NORMALIZATION

Presented by
JALLURI N V L KRISHNA CHANDRIKA
ASSISTANT PROFESSOR
Aditya degree college, kakinada.

Normalization is a process for evaluating and correcting table structures to minimize data redundancies, there by reducing the data anomalies. 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.

Normalization works through a series of stages called Normal Forms.

- ***First normal form***
- ***Second normal form***
- ***Third normal form***
- ***Boyce code normal form***
- ***Fourth normal form***
- ***Fifth 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

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

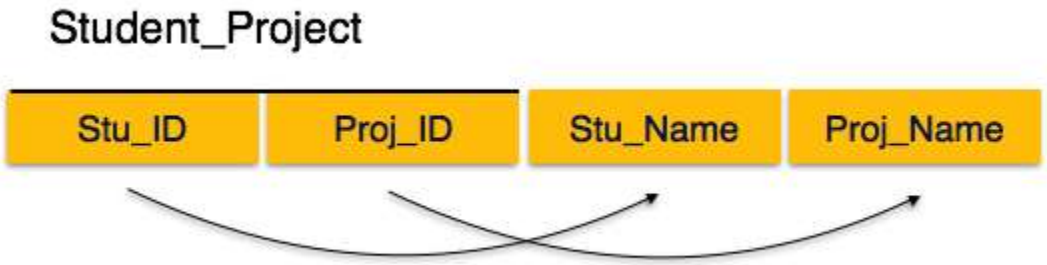
Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form

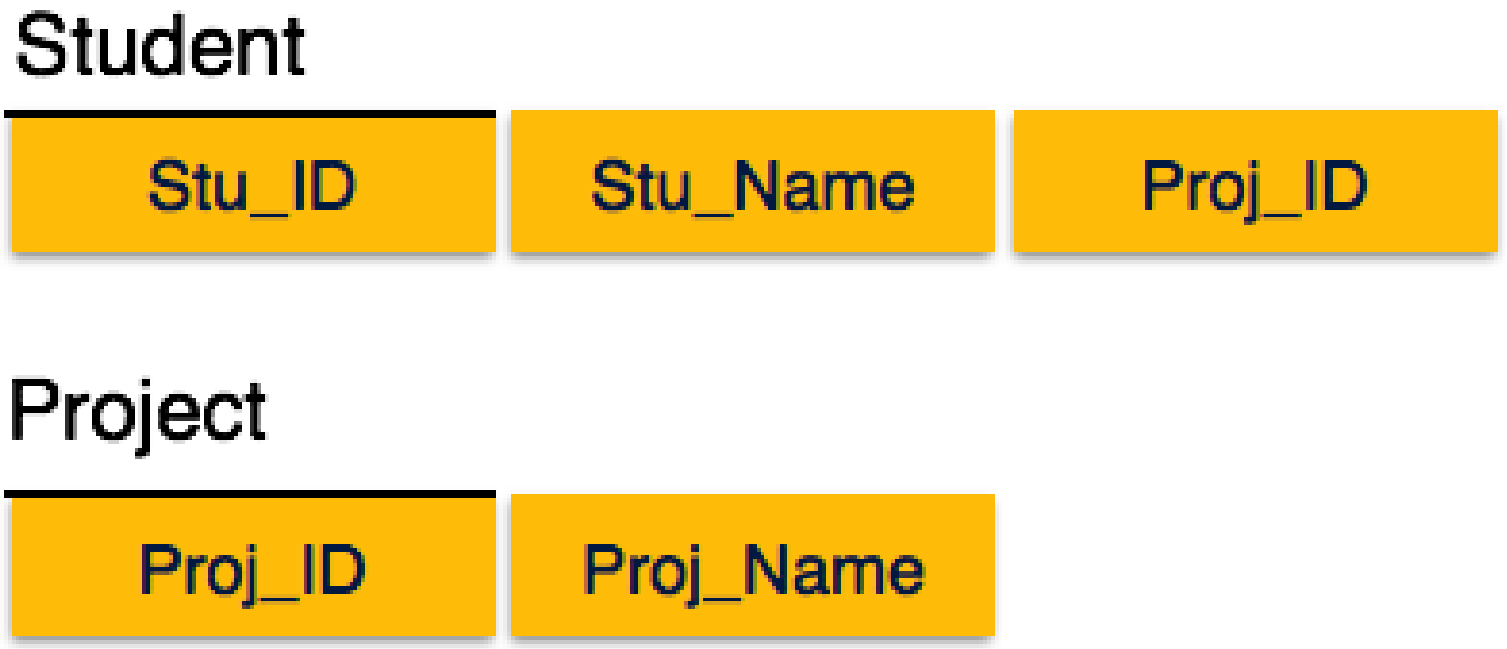
Before we learn about the second normal form, we need to understand the following –

- ***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-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*

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy –

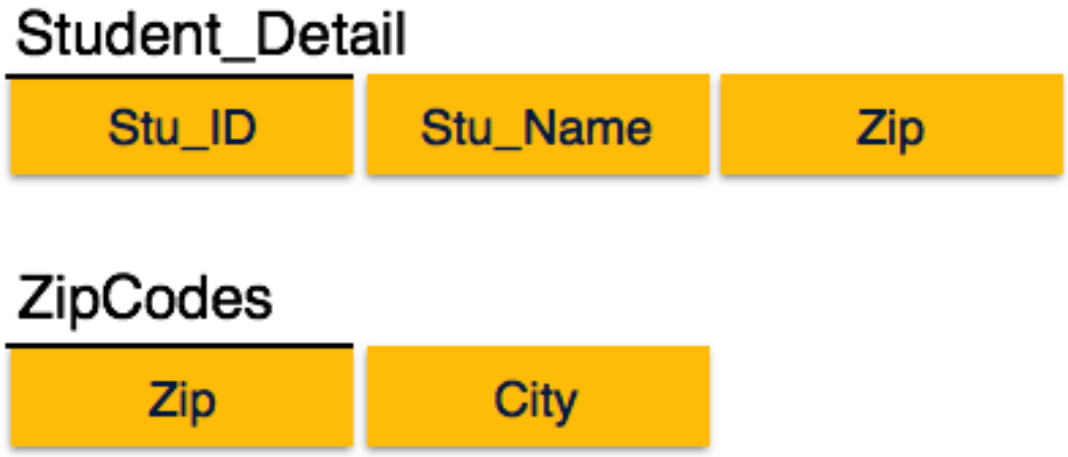
No non-prime attribute is transitively dependent on prime key attribute.

For any non-trivial functional dependency, $X \rightarrow A$, then either –

- *X is a supe rkey or,*
- *A is prime attribute.*

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 superkey 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 relation into two relations as follows –




THANK YOU