DBMS Normalization

Functional Dependency

Functional dependency (FD) is set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by arrow sign (→), that is X→Y, where X functionally determines Y. The left hand side attributes determines the values of attributes at right hand side.

Armstrong's Axioms

If F is set of functional dependencies then the closure of F, denoted as F^+, is the set of all functional dependencies logically implied by F. Armstrong's Axioms are set of rules, when applied repeatedly generates closure of functional dependencies.

- **Reflexive rule:** If alpha is a set of attributes and beta is_subset_of alpha, then alpha holds beta.
- **Augmentation rule:** if a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.
- **Transitivity rule:** Same as transitive rule in algebra, if a → b holds and b → c holds then a → c also hold. a → b is called as a functionally determines b.

Trivial Functional Dependency

- **Trivial:** If an FD X → Y holds where Y subset of X, then it is called a trivial FD. Trivial FDs are always hold.
- **Non-trivial:** If an FD X → Y holds where Y is not subset of X, then it is called non-trivial FD.
- **Completely non-trivial:** If an FD X → Y holds where x intersect Y = Φ, is said to be completely non-trivial FD.

Normalization

If a database design is not perfect it may contain anomalies, which are like a bad dream for database itself. 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 there may be instances when we try to update one data item that has copies of it scattered at several places, few instances of it get updated properly while few are left with there old values. This leaves database in an inconsistent state.
- **Deletion anomalies:** we tried to delete a record, but parts of it 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 database to consistent state and free from any kinds of anomalies.

First Normal Form:

This is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. Values in atomic domain are indivisible units.

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

[*Image: Unorganized relation*]

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

[*Image: Relation in 1NF*]

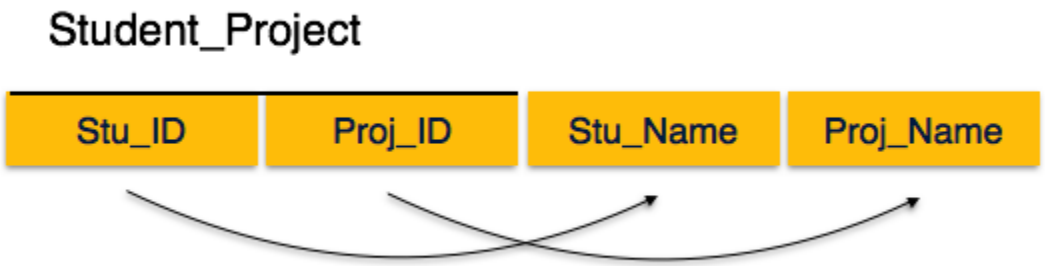
Each attribute must contain only single value from its pre-defined domain.

Second Normal Form:

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

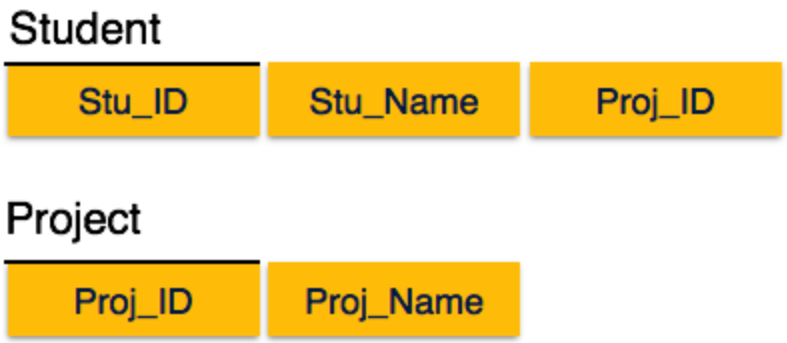
- **Prime attribute:** an attribute, which is part of prime-key, is prime attribute.
- **Non-prime attribute:** an attribute, which is not a part of prime-key, is said to be a non-prime attribute.

Second normal form says, that 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 that Y → A also holds.



[*Image: Relation not in 2NF*]

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.



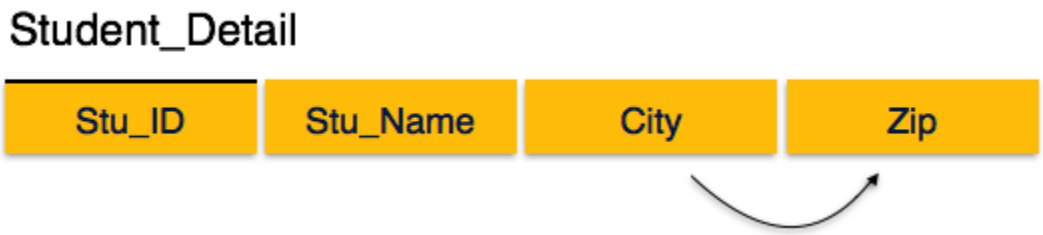
[*Image: Relation in 2NF*]

We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form:

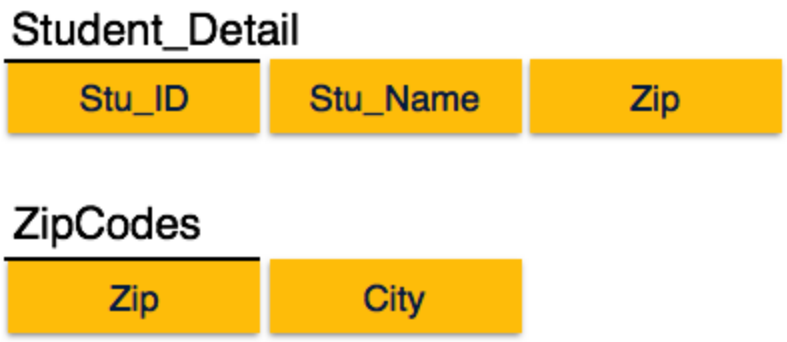
For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy:

- No non-prime attribute is transitively dependent on prime key attribute
- For any non-trivial functional dependency, X → A, then either
- X is a superkey or,
- A is prime attribute.



[*Image: Relation not in 3NF*]

We find that in above depicted Student_detail relation, Stu_ID is 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 City is a prime attribute. Additionally, Stu_ID → Zip → City, so there exists ***transitive dependency***.



[*Image: Relation in 3NF*]

We broke the relation as above depicted two relations to bring it into 3NF.

Boyce-Codd Normal Form:

BCNF is an extension of Third Normal Form in strict way. BCNF states that

- For any non-trivial functional dependency, X → A, then X must be a super-key.

In the above depicted picture, Stu_ID is super-key in Student_Detail relation and Zip is super-key in ZipCodes relation. So,

Stu_ID → Stu_Name, Zip

And

Zip → City

Confirms, that both relations are in BCNF.

Source:

http://www.tutorialspoint.com/dbms/database_normalization.htm