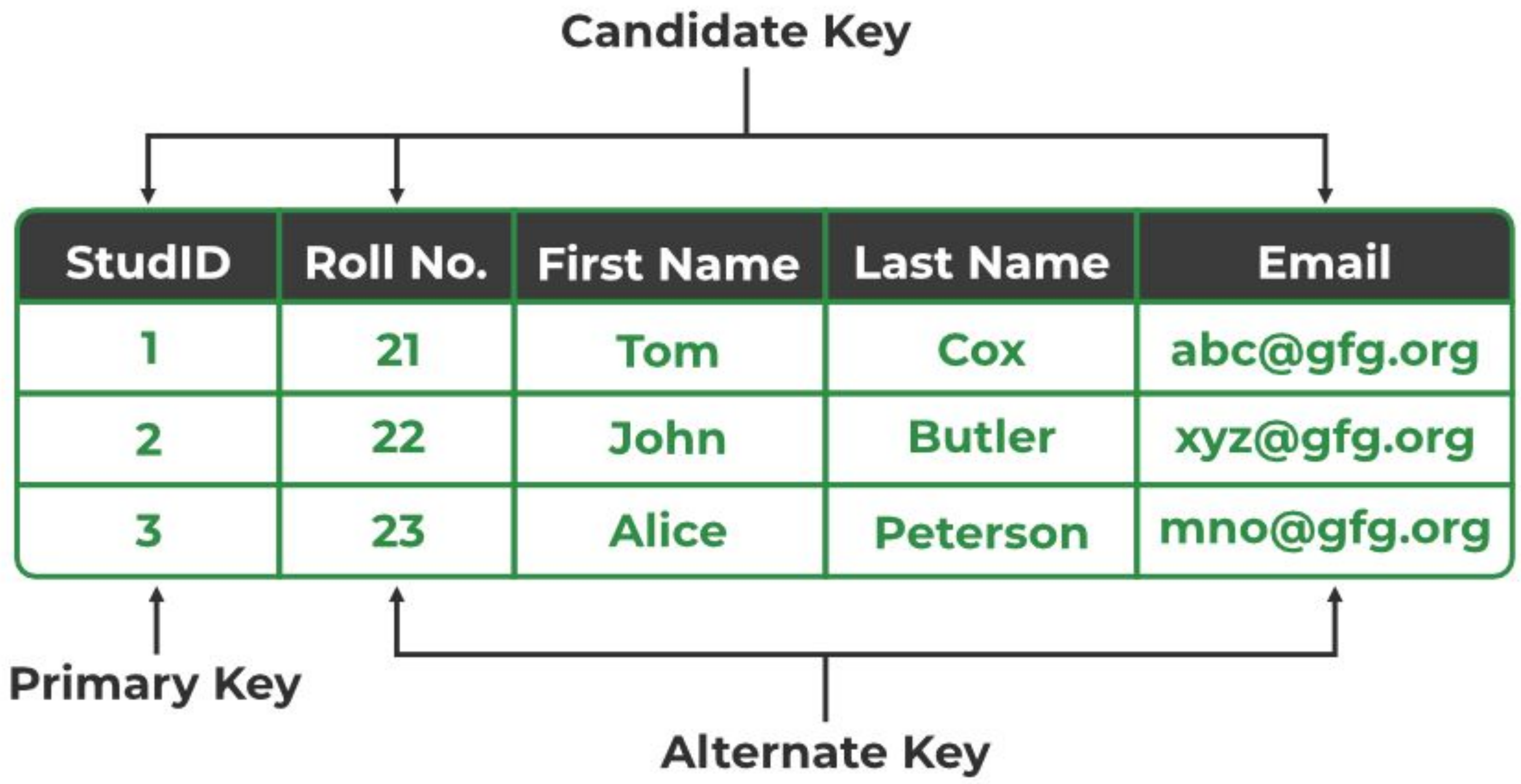
- All the keys which are not primary keys are called alternate keys.
- It is a secondary key.
- It contains two or more fields to identify two or more records.
- These values are repeated.
- Eg:- SNAME, and ADDRESS is Alternate keys

- Example:

- Consider the table shown below.
- STUD_NO, as well as PHONE both,
- are candidate keys for relation STUDENT but
- PHONE will be an alternate key
- (only one out of many candidate keys).

Table STUDENT

STUD_NO	SNAME	ADDRESS	PHONE
1	Shyam	Delhi	123456789
2	Rakesh	Kolkata	223365796
3	Suraj	Delhi	175468965



StudID	Roll No.	First Name	Last Name	Email
1	21	Tom	Cox	abc@gfg.org
2	22	John	Butler	xyz@gfg.org
3	23	Alice	Peterson	mno@gfg.org

5. Foreign Key:

- If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers.
- The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute the relation which refers to the referenced relation is called referencing relation and the corresponding attribute is called referencing attribute.
- The referenced attribute of the referenced relation should be the primary key to it.

- It is a key it acts as a primary key in one table and it acts as
- secondary key in another table.
- It combines two or more relations (tables) at a time.
- They act as a cross-reference between the tables.
- For example, DNO is a primary key in the DEPT table and a non-key in EMP

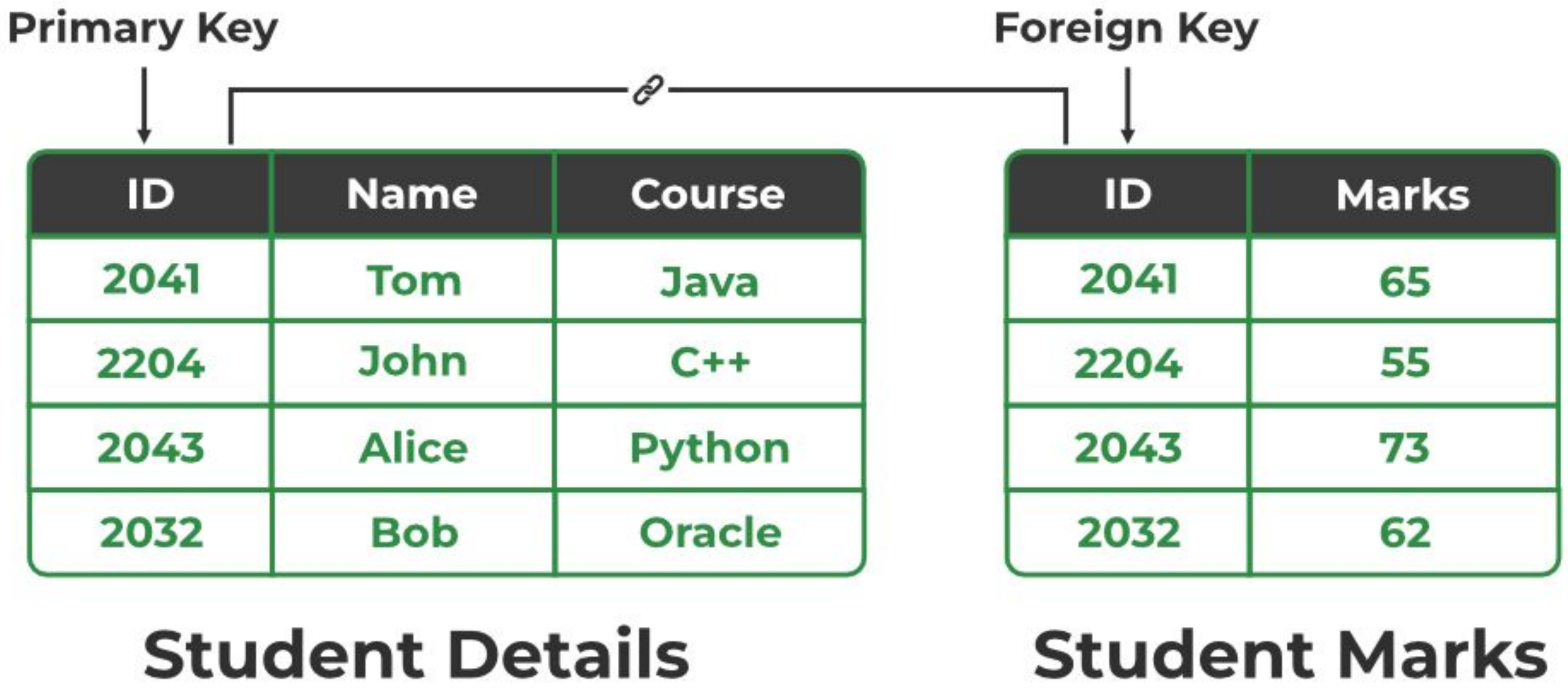
Example:

- Refer Table STUDENT shown above.
- STUD_NO in STUDENT_COURSE is a foreign key to STUD_NO in STUDENT relation.

Table STUDENT_COURSE

STUD_NO	TEACHER_NO	COURSE_NO
1	005	C001
2	056	C005

- It may be worth noting that, unlike the Primary Key of any given relation, Foreign Key can be NULL as well as may contain duplicate tuples i.e. it need not follow uniqueness constraint.
- For Example, STUD_NO in the STUDENT_COURSE relation is not unique. It has been repeated for the first and third tuples.
- However, the STUD_NO in STUDENT relation is a primary key and it needs to be always unique, and it cannot be null.



ID	Name	Course
2041	Tom	Java
2204	John	C++
2043	Alice	Python
2032	Bob	Oracle

ID	Marks
2041	65
2204	55
2043	73
2032	62

6. Composite Key

- Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table.
- To uniquely identify rows of a table, a combination of two or more columns/attributes can be used. It still can give duplicate values in rare cases.
- So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

- It acts as a primary key if there is no primary key in a table
- Two or more attributes are used together to make a composite key.
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

Example:

- FULLNAME + DOB can be combined together to access the details of a student.

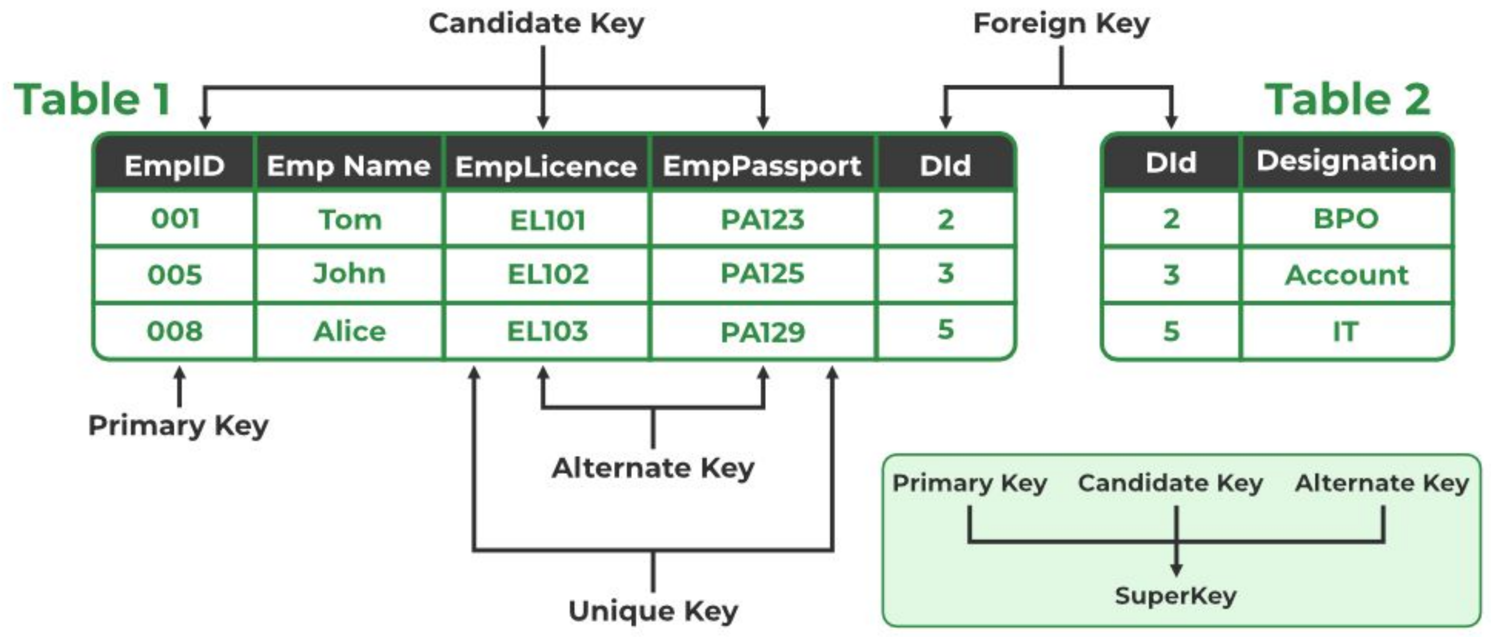


Table 1

EmpID	Emp Name	EmpLicence	EmpPassport	DId
001	Tom	EL101	PA123	2
005	John	EL102	PA125	3
008	Alice	EL103	PA129	5

Table 2

DId	Designation
2	BPO
3	Account
5	IT

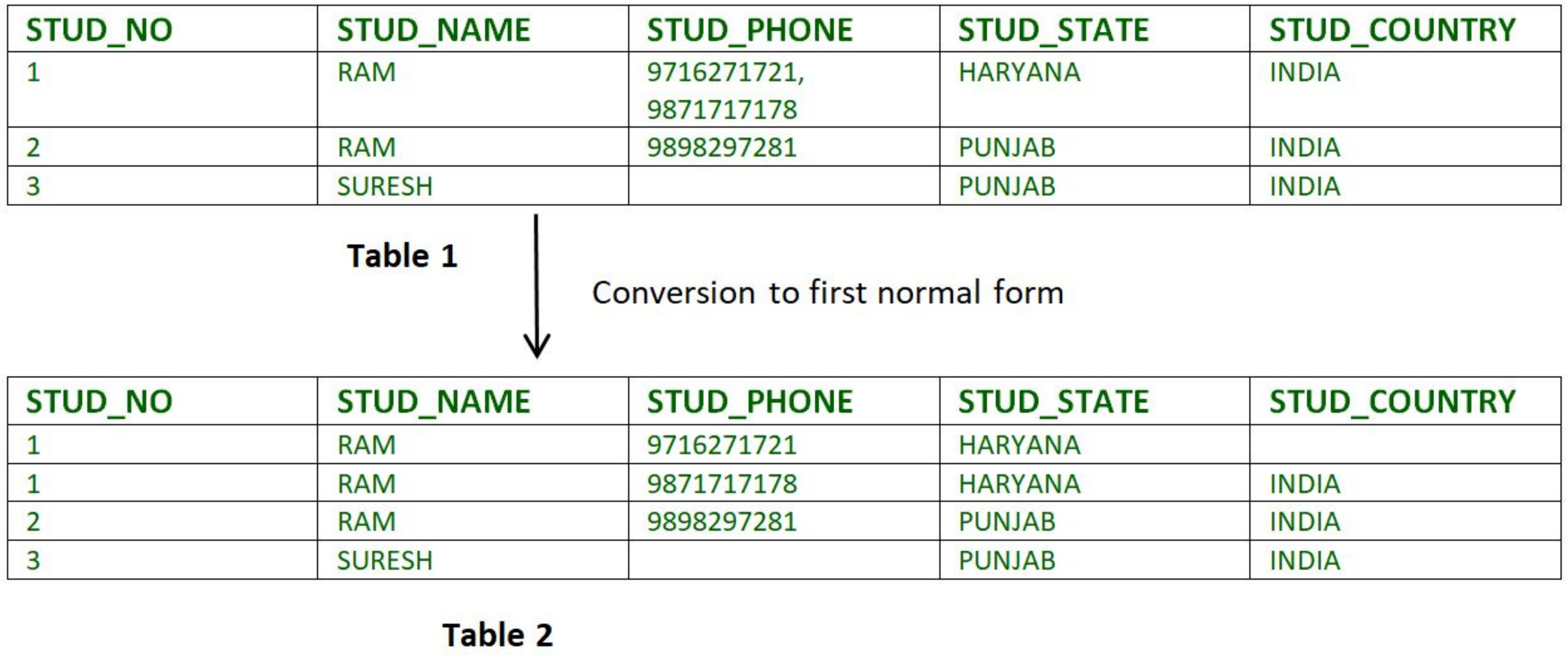
First Normal Form (1NF)

- If a relation contains a composite or multi-valued attribute, it violates the first normal form, or the relation is in first normal form if it does not contain any composite or multi-valued attribute.
- A relation is in first normal form if every attribute in that relation is single valued attribute.

- A table is in 1 NF if:
  - There are only Single Valued Attributes.
  - Attribute Domain does not change.
  - There is a unique name for every Attribute/Column.
  - The order in which data is stored does not matter.

Example-1:

- Relation STUDENT in table 1 is not in 1NF because of multi-valued attribute STUD_PHONE.
- Its decomposition into 1NF has been shown in table 2.



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

$\downarrow$ 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

Example-2

ID	Name	Courses
1	A	c1, c2
2	E	c3
3	M	c2, c3

In the above table, Course is a multi-valued attribute so it is not in 1NF.

Below Table is in 1NF as there is no multi-valued attribute:

ID	**Name**	**Course**
1	**A**	**c1**
1	**A**	**c2**
2	**E**	**c3**
3	**M**	**c2**
3	**M**	**c3**

- First Normal Form (1NF) does not eliminate redundancy, but rather, it’s that it eliminates repeating groups.
- Instead of having multiple columns of the same kind of data in a record, (0NF or Unnormalized form) you remove the repeated information into a separate relation and represent them as rows.
- This is what constitutes 1NF.

Second Normal Form (2NF)

- Second Normal Form (2NF) is based on the concept of full functional dependency.
- Second Normal Form applies to relations with composite keys, that is, relations with a primary key composed of two or more attributes.
- A relation with a single-attribute primary key is automatically in at least 2NF. A relation that is not in 2NF may suffer from the update anomalies.
- To be in second normal form, a relation **must be in the first normal form and the relation must not contain any partial dependency.**
- A relation is in 2NF if it has 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. In other words,

A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in Second Normal Form (2NF).

Note

- If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.
- The normalization of 1NF relations to 2NF involves the removal of partial dependencies.
- If a partial dependency exists, we remove the partially dependent attribute(s) from the relation by placing them in a new relation along with a copy of their determinant.

Example-1: Consider table as following below.

STUD_NO	COURSE_NO	COURSE_FEE
1	C1	1000
2	C2	1500
1	C4	2000
4	C3	1000
4	C1	1000
2	C5	2000

{Note that, there are many courses having the same course fee. }

- Here, COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO;
- COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO;
- COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO;
- Hence, COURSE_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD_NO, COURSE_NO}
- But, COURSE_NO -> COURSE_FEE, i.e., COURSE_FEE is dependent on COURSE_NO, which is a proper subset of the candidate key.
- Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF.
- To convert the above relation to 2NF, ***we need to split the table into two tables*** such as :

Table 1: STUD_NO, COURSE_NO **Table 2: COURSE_NO, COURSE_FEE**

Table 1

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

Table 2

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

Note –

- 2NF tries to reduce the redundant data getting stored in memory.
- For instance, if there are 100 students taking C1 course, we dont need to store its Fee as 1000 for all the 100 records, instead once we can store it in the second table as the course fee for C1 is 1000.

Example-2

- Consider following functional dependencies in relation R (A, B, C, D)

  AB -> C [A and B together determine C]

  BC -> D [B and C together determine D]

- In this case, we can see that the relation R has a composite candidate key {A,B} as AB->C.
- Therefore, A and B together uniquely determine the value of C. Similarly, BC -> D shows that B and C together uniquely determine the value of D.
- The relation R is already in 1NF because it does not have any repeating groups or nested relations.
- However, we can see that the non-prime attribute D is functionally dependent on only part of a candidate key, BC. This violates the 2NF condition.

Third Normal Form (3NF)

- Although Second Normal Form (2NF) relations have less redundancy than those in 1NF, they may still suffer from update anomalies.
- If we update only one tuple and not the other, the database would be in an inconsistent state.
- This update anomaly is caused by a transitive dependency. We need to remove such dependencies by progressing to Third Normal Form (3NF).

Third Normal Form (3NF)

- A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form.
- A relation is in 3NF if at least one of the following conditions holds in every non-trivial function dependency X –> Y:
  - X is a super key.
  - Y is a prime attribute (each element of Y is part of some candidate key).

In other words,

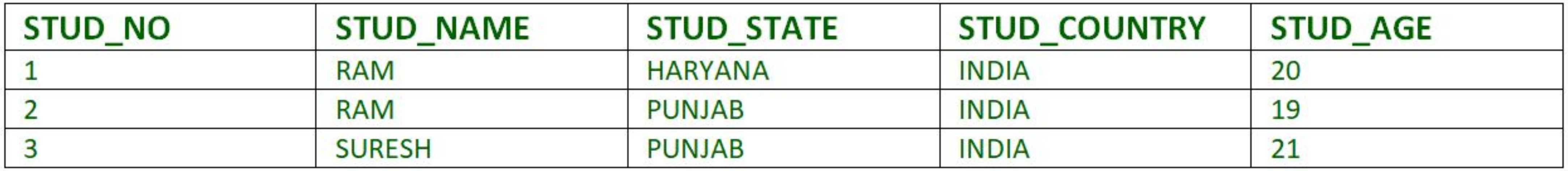
- ***A relation that is in First and Second Normal Form and in which no non-primary-key attribute is transitively dependent on the primary key, then it is in Third Normal Form (3NF).***

- Note – If A->B and B->C are two FDs then A->C is called transitive dependency.

- The normalization of 2NF relations to 3NF involves the removal of transitive dependencies.
- If a transitive dependency exists, we remove the transitively dependent attribute(s) from the relation by placing the attribute(s) in a new relation along with a copy of the determinant.

Example-1:

- In relation STUDENT given in Table 4,



STUD_NO	STUD_NAME	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	HARYANA	INDIA	20
2	RAM	PUNJAB	INDIA	19
3	SURESH	PUNJAB	INDIA	21

Table 4

- FD set:
  {STUD_NO -> STUD_NAME, STUD_NO -> STUD_STATE, STUD_STATE -> STUD_COUNTRY, STUD_NO -> STUD_AGE}
- Candidate Key:
  {STUD_NO}

- For this relation in table 4, STUD_NO -> STUD_STATE and STUD_STATE -> STUD_COUNTRY are true. So STUD_COUNTRY is transitively dependent on STUD_NO. It violates the third normal form.
- To convert it in third normal form, we will decompose the relation STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_COUNTRY_STUD_AGE) as:

STUDENT (STUD_NO, STUD_NAME, STUD_PHONE, STUD_STATE, STUD_AGE)
STATE_COUNTRY (STATE, COUNTRY)

Example-2:

- Consider relation R(A, B, C, D, E)

A -> BC,
CD -> E,
B -> D,
E -> A

All possible candidate keys in above relation are {A, E, CD, BC} All attribute are on right sides of all functional dependencies are prime.

Note –

- Third Normal Form (3NF) is considered adequate for normal relational database design because most of the 3NF tables are free of insertion, update, and deletion anomalies.
- Moreover, 3NF always ensures functional dependency preserving and lossless.

Boyce-Codd Normal Form (BCNF)

- Application of the general definitions of 2NF and 3NF may identify additional redundancy caused by dependencies that violate one or more candidate keys.
- However, despite these additional constraints, dependencies can still exist that will cause redundancy to be present in 3NF relations.
- This weakness in 3NF resulted in the presentation of a stronger normal form called the Boyce-Codd Normal Form (Codd, 1974).
- Although, 3NF is an adequate normal form for relational databases, still, this (3NF) normal form may not remove 100% redundancy because of X–>Y functional dependency if X is not a candidate key of the given relation.
- This can be solved by Boyce-Codd Normal Form (BCNF).

Boyce-Codd Normal Form (BCNF)

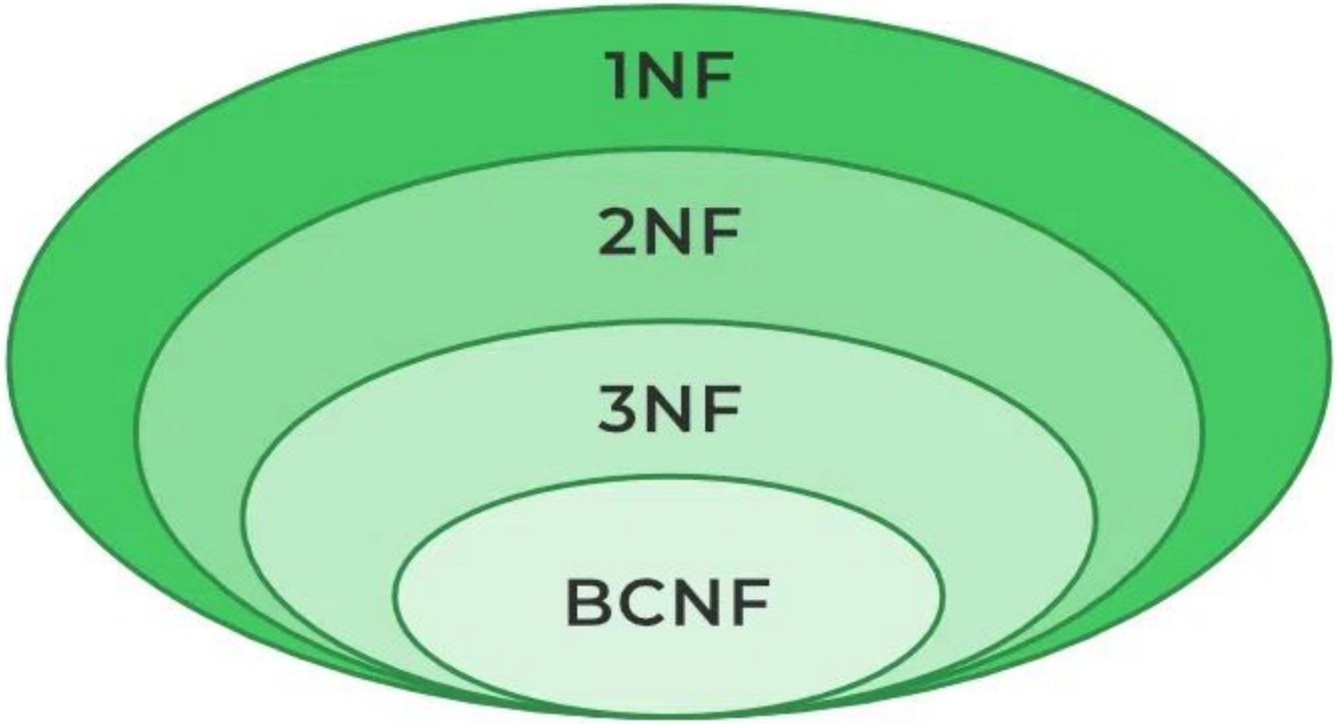
- Boyce–Codd Normal Form (BCNF) is based on functional dependencies that take into account all candidate keys in a relation;
- however, BCNF also has additional constraints compared with the general definition of 3NF.

Rules for BCNF

- Rule 1: The table should be in the 3rd Normal Form.
- Rule 2: X should be a superkey for every functional dependency (FD) X−>Y in a given relation.

Note: To test whether a relation is in BCNF, we identify all the determinants and make sure that they are candidate keys.

- It can be inferred that every relation in BCNF is also in 3NF. To put it another way, a relation in 3NF need not be in BCNF.



- To determine the highest normal form of a given relation R with functional dependencies, the first step is to check whether the BCNF condition holds.
- If R is found to be in BCNF, it can be safely deduced that the relation is also in 3NF, 2NF, and 1NF as the hierarchy shows.
- The 1NF has the least restrictive constraint – it only requires a relation R to have atomic values in each tuple.
- The 2NF has a slightly more restrictive constraint.
- The 3NF has a more restrictive constraint than the first two normal forms but is less restrictive than the BCNF.
- In this manner, the restriction increases as we traverse down the hierarchy.

Examples

- Here, we are going to discuss some basic examples which let you understand the properties of BCNF. We will discuss multiple examples here.

Example 1

- Let us consider the student database, in which data of the student are mentioned.

Stu_ID	Stu_Branch	Stu_Course	Branch_Number	Stu_Course_No
101	Computer Science & Engineering	DBMS	B_001	201
101	Computer Science & Engineering	Computer Networks	B_001	202
102	Electronics & Communication Engineering	VLSI Technology	B_003	401
102	Electronics & Communication Engineering	Mobile Communication	B_003	402

- Functional Dependency of the above is as mentioned:

  Stu_ID –> Stu_Branch

  Stu_Course –> {Branch_Number, Stu_Course_No}

- Candidate Keys of the above table are: {Stu_ID, Stu_Course}

Analysis

- Why this Table is Not in BCNF?
  - The table present above is not in BCNF, because as we can see that neither Stu_ID nor Stu_Course is a Super Key.
  - As the rules mentioned above clearly tell that for a table to be in BCNF, it must follow the property that for functional dependency X–>Y, X must be in Super Key and here this property fails, that's why this table is not in BCNF.

- How to Satisfy BCNF?
  - For satisfying this table in BCNF, we have to decompose it into further tables. Here is the full procedure through which we transform this table into BCNF.
  - Let us first divide this main table into two tables **Stu_Branch and Stu_Course** Table.

- Stu_Branch Table

Stu_ID	Stu_Branch
101	Computer Science & Engineering
102	Electronics & Communication Engineering

Candidate Key for this table: Stu_ID

- Stu_Course Table

Stu_Course	Branch_Number	Stu_Course_No
DBMS	B_001	201
Computer Networks	B_001	202
VLSI Technology	B_003	401
Mobile Communication	B_003	402

Candidate Key for this table: Stu_Course.

- Stu_ID to Stu_Course_No Table

Stu_ID	Stu_Course_No
101	201
101	202
102	401
102	402

- Candidate Key for this table: {Stu_ID, Stu_Course_No}.
- After decomposing into further tables, now it is in BCNF, as it is passing the condition of Super Key, that in functional dependency X−>Y, X is a Super Key.

Exercise

UserID	U_email	Fname	Lname	City	State	Zip
MA12	Mani@ymail.com	MANISH	JAIN	BILASPUR	CHATISGARH	458991
PO45	Pooja.g@gmail.co	POOJA	MAGG	KACCH	GUJRAT	832212
LA33	Lavle98@jj.com	LAVLEEN	DHALLA	RAIPUR	CHATISGARH	853578
CH99	Cheki9j@ih.com	CHIMAL	BEDI	TRICHY	TAMIL NADU	632011
DA74	Danu58@g.com	DANY	JAMES	CHENNAI	TAMIL NADU	645018

Multivalue Dependency

- If two or more independent relations are kept in a single relation or we can say multivalue dependency occurs when the presence of one or more rows in a table implies the presence of one or more other rows in that same table.
- Put another way, two attributes (or columns) in a table are independent of one another, but both depend on a third attribute.
- A multivalued dependency always requires at least three attributes because it consists of at least two attributes that are dependent on a third.

- For a dependency A -> B, if for a single value of A, multiple values of B exist, then the table may have a multi-valued dependency.
- The table should have at least 3 attributes and B and C should be independent for A ->> B multivalued dependency.

Example:

Person	Mobile	Food_Likes
Mahesh	9893/9424	Burger/Pizza
Ramesh	9191	Pizza

Person->-> mobile,
Person ->-> food_likes

- This is read as “person multi determines mobile” and “person multi determines food_likes.”
- Note that a functional dependency is a special case of multivalued dependency.
- In a functional dependency X -> Y, every x determines exactly one y, never more than one.

Fourth Normal Form (4NF)

- The Fourth Normal Form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than a candidate key.
- It builds on the first three normal forms (1NF, 2NF, and 3NF) and the Boyce-Codd Normal Form (BCNF).
- It states that, in addition to a database meeting the requirements of BCNF, it must not contain more than one multivalued dependency.

- Properties
    - A relation R is in 4NF if and only if the following conditions are satisfied:

      1. It should be in the Boyce-Codd Normal Form (BCNF).

      2. The table should not have any Multi-valued Dependency.

- A table with a multivalued dependency violates the normalization standard of the Fourth Normal Form (4NF) because it creates unnecessary redundancies and can contribute to inconsistent data. To bring this up to 4NF, it is necessary to break this information into two tables.

- Example: Consider the database table of a class that has two relations R1 contains student ID(SID) and student name (SNAME) and R2 contains course id(CID) and course name (CNAME).

Table R1

SID	SNAME
S1	A
S2	B

Table R2

CID	CNAME
C1	C
C2	D

- When their cross-product is done it results in multivalued dependencies.

Table R1 X R2

SID	SNAME	CID	CNAME
S1	A	C1	C
S1	A	C2	D
S2	B	C1	C
S2	B	C2	D

- **Multivalued dependencies (MVD) are:**

  SID->->CID;   SID->->CNAME;   SNAME->->CNAME

Fifth normal form (5NF)

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

Example

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
Chemistry	Praveen

P3

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

Relational Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.