Database Management Systems
(DBMS)

(Functional Dependency & Normalization)

For

BCA Part-III (Session 2017-20) Students

BY

ANANT KUMAR
MCA, M. Phil, M. Tech.
Faculty Member
Department of Computer Science
J. D. Women's College, Patna

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 A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

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

Armstrong's Axioms

If F is a 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 a set of rules, that when applied repeatedly, generates a 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 holds. a → b is called as a functionally that determines b.

Trivial Functional Dependency

- **Trivial** − If a functional dependency (FD) X → 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 → 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 → Y holds, where x intersect Y = Φ, 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.

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

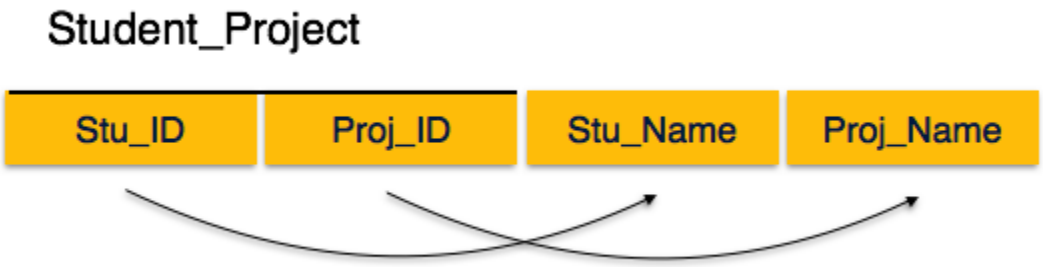
Each attribute must contain only a single value from its pre-defined domain.

Second Normal Form

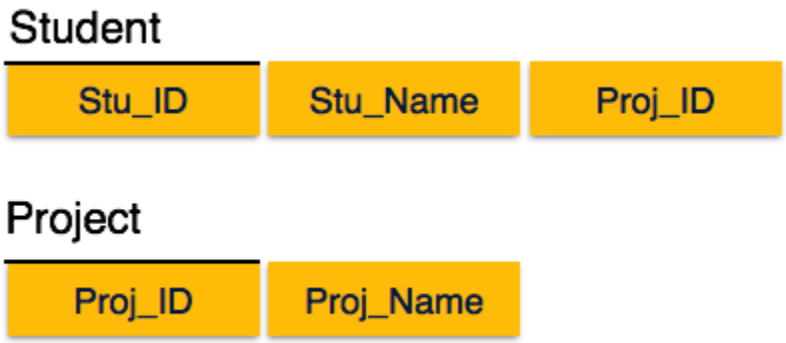
Before we learn about the second normal form, we need to understand the following −

- **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.

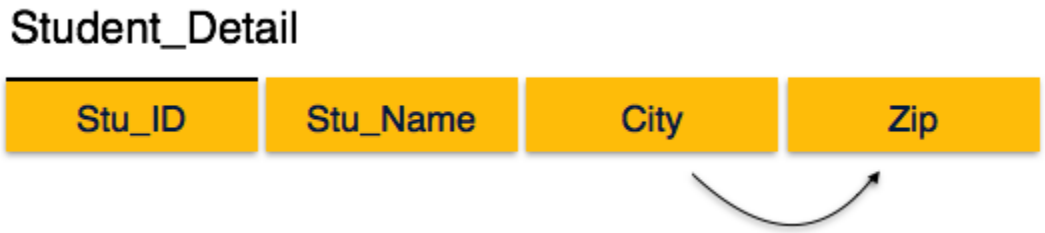


We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

Third Normal Form

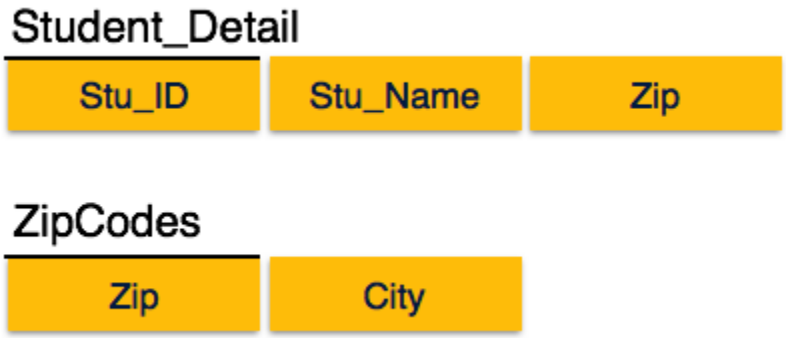
For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → 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 → Zip → City, so there exists **transitive dependency**.

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



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 → 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,

Stu_ID → Stu_Name, Zip

and

Zip → City

Which confirms that both the relations are in BCNF.

Keys in DBMS

Keys in DBMS are −

- **Candidate Key -** The candidate keys in a table are defined as the set of keys that is minimal and can uniquely identify any data row in the table.
- **Primary Key -** The primary key is selected from one of the candidate keys and becomes the identifying key of a table. It can uniquely identify any data row of the table.
- **Super Key -** Super Key is the superset of primary key. The super key contains a set of attributes, including the primary key, which can uniquely identify any data row in the table.
- **Composite Key -** If any single attribute of a table is not capable of being the key i.e it cannot identify a row uniquely, then we combine two or more attributes to form a key. This is known as a composite key.
- **Secondary Key -** Only one of the candidate keys is selected as the primary key. The rest of them are known as secondary keys.
- **Foreign Key -** A foreign key is an attribute value in a table that acts as the primary key in another another. Hence, the foreign key is useful in linking together two tables. Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.

An example to explain the different keys is −

<STUDENT>

Student_Number	Student_Name	Student_Phone	Subject_Number
1	Andrew	6615927284	10
2	Sara	6583654865	20
3	Harry	4647567463	10

<SUBJECT>

Subject_Number	Subject_Name	Subject_Instructor
10	DBMS	Korth
20	Algorithms	Cormen
30	Algorithms	Leiserson

<ENROLL>

Student_Number	Subject_Number
1	10
2	20
3	10

The Super Keys in **<Student>** table are −

```
{Student_Number}
{Student_Phone}
{Student_Number,Student_Name}
{Student_Number,Student_Phone}
{Student_Number,Subject_Number}
{Student_Phone,Student_Name}
{Student_Phone,Subject_Number}
{Student_Number,Student_Name,Student_Phone}
{Student_Number,Student_Phone,Subject_Number}
{Student_Number,Student_Name,Subject_Number}
{Student_Phone,Student_Name,Subject_Number}
```

The Super Keys in **<Subject>** table are −

```
{Subject_Number}
{Subject_Number,Subject_Name}
{Subject_Number,Subject_Instructor}
{Subject_Number,Subject_Name,Subject_Instructor}
{Subject_Name,Subject_Instructor}
```

The Super Key in **<Enroll>** table is −

```
{Student_Number,Subject_Number}
```

The Candidate Key in <Student> table is {Student_Number} or {Student_Phone}

The Candidate Key in <Subject> table is {Subject_Number} or {Subject_Name,Subject_Instructor}

The Candidate Key in <Student> table is {Student_Number, Subject_Number}

The Primary Key in <Student> table is {Student_Number}

The Primary Key in <Subject> table is {Subject_Number}

The Primary Key in <Enroll> table is {Student_Number, Subject_Number}

The Composite Key in <Enroll> table is {Student_Number, Subject_Number}

The Secondary Key in <Student> table is {Student_Phone}

The Secondary Key in <Subject> table is {Subject_Name,Subject_Instructor}

{Subject_Number} is the Foreign Key of <Student> table and Primary key of <Subject> table.