SCHEMA

Schema can be defined as the design of a database. A database schema defines its entities and the relationship among them. Schema is design to help programmers understand the database and make it useful.

Design of a database is called the schema. Schema is of three types:

- Physical Schema
- Logical Schema
- View Schema

Physical Schema :-

Physical schema can be defined as the design of a database at its physical level. In this level, it is expressed how data is stored in blocks of secondary storage device. Pertains to the actual storage of data like file, indices etc. For example At physical level records are described as chunks of storage (in bytes, gigabytes, terabytes or higher) in memory and these elements often remain hidden from the programmers.

Logical Schema :-

Design at logical level is called **logical schema**, It defines tables, views, and integrity of database constraints. In this level, the programmers as well as the database administrator (DBA) work. The internal details such as implementation of data structure is hidden at this level (available at physical level). In logical level records can be illustrated as fields and attributes along with their data type(s), their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

View Schema :-

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems. At view level, user can able to interact with system, with the help of GUI and enter the details at the screen. The users are not aware of the fact how the data is stored and what data is stored; such details are hidden from them.

Relational Schema

A relational database schema is the tables, columns and relationships that make up a relational database. A relational database schema helps you to organize and understand the structure of a database. This is particularly useful when designing a new database, modifying an existing database to support more functionality, or building integration between databases.

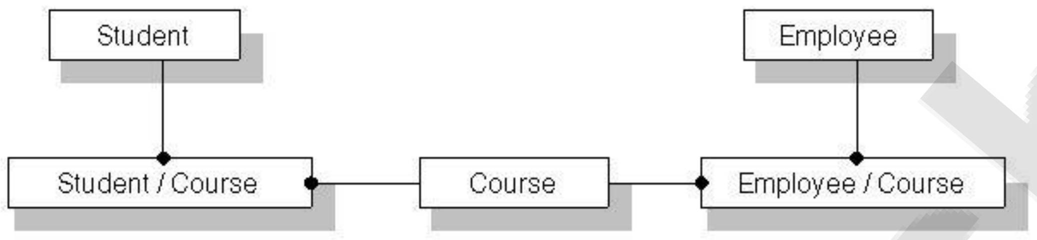
Relational Schema are created in two steps:-

Logical Relational Schema:- The logical schema depicts the structure of the database, showing the tables, columns and relationships with other tables in the database by using constraints and joins.

Physical Relational Schema:- The physical schema is created by actually generating the tables, columns and relationships in the relational database management software (RDBMS).

NOTE:- You could say that a database schema is made up of lots of relation schema and shows how they work together.

For Example:-



There are two intersection entities in this schema: Student/Course and Employee/Course. These handle the two many-to-many relationship:

1) Between Student and Course, and

2) Between Employee and Course.

In the first case, a Student may take many Courses and a Course may be taken by many Students. Similarly, in the second case, an Employee (one of the types of Teachers) may teach many Courses and a Course may be taught by many.

Functional Dependency

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute. Functional dependency (FD) is a set of constraints between two attributes in a relation. One of the attributes is called the determinant and the other attribute is called the determined.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

Types of Functional Dependencies

- Trivial functional dependency
- non-trivial functional dependency
- Multivalued dependency
- Transitive dependency

Inference Rules For Functional Dependencies –

Let S be the set of functional dependencies that are specified on relation schema R. Numerous other dependencies can be inferred or deduced from the functional dependencies in S.

Example :

Let S = {A → B, B → C}

We can infer the following functional dependency from S:

A → C

Armstrong's Inference Rules –

Let A, B and C and D be arbitrary subsets of the set of attributes of the giver relation R, and let AB be the union of A and B. Then,⇒→

- **Reflexivity :**
  If B is subset of A, then A → B
- **Augmentation :**
  If A → B, then AC → BC
- **Transitivity :**
  If A → B and B → C, then A → C.
- **Projectivity or Decomposition Rule :**
  If A → BC, Then A → B and A → C

Proof :

Step 1 : A → BC (GIVEN)

Step 2 : BC → B (Using Rule 1, since B ⊆ BC)

Step 3 : A → B (Using Rule 3, on step 1 and step 2)

- **Union or Additive Rule :**
  If A→B, and A→C Then A→BC.

Proof :

Step 1 : A → B (GIVEN)

Step 2 : A → C (given)

Step 3 : A → AB (using Rule 2 on step 1, since AA=A)

Step 4 : AB → BC (using rule 2 on step 2)

Step 5 : A → BC (using rule 3 on step 3 and step 4)

- **Pseudo Transitive Rule :**
  If A → B, DB → C, then DA → C

Proof :

Step 1 : A → B (Given)

Step 2 : DB → C (Given)

Step 3 : DA → DB (Rule 2 on step 1)

Step 4 : DA → C (Rule 3 on step 3 and step 2)

- **These are not commutative as well as associative.**
  i.e. if X → Y then Y → X x (not possible)

- **Composition Rule :**
  If A → B, and C → D, then AC → BD.
- **Self Determination Rule :**
  A → A is a self determination rule.

Question 1: Prove or disprove the following inference rules for functional dependencies. Note: Read "⇒" as implies a. { X → Y, Z → W} ⇒ XZ → YW ?? b. {X → Y, XY → Z} ⇒ X → Z c. {XY → Z, Y → W} ⇒ XW → Z

Solution :

Method : Use Armstrong's Axioms or Attribute closure to pr ove or disprove.

a. {X → Y, Z → W} ⇒ XZ → YW ??

XZ → XZ

XZ → XW (Z -> W)

XZ → W (decomposition rule)

XZ → XZ

XZ → YZ (X -> Y)

XZ → Y (decomposition rule)

⇒ XZ → YW (union rule)

Hence True.

b. {X → Y, XY → Z} ⇒ X → Z ??

XY→Z

XX → Z (pseudotransitivity rule as X → Y)

⇒ X → Z

Hence True.

c. {XY → Z, Y → W} ⇒ XW → Z ??

W → W

X → X

Y → YW

Z → Z

WX → WX

WY → WY

WZ → WZ

XY → WXYZ

XZ → XZ

YZ → WYZ

Therefore WX → Z is not true

You can also find the attribute closure for WX and show th at closure set does not contain Z.

Question 2: Consider a relational scheme R with attributes A,B,C,D,F and the FDs A → BC B → E CD → EF Prove that fun ctional dependency AD → F holds in R.

Step 1 : A → BC (Given)

Step 2 : A → C (Decomposition Rule applied on step 1)

Step 3 : AD → CD (Augmentation Rule applied on step 2)

Step 4 : CD → EF (Given)

Step 5 : AD → EF (transivity Rule applied on step 3 and 4)

Step 6 : AD → F (Decomposition Rule applied on step 5)

NORMALIZATION

Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data. It was developed by **E. F. Codd.**
It is a systematic approach of decomposing tables to eliminate data redundancy. Normalization is a multi-step process that puts the data into a tabular form by removing the duplicate data from the relation tables. It removes all the duplication issues and incorrect data issues, helping to have a well designed database.

Features of Normalization

- Normalization avoids the data redundancy.
- It is a formal process of developing data structures.
- It promotes the data integrity.
- It ensures data dependencies make sense that means data is logically stored.
- It eliminates the undesirable characteristics like Insertion, Updation and Deletion Anomalies.

Types of Normalization

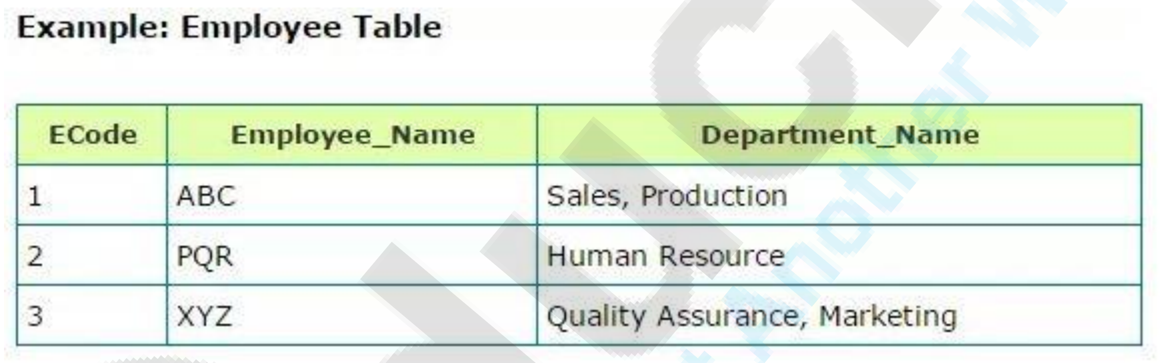
Following are the types of Normalization:
1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. Fourth Normal Form

5. Fifth Normal Form
6. BCNF (Boyce – Codd Normal Form)

<u>First Normal Form:-</u>

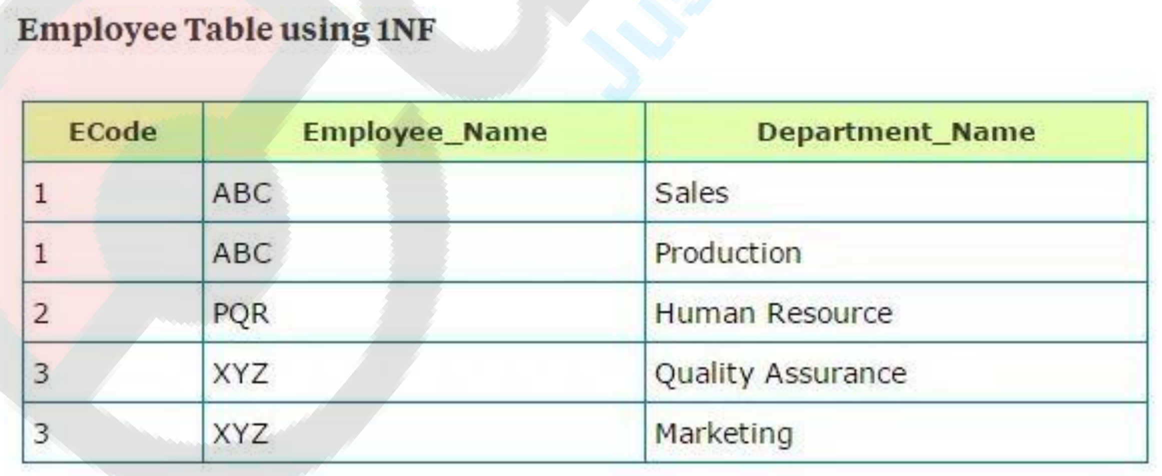
First Normal Form (1NF) is a simple form of Normalization. A table is said to be in First Normal Form (1NF) if and only if each attribute of the relation is atomic. It simplifies each attribute in a relation. Each row in a table should be identified by primary key it means each set of column must have a unique value, it also specifies that No rows of data should have repeating group of column values. A relation is in first normal form if it does not contain any composite or multi-valued attribute.

Example:-



Example: Employee Table

ECode	Employee_Name	Department_Name
1	ABC	Sales, Production
2	PQR	Human Resource
3	XYZ	Quality Assurance, Marketing

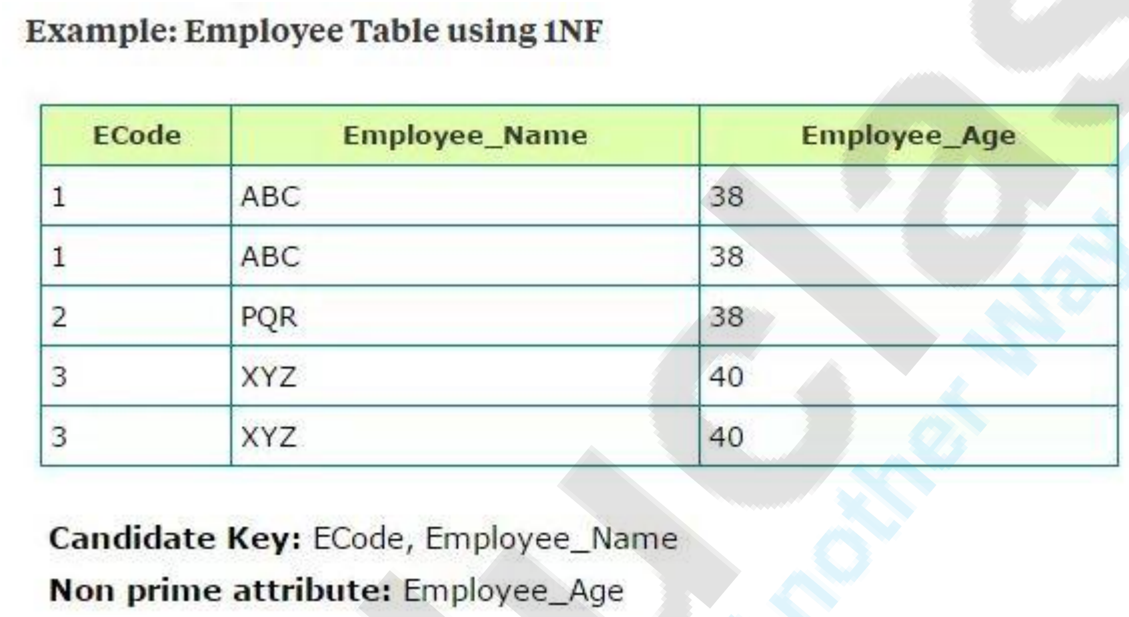


Employee Table using 1NF

ECode	Employee_Name	Department_Name
1	ABC	Sales
1	ABC	Production
2	PQR	Human Resource
3	XYZ	Quality Assurance
3	XYZ	Marketing

Second Normal Form:-

A relation is said to be in a second normal form if and only if, it's in first normal form. In 2NF, the table is required in 1NF. A relation is in 2NF iff it has **No Partial Dependency.**



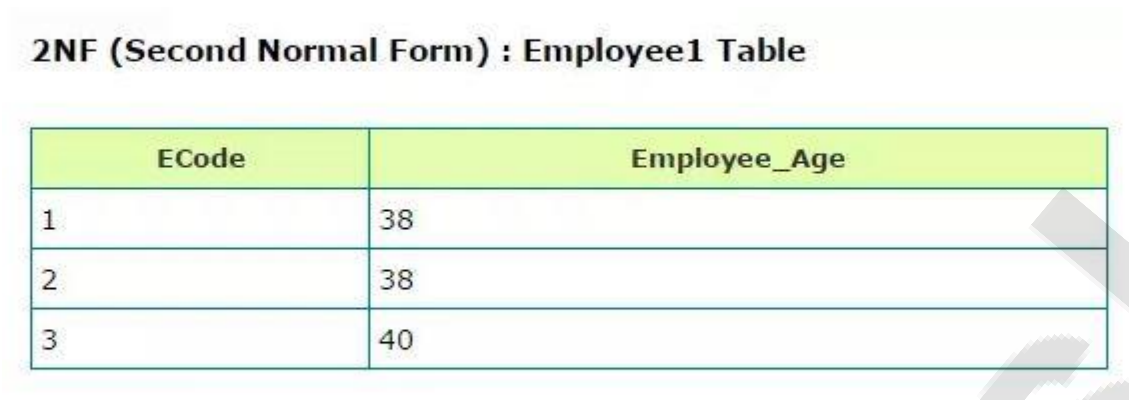
Example: Employee Table using 1NF

ECode	Employee_Name	Employee_Age
1	ABC	38
1	ABC	38
2	PQR	38
3	XYZ	40
3	XYZ	40

Candidate Key: ECode, Employee_Name

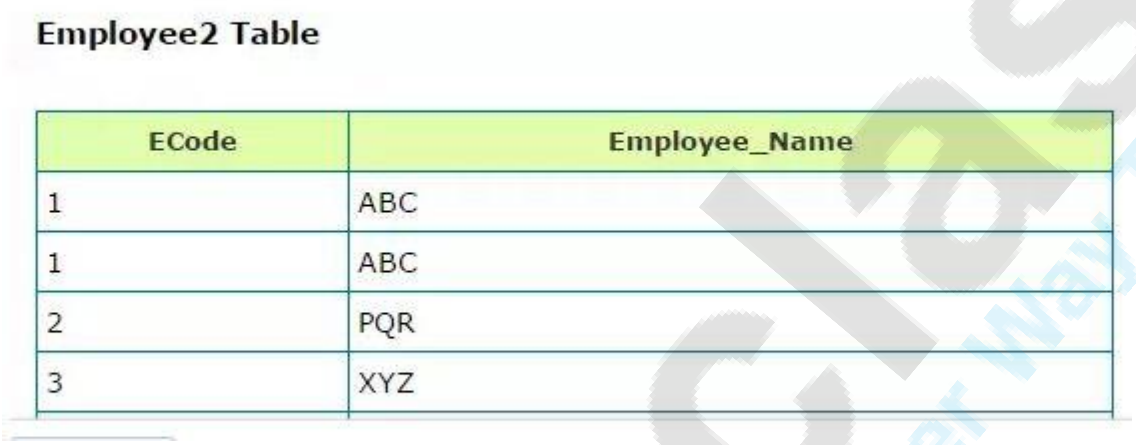
Non prime attribute: Employee_Age

The above table is in 1NF. Each attribute has atomic values. However, it is not in 2NF because non prime attribute Employee_Age is dependent on ECode alone, which is a proper subset of candidate key. This violates the rule for 2NF as the rule says 'No non-prime attribute is dependent on the proper subset of any candidate key of the table'.



2NF (Second Normal Form) : Employee1 Table

ECode	Employee_Age
1	38
2	38
3	40



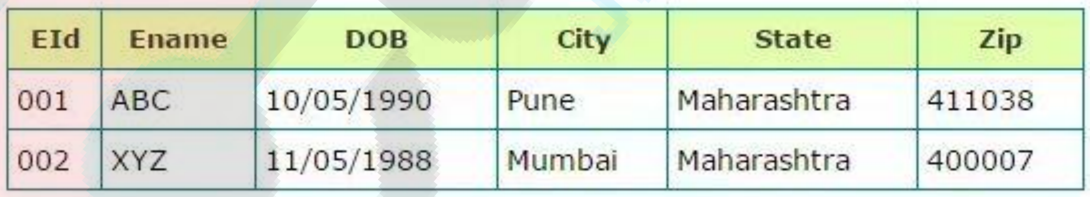
Employee2 Table

ECode	Employee_Name
1	ABC
1	ABC
2	PQR
3	XYZ

Third Normal Form:-

For a relation to be in third normal form it should meet all the requirements of both 1NF and 2NF. Third Normal Form (3NF) is used to minimize the transitive redundancy. 3NF reduces the duplication of data and also achieves the data integrity.

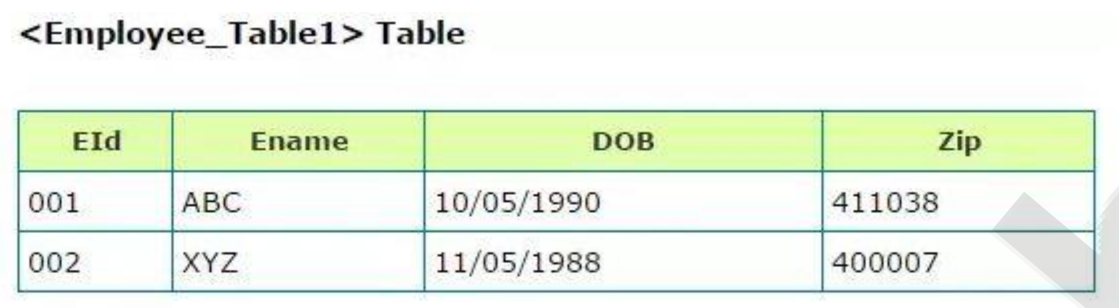
Example:-



EId	Ename	DOB	City	State	Zip
001	ABC	10/05/1990	Pune	Maharashtra	411038
002	XYZ	11/05/1988	Mumbai	Maharashtra	400007

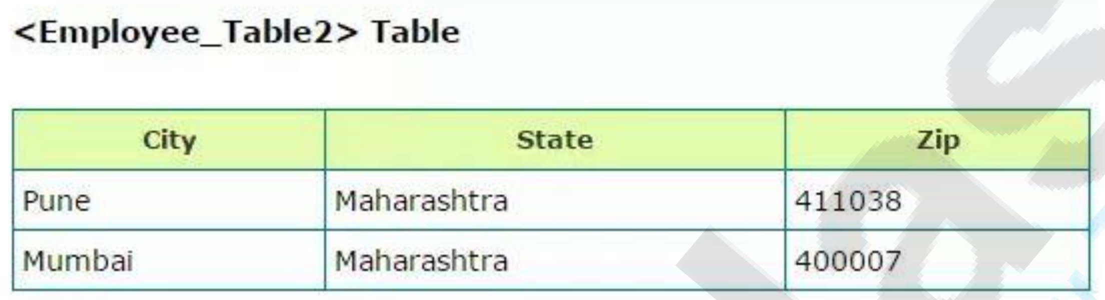
In the above <Employee> table, EId is a primary key but City, State depends upon Zip code.

The dependency between Zip and other fields is called Transitive Dependency. Therefore we apply 3NF. So, we need to move the city and state to the new <Employee_Table2> table, with Zip as a Primary key.



<Employee_Table1> Table

EId	Ename	DOB	Zip
001	ABC	10/05/1990	411038
002	XYZ	11/05/1988	400007



<Employee_Table2> Table

City	State	Zip
Pune	Maharashtra	411038
Mumbai	Maharashtra	400007

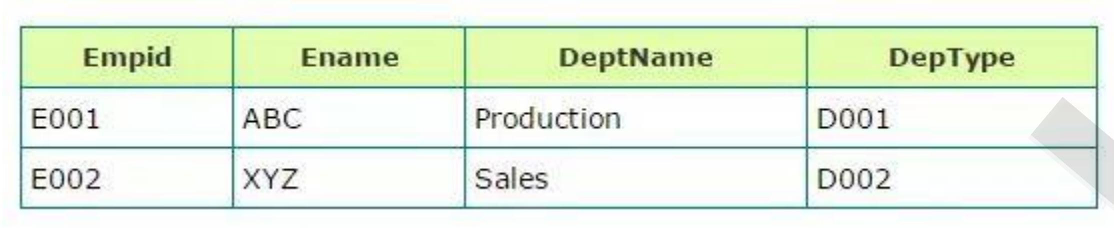
In the above example, using with the 3NF, there is no redundancy of data while inserting the new records.T he City, State and Zip code will be stored in the separate table. And therefore the updation becomes more easier because of no data redundancy.

BCNF (Boyce – Code Normal Form)

Boyce and Codd Normal Form is a higher version of the Third Normal form. BCNF is free from redundancy.

Any table is said to be in BCNF, if its candidate keys do not have any partial dependency on the other attributes. i.e.; in any table with (x, y, z) columns, if (x, y)->z and z->x then it's a violation of BCNF. If (x, y) are composite keys and (x, y)->z, then there should not be any reverse dependency, directly or partially.

Example:-



Empid	Ename	DeptName	DepType
E001	ABC	Production	D001
E002	XYZ	Sales	D002

The functional dependencies are

Empid → EmpName

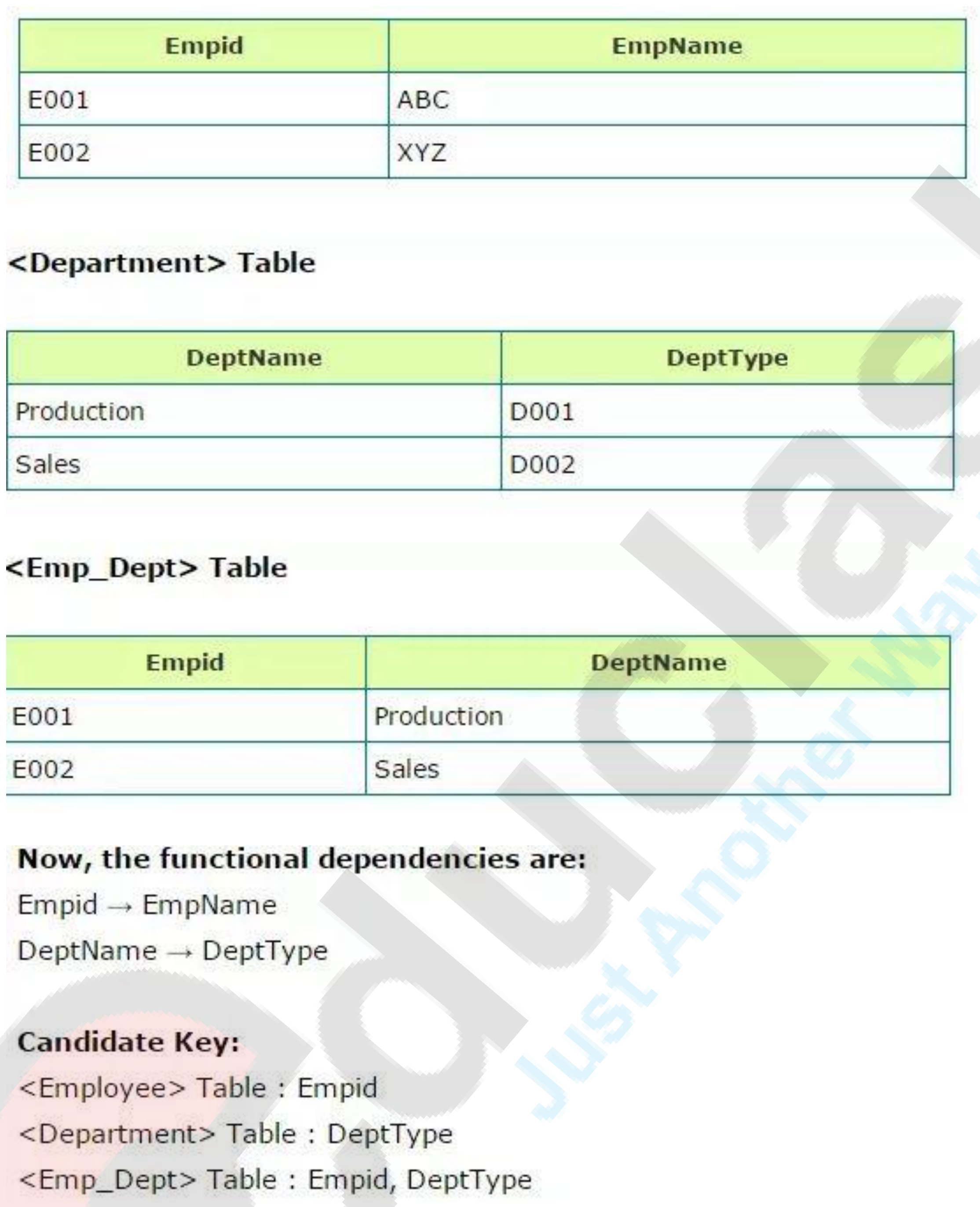
DeptName → DeptType

Candidate Key:

Empid

DeptName

The above table is not in BCNF as neither Empid nor DeptName alone are keys. w e can break the table in three tables to make it comply with BCNF.



Empid	EmpName
E001	ABC
E002	XYZ

<Department> Table

DeptName	DeptType
Production	D001
Sales	D002

<Emp_Dept> Table

Empid	DeptName
E001	Production
E002	Sales

Now, the functional dependencies are:

Empid → EmpName

DeptName → DeptType

Candidate Key:

<Employee> Table : Empid

<Department> Table : DeptType

<Emp_Dept> Table : Empid, DeptType

So, now both the functional dependencies left side part is a key, so it is in the BCNF.

Denormalization

- Denormalization is the process of increasing the redundancy in the database. It is the opposite process of normalization. It is mostly done for improving the performance.
- It is a strategy that database managers use to increase the performance of a database structure.
- Denormalization adds redundant data normalized database for reducing the problems with database queries which combine data from the various tables into a single table.
- The process of adding redundant data to get rid of complex join, in order to optimize database performance. This is done to speed up database access by moving from higher to lower form of normalization.
- Data is included in one table from another in order to eliminate the second table which reduces the number of JOINS in a query and thus achieves performance.

Decomposition

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

Properties of Decomposition

Following are the properties of Decomposition,

1. Lossless Decomposition
2. Dependency Preservation
3. Lack of Data Redundancy

1. Lossless Decomposition

- Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.
- It gives a guarantee that the join will result in the same relation as it was decomposed.
    - **STUDENT :**

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name: **Stu_dept :**

Roll_no	Sname
111	parimal
222	parimal

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

2. Dependency Preservation

- Dependency is an important constraint on the database.
- Every dependency must be satisfied by at least one decomposed table.
- If {A → B} holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in a same relation.

- This decomposition property can only be done by maintaining the functional dependency.
- In this property, it allows to check the updates without computing the natural join of the database structure.

3. Lack of Data Redundancy

- Lack of Data Redundancy is also known as a **Repetition of Information.**
- The proper decomposition should not suffer from any data redundancy.
- The careless decomposition may cause a problem with the data.
- The lack of data redundancy property may be achieved by Normalization process.