Module-5
Relational–Database Design

By

Asst.Prof

Vaishali Patil

Outline of Unit - 5

- Pitfalls in Relational-Database designs ,
- Concept of normalization
- Function Dependencies ,
- First Normal Form,
- 2NF ,
- 3NF,
- BCNF,
- Examples

Pitfalls in Relational Database Design

Introduction

- Pitfalls in database design suggest the effects when database design is wrong or incorrect.
- When we design any relational database schema,we need to take care of many important aspects of databases in order to avoid problem to database.

Pitfalls in Relational Database Design

1. Redundant data
2. Inability to represent some data
3. Dependency of various attributes of relation
4. Loss of information

Pitfalls in Relational Database Design

Redundant Data

1. Redundancy is root of many problem

   Redundant data causes following problems

   a. Insert,Delete or update anomalies
   b. Wastage of Storage
   c. Generation of invalid data

2. Consider a example of Employee_Project table in which employee information and information about project for which he works is kept as follow:

Eid	Ename	Address	Project_id	Project	Salary
1	Sachin	Malad	12	IPF	25000
2	Jayendra	Vashi	44	CAP	30000
3	Suhas	Andheri	55	FID	12000

Pitfalls in Relational Database Design

Eid	Ename	Address	Project_id	Project	Salary
1	Sachin	Malad	24	LEX	25000
2	Jayendra	Vashi	24	LEX	30000
3	Suhas	Andheri	12	IPF	12000

- In above example there is repeatable data present in table which causes wastage of space in database ,also this data complicates the update operations.
- In above design as data for both table employee and project are kept together in one table hence if we want to update a project name then entire record(tuple) which is updated will have all data for both tables.
- In case of adding new row to table we need to give information about tables.
- Hence this approach will lead to redundant data in table.So this design shows bad database design
- To avoid redundancy,we can make use of functional dependencies by which we can perform decomposition of data into multiple table but that causes generation of invalid data

Sid	Sname	Cid	Cname	Fid	Fname	Salary
1	Rahul	C1	DBMS	F1	John	30,000
2	Ravi	C2	Java	F2	Bob	40,000
3	Nitin	C1	DBMS	F1	John	30,000
4	Amrit	C1	DBMS	F1	John	30,000
....						
.....						

Column Level Redundancy

Due to column level redundancy Following problems will occurs

1) Insertion Anomaly
2) Deletion Anomaly
3) Updation Anomaly

Pitfalls in Relational Database Design

Inability to Represent Some data

1. Another problem with design in above ex. is that we cannot represent information directly in any project,unless there is at least one employee working for that project.
2. As above schema requires data about employee such as eid ,ename etc.
3. One solution that we can find is Null values to b inserted for the following attributes:
    i. When attribute are not applicable to this tuples
    ii. When attributes value is unknown
    iii. When value is known but absent(not yet recorded)
4. But these Null values may introduce many problems
5. It is difficult to handle Null values in schema
6. It becomes difficult to specify joins and other operations.

Pitfalls in Relational Database Design

Dependency of various attributes of relations

- Dependency specifies how to interpret attributes values in tuple of each relations, are related with each other.
- If the database design is done carefully followed by a systematic mapping into relations, most of the dependencies will have been accounted for the resulting design should have a simple structure.
    - Ex- EMP(Emp_Id,Name,Salary)
- Salary depend on Emp_Id as each employee will have specific salary.

Pitfalls in Relational Database Design

Loss of Information

- Relational database design gives us summary information in form of tables or relation
- Hence many a time some information is not possible to include in database design which causes a loss of some information or information loss may be the result of some operation on relation
- This information that was lost can be useful or useless.
- If Information is useful then it will be a serve problem for understanding
    - Ex- “every student have different grasping power”

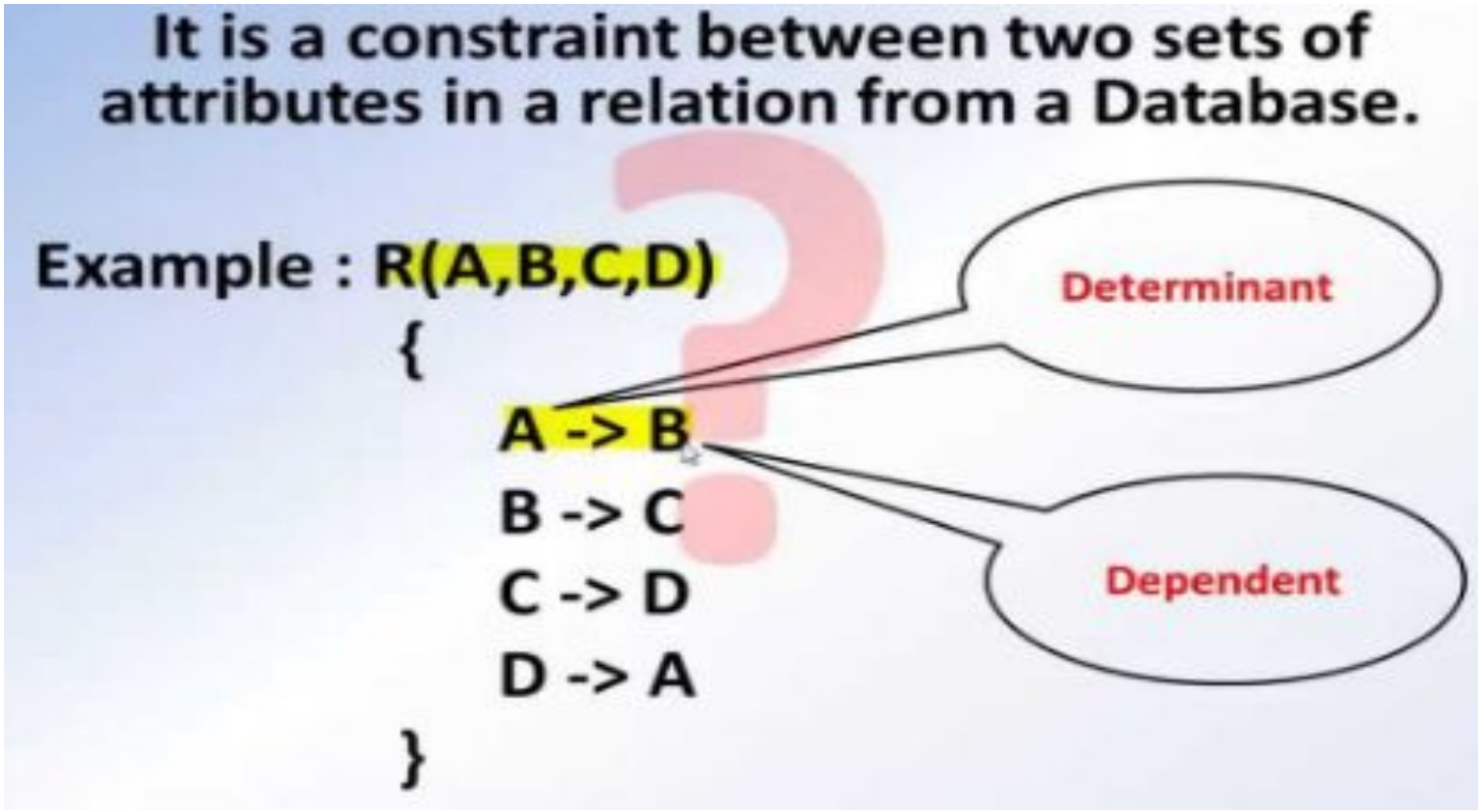
In above statement we cannot have attribute to actually measure grasping power for each student

Function 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**.
- Functional dependencies are used to mathematically express relations among database entities

Function Dependencies

For example:



Function Dependencies

Example:

roll_no	name	dept_name	dept_b
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

- roll_no → { name, dept_name, dept_building },→ Here, roll_no can determine values of fields name, dept_name and dept_building, hence a valid Functional dependency
- roll_no → dept_name , Since, roll_no can determine whole set of {name, dept_name, dept_building}, it can determine its subset dept_name also.
- dept_name → dept_building , Dept_name can identify the dept_building accurately, since departments with different dept_name will also have a different dept_building
- More valid functional dependencies: roll_no → name, {roll_no, name} ⇢ {dept_name, dept_building}, etc.

Function Dependencies

Example:

roll_no	name	dept_name	dept_buildi
42	abc	CO	A4
43	pqr	IT	A3
44	xyz	CO	A4
45	xyz	IT	A3
46	mno	EC	B2
47	jkl	ME	B2

Here are some invalid functional dependencies:

- name → dept_name Students with the same name can have different dept_name, hence this is not a valid functional dependency.
- dept_building → dept_name There can be multiple departments in the same building, For example, in the above table departments ME and EC are in the same building B2, hence dept_building → dept_name is an invalid functional dependency.
- More invalid functional dependencies: name → roll_no, {name, dept_name} → roll_no, dept_building → roll_no, etc.

Function Dependencies

FD Properties (Armstrong Axioms/ Closure of FD)

- 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

Function Dependencies

Armstrong's axioms/properties of functional dependencies:

1. **Reflexivity:** If Y is a subset of X, then X→Y holds by reflexivity rule

   For example, {roll_no, name} → name is valid.

2. **Augmentation:** If X → Y is a valid dependency, then XZ → YZ is also valid by the augmentation rule.

   For example, If {roll_no, name} → dept_building is valid, hence {roll_no, name, dept_name} → {dept_building, dept_name} is also valid.→

3. **Transitivity**: If X → Y and Y → Z are both valid dependencies, then X→Z is also valid by the Transitivity rule.

   For example, roll_no → dept_name & dept_name → dept_building, then roll_no → dept_building is also valid.

Function Dependencies

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

Given Relation R (X,Y,Z,W)

1) Union-

   If X→Y and X→ Z

   Then X→ YZ

2) Decomposition

   If X→YZ

   Then X→Y and X→ Z

3) Pseudo Transitivity

   If X→ Y and YZ→ W Then XZ→ W

Function Dependencies

Types of Functional dependencies in DBMS:

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency
5. Partial Functional dependency

Function Dependencies

1. Trivial Functional Dependency

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

For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **{roll_no, name} → name** is a trivial functional dependency, since the dependent **name** is a subset of determinant set **{roll_no, name}**

Similarly, **roll_no → roll_no** is also an example of trivial functional dependency.

Function Dependencies

2. 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.

For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, **roll_no → name** is a non-trivial functional dependency, since the dependent **name** is **not a subset of** determinant **roll_no**
Similarly, **{roll_no, name} → age** is also a non-trivial functional dependency, since **age** is **not a subset of {roll_no, name}**

Function Dependencies

3. Multivalued Functional Dependency

In **Multivalued functional dependency**, entities of the dependent set are **not dependent on each other.**

i.e. If **a → {b, c}** and there exists **no functional dependency** between **b and c**, then it is called a **multivalued functional dependency.**

For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, **roll_no → {name, age}** is a multivalued functional dependency, since the dependents **name** & **age** are **not dependent** on each other(i.e. **name → age** or **age → name doesn't exist !**)

Function Dependencies

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant.
i.e. If **a** → **b** & **b** → **c**, then according to axiom of transitivity, **a** → **c**. This is a **transitive functional dependency**

For example,

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, **enrol_no** → **dept** and **dept** → **building_no**,
Hence, according to the axiom of transitivity, **enrol_no** → **building_no** is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

Function Dependencies

5. Partial Dependencies

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

Function Dependencies

- Consider an employee table with columns as shown in Table 9.6.2.

Table 9.6.2 : 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.

Function Dependencies

Example-1

Consider relation R=(A,B,C,D,E,F) having FD's

A→ B A→ C

BC→D B→ E

BC→ F AC→F

Calculate some members of Axioms as be below:

1) A→ E
2) BC→ DF
3) AC→D
4) AC→ DF

Function Dependencies

Ex-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.

Keys And Attributes in Keys

- 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 table called as key of table
- any key consisting of single attributes is called a simple key,while that consisting of combination of attributes is called a composite key

Keys And Attributes in Keys

Super Key: Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation. All super keys can't be candidate keys but the reverse is true. In relation, a number of super keys is more than a number of candidate keys.

Prerequisite – Keys in Relational Model

Example: We have a given relation R(A, B, C, D, E, F) and we shall check for super keys by following given dependencies:

Functional dependencies	Super key
AB->CDEF	YES
CD->ABEF	YES
CB->DF	NO
D->BC	NO

By Using key **AB** we can identify the rest of the attributes **(CDEF)** of the table. Similarly, Key **CD.** But, by using key **CB** we can only identify **D** and **F,** not **A** and **E**. Similarly key **D**.

Keys And Attributes in Keys

Candidate Key: A candidate key is a set of attributes (or attributes) that uniquely identify the tuples in relation to or table. As we know the Primary key is a minimal super key, so there is one and only one primary key in any relationship but there is more than one candidate key that can take place. The candidate key's attributes can contain a NULL value which opposes to the primary key.

Example:

```
Student{ID, First_name, Last_name, Age, Sex, Phone_no}
```

Here we can see the two candidate keys **ID** and **{First_name, Last_name, DOB, Phone_no}**. So here, there are present more than one candidate keys, which can uniquely identify a tuple in a relation.

Keys And Attributes in Keys

Super Key	Candidate Key
Super Key is an attribute (or set of attributes) that is used to uniquely identifies all attributes in a relation.	Candidate Key is a subset of a super key.
All super keys can't be candidate keys.	But all candidate keys are super keys.
Various super keys together makes the criteria to select the candidate keys.	Various candidate keys together makes the criteria to select the primary keys.
In a relation, number of super keys is more than number of candidate keys.	While in a relation, number of candidate keys are less than number of super keys.
Super key attributes can contain NULL values.	Candidate key attributes can also contain NULL values.

Keys And Attributes in Keys

Secondary Key

- If a number of candidate keys in a relation schema then one is arbitrarily selected as primary key and other keys are called secondary key
- Secondary key of a table is a column or combination of some columbus used for data retrieval process

Prime attributes

- An attribute in 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 a any candidate key then it is called nonprime attributes

Keys And Attributes in Keys

Employee

Emp_Id	Ename	Salary	Project_Id	Hours	Allowance
10	Mahesh	50000	E001	44	40000
12	Suresh	25000	B506	31	30000
15	Ganesh	26000	C671	23	20000
18	Mahesh	50000	E002	12	15000

Consider Employee Table with FD's.
Then Prime attribute and Non Prime attributes are

Emp_Id→ Ename,Salary

Emp_Id,Project_Id→ Hours,Allowances

Candidate Key is {Emp_Id,Project_Id}

Prime Attribute{Emp_ID,Project_ID}

Non Prime Attribute{Ename,Salary,Hours,allowance}

Design 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

Guideline 2 : Reducing the Redundant Data in Tuples

Guideline 3 : Reducing Null values in Tuples

Guideline 4 : Disallowing Spurious Tuples

Design Guidelines for Relational Schema

Guideline 1: Clear semantics of the attributes in relational schema

- Semantics of attribute should be very clear in relational schema so that relational schemas 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

Design Guidelines for Relational Schema

Guideline 2 : Reducing the Redundant Data in Tuples

1. A relational schema may have some redundancy in database design, if it stores data redundantly
2. If same data is stored at more than one location will leads to redundancy and wastage of memory spaces
3. Data Anamolies: An inconsistent data may cause some problems while adding, updating or deleting in table which is called as data anomalies.

Design Guidelines for Relational Schema

Guideline 2 : Reducing the Redundant Data in Tuples

4. 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 update anomaly.
5. A normalized database stores non-primary key data in only one location.
6. 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, from_Date, to_Date)

Design Guidelines for Relational Schema

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.

- *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

Design Guidelines for Relational Schema

b) Insertion Anomaly

- There is 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.

- *Example*

It is not possible to add a row in Emp_Salary table or Emp_Designation table for an employee who does not exists in employee table.

Design Guidelines for Relational Schema

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.

- *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

Design Guidelines for Relational Schema

Guideline 3 : Reducing Null values in Tuples

1. A value of NULL is different from an empty, White spaces (blank spaces) or zero value
2. Null values in tuple will cause wastage of memory space and it will also create problem of understanding.
3. Relations should be designed in such a way that their tuples should not contain any NULL values
4. We can at least try make number of NULL values as low as possible.

Design Guidelines for Relational Schema

Guideline 3 : Reducing Null values in Tuples

5. Attributes with NULL values can be placed in separate relations with the primary key.
6. There are certain reasons for Null values:
   - Not applicable data
   - Invalid data
   - Unknown data or data not available.

Design Guidelines for Relational Schema

Guideline 4 : Disallowing Spurious Tuples

1. 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
2. The relational schema must be designed to satisfy the property of lossless join
3. If original relation contains fewer number of tuple then tuples generated by doing a natural-join of original relations.

Concept of Normalization:

It is techniques to Remove or Reduced redundancy from table.

ID	Name	Age
1	Mohan	20
2	Rohan	25
1	Mohan	20

Row level Redundancy

Row level redundancy can be remove by applying primary key

Concept of Normalization:

1) Insertion Anomaly(New course introduce)
2) Deletion Anomaly(delete Sid=2)
3) Updation Anomaly(update salary of Fid=F1)

Sid	S Name	Cid	Cname	Fid	Fname	Salary
1	Ram	C1	DBMS	F1	John	3000
2	Ravi	C2	Java	F2	Bob	4000
3	Nitin	C1	DBMS	F1	John	3000
4	Amit	C1	DBMS	F1	John	3000
Insertion Anomaly		C10	MBBS	Insertion Anomaly		

Column Level Redundancy

Concept of Normalization:

Student

Sid	Sname

Course

Cid	Cname

Faculty

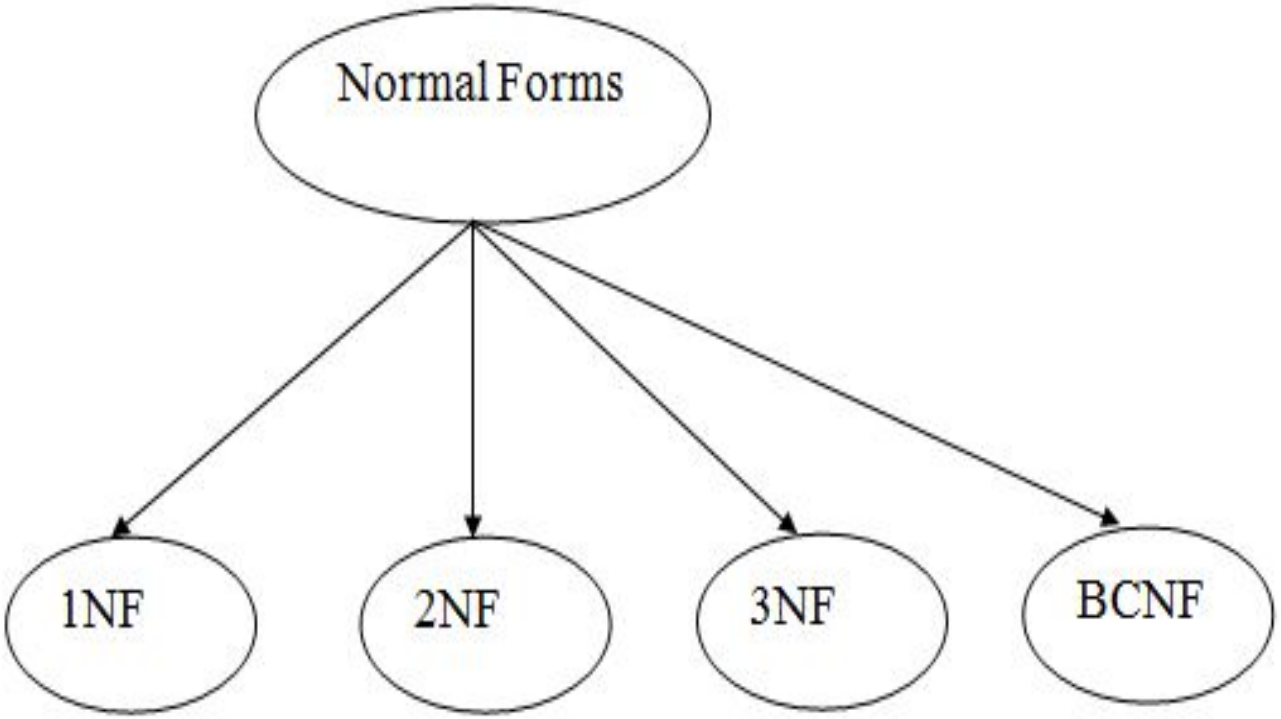
Fid	Fname	Salary

Concept of Normalization:

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- The normal form is used to reduce redundancy from the database table.

Concept of Normalization:

Types of Normal Forms :



Concept of Normalization:

Types of Normal Forms :

Normal Forms	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multivalued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

First Normal Form (1NF) :

Definitions-

1NF states that all attributes in relation must have atomic (indivisible) values and all attribute in a tuple must have single value from the domain of that attribute

First Normal Form (1NF) :

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

First Normal Form (1NF) :

S-id	Name	Credits	Dept	Building	Room_no
1	A	8	COMP	B1	501
2	B	5	COMP	B1	501
3	C	6	IT	B2	601
4	D	6	CIVIL	C2	301
5	E	7	CIVIL	C2	301
6	F	9	COMP	B1	501
7	E	7	IT	B2	601
8	G	5	IT	B2	601

First Normal Form (1NF) :

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

First Normal Form (1NF) :

The decomposition of the EMPLOYEE table into 1NF has been shown below:

E_ID	E_NAME	E_PHONE	E_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Only E_ID,E_Name alone can't be primary key.But we consider (E_ID and E_Phone) can be composite primary key

First Normal Form (1NF) :

The decomposition of the EMPLOYEE table into 1NF has been shown below:

E_ID	E_NAME	E_PHONE1	E_PHONE2	E_STATE
14	John	7272826385	9064738238	UP
20	Harry	8574783832	Null	Bihar
12	Sam	7390372389	8589830302	Punjab

Here maximum 2 values of E_phone is given but,suppose we have column with 2-3 values?

First Normal Form (1NF) :

The decomposition of the EMPLOYEE table into 1NF has been shown below:

E_ID	E_NAME	E_STATE
14	John	UP
20	Harry	Bihar
12	Sam	Punjab
12	Sam	Punjab

E_ID	E_Phone
14	7272826385
14	9064738238
20	8574783832
12	7390372389
12	8589830302

Base table
E_ID is Primary Key

E_ID is Foreign Key
(E_ID,E_Phone)-Primary Key

First Normal Form (1NF) :

Minimizing Domain Redundancy:

- The 1st NF will solve the group redundancy occurs in domain value as it allows only a single value from domain of that attribute.
- 1NF solve all problems related to domain redundancy.

Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key(non prime) attributes are fully functional dependent on the (candidate key) primary key
- There should not be any partial dependency

- **Example:** consider an employee table with following FD.
  - Emp_Id—> Ename,Salary
  - Emp_Id,Project-Id —> Hours ,Allowance
  - {Employee_Id,Project-Id}---->Ename,Salary,Hours,Allowance

Second Normal Form (2NF)

Emp-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

Emp_Id—> Ename,Salary

Emp_Id,Project-Id —> Hours ,Allowance

AS

{Emp_Id,Project-Id}---->Ename,Salary,Hours,Allowance

Therefore, Candidate Key{Emp_Id,Project_Id} selected as primary key

- As attribute Hours,Allowance of employee table are full functionally dependent on primary key whereas attributes Ename Salary are partially dependent on primary key(As Ename,Salary are Depends on part of primary key)

Second Normal Form (2NF)

To normalize above schema to 2NF we can decompose table as

Employee(Emp-Id,Ename,Salary)

Emp-Id—>Ename,Salary

Project(Emp-Id,Project-Id,Hours,Allowance)

Emp-Id,Project-Id—-->Hours,Allowance

Emp-Id	Ename	Salary
10	Mahesh	50000
12	Suresh	25000
15	Ganesh	26000
18	Mahesh	50000

Emp-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

Second Normal Form (2NF)

Minimizing Tuple Redundancy

- 2NF form will avoid the same tuples to be repeated in a table as it force all non key attributes must be full functional depends on primary key of relation
- 2NF will create a new table for each partial key with all its dependent attributes

Second Normal Form (2NF)

Consider,Relation R(A,B,C,D,E,F) and FD's as below,

A—>BC B—>DC D—>EF

i) The candidate key is {AD}-->{A,D,B,C,E,F} selected as primary key

All attributes are partially dependent on the primary key

Hence,Relation R is not in 2NF

ii) The 2NF Relation Schema is,

R1(A,B,C,D) with FDs A—> BC B—>DC

R2(D,E,F) with FDs D—>EF

Third Normal Form (3NF)

- This normal form given by E.F codd in 1971
- This normal form introduced to minimize the transitive redundancy .
- A relation is in 3NF ,if it is in 2NF and all non prime attributes of the relation are non transitively (not transitive)dependent on the every key.
- A relation R is in 3NF if all non prime attributes are
  - Full functionally dependent on primary key
  - Non transitive dependent on every key
- A relational schema R is in 3NF ,if non trivial functional dependency X→A holds true where X is superkey and A is prime attribute.

Third Normal Form (3NF)

Employee

Emp_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

Consider Employee table with following FD’s

Emp_Id—> Ename,Salary,Department_Id

Department_Id—-> Dname

Third Normal Form (3NF)

Therefore,

Candidate key{Emp_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 attribute 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.**
  - **Emp_ID—-> Department_ID**
  - **Depertment_ID—->Dname**

Third Normal Form (3NF)

- To Normalize above schema to 3NF,we can decomposed table as

Employee(Emp_Id,Ename,Salary,Department_Id)

Emp_Id—-> Ename,Salary,Department_Id

Department(Departement_Id,Dname)

Employee_Id—-> Dname

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

Department_Id	Dname
C1	IT
E2	HR

Third Normal Form (3NF)(Ex.)

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

  A—> BC B—>D, D—>EF

i) Candidate key is {A}--> {A,D,B,C,E,F}selected as primary key

- all attribute fully functionally dependent on primary key
- Relation R is in 2NF
- Non prime attribute B,C,D,E,F are transitively depends on key
- So relation R not in 3NF

ii) 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

Third Normal Form (3NF)

Minimizing Group Redundancy

- The 3NF 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 and its dependent attribute.

Boyce Codd normal form (BCNF)

- This normal form is governed by the Raymond F Boyce and E.F Codd in 1974

  Definition

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

Boyce Codd normal form (BCNF)

Ex. Employee table with FDs

E_ID—->Ename

Department_Id—> Dname,Dtype

Therefore,candidate key is {E_Id,Department_Id} is selected as primary key

E_Id	Ename	Department_Id	Dname	D_Type
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

Boyce Codd normal form (BCNF)

- As no attributes in employee table is full functionally dependent on primary .Therefore ,**Relation R is not in 2NF.**
- There should be no non-trivial functional dependencies between the attributes of the relation, where non-trivial means that the dependent attribute is not a subset of any candidate key. **.**
- **In All functional dependencies should have only candidate key on their L.H.S.**

Boyce Codd normal form (BCNF)

It is normalised into

Employee(E_Id,Ename)

E_Id—>Ename

Department(Department_Id,Dname,D_Type)

Department_Id—->Dname,D_Type

Emp_Dept(Emp_Id,Department_Id)

E_Id	Ename
10	Mahesh
12	Ganesh
13	Satish

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

E_Id	Department-Id
10	C1
12	E2
12	C1
10	E2
13	E1

Boyce Codd normal form (BCNF)

Minimizing Key Transitivity(Key Redundancy)

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

Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multivalued dependency.
- **Definition-A relation is said to in fourth normal form if each table contains no more than one multivalued dependency per key attribute**

- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multivalued dependency.

Example : **STUDENT**

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Fourth normal form (4NF)

- The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

- In the STUDENT relation, a student with STU_ID, **21** contains

  two courses, **Computer** and **Math** and

  two hobbies, **Dancing** and **Singing**.

  So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

Fourth normal form (4NF)

So to make the above table into 4NF, we can decompose it into two tables:.

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

Converting Relations to Higher Normal Form: Numerical

- If all candidate keys are simple(single attributes) then it would be in 2NF
- If all attributes of a relation are prime attributes then it would be in 3NF
- If relation is in 3NF and all candidate keys are simple then it is in BCNF

Decomposition

- If a relation is **not in the normal form** and we wish the relation to be normalized so that some of the anomalies can be eliminated, it is necessary to decompose the relation in two or more relations.
- This is a process of dividing one table into multiple tables that can be done using projection operator.
- Decomposed table can be reconstructed using join operation.
- **Desirable Properties or Goals**

  (a) Lossless join decomposition

  (b) Dependency preservation

  (c) No repetition of information

Desirable Properties or Goals of Decomposition

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 **Lossy 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.

Desirable Properties or Goals of Decomposition

Example:

- **Employee** (Employee_Id, Ename, Salary, Department_Id, Dname)

- Can be decomposed using **Lossless decomposition** as:

Employee_desc (Employee_d, Ename, Salary, Department_Id)
Department desc (Department_Id, Dname)

- **Lossy decomposition** would be as joining these tables is not possible to get back original data.

Employee_desc (Employee_Id, Ename, Salary)

Department_desc (Department_Id, Dname)

Desirable Properties or Goals of Decomposition

- **Rules for lossless decompositions are:**

1. The relations to be decomposed must have at least one common attribute in pair of relations. ie. R1**n**R2≠ NULL
2. The attributes in common must be a key for one of the relation for decomposition to be lossless.
3. Union of attributes in all decomposed relation must be equal to attributes in original relation

Desirable Properties or Goals of Decomposition

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→Y holds then we know that the two (sets) attributes are closely related or functionally dependent it would be useful if both attributes in the same relation so that the dependency can be checked easily.

Desirable Properties or Goals of Decomposition

- This can be done by maintaining functional dependency.
- Example Consider relation R(X, Y, Z, W) that has the following dependencies FD:

  $X \rightarrow Y,\ Y \rightarrow ZW$

  - 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.**

Desirable Properties or Goals of Decomposition

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.

Example-1

Consider the universal relation

R = {A, B, C, D, E, F, G, H, I, J} and

the set of functional dependencies

F = { {A, B}→{C},

{A}→{D, E},

{B}→{F},

{F}→{G,H},

{D}→{I, J} }.

What is the key(candidate key) for R?

Example 1-Solution

Example-2

Given a relation R(A, B, C, D) and

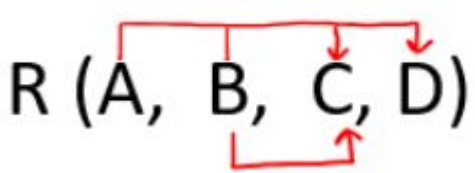
Functional Dependency set FD =

{ AB → CD, B → C },

determine whether the given R is in 2NF? If not convert it into 2 NF.

Example 1-Solution

Let us construct an arrow diagram on R using FD to calculate the candidate key.



AB is candidate key

Let us calculate the closure of AB

AB + = ABCD

Since the closure of AB contains all the attributes of R, hence **AB is Candidate Key**

Since all key will have AB as an integral part, and we have proved that AB is Candidate Key,

Hence there will be only **one candidate key AB**

Example 1-Solution

Definition of 2NF: No non-prime attribute should be partially dependent on Candidate Key

Since R has 4 attributes: - A, B, C, D, and Candidate Key is AB, Therefore, prime attributes (part of candidate key) are A and B while a non-prime attribute are C and D

a) FD: **AB → CD** satisfies the definition of 2NF, that non-prime attribute(C and D) are fully dependent on candidate key AB

b) FD: **B → C** does not satisfy the definition of 2NF, as a non-prime attribute(C) is partially dependent on candidate key AB(i.e. key should not be broken at any cost)

As FD B → C, the above table R(A, B, C, D) is not in 2NF

Convert the table R(A, B, C, D) in 2NF:

Since **FD: B → C,** our table was not in 2NF, let's decompose the table

Example 1-Solution

R1(B, C)

Since the key is AB, and from FD AB → CD, we can create R2(A, B, C, D) but this will again have a problem of partial dependency B → C, hence R2(A, B, D).

Finally, the decomposed table which is in 2NF

a) R1(B, C)

b) R2(A, B, D)

Example 2

Given a relation R(X, Y, Z, W, P) and

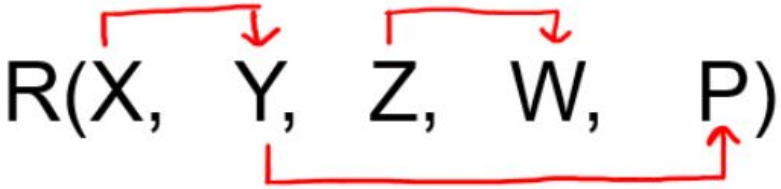
Functional Dependency set

FD = { $X \rightarrow Y$, $Y \rightarrow P$, and $Z \rightarrow W$},

determine whether the given R is in 3NF? If not convert it into 3 NF.

Example 2-Solution

Let us construct an arrow diagram on R using FD to calculate the candidate key.



Let us calculate the closure of XZ

XZ + = XZYPW (from the closure method that we studied earlier)

Since the closure of XZ contains all the attributes of R, hence **XZ is Candidate Key**

Since all key will have XZ as an integral part, and we have proved that XZ is Candidate Key,

Example 2-Solution

Definition of 3NF: First it should be in 2NF and if there exists a non-trivial dependency between two sets of attributes X and Y such that X → Y (i.e., Y is not a subset of X) then

a. Either X is Super Key
b. Or Y is a prime attribute.

Since R has 5 attributes: - X, Y, Z, W, P and Candidate Key is XZ, Therefore, prime attribute (part of candidate key) are X and Z while a non-prime attribute are Y, W, and P

Given FD are X → Y, Y → P, and Z → W and Super Key / Candidate Key is XZ

a. FD: **X → Y** does not satisfy the definition of 3NF, that neither X is Super Key nor Y is a prime attribute.
b. FD: **Y → P** does not satisfy the definition of 3NF, that neither Y is Super Key nor P is a prime attribute.
c. FD: **Z → W** satisfies the definition of 3NF, that neither Z is Super Key nor W is a prime attribute.

Example 2-Solution

Convert the table R(X, Y, Z, W, P) into 3NF:

Since all the FD = { X → Y, Y → P, and Z → W} were not in 3NF, let us convert R in 3NF

R1(X, Y) {Using FD X → Y}

R2(Y, P) {Using FD Y → P}

R3(Z, W) {Using FD Z → W}

And create one table for Candidate Key XZ

R4(X, Z) { Using Candidate Key XZ }

All the decomposed tables R1, R2, R3, and R4 are in 2NF(as there is no partial dependency) as well as in 3NF.

Hence decomposed tables are:

R1(X, Y), R2(Y, P), R3(Z, W), and **R4(X, Z)**

9.11 Converting Relational Schema to Higher Normal Forms

Example 9.11.1 : Relation R(A, B, C, D, E, F, G, H, I, J). Having following set of FD, show convert table to highest normal form.

$AB \rightarrow C$, $C \rightarrow EF$, $AD \rightarrow GH$, $G \rightarrow I$, $H \rightarrow J$

Solution : **(10 Marks)**

a) 1NF

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

b) 2 NF

$\{A, B, D\}^+ \rightarrow \{\underline{A, B, D}, C, E, F, G, H, I, J\}$

{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 ($\underline{A, B}$, C, E, F); $AB \rightarrow C$, $C \rightarrow EF$

R_2 ($\underline{A, D}$, C, E, F); $AD \rightarrow GH$, $G \rightarrow I$, $H \rightarrow J$

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} ($\underline{A, B}$, C); $AB \rightarrow C$

R_{1a} ($\underline{C}$, E, F); $C \rightarrow EF$

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} ($\underline{A, D}$, G, H); $AD \rightarrow GH$

R_{2b} ($\underline{G}$, I); $G \rightarrow I$

R_{2c} ($\underline{H}$, J); $H \rightarrow J$

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

Practices Questions

1. Explain pitfalls in relational database design
2. Explain design guidelines for relational schema
3. What is normalization?Explain 1NF,2NF,3NF, and BCNF with examples.
4. Define normalization.Discuss different normalization techniques with examples.
5. Define Normalization. What is the use of Normal forms?
6. How can you identify the relations is in 1NF or 2NF give suitable examples
7. List and explain different types of Functional dependency.
8. List and explain Armstrong axioms with examples
9. List all functional dependencies satisfied by relation

a	b	c
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c3

1. 73. Consider given relation R(A,B,C,D,E,F) having set of FD's: A→B, A→C, BC→D, B→E,

   BC→F, AC→F

   Calculate some members of axioms as below:

   A→E, BC→DF, AC→D, AC→DF
2. Consider given relation R(A,B,C,D,E,F) having set of FD's: A→B, A→C, C→D, B→E, AC→F

   Calculate attribute closures {A}+, (B}+ and {AC}+ along with all possible candidate keys
3. Relation (A,B,C,D,E,F,G,H,I,J) having following set of FD's show and convert to highest normal form

   AB→C, C→EF, AD→GH, G →I, H→J\
4. Normalize R(A,B,C,D,E,F,G) upto 3NF with following FD's:

   AB→CDEFG, C→B, A→D, E→G