

# DBMS Funktionális függések

Forrás:

[http://www.tutorialspoint.com/dbms/database\\_normalization.htm](http://www.tutorialspoint.com/dbms/database_normalization.htm)

# Functional Dependency

- Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>n</sub>, then those two tuples must have to have same values for attributes B<sub>1</sub>, B<sub>2</sub>, ..., B<sub>n</sub>.
- Functional dependency is represented by an arrow sign ( $\rightarrow$ ) that is,  $X \rightarrow Y$ , where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

# Trivial Functional Dependency

- **Trivial** – If a functional dependency (FD)  $X \rightarrow Y$  holds, where  $Y$  is a subset of  $X$ , then it is called a trivial FD. Trivial FDs always hold.
- **Non-trivial** – If an FD  $X \rightarrow Y$  holds, where  $Y$  is not a subset of  $X$ , then it is called a non-trivial FD.
- **Completely non-trivial** – If an FD  $X \rightarrow Y$  holds, where  $X \cap Y = \emptyset$ , it is said to be a completely non-trivial FD.

# 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.

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

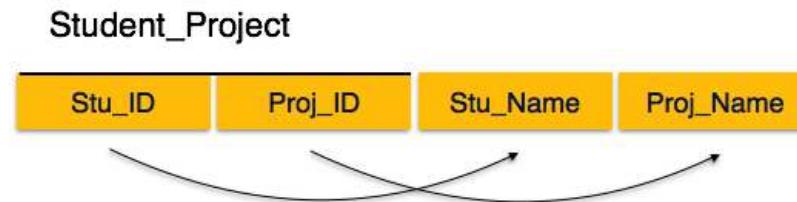
Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

# Second Normal Form

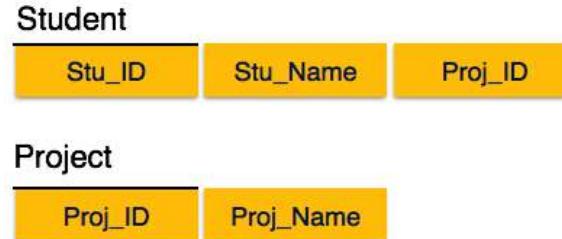
- Before we learn about the second normal form, we need to understand the following –
  - **Prime attribute** – An attribute, which is a part of the prime-key, is known as a prime attribute.
  - **Non-prime attribute** – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

# Second Normal Form

- If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if  $X \rightarrow A$  holds, then there should not be any proper subset Y of X, for which  $Y \rightarrow A$  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 superkey 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,  $Stu\_ID \rightarrow Zip \rightarrow City$ , so there exists transitive dependency.
- To bring this relation into third normal form, we break the relation into two relations as follows –

Student_Detail		
Stu_ID	Stu_Name	Zip
<u>ZipCodes</u>		
Zip	City	

# Boyce-Codd Normal Form

- Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that –
  - For any non-trivial functional dependency,  $X \rightarrow A$ , X must be a super-key.
- In the above image, Stu\_ID is the super-key in the relation Student\_Detail and Zip is the super-key in the relation ZipCodes. So,
  - $\text{Stu\_ID} \rightarrow \text{Stu\_Name}, \text{Zip}$
  - and
  - $\text{Zip} \rightarrow \text{City}$
- Which confirms that both the relations are in BCNF.