3.1 DATABASE DESIGN

The overall design of the database is called the database schema. Database system have several schemas, partitioned according to the level of abstraction.

- Physical schema
- Logical schema
- View schema

The relational schema face the several undesirable problems.

1. **Redundancy**: The aim of the database system is to reduce redundancy, meaning that data is to be stored only once. Storing the data/information many times leads to the wastage of storage space and an increase in the total size of the data stored.

   e.g., The major and phone no. of Vijay stored many times in the database. Thus it is the example of redundancy of data in the database.

2. **Update Anomalies**: Multiple copies of the same fact may lead to update anomalies or inconsistencies. When an update is made and only some of the multiple copies are updated.

   Thus, a change in the phone no. of ‘Vijay’ must be made for consistency, in all tuples pertaining to the student ‘Vijay’. If one-three tuples in table is not change to reflect the new phone-no. of ‘Vijay’, there will be an inconsistency in the data.

   e.g., In the given table the phone-no. of Vijay in all three row is 2374539 if we update the phone-no. of Vijay two rows. Then database will be inconsistence.

3. **Insertion Anomalies**: If this is the only relation in the database showing the association
between a faculty member and the course he or she teaches, the fact that a given professor is teaching in a given course cannot be entered in the database unless a student is registered in the course.

(4) **Deletion Anomalies**: If the only student registered in a given course discontinues the course, the information as to which professor is offering the course will be lost if this is the only relation in the database showing the association between a faculty member and the course she or he teaches.

*Example:*

<table>
<thead>
<tr>
<th>Roll-no.</th>
<th>Name</th>
<th>Course</th>
<th>Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Vijay</td>
<td>DBA</td>
<td>15000</td>
</tr>
<tr>
<td>11</td>
<td>Santosh</td>
<td>VB. Net</td>
<td>5000</td>
</tr>
<tr>
<td>12</td>
<td>Gopal</td>
<td>VC++</td>
<td>8000</td>
</tr>
<tr>
<td>13</td>
<td>Sanjay</td>
<td>Java</td>
<td>7000</td>
</tr>
</tbody>
</table>

Here, we cannot delete the particular attribute Roll-no. = 11, because after this we cannot access the course, name and fee of that student due to lossing of whole information.

### 3.2 Decomposition

The decomposition of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is its replacement by a set of relation schemes $\{R_1, R_2, ..., R_m\}$ such that $R_i \subseteq R$, $1 \leq i \leq m$ and $R_1 \cup R_2 \cup R_3 \cup ... \cup R_m = R$.

A relation scheme $R$ can be decomposed into a collection of relation schemes $\{R_1, R_2, R_3, ..., R_m\}$ to eliminate some of the anomalies contained in the original relation $R$.

The problems in the relation scheme student can be resolved, if we decompose it with the following relation schemes: such as:

- **Student-INFO** (Name, Phone no. Major)
- **Transcript** (Name, Course, Grade)
- **Teacher** (Course Professor)

These decomposition are bad for the following reasons:

(i) Redundancy and update anomaly, because the data for the attributes phone no. and major are repeated.

(ii) Loss of information, because we lose the fact that a student has a given grade in a particular course.

### 3.3 Universal Relation

Let us consider the problem of designing a database. Such a design will be required to represent a finite number of entity sets. Each entity set will be represented by a number of its attributes. If we refer to the set of all attributes as the universal scheme $U$ then a relation $R(U)$ is called the universal relation.

The universal relation is a single relation made up of all the attributes in the database.

### 3.4 Functional Dependency

A functional dependency exists when the value of one thing is fully determined by another.

*Example:* Given the relation

EMP (Emp No, Emp Name, Salary), attribute EmpName is functionally dependent on attribute Emp No.
If we know Emp No, we also know the Emp Name. It is represented by

\[
\text{EMP No} \rightarrow \text{Emp Name}.
\]

A Functional dependency is a constraint between two sets of attributes in a relation from a database.

**Types of Functional Dependency:**

1. **Trivial FD**: A functional dependency of the form \(X \rightarrow Y\) is trivial if \(Y \subseteq X\).
2. **Full Functional Dependency**: A FD \(X \rightarrow Y\) is a full functional dependency if removal of any attribute \(A\) from \(X\) means that the dependency does not hold any more. That is,
   
   **Example Full FD**:
   
   for any attribute \(A \in X, (X - \{A\})\) does not functionally determine \(Y\).
   
   \((X - \{A\}) \rightarrow Y\) is called full functional dependency.
3. **Partial FD**: A FD \(X \rightarrow Y\) is a partial dependency if some attribute \(A \in X\) can be removed from \(X\) and then the dependency still hold.
   
   **Example** :
   
   for some \(A \in X, (X - \{A\}) \rightarrow Y\), then it is called partial dependency.
4. **Transitive Dependency**: A functional dependency \(X \rightarrow Y\) in a relation scheme \(R\) is a transitive dependence if there is a set of attributes \(Z\) that is neither a candidate key nor a subset of any key of \(R\) and both \(X \rightarrow Z\) and \(Z \rightarrow Y\) hold.
   
   **Ex** :
   
   The dependency \(SSN \rightarrow DMGRSSN\) is transitive through
   
   \(SSN \rightarrow DNUM\) and \(DNUM \rightarrow DMGRSSN\)
5. **Multivalued Dependency**: Let \(R\) be a relation schema and let \(\alpha \subseteq R\) and \(\beta \subseteq R\).
   
   The multivalued dependency \(\alpha \rightarrow \beta\) hold on \(R\) if any legal relation \(r\ (R)\), for all pairs of tuples \(t_1\) and \(t_2\) in \(r\) s.t. \(t_1[\alpha] = t_2[\alpha]\), there exist tuples \(t_3\) and \(t_4\) in \(r\) s.t.
   
   \[
   t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha] \\
   t_3[\beta] = t_1[\beta] \\
   t_3[R - \beta] = t_2[R - \beta] \\
   t_4[\beta] = t_2[\beta]
   \]
\[ t_4 [R - \beta] = t_1 [R - \beta] \]

*Example*: A table with schema (name, address, car)

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vijay</td>
<td>Noida</td>
<td>Toyota</td>
</tr>
<tr>
<td>Vijay</td>
<td>G. Noida</td>
<td>Honda</td>
</tr>
<tr>
<td>Vijay</td>
<td>Noida</td>
<td>Honda</td>
</tr>
<tr>
<td>Vijay</td>
<td>G. Noida</td>
<td>Toyota</td>
</tr>
</tbody>
</table>

*(name, address, car)* where

- name \(\rightarrow\) address
- name \(\rightarrow\) car

### 3.5 PRIME ATTRIBUTE

An attribute of relation schema \(R\) is called a prime attribute of \(R\) if it is a member of some candidate key of \(R\).

*For Ex.*: Work-on

<table>
<thead>
<tr>
<th>SSN</th>
<th>PNUMBER</th>
<th>HOURS</th>
</tr>
</thead>
</table>

Both SSN and PNUMBER are prime attributes of work-on.

#### 3.5.1 Non Prime Attribute

An attribute of relation schema \(R\) is called non prime attribute if it is not a member of any candidate key.

*e.g.*, Hours is non prime attribute of work-on.

### 3.6 ARMSTRONG'S AXIOMS

There are following rules that logically implied functional dependency.

Suppose \(X, Y, Z\) denotes sets of attributes over a relational schema. Then the rules are:

1. **Reflexivity**: If \(X\) is a set of attributes and \(Y \subseteq X\)
   
   Then
   
   \(X \rightarrow Y\) holds

2. **Augmentation Rule**:
   
   If \(X \rightarrow Y\) holds and \(Z\) is a set of attributes, then
   
   \(ZX \rightarrow ZY\) holds

3. **Transitivity Rule**: If \(X \rightarrow Y\) and \(Y \rightarrow Z\) then
   
   \(X \rightarrow Z\) hold

These rules are called Armstrong’s axioms.

There are some additional rules are:

4. **Union rule**: If \(X \rightarrow Y\) holds and \(X \rightarrow Z\) holds
   
   then
   
   \(X \rightarrow YZ\) holds

5. **Decomposition rule**: if \(X \rightarrow YZ\) holds,
   
   then
   
   \(X \rightarrow Y\) holds and \(X \rightarrow Z\) holds

6. **Pseudo transitivity rule**: If \(X \rightarrow Y\) holds and \(ZY \rightarrow W\) holds
   
   then
   
   \(XZ \rightarrow W\) holds
For Example, a schema $R = (A, B, C, G, H, I)$ and functional dependency are $\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ we can list several members of $P^+$. 

(i) Since $A \rightarrow B$ and $B \rightarrow H$ Applying transitivity
We get $A \rightarrow H$ holds

(ii) :: $CG \rightarrow I$, and $CG \rightarrow H$ Applying union rule
We get $CG \rightarrow HI$ holds

(iii) :: $A \rightarrow C$ and $CG \rightarrow I$ Applying pseudo transitivity
We get $AG \rightarrow I$ holds

3.7 CLOSURE OF A SET OF FUNCTIONAL DEPENDENCIES

The set of functional dependencies that is logically implied by $F$ is called the closure of $F$ and is written as $F^+$

**Algorithm**: Algorithm to compute the closure of $X$.

Let $X^+$ to all the attributes in $X$.

Determining $X^+$, the closure of $X$ under $F$.

\[
X^+ := X;
\]

repeat

old $X^+$ := $X^+$;

for each functional dependency $Y \rightarrow Z$ in $F$ do

if $Y \subseteq X^+$ then

$X^+ := X^+ \cup Z$;

until ($X^+ = \text{old} X^+$);

**Example**: Let $R = \{A, B, C, D, E, F\}$ and a set of FDs.

$A \rightarrow BC, E \rightarrow CF, B \rightarrow E, CD \rightarrow EF, F \rightarrow D$

Compute the closure of a set of attribute $\{A, B\}$ under the given set of FDs.

**Solution**: Let $X = \{A, B\}$

Now initialization of $X^+$

\[
\therefore \quad X^+ := X
\]

\[
\therefore \quad X^+ = \{A, B\}
\]

For $A \rightarrow BC$, :: $A \subseteq X^+$ then

\[
X^+ = \{A, B\} \cup \{B, C\}
\]

\[
X^+ = \{A, B, C\}
\]

For $B \rightarrow E$, :: $B \subseteq X^+$ then

\[
X^+ = X^+ \cup \{E\}
\]

\[
\therefore = \{A, B, C\} \cup \{E\}
\]

\[
X^+ = \{A, B, C, E\}
\]

For $E \rightarrow CF$, :: $E \subseteq X^+$
\[ X^+ = \{A, B, C, E\} \cup \{C, F\} \]
\[ = \{A, B, C, E, F\} \]

For \( F \rightarrow D \), \( F \subseteq X^+ \), then
\[ X^+ = \{A, B, C, E, F\} \cup \{D\} \]
\[ = \{A, B, C, D, E, F\} \]
\[ \therefore \]
\[ X^+ = \{A, B, C, D, E, F\} \]

**Example**: Let \( X = BCD \) and \( F = \{A \rightarrow BC, \ CD \rightarrow E, \ E \rightarrow C, \ D \rightarrow AEH, \ ABH \rightarrow BD, \ DH \rightarrow BC\} \)

Compute the closure \( X^+ \) of \( X \) under \( F \).
\[ \therefore \]
\[ R = \{A, B, C, D, E, H\} \]

**Solution**: We initialize \( X^+ \) to \( X \)
\[ \therefore \]
\[ X^+ = X, \]
\[ \therefore \]
\[ X^+ = \{B, C, D\} \]

For \( CD \rightarrow E \), \( CD \subseteq X^+ \)
\[ \therefore X^+ = X^+ \cup \{E\} = \{B, C, D\} \cup \{E\} = \{B, C, D, E\} \]

For \( D \rightarrow AEH \), \( D \subseteq X^+ \)
\[ \therefore \]
\[ X^+ = X^+ \cup \{A, E, H\} \]
\[ X^+ = \{A, B, C, D, E, H\} \]

Now \( X^+ \) cannot be augmented any further. Because
\[ X^+ = \{A, B, C, D, E, H\} = R = \{A, B, C, D, E, H\} \]

Thus
\[ X^+ = \{A, B, C, D, E, H\} \]

### 3.8 NON REDUNDANT COVERS

**Algorithm**: Input: A set of FDS \( F \)

**Output**: A non redundant cover of \( F \)

- \( G = F \) (Initialize \( G \) to \( F \))
- for each \( FD \ X \rightarrow Y \) in \( G \)
  - do
    - if \( X \rightarrow Y \in \{ F - (X \rightarrow Y) \}^+ \)
      - then \( F = \{F - (X \rightarrow Y)\} \)
      - \( G = F \) (\( G \) is the non redundant cover of \( F \))
  - end;

**Example**: If \( F = \{A \rightarrow BC, \ CD \rightarrow E, \ E \rightarrow C, \ D \rightarrow AEH, \ ABH \rightarrow BD, \ DH \rightarrow BC\} \). Then find the non-redundant cover for \( F \).

**Solution**: We find that \( (A)^+ \) under \( \{F - (A \rightarrow BC)\} \)

Let \( X = A \) \( : R = \{A, B, C, D, E, H\} \)
\[ \therefore \]
\[ X^+ = X \]
\[ \therefore \]
\[ X^+ = \{A\} \]

For all FDS, \( (A)^+ = \{A\} \neq \{A, B, C, D, E, H\} \)

Thus \( A \rightarrow BC \) is non redundant.

Now for \( CD \rightarrow E \), we find \( (CD)^+ \) under
\( \{ F - (CD \to E) \} \)

Now  
\( X^+ = \{ C, D \} \)

For  \( D \to AEH \), ∴  \( D \subseteq X^+ \) 
\[ X^+ = X^+ \cup \{ A, E, H \} \]
= \{ C, D \} \cup \{ A, E, H \}
\[ X^+ = \{ A, C, D, E, H \} \]

For  \( A \to BC \), ∴  \( A \subseteq X^+ \) 
\[ X^+ = X^+ \cup \{ B, C \} \]
= \{ A, B, C, D, E, H \} = R = \{ A, B, C, D, E, H \}

Thus  \( CD \to E \) is redundant so it is removed.

Now for  \( DH \to BC \), we find  \((DH)^+\) under  \( \{ F - (DH \to BC) \} \)
\[ X^+ = \{ D, H \} \]

\( D \to AEH \), ∴  \( D \subseteq X^+ \) 
\[ X^+ = X^+ \cup \{ A, E, H \} \]
\[ X^+ = \{ A, D, E, H \} \]

\( A \to BC \), ∴  \( A \subseteq X^+ \) 
\[ X^+ = X^+ \cup \{ B, C \} \]
\[ X^+ = \{ A, B, C, D, E, H \} = R = \{ A, B, C, D, E, H \} \]

Thus  \( DH \to BC \) is redundant removed it.

No remaining FDS can be from the modified  \( F \).

Thus a non redundant cover for  \( F \) is  \( \{ A \to BC, E \to C, D \to AEH, ABH \to BD \} \)

### 3.9 CANONICAL COVER OR MINIMAL SET OF FD'S

A minimal cover of a set of functional dependencies  \( E \) is a set of functional dependencies  \( F \) that satisfies the property that every dependency in  \( E \) is in the closure  \( F^+ \) of  \( F \).

A set of functional dependencies  \( F \) to be minimal if it satisfies the following condition.

(i) Every dependency in  \( F \) has a single attribute for its right-hand side.

(ii) We cannot replace any dependency  \( X \to A \) in  \( F \) with a dependency  \( Y \to A \), where  \( Y \) is a proper subset of  \( X \) and still have a set of dependencies that is equivalent to  \( F \).

(iii) We cannot remove any dependency from  \( F \) and still have a set of dependencies that is equivalent to  \( F \).

We can think of a minimal set of dependencies as being a set of dependencies in canonical form and with no redundancies.

A canonical cover is sometimes called minimal.

**Example** : If  \( F = \{ A \to BC, CD \to E, E \to C, D \to AEH, ABH \to BD, DH \to BC \} \)

Find the canonical cover.

**Solution** : First of all find the non redundant cover  
\[ \{ A \to BC, E \to C, D \to AEH, ABH \to BD \} \]

Because  \( CD \to E \) and  \( DH \to BC \) are redundant FDS.

Now the FD  \( ABH \to BD \) can be decomposed into the FDS
Similarly $A \rightarrow BC$ decompose into

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

Since $A \rightarrow B$ is in $F$, we can left reduce the decomposition $ABH \rightarrow B$ and $ABH \rightarrow D$ into

$AH \rightarrow B$ and $AH \rightarrow D$

Now we also notice that $AH \rightarrow B$ is redundant because $A \rightarrow B$ is already in $F$.

Now we decompose the FD $D \rightarrow AEH$ into the FDS

$D \rightarrow A, D \rightarrow E, D \rightarrow H$

Thus the canonical over is

$\{A \rightarrow B, A \rightarrow C, E \rightarrow C, D \rightarrow A, D \rightarrow E, D \rightarrow H, AH \rightarrow D\}$

### 3.10 NORMALIZATION

The goal of a relational database design is to generate a set of relation schemas that allows us to store information without any redundant (repeated) data. It also allows us to retrieve information easily and more efficiently.

For this we use a approach normal form as the set of rules. These rules and regulations are known as Normalization.

Database normalization is data design and organization process applied to data structures based on their functional dependencies and primary keys that help build relational databases.

**Normalization Helps:**

- Minimizing data redundancy.
- Minimizing the insertion, deletion and update anomalies.
- Reduces input and output delays
- Reducing memory usage.
- Supports a single consistent version of the truth.
- It is an industry best method of tables or entity design.

**Uses:** Database normalization is a useful tool for requirements analysis and data modelling process of software development. Thus

The normalization is the process to reduce the all undesirable problems by using the functional dependencies and keys.

Here we use the following normal forms:

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal form (3NF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)
- Sixth Normal Form (6NF)
3.10.1 First Normal Form (1NF)

A relation schema $R$ is said to be in First Normal Form (1NF) if the values in the domain of each attribute of the relation are atomic.

*For Ex.*: Course-INFO

<table>
<thead>
<tr>
<th>Fact-Dept</th>
<th>Professor</th>
<th>Course</th>
<th>Course-Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp-Sci</td>
<td>Vijay</td>
<td>353</td>
<td>Comp Sci</td>
</tr>
<tr>
<td></td>
<td></td>
<td>370</td>
<td>Comp Sci</td>
</tr>
<tr>
<td></td>
<td></td>
<td>310</td>
<td>Physics</td>
</tr>
<tr>
<td></td>
<td>Santosh</td>
<td>353</td>
<td>Comp Sci</td>
</tr>
<tr>
<td></td>
<td></td>
<td>320</td>
<td>Comp Sci</td>
</tr>
<tr>
<td></td>
<td></td>
<td>370</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Chemistry</td>
<td>Gopal</td>
<td>456</td>
<td>Chemistry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>410</td>
<td>Mathematics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>370</td>
<td>Comp Sci</td>
</tr>
</tbody>
</table>

*In 1NF*: Course INFO

<table>
<thead>
<tr>
<th>Professor</th>
<th>Course</th>
<th>Fact-Dept</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vijay</td>
<td>353</td>
<td>Comp Sci</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Vijay</td>
<td>370</td>
<td>Comp Sci</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Vijay</td>
<td>310</td>
<td>Comp Sci</td>
<td>Physics</td>
</tr>
<tr>
<td>Santosh</td>
<td>353</td>
<td>Comp Sci</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Santosh</td>
<td>320</td>
<td>Comp Sci</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Santosh</td>
<td>370</td>
<td>Comp Sci</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>Gopal</td>
<td>456</td>
<td>Chemistry</td>
<td>Chemistry</td>
</tr>
<tr>
<td>Gopal</td>
<td>410</td>
<td>Chemistry</td>
<td>Mathematics</td>
</tr>
<tr>
<td>Gopal</td>
<td>370</td>
<td>Chemistry</td>
<td>Comp Sci</td>
</tr>
</tbody>
</table>

3.10.2 Second Normal Form (2NF)

- 2NF is a normal form in database normalization. It requires that all data elements in a table are full functionally dependent on the table's primary key.
- If data element only dependent on part of primary key, then they are parsed out to separate tables.
• If the table has a single field as the primary key, it is automatically in 2NF.

A table is in 2NF if and only if
(i) It is in 1NF
(ii) Each non primary key attribute is full functionally dependent on the primary key.

Example 1: EMP-PROJ

```
SSN   ENAME
FD1

\{SSN, PNUMBER\} is a primary key and Hours is non key attribute.
(\{SSN, PNUMBER\} \rightarrow \text{HOURS})
Thus HOURS is full FDS on primary key (\{SSN, PNUMBER\})
```

EP2

```
SSN \rightarrow ENAME
```

\cdot SSN is primary key and NAME is a non key attribute
Since non key attribute ENAME is full FDS on primary key attribute SSN. Thus, it is in 2NF.

EP3

```
PNUMBER \rightarrow \{PNAME, PLOCATION\}
```
Example 2:

**TEACHER**:

<table>
<thead>
<tr>
<th>Course</th>
<th>Prof</th>
<th>Room</th>
<th>Room-Cap</th>
<th>Enrol–Unt</th>
</tr>
</thead>
<tbody>
<tr>
<td>353</td>
<td>Vijay</td>
<td>A532</td>
<td>45</td>
<td>40</td>
</tr>
<tr>
<td>351</td>
<td>Vijay</td>
<td>C320</td>
<td>100</td>
<td>60</td>
</tr>
<tr>
<td>355</td>
<td>Santosh</td>
<td>H940</td>
<td>50</td>
<td>45</td>
</tr>
<tr>
<td>456</td>
<td>Santosh</td>
<td>B278</td>
<td>50</td>
<td>45</td>
</tr>
<tr>
<td>459</td>
<td>Gopal</td>
<td>D110</td>
<td>300</td>
<td>200</td>
</tr>
</tbody>
</table>

**TEACHER Relation in Second Normal Form**

<table>
<thead>
<tr>
<th>Course</th>
<th>Prof</th>
<th>Enrol–Unt</th>
</tr>
</thead>
<tbody>
<tr>
<td>353</td>
<td>Vijay</td>
<td>40</td>
</tr>
<tr>
<td>351</td>
<td>Vijay</td>
<td>60</td>
</tr>
<tr>
<td>355</td>
<td>Santosh</td>
<td>45</td>
</tr>
<tr>
<td>456</td>
<td>Santosh</td>
<td>45</td>
</tr>
<tr>
<td>459</td>
<td>Gopal</td>
<td>200</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>Course</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>353</td>
<td>A532</td>
</tr>
<tr>
<td>351</td>
<td>C320</td>
</tr>
<tr>
<td>355</td>
<td>H940</td>
</tr>
<tr>
<td>456</td>
<td>B278</td>
</tr>
<tr>
<td>459</td>
<td>D110</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>Room</th>
<th>Room-Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>A532</td>
<td>45</td>
</tr>
<tr>
<td>C320</td>
<td>100</td>
</tr>
<tr>
<td>H940</td>
<td>50</td>
</tr>
<tr>
<td>B278</td>
<td>50</td>
</tr>
<tr>
<td>D110</td>
<td>300</td>
</tr>
</tbody>
</table>

(c)

3.10.3 Third Normal Form (3NF)

The 3NF is a normal form used in database normalization to check if all the non key attributes of a relation depend only on the candidate keys of the relation.

This means that all non-key attributes are mutually independent or in other words that a non key attribute cannot be transitively dependent on another non-key attribute.

A relation schema $R$ is in 3NF if every non prime attribute of $R$ meets both of the following.

- It is full functionally dependency on every key of $R$.
- It is non transitively dependent on every key of $R$. 


OR we can say: A relation schema $R$ is in 3NF if, whenever a non trivial functional dependency $X \rightarrow A$ holds in $R$.

Either
(a) $X$ is a super key of $R$. OR
(b) $A$ is a prime attribute of $R$.

Example 1: EMP-DEP
The dependency SSN → DMGRSSN is transitive through the FDS

$SSN \rightarrow DNUM$ and $DNUM \rightarrow DMGRSSN$

Thus EMP-DEP is not in 3NF because of the transitive dependency of DMGRSSN on SSN via DNUM.

We can normalize EMP-DEP by decompose into two 3NF said $ED_1$ and $ED_2$ respectively. Hence in 3NF:

(i) $ED_1$
(ii) $ED_2$

Example: Student relation

<table>
<thead>
<tr>
<th>Roll_No</th>
<th>Name</th>
<th>Dept</th>
<th>Year</th>
<th>Hostel_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1784</td>
<td>Raman</td>
<td>Physics</td>
<td>1</td>
<td>Ganga</td>
</tr>
<tr>
<td>1648</td>
<td>Krishnan</td>
<td>Chemistry</td>
<td>1</td>
<td>Ganga</td>
</tr>
<tr>
<td>1768</td>
<td>Gopal</td>
<td>Maths</td>
<td>2</td>
<td>Kaveri</td>
</tr>
<tr>
<td>1848</td>
<td>Raja</td>
<td>Botany</td>
<td>2</td>
<td>Kaveri</td>
</tr>
<tr>
<td>1682</td>
<td>Maya</td>
<td>Geology</td>
<td>3</td>
<td>Krishna</td>
</tr>
<tr>
<td>1485</td>
<td>Singh</td>
<td>Zoology</td>
<td>4</td>
<td>Godavari</td>
</tr>
</tbody>
</table>

Here the dependency Roll-No → HOSTAL NAME is transitive through
ROLL-NO → YEAR AND YEAR → HOSTAL NAME

Thus the student Relation is not 3NF.

So we can normalize student Relation by decomposition into two 3NF, STUD1 AND STUD2 respectively.
(i) STUD1 relation

<table>
<thead>
<tr>
<th>Roll-No</th>
<th>Name</th>
<th>Dept</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1784</td>
<td>Raman</td>
<td>Physics</td>
<td>1</td>
</tr>
<tr>
<td>1648</td>
<td>Krishnan</td>
<td>Chemistry</td>
<td>1</td>
</tr>
<tr>
<td>1768</td>
<td>Gopal</td>
<td>Maths</td>
<td>2</td>
</tr>
<tr>
<td>1848</td>
<td>Raja</td>
<td>Botany</td>
<td>2</td>
</tr>
<tr>
<td>1682</td>
<td>Maya</td>
<td>Geology</td>
<td>3</td>
</tr>
<tr>
<td>1485</td>
<td>Singh</td>
<td>Zoology</td>
<td>4</td>
</tr>
</tbody>
</table>

(ii) STUD2 relation

<table>
<thead>
<tr>
<th>Year</th>
<th>Hostel_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ganga</td>
</tr>
<tr>
<td>2</td>
<td>Kaveri</td>
</tr>
<tr>
<td>3</td>
<td>Krishna</td>
</tr>
<tr>
<td>4</td>
<td>Godavari</td>
</tr>
</tbody>
</table>

3.10.4 Boyce-Codd Normal Form (BCNF)

BCNF is a normal form used in database normalization. It is slightly stronger version of the 3NF. A table is in BCNF if and only if:

(a) It is in 3NF and
(b) For every of its nontrivial functional dependency \( X \rightarrow Y \), \( X \) is a super key.

OR A relation schema \( R \) is in BCNF if whenever a nontrivial functional dependency \( X \rightarrow A \) holds in \( R \) then

(1) \( X \) is a super key of \( R \).

Ex: Student

<table>
<thead>
<tr>
<th>Stud_Id</th>
<th>SName</th>
<th>Subject</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Vijay</td>
<td>Computer</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>Vijay</td>
<td>Physics</td>
<td>B</td>
</tr>
<tr>
<td>10</td>
<td>Vijay</td>
<td>Maths</td>
<td>B</td>
</tr>
<tr>
<td>20</td>
<td>Gopal</td>
<td>Computer</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>Gopal</td>
<td>Physics</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>Gopal</td>
<td>Maths</td>
<td>C</td>
</tr>
</tbody>
</table>

There are two candidate keys (Stud_Id, Subject) and (SName, Subject)

In the above relation following FDS are exist:

\( SName, Subject \rightarrow Grade \)

\( Stud_Id, Subject \rightarrow Grade \)

\( Stud_Id \rightarrow SName \)
Now BCNF decompose $R$ into $R_1$ and $R_2$.

$R_1$

<table>
<thead>
<tr>
<th>Stud_Id</th>
<th>Subject</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Computer</td>
<td>A</td>
</tr>
<tr>
<td>10</td>
<td>Physics</td>
<td>B</td>
</tr>
<tr>
<td>10</td>
<td>Maths</td>
<td>B</td>
</tr>
<tr>
<td>20</td>
<td>Computer</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>Physics</td>
<td>A</td>
</tr>
<tr>
<td>20</td>
<td>Maths</td>
<td>C</td>
</tr>
</tbody>
</table>

$R_2$

<table>
<thead>
<tr>
<th>Stud_Id</th>
<th>SName</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Vijay</td>
</tr>
<tr>
<td>20</td>
<td>Gopal</td>
</tr>
</tbody>
</table>

Ex. 2: Normalize the relation professor so as it is in BCNF.

### PROFESSOR

<table>
<thead>
<tr>
<th>Prof Code</th>
<th>Department</th>
<th>HOD</th>
<th>Percent Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Physics</td>
<td>Ghosh</td>
<td>50</td>
</tr>
<tr>
<td>P1</td>
<td>Maths</td>
<td>Krishnan</td>
<td>50</td>
</tr>
<tr>
<td>P2</td>
<td>Chemistry</td>
<td>Rao</td>
<td>25</td>
</tr>
<tr>
<td>P2</td>
<td>Physics</td>
<td>Ghosh</td>
<td>75</td>
</tr>
<tr>
<td>P3</td>
<td>Maths</td>
<td>Krishnan</td>
<td>100</td>
</tr>
<tr>
<td>P4</td>
<td>Maths</td>
<td>Krishnan</td>
<td>30</td>
</tr>
<tr>
<td>P4</td>
<td>Physics</td>
<td>Ghosh</td>
<td>70</td>
</tr>
</tbody>
</table>

The FDS are

Prof Code, Department → Percent time
Department → HOD

The PROFESSOR Relation decompose into two relation PROF 1 and PROF 2 respectively.

### PROF1

<table>
<thead>
<tr>
<th>Professor Code</th>
<th>Department</th>
<th>Percent Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Physics</td>
<td>50</td>
</tr>
<tr>
<td>P1</td>
<td>Maths</td>
<td>50</td>
</tr>
<tr>
<td>P2</td>
<td>Chemistry</td>
<td>25</td>
</tr>
<tr>
<td>P2</td>
<td>Physics</td>
<td>75</td>
</tr>
<tr>
<td>P3</td>
<td>Maths</td>
<td>100</td>
</tr>
<tr>
<td>P4</td>
<td>Maths</td>
<td>30</td>
</tr>
<tr>
<td>P4</td>
<td>Physics</td>
<td>70</td>
</tr>
</tbody>
</table>
Note: Every relation in BCNF is also in 3NF, but a relation is 3NF is not necessarily in BCNF e.g., The above relations are in BCNF and 3NF also.

### TEACH

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Narayan</td>
<td>Database</td>
<td>Pallaw</td>
</tr>
<tr>
<td>Vijay</td>
<td>Database</td>
<td>Navathe</td>
</tr>
<tr>
<td>Vijay</td>
<td>OS</td>
<td>Galvin</td>
</tr>
<tr>
<td>Vijay</td>
<td>Computer</td>
<td>Gopal</td>
</tr>
<tr>
<td>Santosh</td>
<td>OS</td>
<td>Ahmad</td>
</tr>
<tr>
<td>Santosh</td>
<td>Database</td>
<td>Pallaw</td>
</tr>
</tbody>
</table>

The dependencies show as

\[
\{\text{STUDENT, COURSE}\} \rightarrow \text{INSTRUCTOR}
\]

\[
\text{INSTRUCTOR} \rightarrow \text{COURSE}
\]

But the TEACH Relation is in 3NF but not in BCNF.

### 3.10.5 Fourth Normal Form (4NF)

- An entity type is in 4NF if it is BCNF and there are non multivalued dependencies between its attribute types.
- Any entity is BCNF is transformed in 4NF
  (i) Direct any multivalued dependencies.
  (ii) Decompose entity type.
Example:

<table>
<thead>
<tr>
<th>Author_No</th>
<th>Book_No</th>
<th>Subject</th>
<th>Book_Title</th>
<th>Author_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>Comp Sci</td>
<td>Methods</td>
<td>Vijay</td>
</tr>
<tr>
<td>A1</td>
<td>B1</td>
<td>Maths</td>
<td>Methods</td>
<td>Vijay</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
<td>Comp Sci</td>
<td>Methods</td>
<td>Gopal</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
<td>Maths</td>
<td>Methods</td>
<td>Gopal</td>
</tr>
<tr>
<td>A1</td>
<td>B2</td>
<td>Maths</td>
<td>Calculus</td>
<td>Santosh</td>
</tr>
</tbody>
</table>

IN BCNF

AUTHOR (Author-No, Author-Name)

BOOK (Book-No, Book-title)

AUTHOR-BOOK-SUBJECT (Author-No, Book-No, Subject)

- Example models that “each AUTHOR is associated with all the SUBJECTS under which the Book is classified.
- The attribute SUBJECT contains redundant values. If SUBJECT were delete from row 1 and 2 the value could be deduced from row 3 and 4.

Multivalued dependencies:

<table>
<thead>
<tr>
<th>Author_No</th>
<th>Book_No</th>
<th>Subject</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>A1</td>
<td>B1</td>
<td>Maths</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
<td>Comp Sci</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
<td>Maths</td>
</tr>
<tr>
<td>A1</td>
<td>B2</td>
<td>Maths</td>
</tr>
</tbody>
</table>

Now The 4th NF will be

<table>
<thead>
<tr>
<th>Author_No</th>
<th>Book_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>B1</td>
</tr>
<tr>
<td>A2</td>
<td>B1</td>
</tr>
<tr>
<td>A1</td>
<td>B2</td>
</tr>
</tbody>
</table>

AUTHOR (Author-No, Author-Name)

BOOK (Book-No, Book-Title)

AUTHOR-BOOK (Author-No, Book-No)

BOOK-SUBJECT (Book-No, Subject)
Example 2: Faculty Relation (BCNF Form)

<table>
<thead>
<tr>
<th>Faculty</th>
<th>Subject</th>
<th>Institute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vijay Krishna</td>
<td>DBMS</td>
<td>IILM</td>
</tr>
<tr>
<td>Vijay Krishna</td>
<td>Data Structure</td>
<td>IILM</td>
</tr>
<tr>
<td>Vijay Krishna</td>
<td>C++</td>
<td>IILM</td>
</tr>
<tr>
<td>Vijay Krishna</td>
<td>DBMS</td>
<td>GIMT</td>
</tr>
<tr>
<td>Vijay Krishna</td>
<td>Data Structure</td>
<td>GIMT</td>
</tr>
<tr>
<td>Gopal Krishna</td>
<td>Data Structure</td>
<td>GIMT</td>
</tr>
<tr>
<td>Gopal Krishna</td>
<td>Java</td>
<td>IILM</td>
</tr>
</tbody>
</table>

Relation in 4NF

Note: 4NF eliminates MVD relationships. Thus no table can have more than a single many-to-one or many-to-many relationships which are not directly related.

3.10.6 Fifth Normal Form (5NF)

A table is said to be in the 5NF if and only if it is in 4NF and every Join dependency in it is implied by the candidate key.

Consider the following example:

Psychiatrist-to-Insurer-to-Condition

<table>
<thead>
<tr>
<th>Psychiatrist</th>
<th>Insurer</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Vijay</td>
<td>Healthco</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Vijay</td>
<td>Healthco</td>
<td>Depression</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Friendly Care</td>
<td>Dementia</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Friendly Care</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Friendly Care</td>
<td>Depression</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Friendly Care</td>
<td>Mood Disorder</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Friendly Care</td>
<td>Schizophrenia</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Healthco</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Healthco</td>
<td>Dementia</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Victorian Life</td>
<td>Conversion Disorder</td>
</tr>
</tbody>
</table>
To split the relation into three parts

**Psychiatrist-to-Condition**

<table>
<thead>
<tr>
<th>Psychiatrist</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Vijay</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Vijay</td>
<td>Depression</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Dementia</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Depression</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Mood Disorder</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Schizophrenia</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Dementia</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Conversion Disorder</td>
</tr>
</tbody>
</table>

**Psychiatrist-to-Insurer**

<table>
<thead>
<tr>
<th>Psychiatrist</th>
<th>Insurer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Vijay</td>
<td>Healthco</td>
</tr>
<tr>
<td>Dr. Santosh</td>
<td>Friendly Care</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Friendly Care</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Healthco</td>
</tr>
<tr>
<td>Dr. Gopal</td>
<td>Victorian Life</td>
</tr>
</tbody>
</table>

**Insurer-to-Condition**

<table>
<thead>
<tr>
<th>Insurer</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthco</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Healthco</td>
<td>Depression</td>
</tr>
<tr>
<td>Healthco</td>
<td>Dementia</td>
</tr>
<tr>
<td>Friendly Care</td>
<td>Dementia</td>
</tr>
<tr>
<td>Friendly Care</td>
<td>Anxiety</td>
</tr>
<tr>
<td>Friendly Care</td>
<td>Depression</td>
</tr>
<tr>
<td>Friendly Care</td>
<td>Mood Disorder</td>
</tr>
<tr>
<td>Friendly Care</td>
<td>Schizophrenia</td>
</tr>
<tr>
<td>Victorian Life</td>
<td>Conversion Disorder</td>
</tr>
</tbody>
</table>

### 3.10.7 Sixth Normal Form

This normal form was, as of 2005 only recently proposed. The 6NF was only defined when extending the relational modal to take into account the temporal dimension unfortunately, most current SQL technologies as of 2005 do not take into account this work, and most temporal extensions to SQL are not relational.
3.10.8 Domain/Key Normal Form

Domain/Key Normal Form is a normal form used in database normalization which required that the database contains no constraints other than domain constraints and key constraints.

A domain constraint specifies the permissible values for a given attribute, while a key constraint specifies the attributes that uniquely identify a row in a given table.

The domain/Key NF is the Holy Grail of relational database design, achieved when every constraint on the relation is a logical consequence of the definition of keys and domains, and enforcing key and domain restraints and conditions causes all constraints to be met.

Thus it avoids all non-temporal anomalies. It's much easier to build a database in domain/key normal form than it is to convert lesser databases which may contain numerous anomalies.

However, successfully building a domain/key normal form database remains a difficult task, even for experienced database programmers.

Thus, while the domain/key Normal form eliminates the problems found in most databases it tends to be the most costly normal form to achieve.

3.10.9 Conclusion of Database Normalization

Data Normalization is a technique that ensures some basic properties:
- No duplicate tuples.
- No Nested Relations.

Data Normalization is often used as the only technique for database design-implementation view. A more appropriate approach is to complement conceptual modelling with data normalization.

3.11 LOSSLESS-JOIN DECOMPOSITION

- Let \( R \) be a relation schema.
- Let \( F \) be a set of functional dependency of \( R \).
- Let \( R_1 \) and \( R_2 \) from decomposition of \( R \).

The decomposition is a loseless-Join decomposition of \( R \) if at least one of the following FDS are in \( F^+ \).

1. \( R_1 \cap R_2 \rightarrow R_1 \)
2. \( R_1 \cap R_2 \rightarrow R_2 \)

Why is this true? Simply put, it ensures that the attributes involved in the natural join \((R_1 \cap R_2)\) are a candidate key for at least one of the two relations.

This ensures that we can never get the situation where spurious tuples are generated, as for any value on the join attributes there will be a unique tuple in one of the relations.

Ex.: Let \( R = \{A, B, C, D\} \)
\( F = \{A \rightarrow B, B \rightarrow C\} \)
Suppose decompositions \( R_1 = \{A, B\}, R_2 = \{B, C\} \) and \( R_3 = \{A, D\} \). Find the decompositions is lossless or lossy.

Ans. (1) Consider \( R_1 \) and \( R_3 \)

\[ R_1 \cap R_3 = \{A, B\} \cap \{A, D\} = \{A\} \]

Since \( A \rightarrow B \) and \( A \) is a key in \( R_1 \).

\[ R_1 \cap R_3 \rightarrow R_1 = \{A, B\} \quad \because A \rightarrow B \]

Let us union \( R_1 \) and \( R_3 \) and form \( R_4 \).

\[ R_4 = \{A, B\} \cup \{A, D\} = \{A, B, D\} \]
The decomposition of \((A, B, D)\) into \(R_1\) and \(R_3\) is lossless-Join.

(2) Next consider \(R_4\) and \(R_2\)

Now \(R_4 \cap R_2 = \{A, B, D\} \cap \{B, C\} = \{B\}\)

Since \(B \rightarrow C\) and \(B\) is a key in \(R_2\)

\[ \therefore R_4 \cap R_2 \rightarrow R_2 = \{B, C\} \]

\[ \therefore B \rightarrow C \]

The decomposition of \((A, B, C, D)\) into \(R_2\) and \(R_4\) is lossless-Join.

---

**Solved Problems**

Q. 1. Consider the schema \(R = (V, W, X, Y, Z)\) suppose the following FDs hold :

\[ F = \{Z \rightarrow V, W \rightarrow Y, XY \rightarrow Z, V \rightarrow WX\} \]

State whether the following decomposition of schema \(R\) is loss-less join decomposition. Justify your answer.

(UPTU 2003, 05)

Ans. For the decomposition of relation schema \(R\) into \(R_1\) and \(R_2\) to lossy or lossless either any one of the following conditions hold.

(1) \(R_1 \cap R_2 \rightarrow R_1\)

(2) \(R_1 \cap R_2 \rightarrow R_2\)

(i) Considering the first decomposition

\[ R_1 = (V, W, X) \]

\[ R_2 = (V, Y, Z) \]

\[ R_1 \cap R_2 = \{V\} \]

Since \(V \rightarrow WX\) and \(V\) is a key in \(R_1\)

\[ \therefore R_1 \cap R_2 \rightarrow R_1 \text{ because } V \rightarrow WX = R_1 \]

Thus we can say the decomposition

\[ R_1 = (V, W, X) \]

\[ R_2 = (V, Y, Z) \text{ is lossless-decomposition.} \]

(ii) Now considering the second decomposition

\[ R_1 = (V, W, X) \]

\[ R_2 = (X, Y, Z) \]

\[ R_1 \cap R_2 = \{X\} \]

So either

\[ X \rightarrow VWX \]

\[ \therefore R_1 \cap R_2 \rightarrow R_1 \]

or

\[ X \rightarrow XYZ \]

\[ \therefore R_1 \cap R_2 \rightarrow R_2 \]

But using the given set of FDS, we can not get.

Either \(X \rightarrow VWX\)

or \(X \rightarrow XYZ\)

Thus the decomposition

\[ R_1 = (V, W, X) \]

\[ R = (X, Y, Z) \text{ is lossy decomposition.} \]
An other method: For lossless-join decomposition:

Q. 2. Consider the scheme \( R = (A, B, C, D, E) \) suppose following FDS hold:
\[
\begin{align*}
E & \rightarrow A \\
CD & \rightarrow E \\
A & \rightarrow BC \\
B & \rightarrow D
\end{align*}
\]

State, whether the following decomposition of \( R \) are lossless join decomposition or not, justify
1. \( (A, B, C), (A, D, E) \)
2. \( (A, B, C), (C, D, E) \)

Ans. (1) \( (A, B, C), (A, D, E) \)

Let \( R_1 = (A, B, C) \)
\[
R_2 = (A, D, E)
\]

Put \( \alpha \) in table where attribute is exist in relation and put \( \beta \) where they do not exist in relation.
Now we get the following table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \beta_1 )</td>
<td>( \beta_1 )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( \alpha_A )</td>
<td>( \beta_2 )</td>
<td>( \beta_2 )</td>
<td>( \alpha_D )</td>
<td>( \alpha_E )</td>
</tr>
</tbody>
</table>

After seeing the table we came to know that column A have common \( \alpha \) and \( A \rightarrow BC \)
So we can put \( \alpha \) in Row \( R_2 \) column B and C. After that we get the following table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \beta_1 )</td>
<td>( \beta_1 )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \alpha_D )</td>
<td>( \alpha_E )</td>
</tr>
</tbody>
</table>

Since Row \( R_2 \) has all \( \alpha \).
So the decomposition is lossless.

(2) \( (A, B, C), (C, D, E) \)
\[
\begin{align*}
R_1 & = (A, B, C) \\
R_2 & = (C, D, E)
\end{align*}
\]

- Put \( \alpha \) in table where attribute is exist in relation.
- Put \( \beta \) where they do not exist in relation.

Now we get the following table:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \beta_1 )</td>
<td>( \beta_1 )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( \beta_2 )</td>
<td>( \beta_2 )</td>
<td>( \alpha_C )</td>
<td>( \alpha_D )</td>
<td>( \alpha_E )</td>
</tr>
</tbody>
</table>

After seeing the table we came to know that, column C have common \( \alpha \).
In given FDS, \( C \) does not implies any value, so we can not update table.
So the decomposition
\[
\begin{align*}
R_1 & = (A, B, C) \\
R_2 & = (C, D, E)
\end{align*}
\]
is lossy.
Q. 3. Given \( R = (A, B, C, D, E) \) with the FDs.

\[ F = \{ AB \rightarrow CD, A \rightarrow E, C \rightarrow D \} \]

Find, if the decomposition of \( R \) into
\( R_1 = (A, B, C), R_2 = (B, C, D) \) and \( R_3 = (C, D, E) \) is lossy or not.

**Ans.** Given \( R = (A, B, C, D, E) \)

The decomposition of \( R \) into three relations

\[ R_1 = (A, B, C) \]
\[ R_2 = (B, C, D) \]
\[ R_3 = (C, D, E) \]

and

\[ F = \{ AB \rightarrow CD, A \rightarrow E, C \rightarrow D \} \]

- Put \( \alpha \) in table where attribute is exist in relation and
- Put \( \beta \) where they do not exist in relation.

So we get the table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \beta_1 )</td>
<td>( \beta_2 )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( \beta_2 )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \alpha_D )</td>
<td>( \beta_2 )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( \beta_3 )</td>
<td>( \beta_3 )</td>
<td>( \alpha_C )</td>
<td>( \alpha_D )</td>
<td>( \alpha_E )</td>
</tr>
</tbody>
</table>

After seeing the table we came to know that, column \( C \) have common \( \alpha \) and FD \( C \rightarrow D \).

Thus this decomposition is lossy.

Q4. Given \( R \ (A, B, C, D) \) with the FDs

\[ F = \{ A \rightarrow B, A \rightarrow C, C \rightarrow D \} \]

Find if the decomposition of \( R \) into \( R_1 (A, B, C) \) and \( R_2 (C, D) \) is lossy or not.

**Ans.**

\[ R_1 = (A, B, C) \]
\[ R_2 = (C, D) \]

- Put \( \alpha \) in table where attribute is exist in Relation and
- Put \( \beta \) where they do not exist in Relation.

Now we get the following table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( \alpha_A )</td>
<td>( \alpha_B )</td>
<td>( \alpha_C )</td>
<td>( \beta_1 )</td>
<td></td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( \beta_2 )</td>
<td>( \beta_2 )</td>
<td>( \alpha_C )</td>
<td>( \alpha_D )</td>
<td></td>
</tr>
</tbody>
</table>

After seeing the table we came to know that column \( C \) have common \( \alpha \) and FD \( C \rightarrow D \).
So we can put $\alpha$ in Row $R_1$ column $D$.
So after that we get the table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$\alpha_A$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
</tr>
</tbody>
</table>

Since Row $R_1$ has all $\alpha$.
So the decomposition $R_1 (A, B, C)$ and $R_2 (C, D)$ is lossless.

**Q. 5. Given $R (A, B, C, D, E)$ with FDs**

$$F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\},$$

the decomposition of $R$ into $R_1 (A, B, C), R_2 (B, C, D)$ and $R_3 (C, D, E)$ is lossless or lossy.

**Ans.** Given $R (A, B, C, D, E)$

Decomposition of $R$ into three relations

- $R_1 = (A, B, C)$
- $R_2 = (B, C, D)$
- $R_3 = (C, D, E)$

The FD $F = \{AB \rightarrow CD, A \rightarrow E, C \rightarrow D\}$

- Put $\alpha$ in the table where attribute is exist in relation and
- Put $\beta$ where they do not exist in relation.

We get the following table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$\alpha_A$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\beta_2$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\beta_2$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\alpha_E$</td>
</tr>
</tbody>
</table>

After seeing the table we came to know that column $C$ have common $\alpha$ and FD $C \rightarrow D$.
So we can put $\alpha$ in row $R_1$ column $D$.
After that we get the table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$\alpha_A$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\beta_2$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\beta_2$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\alpha_E$</td>
</tr>
</tbody>
</table>

No further changes are possible and the final version of the table is the same as the table above.
Finally we find no rows in the table with all $\alpha$s.
Hence the decomposition is lossy.

**Q. 6. Consider the relational schema**

$R (A, B, C, D, E, F, G, H)$ with the FDs

$$F = \{AB \rightarrow C, BC \rightarrow D, E \rightarrow F, G \rightarrow F, H \rightarrow A, FG \rightarrow H\}$$

Is the decomposition of $R$ into $R_1 (A, B, C, D), R_2 (A, B, C, E, F)$ and $R_3 (A, D, F, G, H)$ lossless ?

**Ans.** Given $R (A, B, C, D, E, F, G, H)$

The decomposition of $R$ into $R_1, R_2, R_3$. S.T.
$R_2 = (A, B, C, E, F)$, $R_1 = (A, B, C, D)$

$R_3 = (A, D, F, G, H)$

FDs $F = \{AB \rightarrow C, BC \rightarrow D, E \rightarrow F, G \rightarrow F, H \rightarrow A, FG \rightarrow H\}$

* Put $\alpha$ in table where attribute is exist in Relation, and
* Put $\beta$ where they do not exist in relation.

We get the following table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$\alpha_A$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\alpha_D$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
<td>$\beta_1$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$\alpha_A$</td>
<td>$\alpha_B$</td>
<td>$\alpha_C$</td>
<td>$\beta_2$</td>
<td>$\alpha_E$</td>
<td>$\alpha_F$</td>
<td>$\beta_2$</td>
<td>$\beta_2$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$\alpha_A$</td>
<td>$\beta_3$</td>
<td>$\beta_3$</td>
<td>$\alpha_D$</td>
<td>$\beta_3$</td>
<td>$\alpha_F$</td>
<td>$\alpha_G$</td>
<td>$\alpha_H$</td>
</tr>
</tbody>
</table>

After seeing the table we came to know that column $A$ have common $\alpha$. But in given FDs $A$ does not implies any value.

So we cannot update table.

Since any row in this table does not contain all $\alpha$s.

Hence, this decomposition is lossy.

Q. 7. Given the relational schema $R$ $(A, B, C, D, E)$ and given the following set of FDs defined on $R$.

$$F = \{A \rightarrow BC, CD \rightarrow E, AC \rightarrow E, B \rightarrow D, E \rightarrow AB\}$$

(a) Determine a lossless-join decomposition of $R$.

(b) Determine a decomposition of $R$ which is not lossless-join.

(c) Determine if $R$ is in 3NF w.r.t $F$ if it is not violating the 3NF.

Ans. (a) One out of many possible lossless-join decomposition

Let $R_1 = (A, D, E)$, $R_2 (A, B, C)$

These two decompositions $R_1$ and $R_2$ of $R$ is lossless-join decomposition.

This is because

\[ R_1 \cap R_2 = \{A, D, E\} \cap \{A, B, C\} = \{A\} \]

Since $A \rightarrow BC$ and $A$ is a key in $R_2$.

\[ R_1 \cap R_2 = \{A, B, C\} = R_2. \]

(b) The decomposition of $R$ is $R_1$ and $R_2$:

Let $R_1 = \{A, B, C, D\}$ and $R_2 = \{B, E\}$

This decomposition is a lossy decomposition because

\[ R_1 \cap R_2 = \{A, B, C, D\} \cap \{B, E\} = \{B\} \]

\[ B \rightarrow D \]

Therefore, $B \rightarrow R_1$ or $R_2$, i.e., $\{BD \rightarrow R_1 \text{ or } R_2\}$

(c) The candidates key of $R$ are $A$

because $A \rightarrow BC$

\[ A^+ = \{A, B, C\} \]
For each of the following relation schemas and set of FDs.

(i) Identify candidate keys for R.
(ii) Indicate BCNF violations and decompose if necessary.
(iii) Indicate 3NF violations and decompose if necessary.

Ans. (1) Given R \( (A, B, C, D) \) with FDs
\[ F = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \} \]

(i) For \( A \rightarrow B \), \( A^+ = \{A, B\} \)

\( B \rightarrow C \), \( B \subseteq A^+ \)
\[ A^+ = A^+ \cup \{C\} = \{A, B, C\} \]

\( C \rightarrow D \), \( C \subseteq A^+ \)
\[ A^+ = \{A, B, C, D\} = R. \]

\( A \rightarrow ABCD \)

Similarly,
\( B \rightarrow ABCD \)
\( C \rightarrow ABCD \)
\( D \rightarrow ABCD \)

Thus the candidate keys are \( A, B, C, D \).

(ii) Since all \( \{A, B, C, D\} \) are candidate keys. Thus there is no BCNF violation.
Hence no decomposition is necessary.

(iii) Since \( A, B, C, D \) all are candidate keys.
Thus there is no 3NF violation
Hence no decomposition is necessary.

(2) Given R \( (A, B, C, D) \) with FDs
\[ F = \{ B \rightarrow C, B \rightarrow D \} \]

(i) \( B^+ = \{B, C, D\} \)

Since the closure of \( B (B^+) \) does not include all R's attribute
i.e., \( B^+ = \{B, C, D\} \neq R = \{A, B, C, D\} \)

Hence there are no any candidate key.

(ii) \( B \rightarrow C, B \rightarrow D \) both violate BCNF.
because there is no any candidate key.
Thus decomposition is necessary.

Since \( AB \rightarrow ABCD \). Thus \( AB \) is candidate key.

Therefore one possible BCNF decomposition is

\( \{(A, B), (B, C, D)\} \)

(iii) \( B \rightarrow C, B \rightarrow D \) both violate 3NF

Because \( B \) is not a super key and \( CD \) are not part of a candidate key.

One possible 3NF decomposition is

\( \{(A, B), (B, C, D)\} \) or \( \{(A, B), (B, C), (B, D)\} \)

Because \( AB \rightarrow ABCD \). Thus \( AB \) is candidate key.

Q. 9. Are these schema in 3NF?

(a) \( R = \{\text{city, street, zip}\} \)

\( F = \{\text{city, street} \rightarrow \text{zip, zip} \rightarrow \text{city}\} \)

(b) \( R = (A, B, C) \)

\( F = \{A \rightarrow B, B \rightarrow C\} \)

(c) \( R = (A, B, C, D) \)

\( F = \{B \rightarrow C, B \rightarrow D\} \)

(d) \( R = (A, B, C, D) \)

\( F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A\} \)

(e) \( R = (A, B, C, D) \)

\( F = \{AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B\} \)

Ans. (a) \( R = \{\text{city, street, zip}\} \)

\( F = \{\text{city, street} \rightarrow \text{zip, zip} \rightarrow \text{city}\} \)

For city, street \( \rightarrow \) zip

\( \therefore \) \( (\text{city, street})^+ = \{\text{city, street, zip}\} = R. \)

Thus, \( \{\text{city, street}\} \) is the keys.

Similarly, \( \{\text{zip, street}\} \) is also the key.

Therefore, \( \{\text{city, street}\} \) and \( \{\text{zip, street}\} \) are the keys.

The LHS of city, street \( \rightarrow \) zip is a key.

So its OK.

The RHS of zip \( \rightarrow \) city is part of a key so its OK.

Therefore, the schema is in 3NF.

(b) \( R = (A, B, C) \)

\( F = \{A \rightarrow B, B \rightarrow C\} \)

The only key is \( A \). The LHS of \( B \rightarrow C \) is not a super key.

The RHS is not part of a key.

Therefore, the schema is not in 3NF.
(c) Given \( R = (A, B, C, D) \)
\[
F = \{ B \rightarrow C, B \rightarrow D \}
\]
The only key is \( AB \).
The LHS of \( B \rightarrow C \) is not a super key.
The RHS is not part of a key.
Therefore, the schema is not in 3NF.

(d) Given \( R = (A, B, C, D) \)
\[
F = \{ AB \rightarrow C, C \rightarrow D, D \rightarrow A \}
\]
The keys are \( AB, BC \) and \( BD \).
The LHS of \( AB \rightarrow C \) is a key and the RHS of \( C \rightarrow D \) is a part of a key. The RHS of \( D \rightarrow A \) is a key.
Therefore, the schema is in 3NF.

(c) Given \( R = (A, B, C, D) \)
\[
F = \{ AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B \}
\]
The only keys are \( \{ AB, BC, CD, AD \} \)
All the FDs have their LHS as keys.
and RHS have a part of a key.
Therefore the schema is in 3NF.

Q. 10. Give a lossless-join, dependency-preserving decomposition into 3NF of schema \( R \).
\[
R = (A, B, C, D, E)
\]
\[
F = \{ A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A \}
\]
Ans. The schema \( R = (A, B, C, D, E) \) is already into 3NF because the candidate key of \( R \) are \( (A, BC, CD, E) \).
We may also create a schema from the algorithm.
\[
R = \{ (A, B, C), (C, D, E), (B, D), (E, A) \}
\]
schema \( (A, B, C) \) contains a candidate key.
\[ \therefore \] \( R \) is a 3NF dependency-preserving lossless join dependency.

Q. 11. Give a lossless-join dependency preserving decomposition into BCNF of schema \( R \) of above question.
Ans. We know that FD \( B \rightarrow D \) is nontrivial because \( D \subseteq B \), and the LHS is not a superkey. By the algorithm we derive the relation \( \{ (A, B, C, E), (B, D) \} \) is in BCNF.

Q. 12. Given \( R = (A, B, C, D) \)
\[
F = \{ A \rightarrow B, B \rightarrow C \}
\]
Ans. (a) Using \( A \rightarrow B \) First
\[
R_1 = (A, B), R_2 = (A, C, D)
\]
By \( A \rightarrow C \) (which is in \( F^+ \)) decompose \( R_2 \)
\[
R_3 = (A, C), R_4 = (A, D)
\]
The resulting relation schemas are \( R_1, R_3 \) and \( R_4 \).
\[ \therefore \] Result: \( (A, B), (A, C), (A, D) \)
The result is not dependency preserving.
(b) Using $B \rightarrow C$ First

$$R_1 = (B, C), R_2 = (A, B, C)$$

Decompose $R_2$

$$R_3 = (A, B), R_4 = (A, D)$$

**Result:** $BC, AB, AD$

The resulting relation schemas are $R_1, R_3, R_4$.

**Result:** $(B, C), (A, B), (A, D)$

The result is dependency preserving.

Q.13. Compute the closure of the following set $F$ of functional dependencies for relation schema $R = (A, B, C, D, E)$.

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

**List the candidate keys for $R$.**

**Ans.** Starting with $A \rightarrow BC$

We can conclude $A \rightarrow B$ and $A \rightarrow C$ ...

$\therefore A \rightarrow B$ and $B \rightarrow D$, then $A \rightarrow D$ (Transitivity) ...

$\therefore A \rightarrow BC, B \rightarrow D$, then $A \rightarrow CD$. ...

$\therefore A \rightarrow CD$ and $CD \rightarrow E$, then $A \rightarrow E$ (Transitivity) ...

$\therefore A \rightarrow A$ we have (Reflexive) ...

From the above step (i), (ii), (iii), (iv) taking union. We get $A \rightarrow ABCDE$. Thus $A$ is candidate key.

$\therefore E \rightarrow A$ and $A \rightarrow ABCDE$, then

$E \rightarrow ABCDE$ (Transitivity)

Thus $E$ is a candidate key.

For FD $CD \rightarrow E$

$\therefore CD \rightarrow E$ and $E \rightarrow ABCDE$, then

$CD \rightarrow ABCDE$ (Transitivity)

So $CD$ is also a candidate key.

$\therefore B \rightarrow D$ and $BC \rightarrow CD$, then

$BC \rightarrow ABCDE$ (augmentation and transitivity).

So $BC$ is a candidate key.

Therefore the candidate keys are

$(A, BC, CD, E)$

Therefore, any FD with $A, E, BC$ or $CD$ on LHS of the arrow is in $F^+$, no matter with other attributes appear in the FD.

Allow to represent any set of attributes in $R$, then $F^+$ is $BD \rightarrow B$, $BD \rightarrow D$, $C \rightarrow C$, $D \rightarrow D$ $BD \rightarrow BD$, $B \rightarrow D$, $B \rightarrow B$, $B \rightarrow BD$ and all the FDs of the form
Q. 14. Consider relation \( R = (A, B, C, D, E) \)
\( M \) is the set of multivalued functional dependencies.
\( M = (A \rightarrow BC, B \rightarrow CD, E \rightarrow AD) \)
Give a lossless join decomposition of schema \( R \) into 4NF.

**Ans.**

Given \( A \rightarrow BC \),
\[ A \rightarrow BC \text{ and } B \rightarrow CD \]
\[ A \rightarrow CD \]

by union of (i) and (ii) \( A \rightarrow BCD \)

Given \( R \) \( (A, B, C, D, E) \)
Let the decomposition \( R \) into \( R_1 \) and \( R_2 \)
\[ R_1 = (A, B, C, D) \text{ (A is key attribute)} \]
\[ R_2 = (A, D, E) \text{ (E is key attribute)} \]

For lossless-join decomposition
either \( R_1 \cap R_2 \rightarrow R_1 \)
or \( R_1 \cap R_2 \rightarrow R_2 \)
\[ R_1 \cap R_2 = (A, B, C, D) \cap (A, D, E) = \{A, D\} \]
Since \( A \rightarrow BCD \).
\[ R_1 \cap R_2 = (A, B, C, D) \cap (A, D, E) \rightarrow (A, B, C, D) = R_1 \]
Hence, we can say that schema \( R \) can be decomposed into \((A, B, C, D)\) and \((A, E, D)\), which are lossless-join decomposition and are in 4NF.

Q. 15. Given \( R = \{A, B, C, D, E\} \) and set \( M \) of multivalued dependency.
\( M = \{A \rightarrow BC, B \rightarrow CD, E \rightarrow AD\} \)

List all non-trivial MUD in \( M^+ \).

**Ans.**

\[ M^+ = \{A \rightarrow BCD\} \]
For \( A \rightarrow BCD \)
given \( A \rightarrow BC \)
\[ A \rightarrow B \]
\[ A \rightarrow C \]
Since \( B \rightarrow CD \)
\[ A \rightarrow B \text{ and } B \rightarrow CD \]
\[ A \rightarrow CD \]
\[ A \rightarrow B \text{ and } B \rightarrow CD \]
\[ A \rightarrow CD \]
By Union of (i) and (ii) \( A \rightarrow BCD \)
Since \( E \rightarrow AD \)
Applying decomposition
\[ E \rightarrow A \text{ and } E \rightarrow D \]
Q. 16. Explain how FDs can be used to indicate the following:

• A one-to-one relationship set exists between entity set account and customer.
• A many-to-one relationship set exists between entity sets account and customers.

Ans. Let \( PK(r) \) denote the primary key attribute of relation \( r \).

• The FDs \( PK(\text{account}) \rightarrow PK(\text{customer}) \) and \( PK(\text{customer}) \rightarrow PK(\text{account}) \) indicate a one-to-one relationship, because any two tuples with the same value for account must have the same value for customer and any two tuples agreeing on customer must have the same value for account.
• The FDs \( PK(\text{account}) \rightarrow PK(\text{customer}) \) indicates a many-to-one relationship, because any account value which is repeated will have the same customer value but many account values may have the same customer value.

Q. 17. Consider the following collection of relations and dependencies. Assume that each relation is obtained through decomposition from a relation with attributes ABCDEFGHI and that all the known dependencies over relation ABCDEFGHI are listed for each question. (The questions are independent of each other; obviously, since the given dependencies over ABCDEFGHI are different.) For each (sub) relation:

(a) State the strongest normal form that the relation is in.
(b) If it is not in BCNF, decompose it into a collection of BCNF relations.

1. \( R_1(A, C, B, D, E), A \rightarrow B, C \rightarrow D \)
2. \( R_2(A, B, F), AC \rightarrow E, B \rightarrow F \)
3. \( R_3(A, D), D \rightarrow G, G \rightarrow H \)
4. \( R_4(D, C, H, G), A \rightarrow I, I \rightarrow A \)
5. \( R_5(A, I, C, E) \)

Ans. 1. 1NF. BCNF decomposition: AB, CD, ACE.
2. 1NF. BCNF decomposition: AB, BF
3. BCNF.
4. BCNF.
5. BCNF.

Q. 18. Suppose you are given a relation \( R \) with four attributes ABCD. For each of the following sets of FDs, assuming those are the only dependencies that hold for \( R \), do the following:

(a) Identify the candidate key(s) for \( R \).
(b) Identify the best normal form that \( R \) satisfies (1NF, 2NF, 3NF, or BCNF).
(c) If \( R \) is not in BCNF, decompose it into a set of BCNF relations that preserve the dependencies.

1. \( C \rightarrow D, C \rightarrow A, B \rightarrow C \)
2. \( B \rightarrow C, D \rightarrow A \)
3. \( ABC \rightarrow D, D \rightarrow A \)
4. \( A \rightarrow B, BC \rightarrow D, A \rightarrow C \)
5. \( AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B \)

Ans.

1. (a) Candidate keys: \( B \)
   
   (b) \( R \) is in 2NF but not 3NF.

(c) \( C \rightarrow D \) and \( C \rightarrow A \) both cause violations of BCNF. One way to obtain a (lossless) join preserving decomposition is to decompose \( R \) into \( AC, BC, \) and \( CD \).
2. (a) Candidate keys: BD
(b) \( R \) is in 1NF but not 2NF.
(c) Both \( B \rightarrow C \) and \( D \rightarrow A \) cause BCNF violations. The decomposition: \( AD, BC, BD \)
   (obtained by first decomposing to \( AD, BCD \)) is BCNF and lossless and join-preserving.
3. (a) Candidate keys: \( ABC, BCD \)
(b) \( R \) is in 3NF but not BCNF.
(c) \( ABCD \) is not in BCNF since \( D \rightarrow A \) and \( D \) is not a key. However, if we split up \( R \) as \( AD, BCD \),
   we cannot preserve the dependency \( ABC \rightarrow D \). So there is no BCNF decomposition.
4. (a) Candidate keys: A
(b) \( R \) is in 2NF but not 3NF (because of the FD \( : BC \rightarrow D \)).
(c) \( BC \rightarrow D \) violates BCNF since \( BC \) does not contain a key. So we split up \( R \) as in: \( BCD, ABC \).
5. (a) Candidate keys: \( AB, BC, CD, AD \)
(b) \( R \) is in 3NF but not BCNF (because of the FD \( : C \rightarrow A \)).
(c) \( C \rightarrow A \) and \( D \rightarrow B \) both cause violations. So decompose into: \( AC, BCD \) but this does not
   observe \( AB \rightarrow C \) and \( AB \rightarrow D \), and \( BCD \) still not BCNF because \( D \rightarrow B \). So we need to decompose
   further into: \( AC, BD, CD \). However, when we attempt to revive the lost functional dependencies by
   adding \( ABC \) and \( ABD \), we see that these relations are not in BCNF form. Therefore, there is no BCNF
   decomposition.

**Review Questions**

1. What do you mean by functional dependency? Explain with an example and a functional dependency diagram.
2. What is the importance of functional dependencies in database design?
3. What are the main characteristics of functional dependencies?
4. Describe Armstrong’s axioms. What are derived rules?
5. Let us assume that the following is given:
   Attribute set \( R = ABCDEFGH \)
   \( FD \) set of \( F = \{ AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G \} \)
   Which of the following decompositions of \( R = ABCDEG \), with the same set of dependencies \( F \), is
   (a) dependency-preserving and
   (b) lossless-join
   (a) \( \{ AB, BC, ABDE, EG \} \)
   (b) \( \{ ABC, ACDE, ADG \} \)
6. What is the lossless or non-additive join property of decomposition? Why is it important?
7. What do you understand by the term normalization? Describe the data normalization process. What does it accomplish?
8. Describe the purpose of normalising data.
9. What are different normal forms?
10. Define 1 NF, 2 NF and 3 NF.
11. Given a relation $R (A, B, C, D, E)$ and $F = (A \rightarrow B, BC \rightarrow D, D \rightarrow BC, DE \rightarrow \phi)$, synthesis a set of 3 NF relation schemes.

12. Define Boyce–Codd normal form (BCNF). How does it differ from 3 NF? Why is it considered a stronger from 3 NF? Provide an example to illustrate.

13. Why is 4 NF preferred to BCNF?

14. A relation $R (A, B, C)$ has FDs $AB \rightarrow C$ and $C \rightarrow A$. Is $R$ is in 3 NF or in BCNF? Justify your answer.

15. Explain the following:
   (a) Why $R_2$ is in 2 NF but not 3 NF, where
   \[ R_2 = (\{A, B, C, D, E\}, \{AB \rightarrow CE, E \rightarrow AB, C \rightarrow D\}) \]
   (b) Why $R_3$ is in 3 NF but not BCNF, where
   \[ R_3 = (\{A, B, C, D\}, \{A \rightarrow C, D \rightarrow B\}) \]