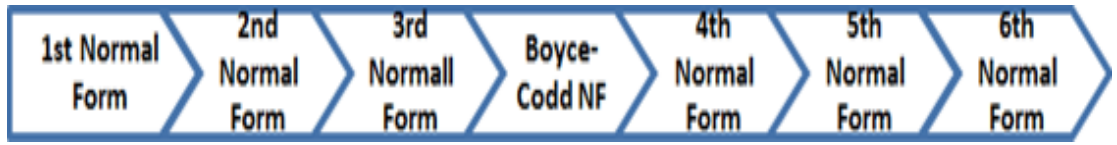
UNIT IV – DATABASE NORMALIZATION
4 HRS

Notes By: Raju Poudel (Mechi Multiple Campus)

Introduction

- ✓ Database normalization, or simply normalization, is the process of restructuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity.
- ✓ **Data redundancy** is the existence of data that is additional to the actual data.
- ✓ **Data integrity** is the maintenance of, and the assurance of the accuracy and consistency of, data over its entire life-cycle.
- ✓ It divides larger tables to smaller tables and links them using relationships.



Advantages of Normalization

- A smaller database can be maintained as normalization eliminates the duplicate data. Overall size of the database is reduced as a result.
- As databases become lesser in size, the passes through the data becomes faster and shorter thereby improving response time and speed.
- Avoid redundant fields or columns.
- More flexible data structure i.e. we should be able to ad new rows and data values easily
- Better understanding of data.
- Easier to maintain data structure i.e. it is easy to perform operations and complex queries can be easily handled.

Disadvantages of Normalization

- Database systems are complex, difficult, and time-consuming to design.
- Substantial hardware and software start-up costs.
- Initial training required for all programmers and users.
- On Normalizing the relations to higher normal forms i.e. 4NF, 5NF the performance degrades.
- It is very time consuming and difficult process in normalizing relations of higher degree.
- Careless decomposition may lead to bad design of database which may leads to serious problems.

Functional Dependencies

- Functional dependency in DBMS, as the name suggests is a **relationship between attributes** of a table dependent on each other.
- Introduced by E. F. Codd, it helps in preventing data redundancy and gets to know about bad designs.
- For Example, consider the following table,

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

- In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc.
- By this, we can say that the **city, Employee Name, and salary** are functionally depended on **Employee number**.

Functional Dependencies

- A functional dependency is denoted by an arrow →
- The functional dependency of X on Y is represented by **X →Y**
- If column A of a table uniquely identifies the column B of same table then it can represented as **A->B** (Attribute B is functionally dependent on attribute A)

Types of Functional Dependencies

- Multivalued dependency
- Trivial functional dependency
- Non-trivial functional dependency
- Transitive dependency

Multivalued Dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.
- For example,

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

BIKE_MODEL → → MANUF_YEAR

BIKE_MODEL → → COLOR

- Here columns **COLOR** and **MANUF_YEAR** are dependent on **BIKE_MODEL** and independent of each other.
- In this case, these two columns can be called as **multivalued dependent** on **BIKE_MODEL.**

Trivial Functional Dependency

- The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.
- So, X -> Y is a trivial functional dependency if Y is a subset of X.
- A trivial functional dependency is a database dependency that occurs when describing a functional dependency of an attribute or of a collection of attributes that includes the original attribute.
- Consider this table with two columns emp_id and emp_name.

emp_id	emp_name
AS555	Harry
AS222	George
AS999	Kevin

- {emp_id, emp_name} -> emp_name [emp_name is a subset of {emp_id, emp_name}]

Non-Trivial Functional Dependency

- If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.

- For example: An employee table with three attributes: emp_id, emp_name, emp_address.

- The following functional dependencies are non-trivial:

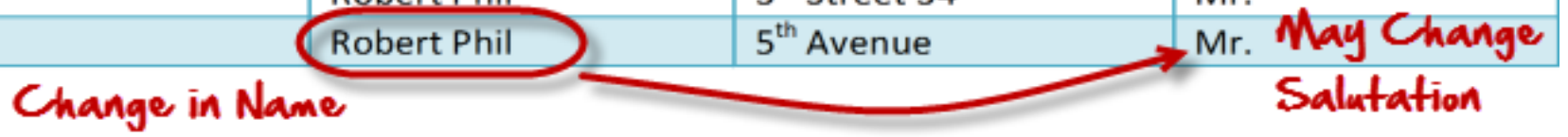
  emp_id -> emp_name (emp_name is not a subset of emp_id)

  emp_id -> emp_address (emp_address is not a subset of emp_id)

Transitive Dependency

- If non-primary key attributes depends upon other non-primary key attributes than there occurs transitive dependency.
- A transitive is a type of functional dependency which happens when t is indirectly formed by two functional dependencies.
- A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change.
- Consider the table, Changing the non-key column **Full Name** may change **Salutation**.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.



Transitive Dependency

- For example,

Company	CEO	Age
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

- {Company} -> {CEO} (if we know the company, we know its CEO's name)
- {CEO } -> {Age} If we know the CEO, we know the Age

- Therefore according to the rule of rule of transitive dependency:
- { Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.
- **Note:** You need to remember that transitive dependency can only occur in a relation of three or more attributes.

Database Anomalies

- Database anomalies are the problems in relations that occur due to redundancy in the relations.
- They can occur in poorly planned, un-normalised databases where all the data is stored in one table.
- These anomalies affect the process of inserting, deleting and modifying data in the relations.
- Some important data may be lost if a relation is updated that contains database anomalies.
- It is important to remove these anomalies in order to perform different processing on the relations without any problem.

Types of Anomalies

- Insertion Anomalies
- Deletion Anomalies
- Modification Anomalies

Insertion Anomaly

- An Insert Anomaly occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

sid	sname	cid	cname
S10	Ram	C03	Java
S11	Shyam	C03	Java
S12	Hari	C04	C
S13	Sita	C05	C++

- For example, we can't add a new course unless we have at least one student enrolled on the course.

- If we want to add a new course then student details will become null. So, course can’t be inserted without having student details. This scenario forms insertion anomaly.

Deletion Anomaly

- A Delete Anomaly exists when certain attributes are lost because of the deletion of other attributes.

sid	sname	cid	cname
S10	Ram	C03	Java
S11	Shyam	C03	Java
S12	Hari	C04	C
S13	Sita	C05	C++

- For example, consider what happens if Student S13 is the last student to leave the course - All information about the course is lost.

Modification Anomaly

- The modification anomaly occurs when the record is updated in the relation. In this anomaly, the modification in the value of specific attribute requires modification in all records in which that value occurs.

sid	sname	cid	cname
S10	Ram	C03	Java
S11	Shyam	C03	Java
S12	Hari	C04	C
S13	Sita	C05	C++

- For example, if we update cid of student then we need to update cname of student too.

- So, normalization process is required to eliminate anomalies from database.

First Normal Form (1NF)

- For a table to be in the First Normal Form, it should follow the following rules:
  a) It should only have single(atomic) valued attributes/columns.
  b) All the columns in a table should have unique names.
- Consider the following table **Student**,

sid	sname	salutation	address	phone	sub_id	sub_name
1	Ram	Mr	Btm	9864912123, 9854964126	1	Database
2	Shyam	Mr	Ktm	9824912345	1	Database
3	Gita	Mrs	Btm	9824923456	2	Java
4	Sita	Ms	Ktm	9825612723	3	C
2	Shyam	Mr	Ktm	9824912345	2	Java

- Above table does not satisfy 1NF because column phone contains multiple values. Hence, we need to create new table **contact** to store phone numbers.

First Normal Form (1NF)

- Following are the normalized tables that satisfy 1NF,

Student

sid	sname	salutation	address	sub_id	sub_name
1	Ram	Mr	Btm	1	Database
2	Shyam	Mr	Ktm	1	Database
3	Gita	Mrs	Btm	2	Java
4	Sita	Ms	Ktm	3	C
2	Shyam	Mr	Ktm	2	Java

Contact

contact_id	phone	sid
1	9864912123	1
2	9854964126	1
3	9824912345	2
4	9824923456	3
5	9825612723	4

Second Normal Form (2NF)

- For a table to be in the Second Normal Form, it must satisfy two conditions:
  a) The table should be in the First Normal Form.
  b) There should be no Partial Dependency
- **Partial Functional Dependency** occurs only in relation with composite keys.
- **Partial functional dependency** occurs when one or more non key attribute are depending on a part of the primary key.

Example:

Table: Stud_id, Course_id, Stud_name, Course_Name

Where: Primary Key = Stud_id + Course_id

Then: To determine name of student we use only Stud_id, which is part of primary key.

{Stud_id} -> {Stud_Name}

Hence,Stud_name is partially dependent on Stud_id. This is called **partial dependency.**

Second Normal Form (2NF)

Student

sid	sname	salutation	address	sub_id	sub_name
1	Ram	Mr	Btm	1	Database
2	Shyam	Mr	Ktm	1	Database
3	Gita	Mrs	Btm	2	Java
4	Sita	Ms	Ktm	3	C
2	Shyam	Mr	Ktm	2	Java

- Our **Student** table does not satisfy 2NF, because we have,

  Primary Key: sid + sub_id
- Here, to determine name of subject, we use sub_id which is a part of primary key. Hence, sub_name is partially dependent on sub_id.
- So, we need to remove partial dependency from Student table and create new table subject to store subject details.

Second Normal Form (2NF)

Student

sid	sname	salutation	address	sub_id
1	Ram	Mr	Btm	1
2	Shyam	Mr	Ktm	1
3	Gita	Mrs	Btm	2
4	Sita	Ms	Ktm	3
2	Shyam	Mr	Ktm	2

Subject

sub_id	sub_name
1	Database
2	Java
3	C

Third Normal Form (3NF)

- The official qualifications for 3NF are:
  a) A table is already in 2NF.
  b) Non primary key attributes do not depend on other non primary key attributes (i.e. no transitive dependencies)
- All transitive dependencies are removed to place in another table.
- Consider the following student table,

sid	sname	salutation	address	sub_id
1	Ram	Mr	Btm	1
2	Shyam	Mr	Ktm	1
3	Gita	Mrs	Btm	2
4	Sita	Ms	Ktm	3
2	Shyam	Mr	Ktm	2

- In above table, there is transitive dependency on sname and salutation. Both are non-primary key attributes. Change in sname might cause change in salutiation. For eg, if we change name **Ram to Maya** then we need to change salutiation too.

Third Normal Form (3NF)

Original table

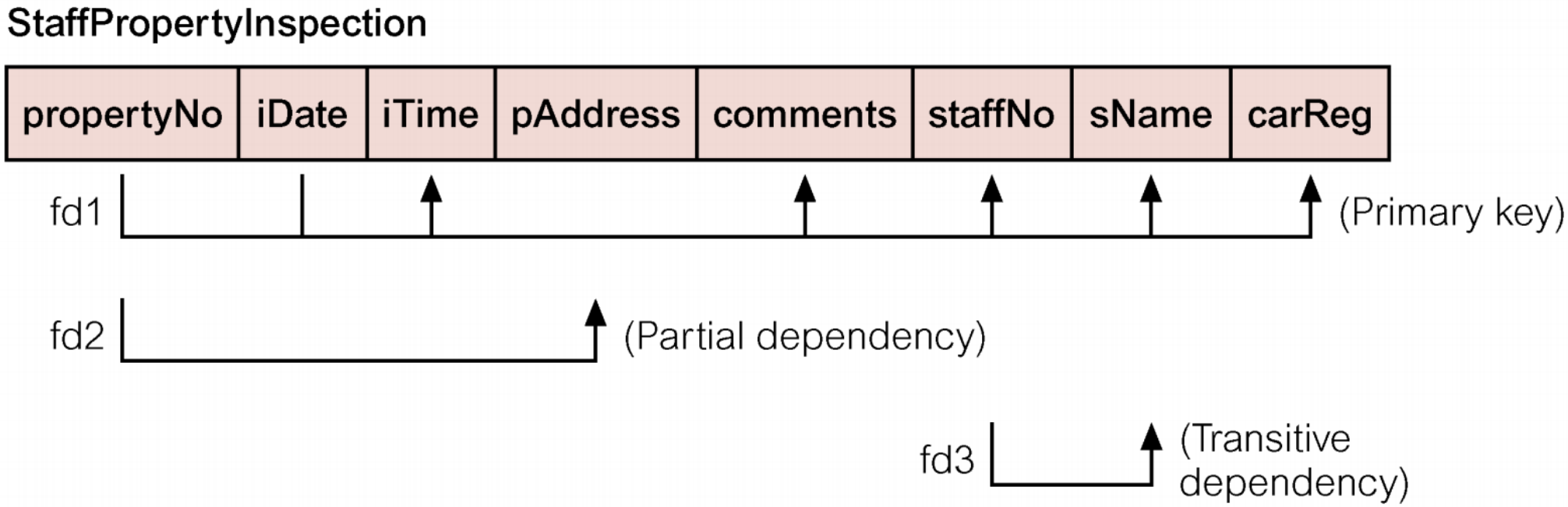
sid	sname	salutation	address	sub_id
1	Ram	Mr	Btm	1
2	Shyam	Mr	Ktm	1
3	Gita	Mrs	Btm	2
4	Sita	Ms	Ktm	3
2	Shyam	Mr	Ktm	2

Normalized tables.

sid	sname	salutation_id	address	sub_id
1	Ram	1	Btm	1
2	Shyam	1	Ktm	1
3	Gita	3	Btm	2
4	Sita	2	Ktm	3
2	Shyam	1	Ktm	2

salutation_id	salutation
1	Mr
2	Ms
3	Mrs

Functional Dependencies



Fourth Normal Form (4NF)

- A table is in the 4NF if it is in 3NF and has no **multivalued dependencies.**
- A **multivalued dependency** exists when there are at least 3 attributes (like X,Y and Z) in a relation and for value of X there is a well defined set of values of Y and a well defined set of values of Z. However, the set of values of Y is independent of set Z and vice versa.
- Suppose a student can have more than one subject and more than one activity.

Student_Info		
Student_Id	**Subject**	**Activity**
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

- Note that all three attributes make up the Primary Key.
- Note that Student_Id can be associated with many subject as well as many activities. This scenario is **multi valued dependency**.
- Databases with multivalued dependencies thus exhibit **redundancy**.

Fourth Normal Form (4NF)

Original Table:

Student_Info		
Student_Id	**Subject**	**Activity**
100	Music	Swimming
100	Accounting	Swimming
100	Music	Tennis
100	Accounting	Tennis
150	Math	Jogging

Here are the tables Normalized-

Student_Id	Subject
100	Music
100	Accounting
150	Math

StudentId	Activity
100	Swimming
100	Tennis
150	Jogging

Fourth Normal Form (4NF)

Original Table:

University courses

Course	Book	Lecturer
AHA	Silberschatz	John D
AHA	Nederpelt	John D
AHA	Silberschatz	William M
AHA	Nederpelt	William M
AHA	Silberschatz	Christian G
AHA	Nederpelt	Christian G
OSO	Silberschatz	John D
OSO	Silberschatz	William M

Here are the tables Normalized-

Course	Book
AHA	Silberschatz
AHA	Nederpelt

Course	Lecturer
AHA	John D
AHA	William M
AHA	Christian G
OSO	John D
OSO	William M

Boyce-Codd Normal Form (BCNF)

- **Boyce-Codd Normal Form (BCNF)** is one of the forms of database normalization. A database table is in BCNF if and only if there are **no non-trivial functional dependencies of attributes** on anything other than a superset of a candidate key.
- **BCNF** is also sometimes referred to as 3.5NF, or 3.5 Normal Form.

- For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:

  a) It should be in the Third Normal Form.

  b) And, for any dependency A → B, A should be a super key.

- The second point sounds a bit tricky, right? In simple words, it means, that for a **dependency A → B, A cannot be a non-prime attribute, if B is a prime attribute**.

Boyce-Codd Normal Form (BCNF)

- Below we have a college enrolment table with columns **student_id, subject and professor**.

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- In the above table student_id, subject together form the primary key, because using student_id and subject, we can find all the columns of the table.
- Also, there is a dependency between subject and professor, where subject depends on the professor name.

Boyce-Codd Normal Form (BCNF)

- This table satisfies the **1st Normal form** because all the values are atomic, column names are unique and all the values stored in a particular column are of same domain.
- This table also satisfies the **2nd Normal Form** as their is no **Partial Dependency**.
- And, there is no **Transitive Dependency**, hence the table also satisfies the **3rd Normal Form**.
- But this table is not in **Boyce-Codd Normal Form**.

Why this table doesn't satisfy BCNF?

- In the table above, student_id, subject form primary key, which means subject column is a prime attribute.
- But, there is one more dependency, professor → subject.
- And while subject is a prime attribute, professor is a non-prime attribute, which is not allowed by BCNF.

Boyce-Codd Normal Form (BCNF)

- To make this relation(table) satisfy BCNF, we will decompose this table into two tables, **student table** and **professor table**.
- Below we have the structure for both the tables.

Student Table

student_id	p_id
101	1
101	2
and so on...	

Professor Table

p_id	professor	subject
1	P.Java	Java
2	P.Cpp	C++
and so on...		

Original Table

student_id	subject	professor
101	Java	P.Java
101	C++	P.Cpp
102	Java	P.Java2
103	C#	P.Chash
104	Java	P.Java

- And now, this relation satisfy **Boyce-Codd Normal Form**.

Assignment

1. **Convert the following Student table to 2NF.**

sid	name	course_id	course	phone_no
1	Rohit	101	BCA	9833888831
2	Ravi	102	BBA	3555939392
3	Shyam	103	MCA	4994994993
4	Hari	104	MBA	4884484844
5	Ram	105	BBS	4908590495

2. **Convert the following Staff Table to 3NF.**

staff_id	name	address	department	department _contact
1	Rohit	Btm	Accounting	9866666666
2	Ravi	Btm	HR	9877777777
3	Shyam	Ktm	Accounting	9866666666
4	Hari	Brt	HR	9877777777
5	Ram	Ktm	Marketing	9833333333

3. **Normalize the following Employee table upto 3NF.**

emp_id	name	address	contact	branch_code	branch
1	Rohit	Btm	9866666,985555	101	Dhulabari
2	Ravi	Btm	983333333	102	Surunga
3	Shyam	Ktm	98777777	102	Surunga
4	Hari	Brt	98334444,9874774	101	Dhulabari
5	Ram	Ktm	98123444	103	Bhadrapur