Unit 4 – Relational Database Design

Prof. Sumit P. Makwana

Table of Contents

Functional Dependencies

A functional dependency is a constraint that specifies the relationship between two sets of attributes where one set can accurately determine the value of other sets.

It is denoted as **X → Y**, where X is a set of attributes that is capable of determining the value of Y

The attribute set on the left side of the arrow, **X** is called **Determinant**, while on the right side, **Y** is called the **Dependent.**

Example

Enrolment no	Name	Mark	Pin code	City
1	Arjun	30	361001	Jamnagar
2	Bhim	25	361002	Rajkot
3	Arjun	35	361003	Ahmedabad
4	Nakul	25	361002	Rajkot

Table 1.1 - Student table

From above we can conclude following dependencies

Enrolment No. → Name

Enrolment No. → Mark

Enrolment No → City

Pin code → City

Some Invalid Functional Dependencies

Name → Mark

Name → Pin code

Armstrong's Axioms/Properties of Functional Dependency

William Armstrong in *1974* suggested a few rules related to functional dependency. They are known as **RAT** rules. Where RAT stands for Reflexivity, Augmentation, and Transitivity

Reflexivity

If **A** is a set of attributes and **B** is a subset of **A**, then the functional dependency **A → B** always holds true.

For example, **{ Employee_Id, Name } → Name** is valid.

Augmentation

If a functional dependency **A → B** holds true, then appending any number of the attribute to both sides of dependency doesn't affect the dependency. It remains true.

- For example, **X → Y** holds true then, **ZX → ZY** also holds true.
- For example, if **{ Employee_Id, Name } → { Name }** holds true then, **{ Employee_Id, Name, Age } → { Name, Age }**

Transitivity

If two functional dependencies **X → Y** and **Y → Z** hold true, then **X → Z** also holds true by the rule of Transitivity.

- For example, if **{ Employee_Id } → { Name }** holds true and **{Name } → { Department }** holds true, then **{Employee_Id } → { Department }** also holds true.

Functional dependencies have many advantages, keeping the database design clean, defining the meaning and constraints of the databases, and removing data redundancy are a few of them.

Types of Functional Dependencies

Trivial Functional Dependencies

In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If **X → Y** and **Y is the subset of X**, then it is called trivial functional dependency

Example

Enrolment no	Name	Mark	Pin code	City
1	Arjun	30	361001	Jamnagar
2	Bhim	25	361002	Rajkot
3	Arjun	35	361003	Ahmedabad
4	Nakul	25	361002	Rajkot

Table 2.1 - Student table

In above table below can be the trivial functional dependencies

{Enrolment No., Name} → Name

Name → Name

Non-trivial Functional Dependency

In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant. i.e. If **X → Y** and **Y is not a subset of X**, then it is called Non-trivial functional dependency.

Example

Enrolment no	Name	Mark	Pin code	City

1	Arjun	30	361001	Jamnagar
2	Bhim	25	361002	Rajkot
3	Arjun	35	361003	Ahmedabad
4	Nakul	25	361002	Rajkot

Table 2.2 - Student table

In above table below can be the non-trivial functional dependencies.

Enrolment No. → Name

Enrolment No. → Mark

Transitive Functional Dependency

n transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If a → b & b → c, then a → c. This is a transitive functional dependency

Example

Enrolment no	Name	Mark	Pin code	City
1	Arjun	30	361001	Jamnagar
2	Bhim	25	361002	Rajkot
3	Arjun	35	361003	Ahmedabad
4	Nakul	25	361002	Rajkot

Table 2.3 - Student table

In above table below can be transitive functional dependency.

if **Enrolment No. → Pin code** & **Pin code → city** then **Enrolment No. → city**

Multivalued Functional Dependency

Multivalued Dependency (MVD) exists when one set of attributes determines another set of attributes, and this determination is not based on functional dependency (FD) but rather on the presence of multiple values in the related attributes.

Let R be a relation (table) with attributes A, B, and C. An MVD between attributes B and C in R can be represented as:

A →→ B

A →→ C

This notation indicates that for every distinct value of A, there is well defined set of values for B and a well-defined set of values for C. However, set of values of B is independent of set of values of C and vice versa. (i.e., There is no functional dependency between B and C)

In simpler terms, an MVD states that for a given value in one attribute (A), there can be multiple sets of values in two other attributes (B and C) that are not associated with each other.

Here's an example to illustrate multivalued dependency:

Consider a relation "CourseInfo" with the following attributes:

- CourseID (Primary Key)
- CourseName
- Instructors (a set of instructors teaching the course)
- Textbooks (a set of textbooks used for the course)

In this case, if we have a course with a unique CourseID, there can be multiple instructors and multiple textbooks associated with that course. This creates an MVD:

CourseID →→ Instructors

CourseID →→ Textbooks

This means that for each course (CourseID), there can be multiple sets of instructors and textbooks associated with it.

Normalization

Normalization is the process of organizing data in a database to reduce data redundancy and improve data integrity. Redundancy in relation may cause insertion, deletion and updation anomalies. The main goal of normalization is to eliminate data anomalies and ensure that the data is stored efficiently and accurately. It involves breaking down a large, complex table into smaller, related tables and establishing relationships between them.

Normalization is achieved through steps known as Normal forms to eliminate or reduce redundancy in database tables. Normal form spans from 1st Normal Form to 6th Normal Form.

Benefits of Normalization

- Minimize data redundancy
- Improve data integrity
- Solve insert, update, delete anomalies which can arise through bad relation (table) design.
- Enhance query performance: by reducing size of table the query may run faster and executes in less time
- Easier Maintenance: modifications and updates are easier to perform because changes typically only need to be made in one place
- Flexibility and Scalability: Normalized databases are often more flexible and scalable because data is organized logically.

Insert Anomaly

There are some circumstances when certain fact or record can not be inserted due to bad choice or relation schema.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201
424	Dr. Newsome	29-Mar-2007	?

Source: upload.wikimedia.org/wikipedia/commons/5/5c/Insertion_anomaly.svg

For example, each record in a "Faculty and Their Courses" relation might contain a Faculty ID, Faculty Name, Faculty Hire Date, and Course Code. Therefore, the details of any faculty member who teaches at least one course can be recorded, but a newly hired faculty member who has not yet been assigned to teach any courses cannot be recorded, except by setting the Course Code to null.

Update Anomaly

Some times same information is stored in multiple rows, which may cause update anomaly. Update anomaly can mean after update operation same data can represent multiple fact which leads to logical inconsistency.

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519	94 Chestnut Street	Public Speaking
519	96 Walnut Avenue	Carpentry

Source: upload.wikimedia.org/wikipedia/commons/2/29/Update_anomaly.svg

For example, each record in an "Employees' Skills" relation might contain an Employee ID, Employee Address, and Skill; thus, a change of address for a particular employee may need to be applied to multiple records (one for each skill). If the update is only partially successful – the employee's address is updated on some records but not others – then the relation is left in an inconsistent state. Specifically, the relation provides conflicting answers to the question of what this particular employee's address is.

Deletion anomaly

Deletion anomaly means in some circumstances deletion operation certain data may also leads to deletion of another data which is not intentional and which we do not want to delete.

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

DELETE

Source: upload.wikimedia.org/wikipedia/commons/2/2c/Deletion_anomaly.svg

For example, if we are storing faculty and course data in same relation then this relation is pruned for deletion anomaly. For example, if we want to delete the information of faculty 389, this delete operation will also delete the information of course ENG-206 which we do not want to delete or vice versa.

First Normal Form (1NF)

For a relation to be in first normal form, it must satisfy below conditions.

1. Must not contain any multivalued attribute.
2. Must not contain any composite attribute.

Relation is in first normal for if all of its attributes are single valued attribute.

Example

Enrolment No	Name	Subject
1001	Yudhisthir	Sanskrit, Gujarati
1002	Bhim	Sanskrit, English
1003	Arjun	Gujarati, English, Hindi

Table 3.1 – StudentInfo table

In above relation Subject is multivalued attribute, so this relation is not in first normal form. This can cause update anomalies. To make this relation in first normal form either we can write each subject on different tuple.

Solution: Write each value of multivalued attribute on different rows.

Enrolment No	Name	Subject
1001	Yudhisthir	Sanskrit

1001	Yudhisthir	Gujarati
1002	Bhim	Sanskrit
1002	Bhim	English
1003	Arjun	Gujarati
1003	Arjun	English
1003	Arjun	Hindi

Table 3.2 – StudentInfo table

Now above relation does not contain any multivalued attribute, so it is now in first normal form.

Second Normal Form (2NF)

For a relation to be in second normal form, it must satisfy below conditions.

- Relation must be in first normal form
- Must not contain any partial dependencies.

Partial Dependencies

Partial Dependencies means any non-prime attribute (attribute which not part of any candidate key) is dependent on any proper subset of any candidate key of a relation.

Example

Enrolment No	Certification Course	Fee
1001	Java	5000
1001	.NET Programming	8000
1002	PHP	4000
1003	Java	5000

1003	PHP	4000
1003	.NET Programming	8000

Table 3.3 – StudentCertification table

Here we can see that in above relation

- (Enrolment No, Certification Course) is composite primary key.
- Fee is non-prime attribute because it is not part of any candidate key.

Here Fee can be determined by Certification Course (Functional Dependency is **Certification Course → Fee**)

Hence, we can say that Fee is non-prime attribute which can be determine by Certification Course which is subset of composite candidate key. So, it is partial dependencies. Which can cause data redundancy.

To eliminate this, we can discompose relation into two relations to make these relations into second normal form as follows.

Enrolment No	Certification Course
1001	Java
1001	.NET Programming
1002	PHP
1003	Java
1003	PHP
1003	.NET Programming

Table 3.4 – StudentCertification table

Certification Course	Fee

Java	5000
.NET Programming	8000
PHP	4000

Table 3.5 – CertificationFees table

Note: If table do not have any composite candidate / primary key then table is already in the second normal form.

Third Normal Form (3NF)

For a relation to be in second normal form, it must satisfy below conditions.

- Relation must be in second normal form.
- There must not be any transitive dependencies for any non-prime attribute.

Example

ID	Name	State	Country	Age
1001	Yudhisthir	Gujarat	India	55
1002	Bhim	Madhya Pradesh	India	54
1003	Arjun	Gujarat	India	53

Table 3.6 – PersonDetail table

In above relation ID is candidate key and below listed are various Functional dependencies.

- ID → Name
- ID → State
- ID → Country

- ID → Age
- State → Country

For this relation ID → State and State → Country hold true. So, ID → Country is transitive dependency. Country is non-prime attribute so it violates the rule of third normal form.

To convert this relation into third normal form we can decompose this relation as below.

ID	Name	State	Age
1001	Yudhisthir	Gujarat	55
1002	Bhim	Madhya Pradesh	54
1003	Arjun	Gujarat	53

Table 3.7 – PersonDetail table

State	Country
Gujarat	India
Madhya Pradesh	India

Table 3.8 – StateCountry table

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form is a slightly stronger version of the third normal form (3NF). Sometimes also known as 3.5NF

For a relation to be in Boyce-Codd normal form, it must satisfy below conditions.

- Relation must be in third normal form.

- For every functional dependency, LHS must be super key. (i.e. for every functional dependency X → Y, X must be a super key)

Only in rare cases does a 3NF table not meet the requirements of BCNF. A 3NF table that does not have multiple overlapping candidate keys is guaranteed to be in BCNF.

Fourth Normal Form (4NF)

Fourth Normal Form (4NF) is an advanced level of database normalization that builds upon the concepts of the first three normal forms (1NF, 2NF, and 3NF). 4NF addresses certain types of multi-valued dependencies that can exist in a relational database.

For a relation to be in fourth normal form, it must satisfy below conditions.

- It must be in Third Normal Form (3NF)
- For every non-trivial multi-valued dependency (MVD) X →→ Y (read as "X multi-determines Y"), where X and Y are sets of attributes, X must be a superkey

In simpler terms, 4NF eliminates situations where non-prime attributes depend on other non-prime attributes in a multi-valued way. It ensures that these dependencies are represented correctly and efficiently. Achieving 4NF can lead to a further reduction in data redundancy and better data organization, especially when dealing with complex, multi-valued relationships.

However, it's worth noting that achieving 4NF can be more complex than achieving earlier normal forms, and it might not always be necessary for every database design. Whether or not to apply 4NF depends on the specific requirements and characteristics of your data and database schema.

Example

Restaurant	PizzaVariety	DeliveryArea
Dominos	Margarita Pizza	New Jersey
Dominos	Margarita Pizza	Downtown
Dominos	Margarita Pizza	Manhattan
Dominos	Peperoni Pizza	New Jersey
Dominos	Peperoni Pizza	Downtown
Dominos	Peperoni Pizza	Manhattan
Pizza Hut	Margarita Pizza	Downtown
Pizza Hut	Margarita Pizza	Manhattan
Pizza Hut	Peperoni Pizza	Downtown
Pizza Hut	Peperoni Pizza	Manhattan
Pizza Hut	Detroit Pizza	Downtown
Pizza Hut	Detroit Pizza	Manhattan
Pizza Hut	Chicago Thin Crust Pizza	Downtown
Pizza Hut	Chicago Thin Crust Pizza	Manhattan

Table 3.8 – Restaurant table

For above relation, all the rules for up to BCNF is satisfied so we can say that this relation is in BCNF.

If we assume Pizza Variety offered by a restaurant is not affected by Delivery Area (Restaurant offers all pizza variety to the all-serviceable area) then It does not meet criteria of 4NF. There are two non-trivial multivalued dependency.

{Restaurant} →→ {Pizza Variety}

{Restaurant} →→ {Delivery Area}

These non-trivial multivalued dependencies on a non-superkey reflect the fact that the varieties of pizza a restaurant offers are independent from the areas to which the restaurant delivers. This leads to redundancy of data in a table and also cause insert anomalies.

For example if we are told that a restaurant will offer one more variety of pizza or will be serving one more delivery location then we need to add multiple rows and if we forgot one of the rows then it will cause data integrity issue.

To eliminate these data redundancy and anomalies, we can decompose table into multiple tables. So that there will be no multivalued dependency where determinant is non superkey.

Restaurant	PizzaVariety
Dominos	Margarita Pizza
Dominos	Peperoni Pizza
Pizza Hut	Margarita Pizza
Pizza Hut	Peperoni Pizza
Pizza Hut	Detroit Pizza
Pizza Hut	Chicago Thin Crust Pizza

Table 3.9 – RestaurantPizza table

Restaurant	DeliveryArea
Dominos	New Jersey
Dominos	Downtown
Dominos	Manhattan
Pizza Hut	Downtown
Pizza Hut	Manhattan

Table 3.10 – RestaurantDeliveryArea table