Normalization
Database Management System

Normalization

- Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

- It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization

Normalization is used for mainly two purpose,

- Eliminating redundant (useless) data.
- Ensuring data dependencies make sense i.e. data is logically stored.

Normalization

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

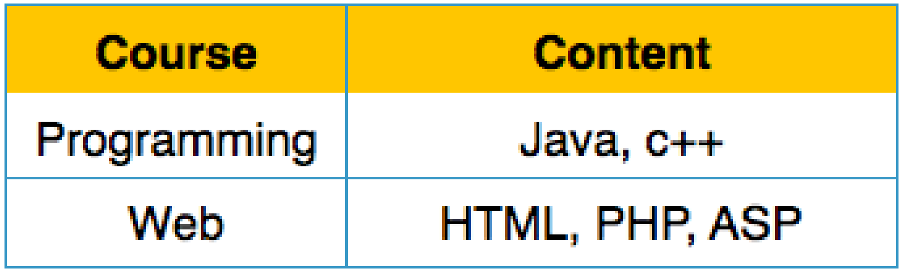
- **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
- **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies** − We tried to insert data in a record that does not exist at all.
- Normalization is a method to remove all these anomalies and bring the database to a consistent state.

Types of Normalization

- 1st Normal Form
- 2nd Normal Form
- 3rd Normal Form
- Boyce Code Normal Form
- 4th Normal Form

First Normal Form

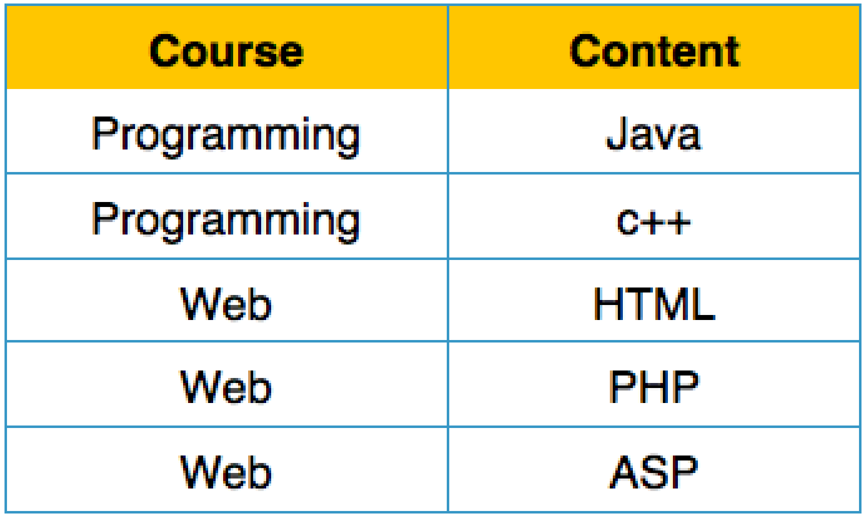
- First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

First Normal Form

- We re-arrange the relation (table) as below, to convert it to First Normal Form.



Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

- Each attribute must contain only a single value from its pre-defined domain.

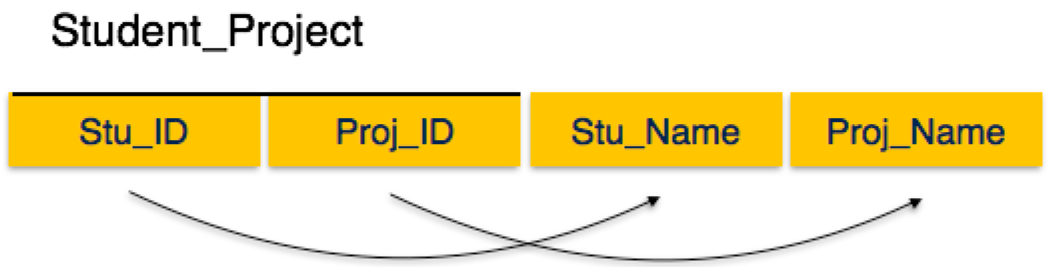
First Normal Form

- After converting non-atomic attribute to atomic if the candidate key breaks it’s rule or fails to create individual relation or rows then create another cast table to solve the problem.

Second Normal Form

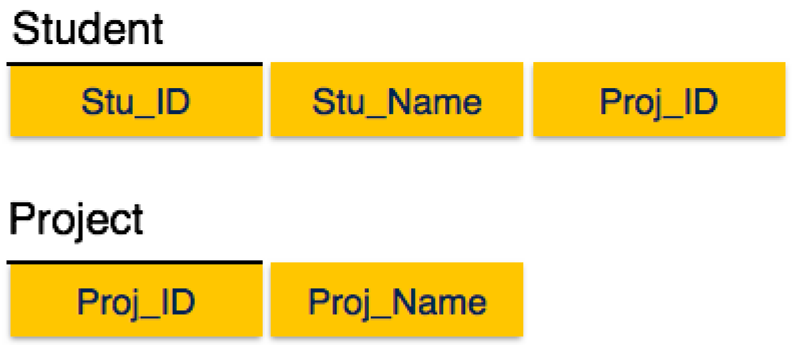
- A relation is in second normal form when it is in 1NF and there is no such non-key attribute that depends on part of the candidate key, but on the entire candidate key.
- It follows from the above definition that a relation that has a single attribute as its candidate key is always in 2NF

Second Normal Form



We see here in Student_Project relation that the **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 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.

Second Normal Form



We broke the relation in two as depicted in the above picture. So there exists no partial dependency. (If you have a relation which contains candidate key with only one column then the relation is already in the 2nd Normal form.)

Third Normal Form

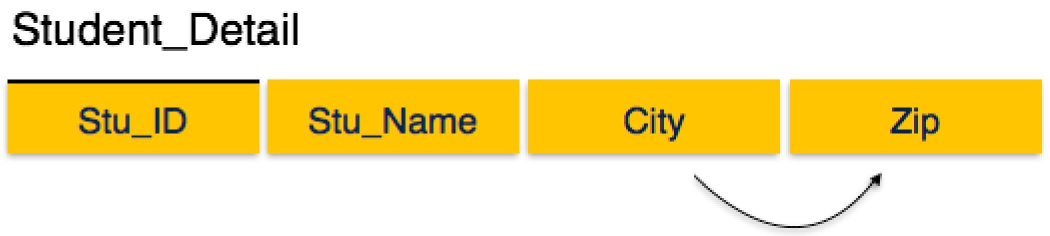
- For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −
- No non-key attribute is transitively dependent on key attribute
- (All we know about transitive dependency that, if

  x -> y and

  y -> z

than x -> z. We have to avoid this type of transitive dependency to convert into 3rd Normal form.)

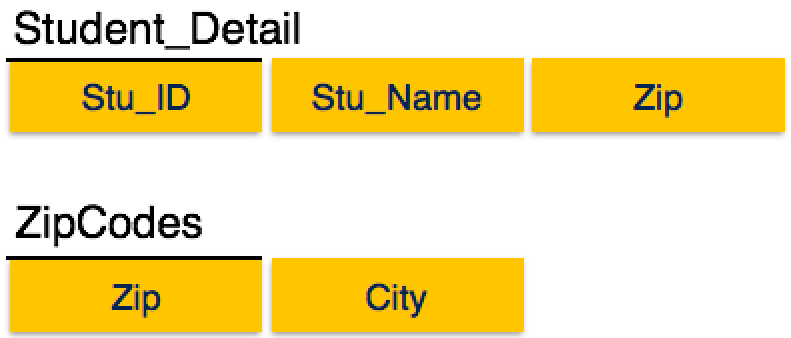
Third Normal Form



We find that in the above Student_detail relation, Stu_ID is the key and only key attribute. We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a key attribute. Additionally, Stu_ID → Zip → City, so there exists **transitive dependency**.

Third Normal Form

To bring this relation into third normal form, we break the relation into two relations as follows −



*****Don’t forget to practice more examples as much as you can.**

Practice

Consider the following table that shows the Movie relation. In the relation, {Movie_Title, Year} form a candidate key.

Movie_Title	Year	Type	Director	Director_DOB	Yr_releases_cnt	Actors
Nothing Hill	1999	Romantic	Roger M	06/06/1956	30	**Hugh G, Rhys**
Lagaan	**2000**	**Drama**	**Ashutosh**	**16/02/1968**	**50**	**Aamir K, Gracy S**