CHAPTER 11

Module V

Relational Database Design

Syllabus

Design guidelines for relational Schema, Functional Dependencies, Database tables and normalization, The need for normalization, The normalization process, Improving the design, Definition of Normal Forms- 1NF, 2NF, 3NF & The Boyce-Codd Normal Form (BCNF).

11.1 Design Guidelines for Relational Schema

Q. Describe various design guidelines for relational schema.

1. Introduction

- In previous chapters, we have already studied various aspects of the relational model and the languages associated with it. Each relation schema consists of a number of attributes, the relational database schema consists of a number of relation schemas.

Relational Schema (Table Structure)	Relational Database Schema (Database Structure)
$R_1 (A_1, A_2,.., A_n)$ $R_2 (B_1, B_2,.., B_n)$ ... $R_m (C_1, C_2,.., C_n)$	$R_1, R_2 ,..., R_n$

Where,

$R_1, R_2,... R_n$ = Relation or Tables

A, B, C = Attributes or Columns

- The attributes are grouped to form a relation schema by mapping a conceptual data model i.e. ER or EER data model to relational schema.
- The relational mapping will identify entity types and relationship types and their respective attributes.

2. Goodness of Relational Design

- In process of database design, we need to develop some measure of goodness for the quality of the design. In this module, we will discuss some measure of goodness to evaluate design quality of relational schemas.
- The above measure will help the developer to analyze why one grouping of attributes is better than another grouping of attributes.
- There are two levels for measuring goodness of relation schemas.
- Logical (or Conceptual) Level
  - The goodness of relational schema depends on how users understand the meaning of various attributes of relation.
  - This will helps users to understand the meaning of the data stored in the relations, and hence it will make easy to formulate relational queries correctly.
- Implementation (or Physical Storage) Level
- It helps user to understand how the tuples in a base relation are stored and updated.
  - This level is applied to base relational schema stored as files.

3. Database Design Approach

- The database design can be done using the bottom-up approach to design relational schema using individual attributes of relation or top-down approach to identify individual attributes from relational schema.
- Bottom-up design methodology / Design by Synthesis
  - The basic relationships among individual attributes are identified then it will be combined to construct relation schemas.
  - This is not very popular approach as it needs to study many binary relationships among attributes at the beginning which is very difficult.
- Top-down design methodology / Design by Analysis
  - The starts with relational schema and it will be further decomposed to a group of attributes to achieve desirable properties.

- This is very practical approach and used in real world database projects.

4. Guidelines for Relational Schema

- To determine the quality of relation schema design some informal guidelines can be used.
- **Guideline 1 :** Clear semantics of the attributes in relational schema
  - Semantics of attribute should be very clear in relational schema so that relational schema will have some real-world meaning associated with it.
  - The relational schema has a clear meaning associated with it.
  - Example,
  - Employee (Emp_Id, Ename, Address, Salary)
  - The Employee table contains information about all employees in company with their address and Salary.
- **Guideline 2 :** Reducing the Redundant Data in Tuples
  - A relational schema may have some redundancy in database design, if it stores data redundantly.
  - If same data is stored at more then one locations will leads to redundancy and wastage of memory space.
  - **Data Anomalies** : An inconsistent data may cause some problems while adding, updating or deleting data in table which is called as data anomalies.
  - Redundant data is more vulnerable to various data anomalies as if data is updated at only one location and not at other locations, then that data becomes inconsistent, and this problem referred to as an update anomaly.
  - A normalized database stores non-primary key data in only one location.
  - A relational database table should avoid all data anomalies.

Example,

Employee (Emp_Id, Ename, Address)

Emp_Salary (Emp_Id, Ename, payScale, grossSalary, netSalary)

Emp_Designation (Emp_Id, Ename, Desg, fromDate, toDate)

a. Update Anomaly

- The relational schema may have same data stored in multiple relations, if we update such data from only one relation may result in logical inconsistencies.
- In above example,
- All 3 tables contain the Ename attribute, thus any change in name of one employee will lead to updating his name in all 3 tables. Otherwise, if all the records are not updated then some tables may leave in an inconsistent state.

b. Insertion Anomaly

- There is a possibility in which certain facts cannot be recorded in database.
- An Insert Anomaly arises when certain attributes cannot be added into the database without the presence of other attributes.
- In above example,
- It is not possible to add a row in Emp_Salary table or Emp_Designation table for an employee who is not exists in employee table.

c. Deletion Anomaly

- If data deleted from one table all relevant data in another related tables must also be deleted otherwise it will create data inconsistency problem.
- Deletion of some data from one relation necessitates the deletion of some other data in other table.
- In above example,
- It is not possible to delete a row in Employee table if Emp_Salary table or Emp_Designation table contains data for respective employee.
- **Guideline 3 :** Reducing Null values in Tuples
  - A value of NULL is different from an empty, White spaces (blank spaces) or zero value.
  - Null values in tuple will cause wastage of memory space and it will also create problem of understanding.
  - Relations should be designed in such a way that their tuples should not contain any NULL values.
  - We can at least try make number of NULL values as low as possible.
  - Attributes with NULL values can be placed in separate relations with the primary key.
  - There are certain resons for Null values,
  - Not applicable data
  - Invalid data
  - Unknown data or data not available

- **Guideline 4 :** Dissallowing Spurious Tuples
  - The bad designs of a relational database may result in erroneous results for some JOIN operation. As it is not possible to get original relation data from new relation after JOIN operation.
  - The relational schema must be designed to satisfy the property of lossless join.
  - If original relation contains fewer number of tuple then tuples generated by doing a natural-join of original relations.

11.2 Functional Dependencies

Q. Attempt the following : Functional Dependencies **MU - Dec. 18. 5 Marks**

Q. Explain concept of functional dependency.

1. Introduction

- The concept of functional dependency is given by E. F. Codd.
- Functional Dependency (FD) provides a constraint between various attributes of a relation.
- Functional dependencies are restrictions imposed between two set of attributes in relation from a database.
- In a Relation R with attributes X and Y represented as R(X, Y), where Y is functionally dependent on other column X or we can say X functionally determines Y.
- This dependency can be denoted with help of arrow (→)

$$\mathbf{X \rightarrow Y}$$

- The data value in column Y must change when data value in another column X is modified.
- All the attributes before arrow is called as **determinant** and attributes after arrow is called as determine.
- Functional Dependency (FD) provides a formal mechanism to express constraints between various attributes of a relation.
- Functional dependency determines the set of values or the attribute based on another attribute.
- Functional data dependencies are restrictions imposed on the data in database.

2. Example

- Consider an employee table with columns as shown in Table 11.2.1

Table 11.2.1 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

Case 1 : (X→Y)

- Consider an Employee table for specific employee_Id there is one and only one Ename whereas for another employee_Id there can be other Ename.

$$Employee\_Id \rightarrow Ename$$

- As per above constraint, it is possible to have multiple employees with same Ename and different Employee_Id. But it is not allowed to have two employees with same Employee_Id and different Ename.

Case 2 : (X→YZ)

- In above Employee table using below given functional dependency, for specific employee_Id there is one and only one set of Ename and Salary whereas for another employee_Id there can be other values of Ename and salary.

$$Employee\_Id \rightarrow Ename, Salary$$

- As per above constraint, it is possible to have multiple employees with same Ename and Salary.

Case 3 : (XY→ZW)

- In above Employee table using below given functional dependency, for one employee_Id and Project_Id pair there is only one amount of time spent (Hours) and allowance given by company whereas for another pair there can be other values of Hours and Allowance.

$$Employee\_Id, Project\_Id \rightarrow Hours, Allowance$$

- As per above constraint, it is possible to have multiple employee_Id and Project_Id pairs with same values of Hours and Allowance.

11.3 Solved Examples on Functional Dependencies

Example 11.3.1 : List all functional dependencies satisfied by the relation. **MU - Dec. 13, May 15, 5 Marks**

A	B	c
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

Solution :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

A	B	C	Tuple
a_1	b_1	c_1	Tuple 1
a_1	b_1	c_2	Tuple 2
a_2	b_1	c_1	Tuple 3
a_2	b_1	c_3	Tuple 4

Step 1 : Test for type 1 FD (X→Y)

Functional Dependency	Violated First by Tuple	Result
A → B	--	Valid FD
A → C	Tuple 2 $a_1 \rightarrow c_1$ $a_1 \rightarrow c_2$	Not Valid FD
B → A	Tuple 3	Not Valid FD
B → C	Tuple 2	Not Valid FD
C → A	Tuple 3	Not Valid FD
C → B	--	Valid FD

Step 2 : Test for type 1 FD (X→YZ)

Functional Dependency	Violated First by Tuple	Result
A → BC	Tuple 2 $a_1 \rightarrow b_1c_1$ $a_1 \rightarrow b_1c_2$	Not Valid FD
B → AC	Tuple 2	Not Valid FD
C → AB	Tuple 3	Not Valid FD

Step 3 : Test for type 1 FD (XY→Y)

Functional Dependency	Violated First by Tuple	Result
AB → C	Tuple 2 $a_1b_1 \rightarrow c_1$ $a_1b_1 \rightarrow c_2$	Not Valid FD
AC → B	--	Valid FD
BC → A	Tuple 3	Not Valid FD

Therefore, all FDs satisfied by relation are,

$$A \rightarrow B;\ C \rightarrow B;\ AC \rightarrow B$$

Example.11.3.2 : Consider the following relation.

A	B	C	Tuple #
10	b_1	c_1	#1
10	b_2	c_2	#2
11	b_4	c_1	#3
12	b_3	c_4	#4
13	b_1	c_1	#5
14	b_3	c_4	#6

Given the previous state which of the following dependencies may hold in the above relation ? If the dependency cannot hold explain why by specifying the tuples that cause the violation :

(i) A →B (ii) B →C (iii) C → B (iv) B → A (v) C → A **MU - May 14, 10 Marks**

Solution :

To find out all functional dependencies satisfied by the relation, we must remember one determinant (attributes before arrow) will give one and only one value of determine (attribute after arrow).

1. A →B

As in above relation to test A → B, {10} → {b1} but in Tuple # 2 {10} → {b2}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #2.

2. B → C

As in above relation to test B → C, {b1}→ {c1}, {b2}→{c2}, {b3}→{c4} for all tuples.

Therefore, Dependency holds for all tuples in above table.

3. C → B

As in above relation to test A → B, {c1} → {b1} but in Tuple #3 {c1} → {b4}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3.

4. **B → A**

As in above relation to test A → B, {b1} → {10} but in Tuple #5 {b1} → {13}

Also {b3} → {13} but in Tuple #6 {b3} → {14}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #5 and Tuple #6.

5. **C → A**

As in above relation to test C → A, In Tuple #1 {C1} → {10,11,13}

But in Tuple #3 {c1} → {11} and in Tuple #5 {c1} → {13}

This violates dependency.

Therefore, Dependency cannot hold in above table values due to Tuple #3 and Tuple #5.

11.4 Types of Functional Dependencies

Q. What are types of functional dependencies ?

1. Full functional dependency

- A functional dependency is a full functional dependency if removal of any attributes from determinant will invalidate the dependency.
- A functional dependency A → B is a full functional dependency if removal of any attributes from A means that the dependency does not hold any more.

Example

- Consider an employee table with columns as shown in Table 11.4.1.

Table 11.4.1 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

- In the above example, Hours and allowance are fully functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Hours, Allowance

- The number of hours spent on the project by a particular employee can not be determined with the project number (Project_no) alone. It needs the employee number (Emp_no) as well.

2. Partial functional dependency

- A partial dependency means that a non key column is depend on some columns in composite primary key of a table.
- An FD A → B is a partial dependency if there is some attribute X ∈ A (X subset of A), that can be removed from A and the dependency will still hold.
- If determine (attributes after arrow) attributes depends on **part of** (partial) determinant attributes. Such dependency is called as partial functional dependency.

Example

- Consider an employee table with columns as shown in Table 11.4.2.

Table 11.4.2 : Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

- In the above example, attribute salary is considered to be functionally dependent on both Employee_Id and Project_Id.

Employee_Id, Project_Id → Salary

- But, Attribute salary functionally dependent on Employee_Id is also holds true.

Employee_Id → Salary

- So, Salary is partial functionally dependent on attribute pair Employee_Id and Project_Id.

3. Transitive dependency

- This concept is used when there is redundancy in database.
- If changing any non key column (column other then key column) causes change in other non key column in such situations you may have transitive dependency.

- If one attribute of relation is functionally dependent on other dependent attribute then such a dependency is called as **transitive dependency**.
- An FD X →Y in a relation R is a transitive dependency, if there is a set of attributes Z that is not a subset of any key of R, and both X → Z and Z → Y holds true.

Example

- Consider an employee table schema,
- Employee_Id → Department_Id
- Department_Id → Dname

Table 11.4.3 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

- Dependency of Department_Id on key attribute Dname is transitive functional dependency as Dname is depends on Department_Id which is depend on Employee_Id. So, Dname is transitive functionally dependent on Employee_Id.

4. Trivial functional dependency

- Functional dependency (FD) X → Y and Y is a subset of X, then it is called as a trivial FD.
- Functional dependency X → Y and Y is not a subset of X, then it is called as a non-trivial functional dependency.

Example

- For employee table the below given FD is trivial as Ename is a subset of {Employee_Id, Ename}
- **Employee_Id, Ename → Ename**
- And below given FD is non-trivial as Hours is not a subset of {Employee_Id, Project_Id}

Employee_Id, Project_Id → Hours

5. Multivalued dependency

- Multivalued dependency defined by X →→ Y is said to hold for a relation R (X, Y, Z) if for a given set of values of X, there is a set of associated values of attribute Y, and X values depend only on X values and have no dependence on the set of attributes Z.
- **Multivalued dependency** in turn, is defined as relationship which accepts the cross product pattern.

Example

- For employees car table as given below,

Table 11.4.4 : Employee_Car Table

Employee_Id	Ename	Car
10	Mahesh	Ertiga
12	Suresh	Zen
15	Ganesh	Sentro
10	Mahesh	Wagon R

- The FDs are given as below,

Employee_Id →→ Car

- α →→ β says relationship between α and β independent of relationship between α and R-β. That means **Employee_Id →→ Car** relationship is independent of Employee_Id and EName relation. i.e. **Employee_Id →→ Car** is independent of Employee_Id →→ Ename

11.5 FD Properties (Armstrong's Axioms / Closures of FD)

Q. Give Armstrong axioms.

Q. List the Armstrong's axioms for functional dependencies. What do you understand by soundness and completeness of these axioms ?

- Given that Relation R(X, Y, Z, W); represents a table R with set of indivisible attributes X, Y, Z and W.
- It is possible to derive many properties of functional dependencies.
- Axioms are nothing but rules of inference which provides a simple technique for reasoning about functional dependencies.

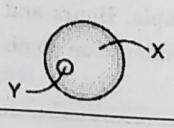
1. Primary Properties

a. Subset property (Axiom of Reflexivity)

For given relation R(X, Y, Z, W),

If Y is a subset of X as shown in diagram,

Then **X → Y**



(Which can be referred as X is functionally dependent on Y)

b. Append Property (Axiom of Augmentation)

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$

Then $XZ \rightarrow YZ$

It is possible to append attribute Z to both sides of FD provided that it is part of same table.

c. Transitivity (axiom of transitivity)

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $Y \rightarrow Z$

Then $X \rightarrow Z$

It is possible to use transitivity if attribute X, Y and Z are part of the same table.

2. Secondary Properties

a. Union

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $X \rightarrow Z$

Then $\mathbf{X \rightarrow YZ}$

b. Decomposition

For given relation R(X, Y, Z, W),

If $X \rightarrow YZ$

Then $X \rightarrow Y$ and $X \rightarrow Z$

c. Pseudo Transitivity

For given relation R(X, Y, Z, W),

If $X \rightarrow Y$ and $YZ \rightarrow W$

Then $XZ \rightarrow W$

3. Canonical cover

A functional depending set X is canonical or minimal if the set has following properties

1. Each right set of a functional dependency of X contains only one attribute.
2. Each left set of a functional dependency of X is minimal. It means there should not be any extraneous attribute is present in dependency.
3. Reducing any functional dependency will change the content of X.

Note : Extraneous attributes are one which can be removed without changing the closure set of functional dependency.

Example 11.5.1 : Consider relation R = (A, B, C, D, E, F) having set of FD's

$A \rightarrow B$ $A \rightarrow C$

$BC \rightarrow D$ $B \rightarrow E$

$BC \rightarrow F$ $AC \rightarrow F$

Calculate some members of Axioms as be below:

1. $A \rightarrow E$ 2. $BC \rightarrow DF$
3. $AC \rightarrow D$ 4. $AC \rightarrow DF$

Solution :

1. $A \rightarrow E$

As $A \rightarrow B$ and $B \rightarrow E$

So using Transitive rule,

$\therefore \mathbf{A \rightarrow E}$

2. $BC \rightarrow DF$

As $BC \rightarrow D$ and $BC \rightarrow$

So using union rule,

$\therefore \mathbf{BC \rightarrow DF}$

3. $AC \rightarrow D$

As $A \rightarrow B$ and $BC \rightarrow$

So using pseudo transitivity,

$\therefore \mathbf{AC \rightarrow D}$

4. $AC \rightarrow DF$

As $AC \rightarrow D$ and AC

So using union rule,

$\therefore \mathbf{AC \rightarrow DF}$

Example 11.5.2 : Consider relation R = (A, B, C, D, E, F) having set of FD's

$A \rightarrow B$ $A \rightarrow C$

$C \rightarrow D$ $B \rightarrow E$

$AC \rightarrow F$

Calculate some closures as {A}+, {B}+, {AC}+ and also find key of above relation.

Solution :

1. $A \rightarrow B, A \rightarrow C$ $\{A\}^+ = \{A, B, C\}$

   So using union rule,

   $A \rightarrow BC$

2. $B \rightarrow E$ $\{B\}^+ = \{B, E\}$

3. $\{A\}^+ = \{A, B, C\}$ and $C \rightarrow D, B \rightarrow E$ and $AC \rightarrow F$ $\{AC\}^+ = \{A, B, C, D, E, F\}$

4. $\{AC\}^+ = \{A, B, C, D, E, F\}$ {AC} can determine all attributes in relation R

   So, {AC} is a key of Relation R

Example 11.5.3 : Consider relation R = (A,B,C,D,E,F) having set of FD's

$A \rightarrow B$ $A \rightarrow C$

$BC \rightarrow D$ $B \rightarrow E$

$BC \rightarrow F$ $AC \rightarrow F$

Calculate some members of Axioms as be below :

1. $A \rightarrow E$ 2. $BC \rightarrow DF$
3. $AC \rightarrow D$ 4. $AC \rightarrow DF$

Soln. :

1. $A \rightarrow E$

As $A \rightarrow B$ and $B \rightarrow E$

So using Transitive rule,

$\therefore$ **$A \rightarrow E$**

2. $BC \rightarrow DF$

As $BC \rightarrow D$ ...(i)

$BC \rightarrow F$ ...(ii)

$\therefore$ Using union rules (i) and (ii)

$\therefore$ **$BC \rightarrow DF$**

3. $AC \rightarrow D$

$A \rightarrow B$ ...(iii)

As $BC \rightarrow D$ ...(iv)

$\therefore$ Using pseudo transitivity

$\therefore$ **$AC \rightarrow D$** ...(v)

4. $AC \rightarrow DF$

Using above rule (v)

$AC \rightarrow D$ ...(vi)

$AC \rightarrow F$ ...(vii)

$\therefore$ **$AC \rightarrow DF$**

11.6 Decomposition

Q. What is decomposition ? Explain various rules of normalization.

1. Introduction

a. If a relation is not in the normal form and we wish the relation to be normalised so that some of the anomalies can be eliminated, it is necessary to decompose the relation in two or more relations.

b. This is a process of dividing one table into multiple tables can be done using projection operator.

c. Decomposed table can be reconstructed using join operation.

The process of decomposition of a relation R into a set of relations R_1, R_2....R_n was based on identifying attributes and using that as a basis of decomposition.

$$R = R_1 \cup R_2 \cup \ldots \cup R_n$$

This is a process of dividing one table into multiple tables using projection operator.

2. Desirable Properties or Goals

Q. What are goals of decomposition ?

Q. What are goals of normalization ?

a. Lossless join decomposition

- The original relation and relation reconstructed from joining decomposed relations must contain same number of tuples if number is increased or decreased then it is Losssy Join decomposition.
- Lossless join decomposition ensures that we can never get the situation where spurious tuple are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations.
- Rules for lossless decompositions are,
  a. The relations to be decomposed must have at least one common attribute in pair of relations.
     i.e. $R_1 \cap R_2 \neq \Phi$
  b. The attributes in common must be a key for one of the relation for decomposition to be lossless.
- It is clear that decomposition must be lossless so that we do not lose any information from the relation that is decomposed.
- Lossless join decomposition ensures that we can never get the situation where spurious tuple are generated in relation, for every value on the join attributes there will be a unique tuple in one of the relations. For above join to become lossless we need to go for following steps.

Steps

1. Let R_1 and R_2 form decomposition of relation R as R_1 and R_2 are both sets of attributes from R.
2. Decompose the relation schema Department_Student into
   Department-schema = (Dept_Id, Dname)
   Student-schema = (Stud_id, Sname, Location)
3. The attributes in common must be a key for one of the relation for decomposition to be lossless. $R_1 \cap R_2 \neq \Phi$ There must not be null value.

Note : You are joining a primary key and a foreign key of table.

Example,

- Employee (Employee_Id, Ename, Salary, Department_Id, Dname)
- Can be decomposed using lossless decomposition as,
- Employee_desc (Employee_Id, Ename, Salary, Department_Id)
- Department_desc (Department_Id, Dname)
- Alternatively the lossy decomposition would be as joining these tables is not possible so not possible to get back original data.
- Employee_desc (Employee_Id, Ename, Salary)
- Department_desc (Department_Id, Dname)

Example

Above relation can be lossless decomposed as follows,

Schema 1 : Department schema contains (Dept_Id, Dname)

$\therefore$ $R_1 \leftarrow \Pi_{Dept\_Id,Dname}$ (Department_student)

Dept_Id	Dname
10	Development
20	Teaching
30	HR

Schema 2 : Student schema contains (Stud_Id, Dept_Id, Sname, Location)

$\therefore$ $R_2 \leftarrow \Pi_{Stud\_Id, Dept\_Id, Location,Sname}$ (Department_student)

Stud_Id	Dept_Id	Location	Sname
1	10	Mahim	Sushant
2	20	Vashi	Snehal
3	30	Worli	Pratiksha
4	20	Dadar	Supraja

This is lossless-join decomposition as $R_1 \cap R_2 \neq \Phi$ common column is Dept_Id

b. Dependency preservation

- All functional dependencies result in just one relation.
- Dependency preservation is another important requirement since a dependency is a very important constraint on the database.
- As a result of any database updates, the database should not result in illegal relation being created. Hence, our design should allow us to check updates without natural joins.
- If $X \rightarrow Y$ holds then we know that the two (sets) attributes are closely related or functionally dependent and it would be useful if both attributes in the same relation so that the dependency can be checked easily.
- This can be done by maintaining functional dependency.

Example,

- Consider relation R(X, Y, Z, W) that has the following dependencies F,

  $X \rightarrow Y$

  $Y \rightarrow ZW$

- If we decompose the above relation into R1 (X, Y) and R2 (X, Z, W) the dependency $Y \rightarrow ZW$ is not preserved.
- But, If we decompose the above relation into R1 (X, Y) and R2 (Y, Z, W) the all dependencies are preserved.

c. No repetition of information

- Decomposition that we have done should not suffer from any repetition of information problem.
- It is desirable not to have any redundancy in database.
- STUDENT and SECTION data are separated into distinct relations. Thus we do not have to repeat STUDENT data for each SECTION.
- If a single SECTION is made into several STUDENTS, we do not have to repeat the SECTION data for each STUDENT.
- IT is desirable not to have any redundancy in database.
- This property may be achieved by normalization process.

11.7 Keys and Attributes in keys

Q. Write a note on : Candidate Key, Secondary Key and Super Key.

1. Introduction

- Normalization process will makes use of some types of keys to remove the redundancies present in data of relational tables.

- The column value that uniquely identifies a single record in a table called as Key of table.
- Any key consisting of a single attribute is called a **simple key,** while that consisting of a combination of attributes is called a **composite key**.
- The non-normalized relations may causes redundancy problems which leads to data anomalies.

2. Super Key

Q. Write a note on : Super Key.

- A **superkey** of a relation is a set of attributes $S \subseteq R$ with the property that no two tuples in relation will have same combination of attribute values in S.
- In other words, a relation will have unique set of attributes S.
- If we add additional attributes to above set of attributes S, the resulting combination would still uniquely identifies a single record in a table. Such augmented keys are also called as **superkey**.

Example,

- Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

$$A \rightarrow BC$$

$$AC \rightarrow DE$$

$$E \rightarrow F$$

- To find out key of relation R,

$$\{A\}^+ = \{A, B, C, D, E, F\}$$

$$\{AC\}^+ = \{A, B, C, D, E, F\}$$

$$\{ACD\}^+ = \{A, B, C, D, E, F\}$$

So {A}, {AC}, {ACD} are all superkey.

Example,

- STUDENT (Id, Name, Class, Branch, Age, Address, Mobile)
- The ID is a key attribute of STUDENT table, so ID and Name can be a superkey.

3. Candidates Key

Q. Write a note on : Candidate Key.

- Superkey concept given above may contain unnecessary attributes to key so the concept of a superkey is not sufficient.
- A candidate key (CK) is a superkey with the minimal attribute from super key.
- A minimal (irreducible) superkey is called as candidates key.
- A superkey that does not contain a subset of attributes that is itself a superkey.
- Minimum attributes of superkey by omitting unnecessary attributes of table which are sufficient for identifying entity (row/record) uniquely are called as **candidate keys**.
- Candidate key is also a potential primary key.

Example,

- Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

$$A \rightarrow BC$$

$$AC \rightarrow DE$$

$$E \rightarrow F$$

To find out key of relation R,

$$\{A\}^+ = \{A, B, C, D, E, F\}$$

$$\{AC\}^+ = \{A, B, C, D, E, F\}$$

$$\{ACD\}^+ = \{A, B, C, D, E, F\}$$

So {A}, {AC}, {ACD} are all superkey and {A} is a candidates key.

Example,

- STUDENT (Id, Name, Class, Branch, Age, Address, Mobile)
- The ID is a candidate key attribute of STUDENT table.

4. Secondary Key

Q. Write a note on : Secondary Key.

- A candidate key selected to uniquely identify all other attribute values in any given row.
- If a number of candidate keys in a relation schema then one is arbitrarily selected as **primary key** and other keys are called as **secondary keys**.
- Secondary key of a table is a column or combination of some columns used for data retrieval process.

Example,

- Consider a Relation R (A, B, C, D, E, F) with Functional dependencies,

$$AB \rightarrow C, BC \rightarrow DE, E \rightarrow AF$$

- To find out key of relation R,

  $\{AB\}^+ = \{A, B, C, D, E, F\}V$

  $\{BC\}^+ = \{A, B, C, D, E, F\}$

  So {AB} and {BC} are all candidates key.

  If {AB} is selected as primary key then {BC} is called as secondary key.

5. Prime Attributes

Q. Write a note on : Prime Attributes.

- An attribute in a relation schema R is called as prime attribute, if it is a member of any of the candidate key present in a relation.
- If an attribute is not a member of any candidate key then it is called as nonprime attribute.

Example

- Consider an Employee table with following FDs,
- Employee_Id → Ename, Salary
- Employee_Id, Project_Id → Hours, Allowance

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

In above table candidate key is {Employee_Id, Project_Id}

Prime Attributes : Employee_Id, Project_Id

Non-Prime Attributes : Ename, Salary, Hours, Allowance

11.8 Normalization Process

Q. What is Normalization ?
MU - May 16, May 18, May 19, Dec.19, 2 Marks

Q. Explain need of Normalisation along with all the normal forms. **MU - Dec. 17,Dec. 18, 10 Marks**

1. Introduction

- Normalization is a step by step decomposition of complex records into simple records.
- Normalization is a process of organizing data in database in more efficient form. It results in tables that satisfy some constraints and are represented in a simple manner.
- This process is also called as canonical synthesis.
- Normalization is a step by step decomposition and database designers may not normalize relation to the highest possible normal form.
- The relations may be left in a lower normal form like 2NF, which may cause some penalties like data anomalies.

2. Definition

Normalization is a process of designing a consistent database by minimizing redundancy and ensuring data integrity through decomposition which is lossless.

3. Goals of Database Normalization

1. Ensures data integrity.
   - Data integrity ensures the correctness of data stored within the database.
   - It is achieved by imposing integrity constraints.
   - An integrity constraint is a rule, which restricts values present in the database.
2. Prevents redundancy in data.
   - A non-normalized database is more vulnerable to various problems, if it stores data redundantly.
   - If data is stored in two locations, but the data is updated in only one of the locations, then that data becomes inconsistent; this is referred to as an "update anomaly".
   - A normalized database stores non-primary key data in only one location.

3. To avoid data anomaly

- A non-normalized table can have some inconsistencies which may cause data anomalies.
- A relational database table should avoid all data anomalies.
- The normal forms of relational database decide, whether relational database design is vulnerable to data anomalies.

a. Update Anomaly

Same information can be present in multiple records of various relations; updates to only one table may result in logical inconsistencies.

Example

- Each record in an "Emp_Salary" table might contain an Emp_ID, Ename, Address, Salary. Thus, a change

of address for a particular Employee will potentially need to be applied to multiple tables such as Employee table.

- If all the records are not updated then some tables may leave in an inconsistent state.

b. Insertion Anomaly

There is a possibility in which certain facts cannot be recorded at all or they are not yet recorded.

Example

- Consider a table, Faculty (Faculty_ID, FName, Subject_Code, Subject, Class).
- We can add the details of any faculty member who teaches for a certain subject in a certain class, but we cannot record the details of a new faculty member who has not yet been assigned to teach any subject or class. So, subject and class column may be empty initially.
- If data is deleted from one table all relevant data must also be deleted or redundant.

c. Deletion Anomaly

- If data deleted from one table then all relevant data in another related tables must also be deleted, otherwise it will create redundancy problem.
- Deletion of some data from a relation necessitates the deletion of some unrelated data also called as deletion anomaly.

Example

- In the previous example, the table suffers from this type of anomaly. If a faculty member temporarily ceases to be assigned a subject, we must delete the entire record on which that faculty member appears.
- There are formal methods for quantifying "how normalized" a relational database is, and these classifications are called Normal Forms (or NF).

11.9 First Normal Form (1NF)

Q. Explain 1NF.
MU - May 16, May 18, May 19, Dec.19, 2 Marks

Q. Explain need of Normalisation along with all the normal forms. MU - Dec. 17, Dec. 18, 10 Marks

1. Introduction

- This is simplest form of normalization, simplifies each attribute in relation.
- This normal form given by E.F. Codd (1970) and the later version by C.J. Date (2003).

2. Definition

- A relation is in 1NF, if every row contains exactly one value for each attribute.
- 1NF states that all attributes in relation must have atomic (indivisible) values and all attribute in a tuple must have a single value from the domain of that attribute.
- In short rules for data in 1NF is,
- A columns in a table should contain only indivisible data.

3. Example

- Consider an employee table with columns as shown in diagram,
- The relational schema not in 1 NF is represented as,

Employee Table

Employee_Id	Ename	Salary	Ecity

– The state of Employee relational schema is as given below and it contains the Ecity which is non atomic (divisible) domain.

Table 11.9.1 (Non-Normalised Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai, Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai, Delhi

- To convert relational schema in 1NF, the Ecity attribute is divided in atomic domains it may introduce some data redundancy.

Fig. 11.9.2 1NF Employee Table

Employee_Id	Ename	Salary	Ecity
10	Mahesh	50000	Mumbai
10	Mahesh	50000	Pune
12	Suresh	25000	Mumbai
15	Ganesh	26000	Pune
18	Kasturi	50000	Mumbai
18	Kasturi	50000	Delhi

4. Minimizing Domain Redundancy

- The first normal for will solve the group redundancy occurs in domain value as it allows only a single value from the domain of that attribute.
- So, 1NF will solve all problems related to domain redundancy.
- Nested relations must be removed to convert relation in 1 NF.

11.10 Second Normal Form (2NF)

Q. Explain 2NF. **MU - May 16, May 18, May 19, Dec.19, 2 Marks**

Q. Explain need of Normalisation along with all the normal forms. **MU - Dec. 17, Dec. 18, 10 Marks**

1. Introduction

- This normal form makes use of full functional dependency and tries to remove problem of redundant data introduced by 1NF decomposition.
- Therefore, before applying 2NF to a relation, it needs to satisfy 1NF condition.
- This normal form is given by E.F. Codd in 1971.

2. Definition

- A relation is in 2NF, if it is in 1NF and all non-key attributes in relation are fully functionally dependent on the primary key of the relation.

OR

- A relation is in 2NF, if it is in 1NF and every non-key attribute is fully functionally dependent on the complete primary key of relation (and not depends on part of (partial) primary key).
- In short 2NF means,
- It should be in 1NF.
- There should not be any partial dependency on primary key attributes.

3. Steps

a. Find and remove attributes that are related to only a part of the key or not related to key.

b. Group the removed attributes in another table.

c. Assign the new table a key that consists of that part of the old composite key.

d. If a relation is not in 2NF, it can be further normalized into a number of 2NF relations.

4. Example

- Consider an employee table with columns as shown in diagram,
- The relational schema **not in 2 NF** is represented as,
- Consider an Employee table with following FDs,
- Employee_Id → Ename, Salary
- Employee_Id, Project_Id → Hours, Allowance
- As
- {Employee_Id, Project_Id} → Ename, Salary, Hours, Allowance

Therefore,

- Candidate key {Employee_Id, Project_Id} is selected as primary key.
- As attributes Hours, Allowance of employee table are full functionally dependent on primary key whereas attributes Ename and Salary are partially depends on primary key. (As Ename, Salary are depends **on part of primary key**)
- The state of Employee relational schema is,

Table 11.10.1 : Non-2NF Employee Table

Employee_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B056	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000
15	Ganesh	26000	E001	24	20000
18	Mahesh	50000	B056	11	10000

- To normalize above schema to 2NF we can decompose tables as ,
- Employee (Employee_Id, Ename, Salary)
- Employee_Id → Ename, Salary

Table 11.10.2 : 2NF Employee Table

Employee_Id	Ename	Salary
10	Mahesh	50000
12	Suresh	25000
15	Ganesh	26000
18	Mahesh	50000

- Project (Employee_Id, Project_Id, Hours, Allowance)
- Employee_Id, Project_Id → Hours, Allowance

Table 11.10.3 : 2NF Project Table

Employee_Id	Project_Id	Hours	Allowance
10	E001	44	40000
12	B056	31	30000
15	C671	23	20000
18	E002	12	15000
15	E001	24	20000
18	B056	11	10000

- Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

  $A \rightarrow BC$, $B \rightarrow DC$, $D \rightarrow EF$

  a. The candidate Key is {AD} $\rightarrow$ {A, D, B, C, E, F} selected as primary key.

  All attributes are partially dependent on primary key.

  Hence, Relation R is **not in 2NF**.

  b. The 2NF Relation Schema is,

  R1 (A, B, C, D) with FDs $A \rightarrow BC$, $B \rightarrow DC$

  R2 (D, E, F) with FDs $D \rightarrow EF$

5. Minimizing Tuple Redundancy

- The second normal form will avoid same tuples to be repeated in a table as it forces all non-key attributes must be full functionally depends on primary key of a relation.
- 2NF will create a new table for each partial key with all its dependent attributes.

11.11 Third Normal Form (3NF)

Q. Explain 3NF.

MU - May 16, May 18, May 19, Dec.19, 2 Marks

Q. Explain need of Normalisation along with all the normal forms. MU - Dec. 17, Dec. 18, 10 Marks

1. Introduction

- This normal form is given by E.F. Codd in 1971.
- This normal form introduced to minimize the transitive redundancy.
- To remove the data anomalies left in relational schema even after applying second normal form like transitive dependencies.

2. Definition

- A relation is in 3NF, if it is in 2NF and all non-prime attributes of the relation are non-transitively dependent on the every key.
- A relation R is in 3NF if all non prime attributes are,
  1. Full functionally dependent on primary key.
  2. Non-transitive dependent on every key.
- A relational schema R is in 3NF, if non-trivial functional dependency $X \rightarrow A$ holds true where X is a superkey and A is a prime attribute.

3. Example

- Consider an employee table with columns as shown in diagram,
- The relational schema **not in 2 NF** is represented as,
- Consider an Employee table with following FDs,
- Employee_Id $\rightarrow$ Ename, Salary, Department_Id
- Department_Id $\rightarrow$ Dname
- The state of Employee relational is,

Table 11.11.1 : Employee Table

Employee_Id	Ename	Salary	Department_Id	Dname
10	Mahesh	50000	C1	IT
12	Suresh	25000	E2	HR
15	Ganesh	26000	C1	IT
18	Mahesh	50000	E2	HR

{Employee_Id} $\rightarrow$ Ename, Salary, Department_Id, Dname

Therefore,

- Candidate key {Employee_Id} is selected as primary key.
- As all attributes in employee table are full functionally dependent on primary key. Therefore, **Relation R is in 2NF.**
- Non-prime attributes Ename, Salary, Department_Id are non-transitively dependent on primary key. But, Dname attribute is transitively dependent on key. Therefore, **Relation R is not in 3NF.**

  Employee_Id $\rightarrow$ Department_Id

  Department_Id $\rightarrow$ Dname
- To normalize above schema to 3NF we can decompose tables as ,

  Employee (Employee_Id, Ename, Salary, Department_Id)

  Employee_Id $\rightarrow$ Ename, Salary, Department_Id

Table 11.11.2 3NF Employee Table

Employee_Id	Ename	Salary	Department_id
10	Mahesh	50000	C1
12	Suresh	25000	E2
15	Ganesh	26000	C1
18	Mahesh	50000	E2

Department (Department_Id, Dname)

Employee_Id → Dname

Table 11.11.3 : 3NF Department Table

Department_Id	Dname
C1	IT
E2	HR

- Consider, Relation R(A, B, C, D, E, F) and the FDs as below,

  A → BC , B → D, D → EF

1. The candidate Key is {A} → {A, D, B, C, E, F} selected as primary key.
   - All attributes are full functionally dependent on primary key.
   - Hence, Relation R is **in 2NF**.
   - But, non-prime attributes B, D, E and F are transitively depends on key.
   - So, Relation R is **not in 3NF**.
2. The 3NF Relation Schema is,
   - R1 (A, B, C) with FDs A → BC
   - R2 (B, D) with FDs B → D
   - R3 (D, E, F) with FDs D → EF

Note : In most cases, third normal form is the sufficient level of decomposition. But some case requires the design to be further normalized up to the level of 4th as well as 5th. These are based on the concept of Multi Valued Dependency.

4. Minimizing Group Redundancy

- The third normal form will avoid repeating groups in same table as it forces all non-prime attributes must be non-transitively depends on key of a relation.
- 3NF will create a new table for each transitive attribute and its dependent attributes.

Table 11.11.4 : Summary of normal form based on primary keys and corresponding normalization

Normal form	Test	Remedy (normalization)
First (1NF)	Relation should have to multivalued attributes or nested relations.	Form new relations for each multivalued attributed to nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attributed should be functionally.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original promarykey and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should bot have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determinte(s) other nonkey attribute(s).

11.12 Boyce-Codd Normal Form (BCNF)

Q. Describe BCNF in detail. **MU - Dec. 15, Dec. 16, 5 Marks**

Q. Explain need of Normalisation along with all the normal forms. **MU - Dec. 17, Dec. 18, 10 Marks**

1. Introduction

- This normal form is governed by Raymond F. Boyce and E.F. Codd in 1974.
- BCNF is more rigorous form of 3NF and every relation in BCNF is always in 3NF.
- The intention of Boyce-Codd Normal Form (BCNF) is that 3NF does not satisfactorily handle the case of overlapping candidate keys.

- If transitivity is present in prime attributes of relation may not be removed by 3NF.

2. Definition

- A relation R is said to be in BCNF, if and only if every determinant is a candidate key.
- A relational schema is in BCNF, if a non-trivial functional dependency X → A is true then X is a superkey of relation R.
- In 3NF definition A should be prime attribute, which is not the case in BCNF definition.

3. Example

- Consider an employee table in which employee can work in more than one department,
- The relational schema **not in 2 NF** is represented as,
- Consider an Employee table with following FDs,
- Employee_Id → Ename, Salary, Department_Id
- Department_Id → Dname
- The state of Employee relational is,

Table 11.12.1 : Employee Table

Employee_Id	Ename	Department_Id	Dname	Dtype
10	Mahesh	C1	IT	Technical
12	Ganesh	E2	HR	Skill
12	Ganesh	C1	IT	Technical
10	Mahesh	E2	HR	Skill
13	Satish	E1	TS	Technical

Employee_Id → Ename

Department_Id → Dname, Dtype

Therefore,

- Candidate key {Employee_Id, Department_Id} is selected as primary key.
- As no attribute in employee table is full functionally dependent on primary key. Therefore, **Relation R is not in 2NF.** All non-prime attribute are non-transitively dependent on primary key. Therefore, **Relation R is in 3NF.**
- To normalize above schema to **BCNF** we can decompose tables as ,
- **Employee** (Employee_Id, Ename)
- Employee_Id → Ename
- The determinant Employee_Id is candidate key.

Table 11.12.2 : 3NF Employee Table

Employee_Id	Ename
10	Mahesh
12	Ganesh
13	Satish

- **Department** (Department_Id, Dname, Dtype)
- Department_Id → Dname, Dtype
- The determinant Department_Id is candidate key.

Table 11.12.3 : 3NF Department Table

Department_Id	Dname	Dtype
C1	IT	Technical
E2	HR	Skill
E1	TS	Technical

- **Emp_Dept** (Employee_Id, Department_Id)

Table 11.12.4 : 3NF Emp Dept Table

Employee_Id	Department_Id
10	C1
12	E2
12	C1
10	E2
13	E1

4. Minimizing Key Transitivity (Key redundancy)

- The BCNF will produces a table for each functional dependency to make all determinants as key of relation.
- BCNF will create a new table for each transitive attribute and its dependent attributes.

Example 11.12.1 : Consider the following relation,
CAR-SALE (Car#, Date-sold, Salesman#, commission%, Discount-amt)
Assume that {Car#, Salesman#} is the primary key.
Additional dependencies are,
Date-sold → Discount_amt
Salesman# → commission%
Based on the given primary key, is this relation in 1NF, 2NF or 3NF ?
Why or why not ?
How would you successively normalize it completely ?

MU - Dec. 14,10 Marks

Solution :

- CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)
- Assuming {Car#, Salesman#} is the primary key

Therefore,

Car#, Salesman# → Date-sold, commission%, Discount-amt

Additinally,

Date-sold → Discount_amt

Salesman# → commission%

- The above relation **is in 1NF as** all attributes of relation are atomic domains.
- The above relation **is in 2NF as** primary key is assumed and all non key attributes are full functionally depends on primary key.
- The above relation **is in 3NF as** all non prime attributes are non-transitively depending on primary key.
- Noramlized Relation

CAR-SALE (Car#, Salesman#, Date-sold, commission%, Discount-amt)

Car#, Salesman# → Date-sold, commission%, Discount-amt

Date-sold → Discount_amt

Salesman# → commission%

11.13 Converting Relational Schema to higher normal forms

Example 11.13.1 : Relation R(A, B, C, D, E, F, G, H, I, J). Having following set of FD, show convert table to highest normal form.

AB → C, C → EF, AD → GH, G → I, H → J

Solution :

a. 1NF

Assuming all attributes are atomic domains so relation R is in 1NF.

b. 2 NF

{A, B, D}$^+$ → {**A, B, D**, C, E, F, G, H, I, J}

{A, B, D} is candidate key for relation R.

No attribute in relation is full functionally dependent on above key.

Therefore, Relation R is not in 2NF.

Decomposition to 2NF

R_1 (A, B, C, E, F); AB → C, C → EF

R_2 (A, D, C, E, F); AD → GH, G → I, H → J

c. 3NF

In relation R1, non-prime attributes E, F are transitively depends on key. So, relation is not in 3NF.

The relation R1 in 3NF can be written as,

R_{1a} (A, B, C); AB → C

R_{1a} (C, E, F); C → EF

In relation R2, non-prime attributes I, J are transitively depends on key. So, relation is not in 3NF.

The relation R1 in 3NF can be written as,

R_{2a} (A, D, G, H); AD → GH

R_{2b} (G, I); G → I

R_{2c} (H, J); H → J

c. BCNF

Relation	FDs	Determinant	Key	BCNF?
R_{1a} (A, B, C)	AB → C	AB	AB	Yes
R_{1a} (C, E, F)	C → EF	C	C	Yes
R_{2a} (A, D, G, H)	AD → GH	AD	AD	Yes
R_{2b} (G, I)	G → I	G	G	Yes
R_{2c} (H, J)	H → J	H	H	Yes

So, relational schema in BCNF is as given below,

R_{1a} (A, B, C); AB → C

R_{1a} (C, E, F); C → EF

R_{2a} (A, D, G, H); AD → GH

R_{2b} (G, I); G → I

R_{2c} (H, J); H → J

Example 11.13.2 : We are given Relation R with Attributes A, B, C, D, E, F and the FDs as below, Find and explain which Armstrong's Axioms can be applied here to find closure,

A → BC, B → E, CD → EF

Solution :

1. {A}$^+$

a. Axiom of Pseudo transitivity

A → BC and B → E ...(1)

∴ A → EC ...(2)

c. Decomposition from Equations (1) and (2)

A → B, A → C, A → E

∴ {A}$^+$ = {A, B, C, E}

{A}$^+$ = { A, B, C, E}

2. {A, B} = {A, B, C, E}

3. {C, D} = {C, D, E, F}

4. {A, D} = {A, B, C, D, E, F}

11.14 Solved Examples on Normalization - Real time application of normalization (Self Learning Topic)

Example 11.14.1 : Consider a dependency diagram of relation R and normalize it up to third normal form.

MU - Dec. 13, May 18, 10 Marks

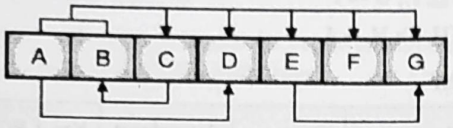


Fig. Ex. 11.14.1

Solution :

Noramlized Relation R (A, B, C, D, E, F, G) with set of FDs

$AB \rightarrow CDEFG$

$C \rightarrow B$

$A \rightarrow D$

$E \rightarrow G$

Example 11.14.2 : Consider a dependency diagram of relation R and normalize it upto third normal form.

MU - Dec. 13, 10 Marks

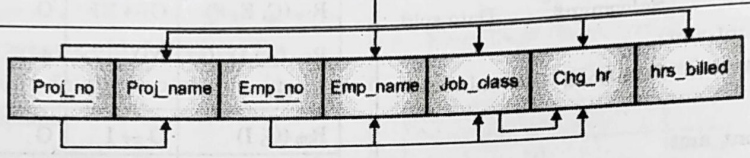


Fig. Ex. 11.14.2

Solution :

- Noramlized Relation,
- Employees (Proj_no, Emp_no, Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed)
- With set of FDs

  Proj_no, Emp_no → Proj_name, Emp_name, Job_Class, Chg_Hr, Hrs_Billed

  Emp_no → Emp_name, Job_Class, Chg_Hr

  Proj_no → Proj_name

  Job_Class → Chg_Hr

Example 11.14.3 : Consider the following dependency diagram of relation R and normalize till 3NF form.

MU - May 17, 10 Marks

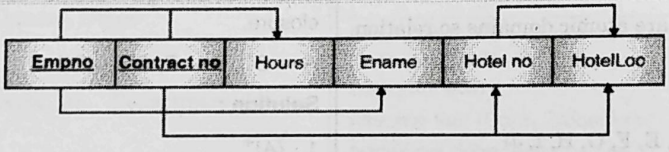


Fig. Ex. 11.14.3

Solution :

- Converting dependency diagram to dependencies,

  Empno,Contactno -> Hours

  Empno ->Ename

  Contactno ->Hotelno,HotelLoc

  Hotelno->HotelLoc
- 3NF schema is as given below,

  Emp(Empno,Contactno,Hours); Empno,Contactno -> Hours

  Emp_Details(Empno,Ename); Empno ->Ename

  Hotel_Contact(Contactno,Hotelno,HotelLoc); Contactno ->Hotelno,HotelLoc

  Hotel(Hotelno,HotelLoc); Hotelno ->HotelLoc

Example 11.14.4 : Relation R(A, B, C, D, E, F, G, H, I, J). Having following set of FDs, show whether it is in 2NF and 3NF.

$AB \rightarrow C$ $\quad$ $BD \rightarrow EF$

$AD \rightarrow GH$ $\quad$ $A \rightarrow I$

$H \rightarrow J$

Solution :

a. Finding key

- For finding key we must find the super key of above relation.
- We start with minimum possible key

Closure of $\{A\}^+ \rightarrow \{A, I\}$

Closure of $\{A, B\}^+ \rightarrow \{A, B, C, I\}$

Closure of $\{B, D\}^+ \rightarrow \{B, D, E, F\}$

Closure of $\{A, D\}^+ \rightarrow \{A, D, G, H, I, J\}$

Closure of $\{A, B, D\} \rightarrow \{A, B, C, D, E, F, G, H, I, J\}$

- So Super key for above relation is A, B, D.

b. Decomposition to 2NF

- **2NF** → Every non key attribute must full functionally dependent on primary key.
- As All other attribute can be determined by A, B, D primary key but each attribute not directly dependent on all A, B and D together.
- Example E, F depend only on B, D and not on A. It's a partial dependency.
- So, above relation not in 2NF.

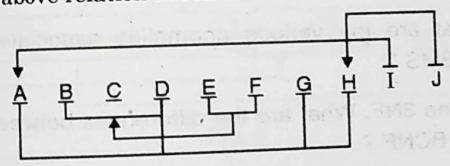


Fig. P. 11.14.4

Example 11.14.5 : Relation R (A, B, C, D, E). Having following set of FDs, convert it to 3NF.

$A \rightarrow BD$ $\quad$ $B \rightarrow C$ $\quad$ $D \rightarrow E$

Solution :

a. Finding key

Closure of A : $\{A\}^+ = \{A,B,D,C,E\}$

A can be the key. No other single attribute can be key.

b. Decomposition to 2NF

- **2NF** → every non key attribute must full functionally dependent on primary key.
- All attribute can be determined by primary key A and each attribute directly dependent on A.

∴ Relation in 2NF

c. Decomposition to 3NF

- For relation to be in 3NF there should not be any Transitive dependency

  In above relation $A \rightarrow D$ and $D \rightarrow E$

  ∴ E is transitively dependent on A

  ∴ Transitive dependency exists.

  ∴ Relation not in 3NF
- Composing to 3NF

  R_1 (A, B, C, D)

  R_2 (D, E)
- Now, relation in 3NF

d. Decomposition to BCNF

- **BCNF** → every determinant must be candidate key.
- In relation R_1 A is Determinant and it is candidate key also.
- In relation R_2 D is Determinant and it is candidate key also.

∴ Relation R is in BCNF

Example 11.14.6 : We are given Relation R with Attributes A, B, C, D, E, F and the FDs as below, Find and explain which Armstrong's Axioms can be applied here to find closure,

$A \rightarrow BC$ $\quad$ $B \rightarrow E$ $\quad$ $CD \rightarrow EF$

Soln. :

1. $\{A\}^+ = \{A, B, C, E\}$

Axioms :

a. Axiom of Reflexivity

$A \rightarrow A$ ...(1)

b. Axiom of Pseudo transitivity

$A \rightarrow BC$ and $B \rightarrow E$ ...(2)

$\therefore A \rightarrow EC$ ...(3)

c. Decomposition from Equations (1), (2) and Equation (3)

$A \rightarrow A$, $A \rightarrow B$; $A \rightarrow C$; $A \rightarrow E$

∴ $\{A\}^+ = \{A, B, C, E\}$

2. $\{A, B\} = \{A, B, C, E\}$
3. $\{A, D\} = \{A, B, C, D, E, F\}$

Example 11.14.7 : Consider relation R with five attribute ABCDE. Your given the following dependencies :

$A \rightarrow B$ $BC \rightarrow E$ $ED \rightarrow A$

1. List all keys for R
2. Is R in 3NF
3. Is R in BCNF **MU - May 19,10 Marks**

Ans. :

1. List all keys for R

$\{A\}^+ = \{A, B\}$

$\{AB\}^+ = \{A, B,\}$

$\{E;D\}^+ = \{E, D, A, B\}$

$\{E, D, C\} = \{E, D, A, B, C\}$

$\therefore$ Keys $\{E, D, C\}$

are $\{E, D, C, B\}$

$\{E, D, C, A\}$

$\{E, D, C, A, B\}$

2. Is R in 3NF : No.
3. IS R in BCNF : No.

Example 11.14.8 : List the functional dependencies which satisfy the relation **MU - Dec. 19,10 Marks**

x	y	z
X_1	Y_1	Z_1
X_1	Y_2	Z_1
X_2	Y_2	Z_1
Z_2	Y_2	Z_1

Ans. :

$x \rightarrow z$
$xy \rightarrow z$
$yz \rightarrow z$
$y \rightarrow z$
$z \rightarrow x$

} not valid

$z \rightarrow y$
$y \rightarrow z$
$xz \rightarrow y$

} not valid

Review Questions

1. Write a short note : Functional dependency
2. What is nformalizations ? What is its importance in DBMS design ? Explain 1NF, 2NF, 3NF and BCNF with suitable example.
3. What are the three data anomalies that are likely to occur as a result of data redundancy ? Can data redundancy be completely eliminated in database approach ? Why or why not ?
4. What is closure set of functional dependency ?
5. We are given Relation R with Attributes A, B, C, D, E, F and the FDs as below, Find and explain which Armstrong's Axioms can be applied here to find Closure,

   $A \rightarrow BC$, $B \rightarrow EC$, $D \rightarrow EF$
6. Describe data redundancy in relational schema.
7. Distinguish between functional dependency and multivalued dependency.
8. List the Armstrong's axioms for functional dependencies ?
9. Why BCNF is more desirable than 3 NF ?
10. State and compare 3 NF and BCNF.
11. Suppose we have a relation ABCD with some FD's F as follows :

    $F = AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$

    Compare closure A+, C+, (AB)+, (AC)+ and all FD'S (functional dependencies) that follow from F.
12. Differentiate between full functional dependency and partial functional dependency.
13. What are the various anomalies associated with RDBMS ?
14. Define 3NF. What are the differences between 3NF and BCNF ? ... (1) ... (2)
15. Give two sets F1 and F2 of FDs for a relation (A, B, C, D, E)

    $F1 : A \rightarrow B$, $AB \rightarrow C$, $D \rightarrow AC$, $D \rightarrow E$

    $F2 : A \rightarrow BC$, $D \rightarrow AE$

    Are F1 and F2 equivalent ? Explain.
16. Consider the relation R = (A, B, C, D, E, F, G, H) with following FDs :

    $F = \{AC \rightarrow G, D \rightarrow EG, BC \rightarrow D, CG \rightarrow BD, ACD \rightarrow B, CE \rightarrow AG\}$

    Find the canonical cover and minimal of F.

11.15 University Questions and Answers

May 2015

1. List all functional dependencies satisfied by the relation. **(5 Marks)**

a	b	c
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

Dec. 2015

2. Describe BCNF in **detail** **(5 Marks)**

May 2016

3. Define normalization ? Explain 1NF, 2NF and 3NF with example. **(10 Marks)**

Dec.2016

4. Describe BCNF in detail. **(5 Marks)**

May 2017

5. Consider the following dependency diagram of relation R and normalize till 3NF form.
(Ans. : Refer Example 11.12.3)

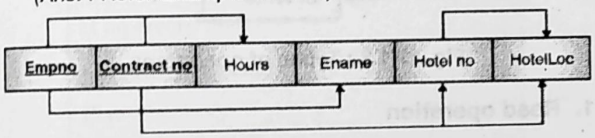


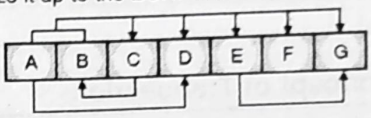
Fig. 3

Dec. 2017

6. Explain need of Normalisation along with all the normal forms. **(10 Marks)**

May 2018

7. Define Normalization? Explain 1NF, 2NF and 3NF with examples? **(10 Marks)**
8. Construct a dependency diagram of relation Rand normalize it up to the BCNF Normal form



Dec. 2018

9. Illustrate the need of normalization? explain all forms with an example. **(10 Marks)**
10. Attempt the following : Functional Dependencies **(5 Marks)**

May 2019

11. Define normal forms and explain with suitable **example** First , Second, Third normal forms. **(10 Marks)**

Dec. 2019

12. Define normalization ? Explain 1NF, 2NF and 3NF with example. **(10 Marks)**

❑❑❑