# Data Normalization

Tushar B. Kute,

http://tusharkute.com

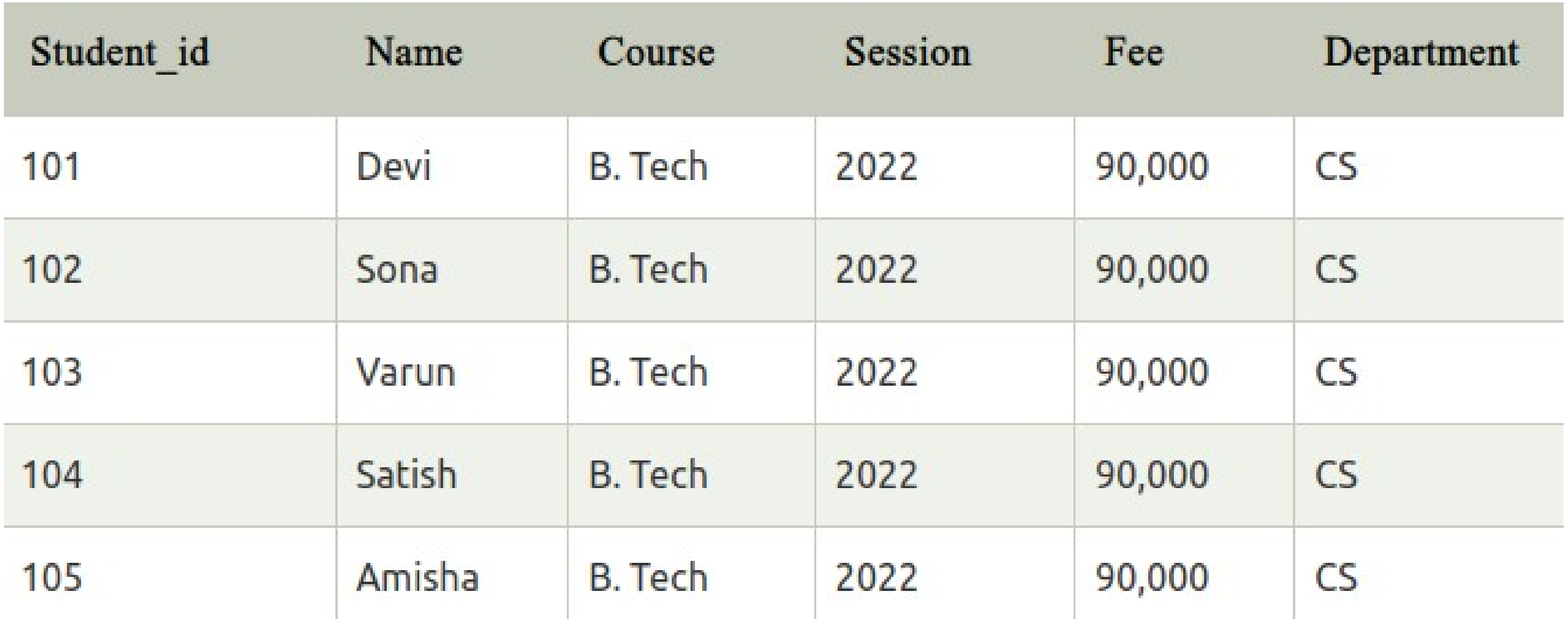
## Data redundancy

- In DBMS, when the same data is stored in different tables, it causes data redundancy.
- Sometimes, it is done on purpose for recovery or backup of data, faster access of data, or updating data easily.
- Redundant data costs extra money, demands higher storage capacity, and requires extra effort to keep all the files up to date.

## Data redundancy

- Sometimes, unintentional duplicity of data causes a problem for the database to work properly, or it may become harder for the end user to access data.
- Redundant data unnecessarily occupy space in the database to save identical copies, which leads to space constraints, which is one of the major problems.

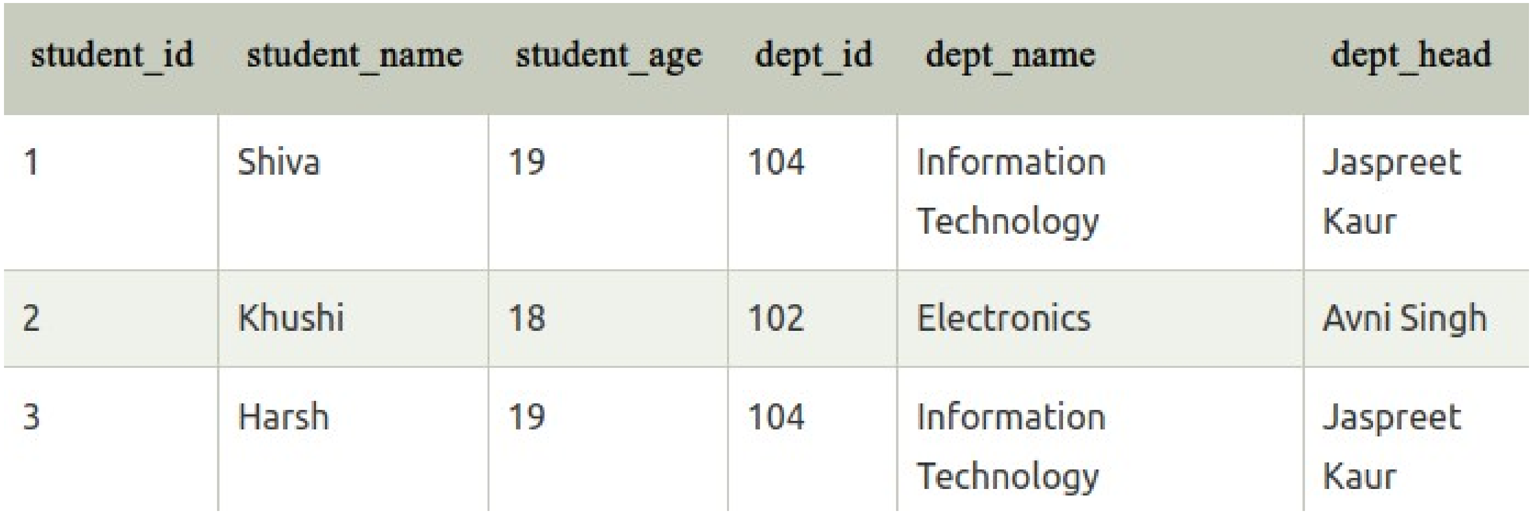
## Data redundancy



| Student_id | Name | Course | Session | Fee | Department |
|---|---|---|---|---|---|
| 101 | Devi | B. Tech | 2022 | 90,000 | CS |
| 102 | Sona | B. Tech | 2022 | 90,000 | CS |
| 103 | Varun | B. Tech | 2022 | 90,000 | CS |
| 104 | Satish | B. Tech | 2022 | 90,000 | CS |
| 105 | Amisha | B. Tech | 2022 | 90,000 | CS |

## Problems caused by redundancy

- Redundancy in DBMS gives rise to anomalies.
- In a database management system, the problems that occur while working on data include inserting, deleting, and updating data in the database.



| student_id | student_name | student_age | dept_id | dept_name | dept_head |
|---|---|---|---|---|---|
| 1 | Shiva | 19 | 104 | Information Technology | Jaspreet Kaur |
| 2 | Khushi | 18 | 102 | Electronics | Avni Singh |
| 3 | Harsh | 19 | 104 | Information Technology | Jaspreet Kaur |

## Insertion Anomaly

- Insertion anomaly arises when you are trying to insert some data into the database, but you are not able to insert it.
- Example:
  - If you want to add the details of the student in the above table, then you must know the details of the department; otherwise, you will not be able to add the details because student details are dependent on department details.

## Deletion Anomaly

- Deletion anomaly arises when you delete some data from the database, but some unrelated data is also deleted; that is, there will be a loss of data due to deletion anomaly.
- Example:
  - If we want to delete the student detail, which has student_id 2, we will also lose the unrelated data, i.e., department_id 102, from the above table.

## Updation Anomaly

- An update anomaly arises when you update some data in the database, but the data is partially updated, which causes data inconsistency.
- Example:
  - If we want to update the details of dept_head from Jaspreet Kaur to Ankit Goyal for Dept_id 104, then we have to update it everywhere else; otherwise, the data will get partially updated, which causes data inconsistency.

## Redundancy: Advantages

- Provides Data Security:
  - Data redundancy can enhance data security as it is difficult for cyber attackers to attack data that are in different locations.
- Provides Data Reliability:
  - Reliable data improves accuracy because organizations can check and confirm whether data is correct.
- Create Data Backup:
  - Data redundancy helps in backing up the data.

## Redundancy: Disadvantages

- Data corruption:
  - Redundant data leads to high chances of data corruption.
- Wastage of storage:
  - Redundant data requires more space, leading to a need for more storage space.
- High cost:
  - Large storage is required to store and maintain redundant data, which is costly.

## How to reduce data redundancy?

- Database Normalization: We can normalize the data using the normalization method. In this method, the data is broken down into pieces, which means a large table is divided into two or more small tables to remove redundancy. Normalization removes insert anomaly, update anomaly, and delete anomaly.
- Deleting Unused Data: It is important to remove redundant data from the database as it generates data redundancy in the DBMS. It is a good practice to remove unwanted data to reduce redundancy.

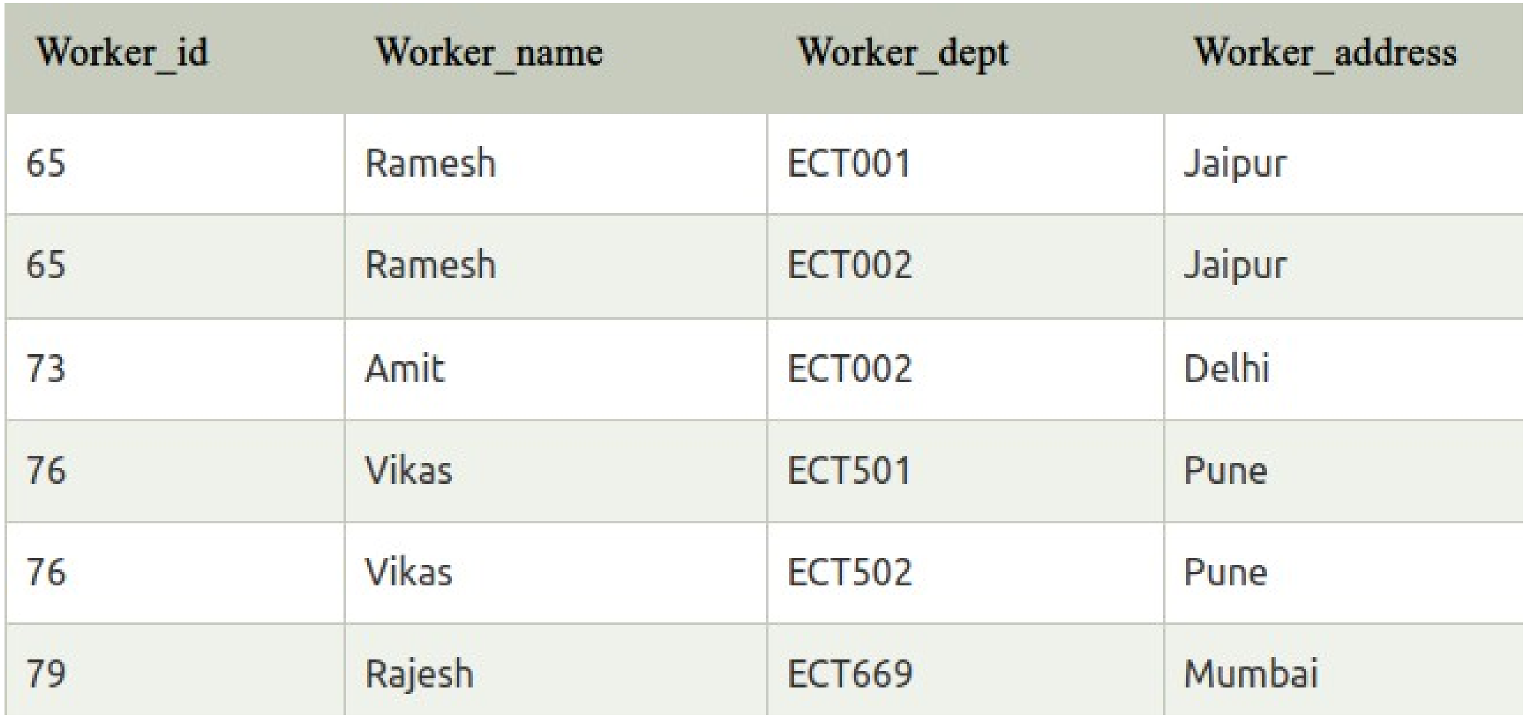
## How to reduce data redundancy?

- Master Data:
  - The data administrator shares master data across multiple systems. Although it does not remove data redundancy, but it updates the redundant data whenever the data is changed.

## Data Anomaly

- Anomaly means inconsistency in the pattern from the normal form. In Database Management System (DBMS), anomaly means the inconsistency occurred in the relational table during the operations performed on the relational table.
- There can be various reasons for anomalies to occur in the database.
  - For example, if there is a lot of redundant data present in our database then DBMS anomalies can occur. If a table is constructed in a very poor manner then there is a chance of database anomaly. Due to database anomalies, the integrity of the database suffers.

# Data Anomaly



| Worker_id | Worker_name | Worker_dept | Worker_address |
|---|---|---|---|
| 65 | Ramesh | ECT001 | Jaipur |
| 65 | Ramesh | ECT002 | Jaipur |
| 73 | Amit | ECT002 | Delhi |
| 76 | Vikas | ECT501 | Pune |
| 76 | Vikas | ECT502 | Pune |
| 79 | Rajesh | ECT669 | Mumbai |

## Updation / Update Anomaly

- When we update some rows in the table, and if it leads to the inconsistency of the table then this anomaly occurs.
- This type of anomaly is known as an updation anomaly. In the above table, if we want to update the address of Ramesh then we will have to update all the rows where Ramesh is present.
- If during the update we miss any single row, then there will be two addresses of Ramesh, which will lead to inconsistent and wrong databases.

## Insertion Anomaly

- If there is a new row inserted in the table and it creates the inconsistency in the table then it is called the insertion anomaly.
- For example, if in the above table, we create a new row of a worker, and if it is not allocated to any department then we cannot insert it in the table so, it will create an insertion anomaly.

## Deletion Anomaly

- If we delete some rows from the table and if any other information or data which is required is also deleted from the database, this is called the deletion anomaly in the database.
- For example, in the above table, if we want to delete the department number ECT669 then the details of Rajesh will also be deleted since Rajesh's details are dependent on the row of ECT669. So, there will be deletion anomalies in the table.

# Example:

| Stu_id | Stu_name | Stu_branch | Stu_club |
|---|---|---|---|
| 2018nk01 | Shivani | Computer science | literature |
| 2018nk01 | Shivani | Computer science | dancing |
| 2018nk02 | Ayush | Electronics | Videography |
| 2018nk03 | Mansi | Electrical | dancing |
| 2018nk03 | Mansi | Electrical | singing |
| 2018nk04 | Gopal | Mechanical | Photography |

## Updation Anomaly

- In the above table, if Shivani changes her branch from Computer Science to Electronics, then we will have to update all the rows.
- If we miss any row, then Shivani will have more than one branch, which will create the update anomaly in the table.

## Insertion Anomaly

- If we add a new row for student Ankit who is not a part of any club, we cannot insert the row into the table as we cannot insert null in the column of stu_club. This is called insertion anomaly.

## Deletion Anomaly

- If we remove the photography club from the college, then we will have to delete its row from the table.
- But it will also delete the table of Gopal and his details. So, this is called deletion anomaly and it will make the database inconsistent.

## Functional Dependency

- The functional dependency is a relationship that exists between two attributes.
- It typically exists between the primary key and non-key attribute within a table.

  $X \rightarrow Y$

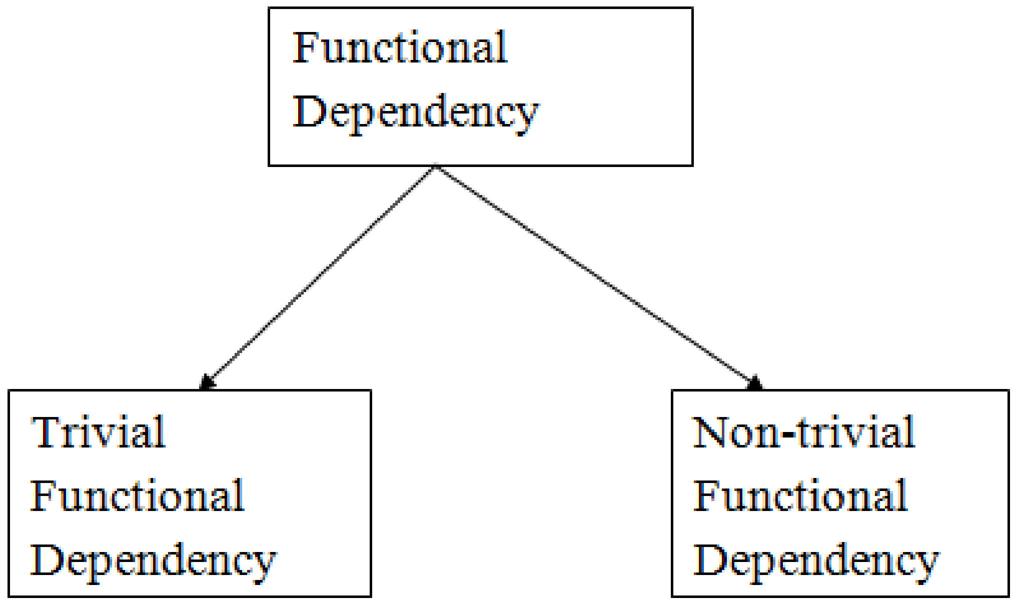
- The left side of FD is known as a determinant, the right side of the production is known as a dependent.

## Functional Dependency

- *For example:*
- Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.
- Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.
- Functional dependency can be written as:

  Emp_Id → Emp_Name
- We can say that Emp_Name is functionally dependent on Emp_Id.

## Functional Dependency


Functional Dependency
Trivial Functional Dependency
Non-trivial Functional Dependency

# Trivial Functional Dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B
- Example:
  - Consider a table with two columns Employee_Id and Employee_Name.
  - {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
  - Employee_Id is a subset of {Employee_Id, Employee_Name}.
  - Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too.

## Trivial Functional Dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.
- When A intersection B is NULL, then A → B is called as complete non-trivial.
- Example:
  - ID → Name,
  - Name → DOB

## Normalization

- A large database defined as a single relation may result in data duplication. This repetition of data may result in:
  - Making relations very large.
  - It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

## Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

## Normalization: Why?

- The main reason for normalizing the relations is removing these anomalies.
- Failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.
- Normalization consists of a series of guidelines that helps to guide you in creating a good database structure.

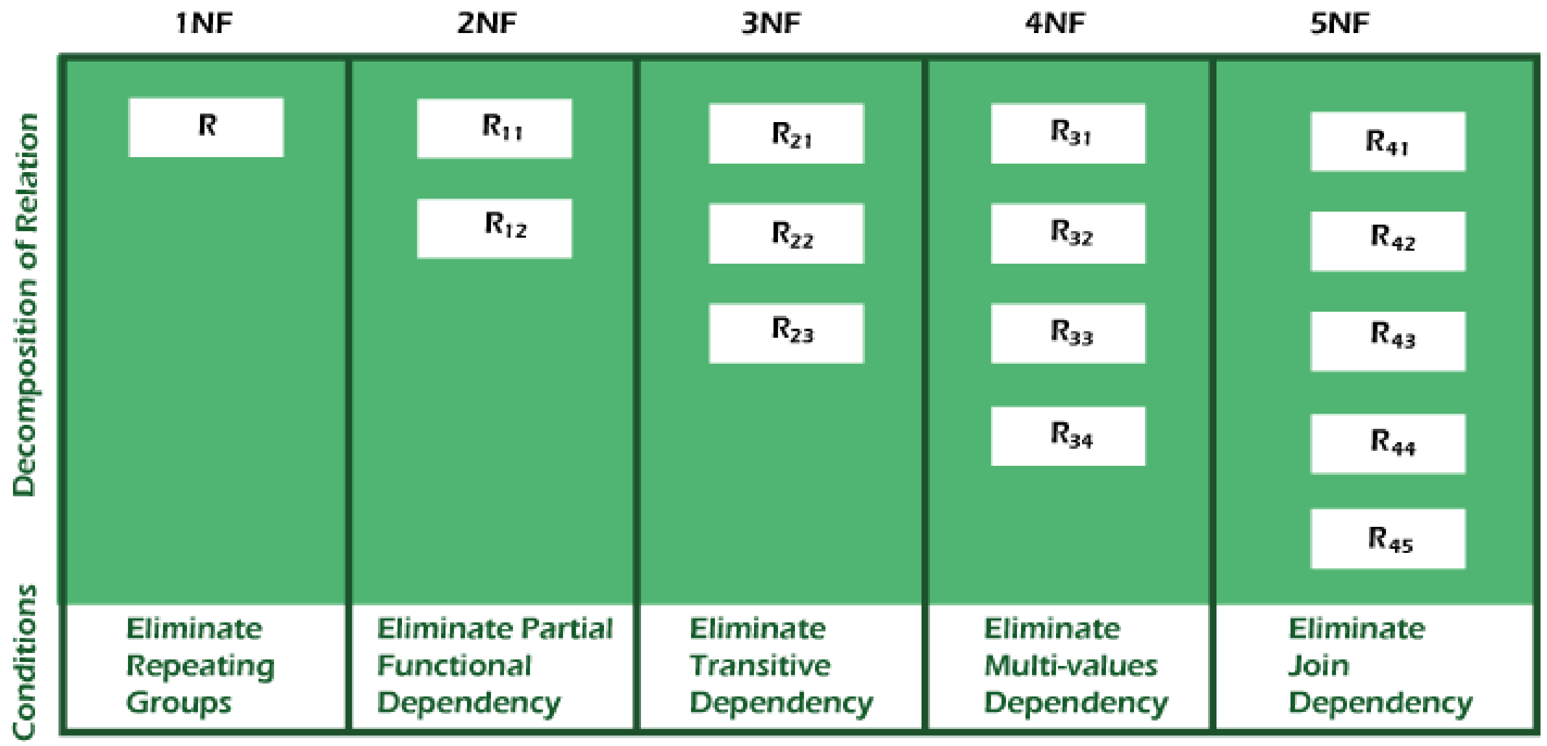
## Normalization

- Data modification anomalies can be categorized into three types:
  - Insertion Anomaly: Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
  - Deletion Anomaly: The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
  - Updation Anomaly: The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

## Normal Forms

- Normalization works through a series of stages called Normal forms.
- The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies constraints.

## Normal Forms


1NF
2NF
3NF
4NF
5NF
Decomposition of Relation
Conditions
R
R11
R12
R21
R22
R23
R31
R32
R33
R34
R41
R42
R43
R44
R45
Eliminate Repeating Groups
Eliminate Partial Functional Dependency
Eliminate Transitive Dependency
Eliminate Multi-values Dependency
Eliminate Join Dependency

## Normal Forms

- 1NF
  - A relation is in 1NF if it contains an atomic value.
- 2NF
  - A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
- 3NF
  - A relation will be in 3NF if it is in 2NF and no transition dependency exists.
- BCNF
  - A stronger definition of 3NF is known as Boyce Codd's normal form.

## Normal Forms

- 4NF
  - A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
- 5NF
  - A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

# Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Enforces the concept of relational integrity.

# Disadvantages of Normalization

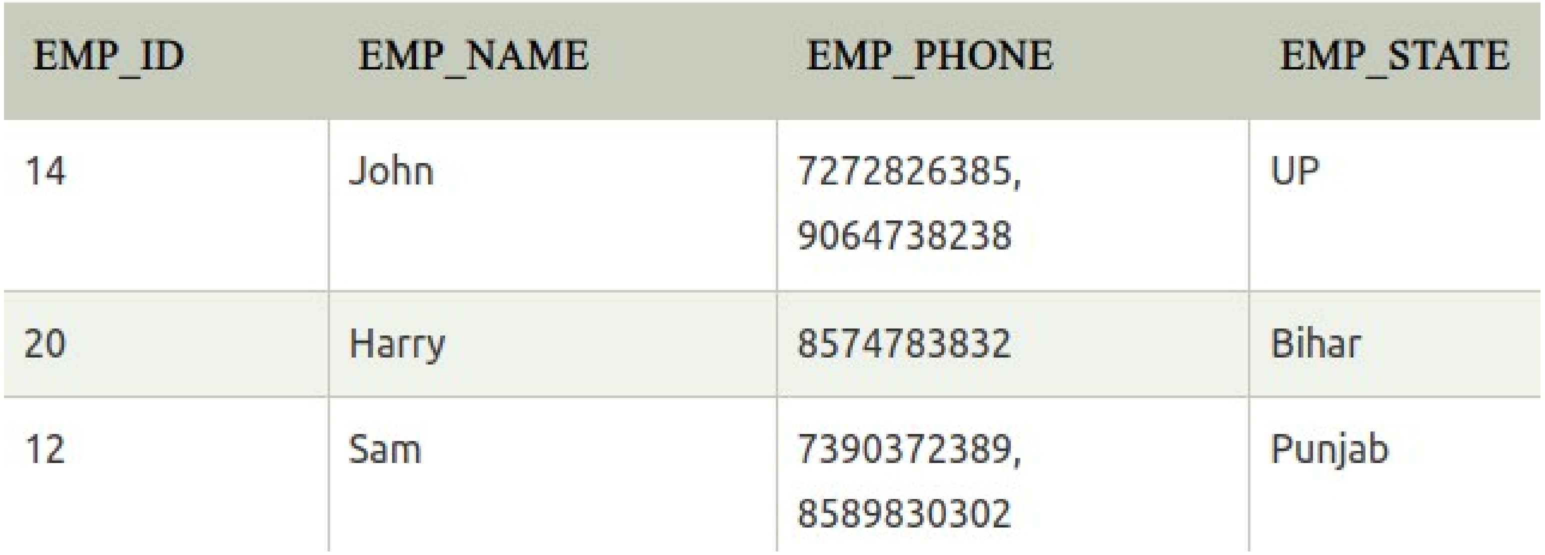
- You cannot start building the database before knowing what the user needs.
- The performance degrades when normalizing the relations to higher normal forms, i.e., 4NF, 5NF.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

## 1NF

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

## 1NF

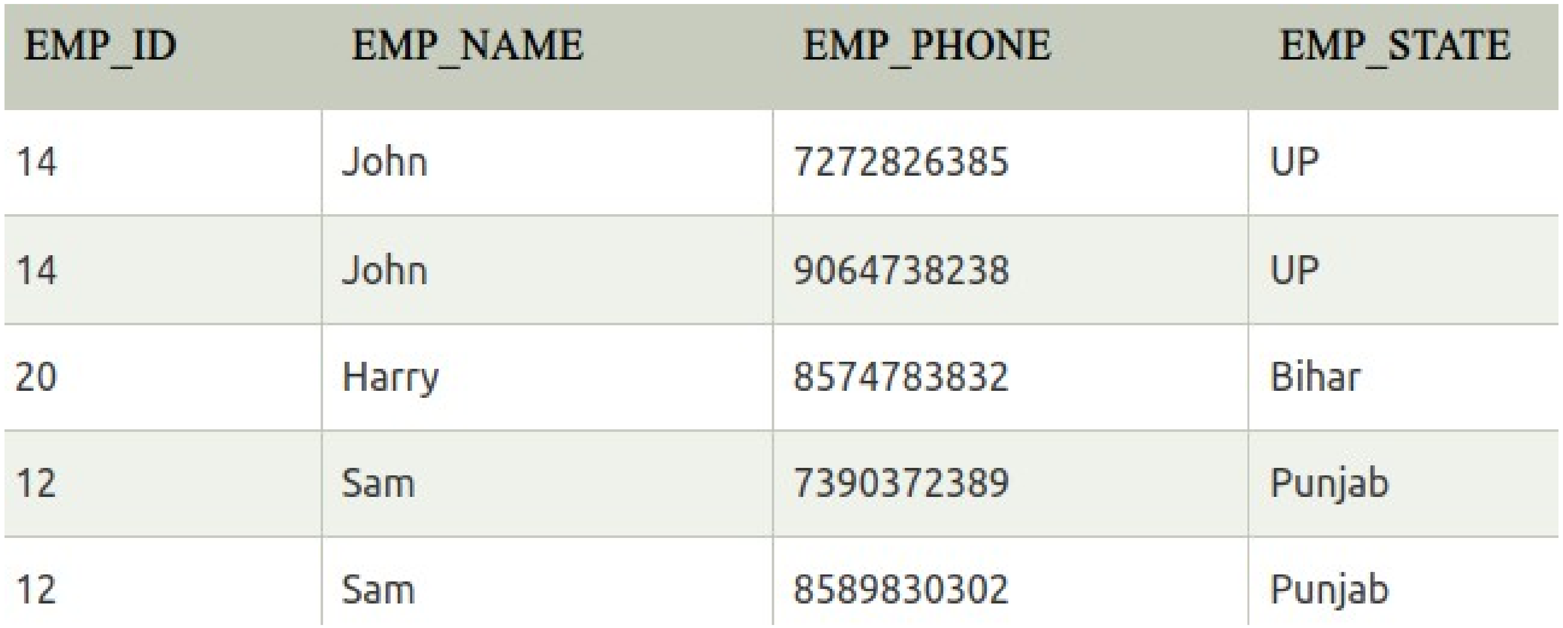
- Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.



| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385, 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389, 8589830302 | Punjab |

# 1NF

- The decomposition of the EMPLOYEE table into 1NF has been shown below:



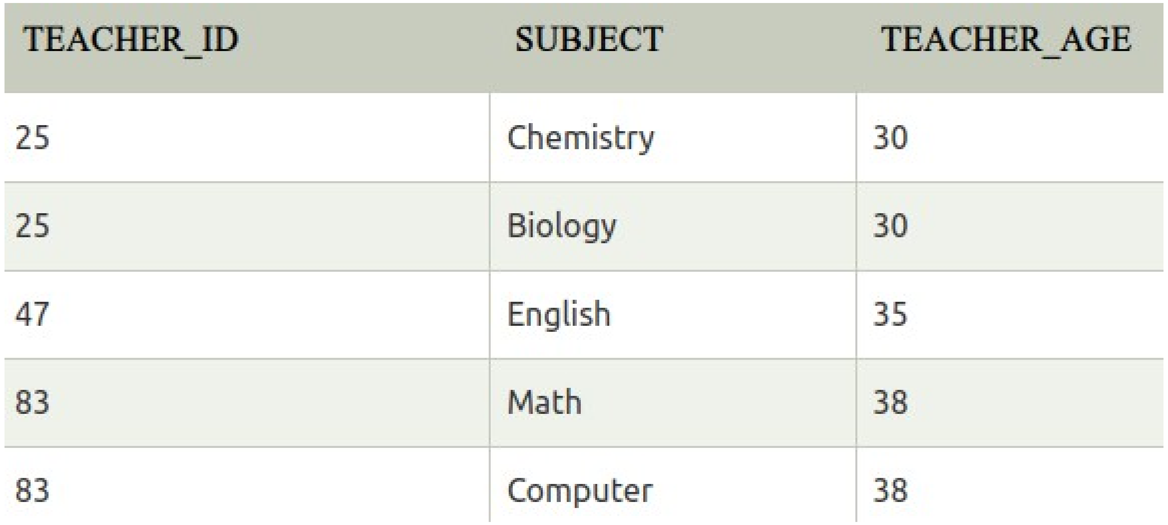
| EMP_ID | EMP_NAME | EMP_PHONE | EMP_STATE |
|---|---|---|---|
| 14 | John | 7272826385 | UP |
| 14 | John | 9064738238 | UP |
| 20 | Harry | 8574783832 | Bihar |
| 12 | Sam | 7390372389 | Punjab |
| 12 | Sam | 8589830302 | Punjab |

## 2NF

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

## 2NF

- Example: Let's assume, a school can store the data of teachers and the subjects they teach.
- In a school, a teacher can teach more than one subject.



| TEACHER_ID | SUBJECT | TEACHER_AGE |
|---|---|---|
| 25 | Chemistry | 30 |
| 25 | Biology | 30 |
| 47 | English | 35 |
| 83 | Math | 38 |
| 83 | Computer | 38 |

## 2NF

- In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.
- To convert the given table into 2NF, we decompose it into two tables:

| TEACHER_ID | TEACHER_AGE |
|---|---|
| 25 | 30 |
| 47 | 35 |
| 83 | 38 |

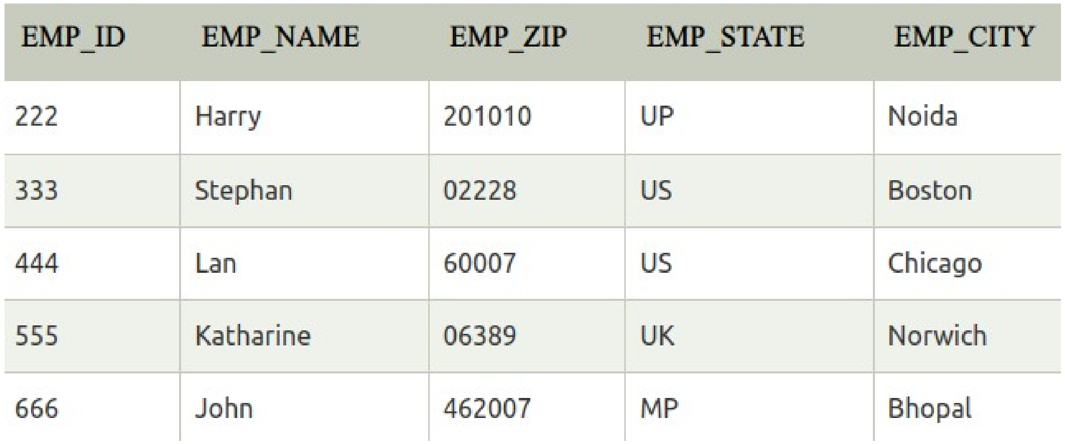
| TEACHER_ID | SUBJECT |
|---|---|
| 25 | Chemistry |
| 25 | Biology |
| 47 | English |
| 83 | Math |
| 83 | Computer |

## 3NF

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.
- A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency X → Y.
  - X is a super key.
  - Y is a prime attribute, i.e., each element of Y is part of some candidate key.

# 3NF

- Super key in the table above:
- {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on
- Candidate key: {EMP_ID}
- Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.



| EMP_ID | EMP_NAME | EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|---|---|
| 222 | Harry | 201010 | UP | Noida |
| 333 | Stephan | 02228 | US | Boston |
| 444 | Lan | 60007 | US | Chicago |
| 555 | Katharine | 06389 | UK | Norwich |
| 666 | John | 462007 | MP | Bhopal |

## 3NF

- Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID.
- The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.
- That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

# 3NF

| EMP_ID | EMP_NAME | EMP_ZIP |
|---|---|---|
| 222 | Harry | 201010 |
| 333 | Stephan | 02228 |
| 444 | Lan | 60007 |
| 555 | Katharine | 06389 |
| 666 | John | 462007 |

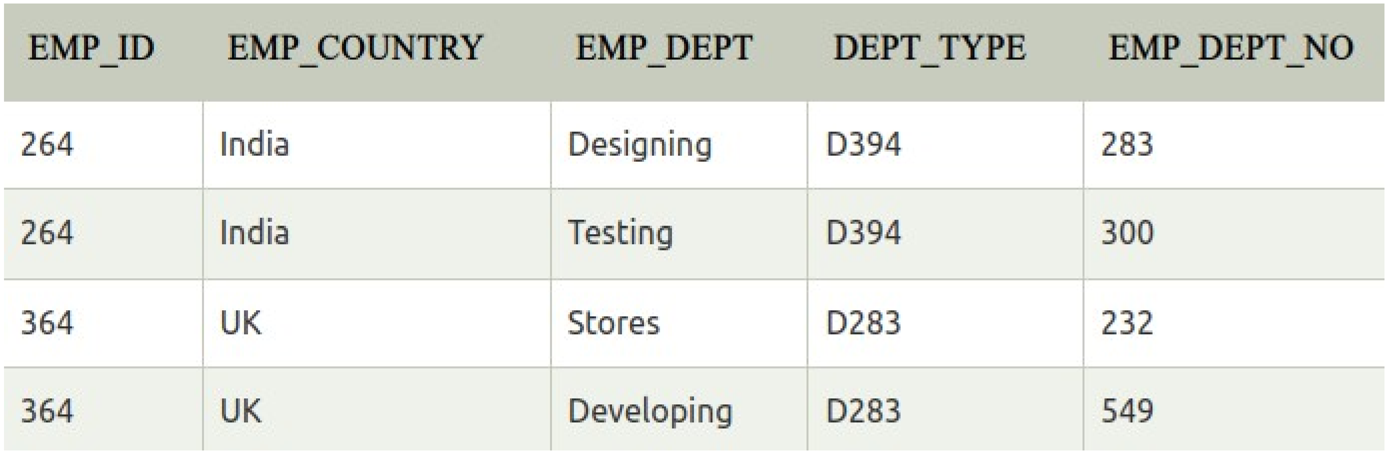
| EMP_ZIP | EMP_STATE | EMP_CITY |
|---|---|---|
| 201010 | UP | Noida |
| 02228 | US | Boston |
| 60007 | US | Chicago |
| 06389 | UK | Norwich |
| 462007 | MP | Bhopal |

## BCNF

- 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.
- 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.



| 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 |

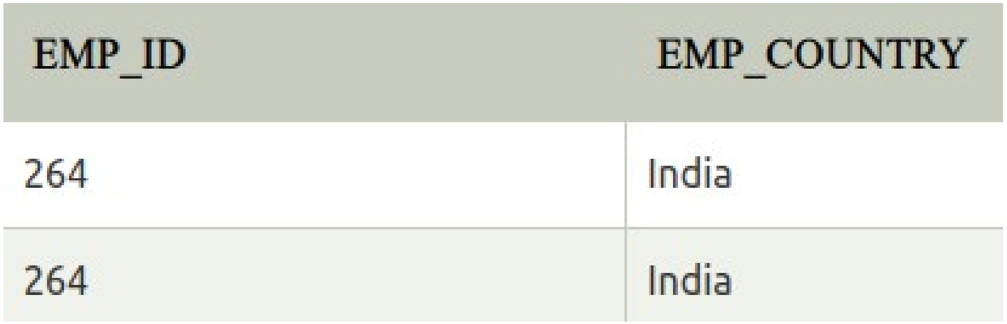
- In the above table Functional dependencies are as follows:

  EMP_ID → EMP_COUNTRY

  EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}
- Candidate key: {EMP-ID, EMP-DEPT}

## BCNF

- The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.
- To convert the given table into BCNF, we decompose it into three tables:
- EMP_COUNTRY table:



| EMP_ID | EMP_COUNTRY |
|---|---|
| 264 | India |
| 264 | India |

## BCNF

| EMP_DEPT | DEPT_TYPE | EMP_DEPT_NO |
|---|---|---|
| Designing | D394 | 283 |
| Testing | D394 | 300 |
| Stores | D283 | 232 |
| Developing | D283 | 549 |

| EMP_ID | EMP_DEPT |
|---|---|
| D394 | 283 |
| D394 | 300 |
| D283 | 232 |
| D283 | 549 |

## BCNF

- Functional dependencies:

  EMP_ID → EMP_COUNTRY

  EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

- Candidate keys:
  - For the first table: EMP_ID
  - For the second table: EMP_DEPT
  - For the third table: {EMP_ID, EMP_DEPT}

## 4NF

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

## 4NF

| STU_ID | COURSE | HOBBY |
|---|---|---|
| 21 | Computer | Dancing |
| 21 | Math | Singing |
| 34 | Chemistry | Dancing |
| 74 | Biology | Cricket |
| 59 | Physics | Hockey |

## 4NF

- The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.
- In the STUDENT relation, a student with STU_ID, 21 contains two courses, Computer and Math and two hobbies, Dancing and Singing.
- So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.
- So to make the above table into 4NF, we can decompose it into two tables:

## 4NF

| STU_ID | COURSE |
|---|---|
| 21 | Computer |
| 21 | Math |
| 34 | Chemistry |
| 74 | Biology |
| 59 | Physics |

| STU_ID | HOBBY |
|---|---|
| 21 | Dancing |
| 21 | Singing |
| 34 | Dancing |
| 74 | Cricket |
| 59 | Hockey |

## 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).

## 5NF

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

## 5NF

- 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:

## 5NF

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

| SUBJECT | LECTURER |
|---|---|
| Computer | Anshika |
| Computer | John |
| Math | John |
| Math | Akash |
| Chemistry | Praveen |

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

# Thank you

@mitu_skillologies @mITuSkillologies @mitu_group @mitu-skillologies @MITUSkillologies

kaggle
@mituskillologies

**Web Resources**
https://mitu.co.in
http://tusharkute.com

@mituskillologies

**contact@mitu.co.in**
**tushar@tusharkute.com**