Parul® University

Unit 5- Relational Database Design

Subject Code: 303105203

Prof. S.W.Thakare
Assistant Professor,
Computer science & Engineering

Parul® University

CHAPTER-5

Relational Database Design

Domain and data dependency:

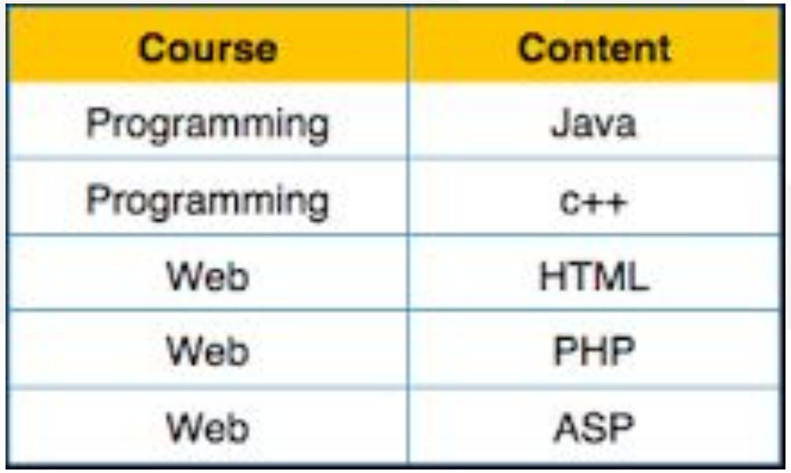
- A domain is a unique set of values that can be assigned to an attribute in a database. For example, a domain of strings can accept only string values.

- Dependencies in DBMS is a relation between two or more attributes. It has the following types in DBMS:

- Functional Dependency (Single and Multivalued Dependency)
- Transitive Dependency
- Partial Dependency

Functional Dependency:

- Functional dependency (FD) is a set of constraints between two attributes in a relation.
- Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.



Course	Content
Programming	Java
Programming	C++
Web	HTML
Web	PHP
Web	ASP

Functional Dependency:

- Let R be a relation schema having n attributes A1, A2, A3,…, An.
- Let attributes X and Y are two subsets of attributes of relation R.
- If the values of the X component of a tuple uniquely (or functionally) determine the values of the Y component, then, there is a functional dependency from X to Y.

This is denoted by $X \rightarrow Y$.

- It is referred as: Y is functionally dependent on the X, or X functionally determines Y.
- The left hand side of the FD is also referred as determinant whereas the right hand side of the FD is referred as dependent.

Parul® University

DIGITAL LEARNING CONTENT

Functional Dependency:

- **EXAMPLE:**

Table 1.1: Example of Functional Dependency

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S3	C	C3

•Question: If sid → sname is a valid FD?

Functional Dependency:

- **EXAMPLE:**
- As we have discussed, for X → Y, for each value of X there should be one Y value.

Sid	→	Sname
S1		A
S1		A
S1		A
S2		B
S3		B

- As you can see over here, for each value of Sid there is only one Sname value. Hence, Sid → Sname is valid FD.

Parul® University

DIGITAL LEARNING CONTENT

Types of Functional Dependency:

Trivial FD:

- X⟶Y is trivial FD if Y is a subset of X.
- Every possible Trivial FD is implied in the relation.
- For example, Consider Table 1.1. Following FDs are Trivial FDs.
    - SidSname ⟶ Sname

Types of Functional Dependency:

Non- Trivial FD:

- X⟶Y is non- trivial FD if there is no common attribute in X, Y attribute set.
- For example, Consider Table 1.1.
- Sid ⟶ Sname is non- trivial FD.

Parul® University
DIGITAL LEARNING CONTENT

Types of Functional Dependency:

What is Transitive Dependency
When an indirect relationship causes functional dependency it is called Transitive Dependency.
If P -> Q and Q -> R is true, then P-> R is a transitive dependency.
To achieve 3NF, eliminate the Transitive Dependency.

What is Partial Dependency?
Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.
The 2nd Normal Form (2NF) eliminates the Partial Dependency.

Parul® University

Armstrong's axioms:

- Armstrong's axioms are a set of rules used to infer (derive) all the functional dependencies on a relational database.
- Let A, B, and C is subsets of the attributes of the relation R.
- Rules are mentioned in the next slide.

Armstrong's axioms:

1. Reflexivity
   – If **B is a subset of A** then **A → B**
2. Augmentation
   – If **A → B** then **AC → BC**
3. Transitivity
   – If **A → B** and **B → C** then A → C
4. Pseudo Transitivity
   – If **A → B** and **BD → C** then **AD → C**
5. Self-determination
   – **A → A**
6. Decomposition
   – If **A → BC** then **A → B** and **A → C**
7. Union
   – If **A → B** and **A → C** then **A → BC**
8. Composition
   – If **A → B** and **C → D** then **AC → BD**

Attribute Closure:

- Given a set of attributes, the closure of any attribute X under F is the set of attributes that are functionally determined by X under F.
- It is denoted by X^+.
- $[X]^+$ = { Set of attributes determined by X.}

Attribute Closure:

Example:

- Given relation R with attributes A,B,C,D,E,F,G and set of FDs,
  F = { $A \rightarrow B$, $B \rightarrow C$, $AC \rightarrow D$, $CD \rightarrow E$, $G \rightarrow A$, $AF \rightarrow G$ }.
  Find the Closure of and AF.

- Answer: A^+ is { A, B, C, D, E)

- $[AF]^+$ = { A, B, C, D, E, F, G}

Closure Of FD:

- The Closure Of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules known as Armstrong's Rules.
- If "F" is a functional dependency then closure of functional dependency can be denoted using "$\{F\}^+$".
- There are three steps to calculate closure of functional dependency. These are:

Step-1 : Add the attributes which are present on Left Hand Side in the original functional dependency.

Step-2 : Now, add the attributes present on the Right Hand Side of the functional dependency.

Step-3 : With the help of attributes present on Right Hand Side, check the other attributes that can be derived from the other given functional dependencies. Repeat this process until all the possible attributes which can be derived are added in the closure.

Closure Of FD:

- Example-2 : Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies.

1) FD1 : A BC, 2) FD2 : C B 3)FD3 : D E 4)FD4 : E D

Now, we need to calculate the closure of attributes of the relation R. The closures will be:

- {A}+ = {A, B, C}
- {B}+ = {B}
- {C}+ = {B, C}
- {D}+ = {D, E}
- {E}+ = {E}

Parul® University

DIGITAL LEARNING CONTENT

Closure Of FD:

Example-1 : Consider the table student_details having (Roll_No, Name,Marks, Location) as the attributes and having two functional dependencies.

- FD1 : Roll_No Name, Marks
- FD2 : Name   Marks, Location
- Now, We will calculate the closure of all the attributes present in the relation using the three steps mentioned below.
- Step-1 : Add attributes present on the LHS of the first functional dependency to the closure.
- {Roll_no}+ = {Roll_No}
- Step-2 : Add attributes present on the RHS of the original functional dependency to the closure.
- {Roll_no}+ = {Roll_No, Marks}

Closure Of FD:

Step-3 : Add the other possible attributes which can be derived using attributes present on the RHS of the closure. So Roll_No attribute cannot functionally determine any attribute but Name attribute can determine other attributes such as Marks and Location using 2nd Functional Dependency(Name [icon name="long-arrow-right" class="" unprefixed_class=""] Marks, Location).

Therefore, complete closure of Roll_No will be :

{Roll_no}+ = {Roll_No, Marks, Name, Location}

Closure Of FD:

Similarly, we can calculate closure for other attributes too i.e “Name”.

Step-1 : Add attributes present on the LHS of the functional dependency to the closure.

{Name}+ = {Name}

Step-2 : Add the attributes present on the RHS of the functional dependency to the closure.

{Name}+ = {Name, Marks, Location}

Step-3 : Since, we don’t have any functional dependency where “Marks or Location” attribute is functionally determining any other attribute , we cannot add more attributes to the closure. Hence complete closure of Name would be :

{Name}+ = {Name, Marks, Location}

Closure Of FD:

NOTE : We don’t have any Functional dependency where marks and location can functionally determine any attribute. Hence, for those attributes we can only add the attributes themselves in their closures. Therefore,

{Marks}+ = {Marks}

and

{Location}+ = { Location}

Closure Of An Attribute:

Closure of an attribute x is the set of all attributes that are functional dependencies on X with respect to F. It is denoted by X^+ which means what X can determine.

Algorithm

Let's see the algorithm to compute X^+

```
• Step 1 − $X^+ = X$
• Step 2 − repeat until $X^+$ does not change
    ◦ For each FD Y->Z in F
        ▪ If $Y \subseteq X^+$ then $X^+ = X^+ \cup Z$
```

Example 1

Consider a relation R(A,B,C,D,E,F)

F: E->A, E->D, A->C, A->D, AE->F, AG->K.

Find the closure of E or E^+

Closure Of An Attribute:

Solution: The closure of E or E+ is as follows –

E+ = E

=EA {for E->A add A}

=EAD {for E->D add D}

=EADC {for A->C add C}

=EADC {for A->D D already added}

=EADCF {for AE->F add F}

=EADCF {for AG->K don't add k AG ⊄ D+)

Example 2 Let the relation R(A,B,C,D,E,F)

F: B->C, BC->AD, D->E, CF->B. Find the closure of B.

Closure Of An Attribute:

Solution

The closure for B is as follows −

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

Closure is used to find the candidate keys of R and compute F^+

Candidate key of R: X is a candidate keys of R if X->{R}

Super Key:

- FD's determine Super key.
- X is a Super Key of R iff X^+ should determine all the attribute of Relation R.
- **Example:** R(ABCD), F = { A $\longrightarrow$ B, B $\longrightarrow$ C, C $\longrightarrow$ D }
- $[A]^+$ = { A, B, C, D}
- Hence, A is one of the super keys.

Candidate Key:

- Minimal Super key is Candidate Key.

- X is candidate key of Relation R if and only if

1. X must be super key of R. AND
2. No proper Subset of attribute X is Super Key.

- **Prime attributes and Non-Prime Attributes:** Attributes of relations which are part of candidate key are known as Prime Attributes and attributes which are not part of candidate keys are known ad non-prime attributes.

Candidate Key:

Example:

- R(ABCD), F = { AB $\rightarrow$ C, C $\rightarrow$ D, B $\rightarrow$ E }
- $[AB]^+$ = { A, B, C, D, E}
- Hence, AB is one of the super keys.

- To check if AB is Candidate key or not, we have to check closure set of minimal subset of AB, i.e $[A]^+$ and $[B]^+$
- $[A]^+$ = {A} (**NOT SUPERKEY/CANDIDATE KEY)**
- $[B]^+$ = {BE} (**NOT SUPERKEY/CANDIDATE KEY)**

Candidate Key:

Note 1: If attribute is in R but not in non-trivial FD set, then that attribute must be part of Candidate key.

Example:

- R(ABCDEF), F = { A $\rightarrow$ D, F $\rightarrow$ E, C $\rightarrow$ F, F $\rightarrow$ A }
- Here, B is not part of non-trivial FD set so it must be part of candidate key.
- $[C]^+$ = { A, C, E, F}
- **Candidate key:** { BC }
- **Prime attributes:** B and C

Parul® University

DIGITAL LEARNING CONTENT

Candidate Key:

Note 2: If some attribute X is only part of determinant side but not belongs to any determiner of non-trivial FD set of relation R then X must be part of Candidate key.

Example:

- R(ABCDEF), F = { AB ⟶ C, E ⟶ F, D ⟶ C, C ⟶ B }
- Here, A,E are at the determinant side only.
- **Candidate key:** { ABE}
- **Prime attributes:** A,B and E

Parul® University

DIGITAL LEARNING CONTENT

Candidate Key:

Exercise: Find Candidate Keys of below relations.

1.R(ABCDE), F = { AB $\longrightarrow$ B, BC $\longrightarrow$ D, CD $\longrightarrow$ E, E $\longrightarrow$ A}

2. R(ABCD), F = { AB $\longrightarrow$ CD, C $\longrightarrow$ A, D $\longrightarrow$ B}

3.R(ABCDEF), F = { AB $\longrightarrow$ C, C $\longrightarrow$ DE, E $\longrightarrow$ F, EF $\longrightarrow$ B, E $\longrightarrow$ A}

4.R(ABCDEFG), F = { AB $\longrightarrow$ CD, AF $\longrightarrow$ D, DE $\longrightarrow$ F, C $\longrightarrow$ G, F $\longrightarrow$ E, G $\longrightarrow$ A}

Normalization:

- Normalization is the procedure to reduce the Redundancy.
- Redundancy can occur in Relation if two or more independent relations are stored in the single table.
- For example, in the given example, two independent relations Student relation and Course relation is stored in single table.

Normalization:

- **Example of Redundancy:**

Table 1.2: Example of Redundancy in Database table.

Sid	Sname	Age	cid	Cname	fees
S1	A	20	C1	DBMS	5000
S2	A	22	C1	DBMS	5000
S3	B	22	C1	DBMS	5000
S2	A	22	C3	DAA	7000
S2	A	22	C4	DS	7000

Redundancy

Redundancy

Normalization:

Anomalies in Database Design:

- Due to redundancy in Database, Inconsistency (Anomalies) occurs.
- Anomalies are problems that can occur in poorly planned, un-normalized database where all the data are stored in one table.
- There are three types of anomalies that can arise in the database because of redundancy are:

1. Insert anomalies, occur due to Insert Operation.
2. Delete anomalies, occur due to Delete Operation.
3. Update / Modification anomalies, occur due to update operation.

Normalization:

Insert Anomaly:

- Consider previous example, Student_Course(Sid, Sname, age, Cid, Cname, fees) as Sid as a primary key.
- Let us assume that a new course has been started by the Institute but initially there are no students enrolled for that course.
- Then the tuple for this course cannot be inserted in to this table as the Sid will have NULL value, which is not allowed because Sid is primary key.
- This kind of problem in the relation where **some tuple cannot be inserted** is known as insert anomaly.

Parul® University

DIGITAL LEARNING CONTENT

Normalization:

Insert Anomaly:

Table 1.3 : Insert Anomaly

	Sid	Sname	Age	cid	Cname	fees
	S1	A	20	C1	DBMS	5000
	S2	A	22	C1	DBMS	5000
	S3	B	22	C1	DBMS	5000
	S2	A	22	C3	DAA	7000
	S2	A	22	C4	DS	7000
Insert Anomaly	NULL	NULL	NULL	C5	CD	8000

Normalization:

Delete Anomaly:

- Consider previous example, Student_Course(Sid, Sname, age, Cid, Cname, fees) as Sid as a primary key.
- Now consider there is one student in and that employee leaves the Institute.
- Then the tuple of that student has to be deleted from the table, but in addition to that information about the course also will be deleted.
- This kind of problem in the relation where **deletion of some tuples can lead to loss of some other data not intended to be removed** is known as delete anomaly.

Normalization:

Delete Anomaly:

- If student C leaves the institute, then we have to delete information related to that particular student. If we delete that, information regarding course C3 will be deleted.

Table 1.4 : Deletion Anomaly

Sid	Sname	Age	cid	Cname	fees
S1	A	20	C1	DBMS	5000
S2	A	22	C1	DBMS	5000
S3	B	22	C1	DBMS	5000
S2	A	22	C3	DAA	7000
S2	A	22	C4	DS	7000

Normalization:

Update Anomaly:

- Consider previous example, Student_Course(Sid, Sname, age, Cid, Cname, fees) as Sid as a primary key.
- Suppose the fees of a course has changed, this requires that the course in all the tuples corresponding to that course must be changed to reflect the new status.
- If we fail to update all the tuples of given course, then two different records of the student enrolled in the same course might show different fees which leads to inconsistency in the database.
- This kind of problem is known as **update or modification anomaly**.

Normalization:

Update Anomaly:

Table 1.5 : Update Anomaly

Sid	Sname	Age	cid	Cname	fees
S1	A	20	C1	DBMS	6000
S2	A	22	C1	DBMS	5000
S3	B	22	C1	DBMS	5000
S2	A	22	C3	DAA	7000
S2	A	22	C4	DS	7000

Update Anomaly

Normalization:

Need of Normalization:

- Eliminates redundant data
- Reduces chances of data errors
- Reduces disk space
- Improve data integrity, scalability and data consistency.

Normalization:

Normal forms:

1 NF (First normal form)
2NF (Second normal form)
3NF (third normal form)
BCNF

} 0% redundancy over Single valued FD’s. (X⟶Y)
Redundancy can be possible because of MVD’s. (X⟶ ⟶ Y)

4NF (forth normal form)
5NF (fifth normal form)

} No Redundancy because of MVD’s also

Normalization:

1NF:

Relational schema R is in 1 NF if no multivalued attribute in Relation R.

Or

Every attribute of relation R must be single valued.

Normalization:

1NF:

- Consider below relation.

Table 1.6: 1 NF Example

CustomerID	Name	Address
C01	A	Karelibag, Vadodara
C02	B	Satellite, Ahmedabad

- Above relation has three attributes CustomerID, Name, Address.
- Here address is composite attribute which is further divided in to sub attributes as Area and City.
- So above relation is not in 1NF.

Normalization:

1NF:

- Consider below relation.

Table 1.6: 1 NF Example

CustomerID	Name	Address
C01	A	Karelibag, Vadodara
C02	B	Satellite, Ahmedabad

- **Problem:**
- It if difficult to retrieve the list of Customers living in ‘Vadodara’ from above table.
- Reason is address attribute is composite attribute which contains name of the area as well as city name in single cell.

Normalization:

1NF:

- **Solution:**
- Divide composite attributes into number of sub- attribute and insert value in proper sub attribute.

Table 1.7: 1 NF Example solution

Customer ID	Name	Area	City
C01	A	Karelibag	Vadodara
C02	B	Satellite	Ahmedabad

Normalization:

1NF:

- Default NF for RDBMS relation is 1 NF.
- Redundancy occurs due to 2NF, 3NF, BCNF in single valued FD’s.

Normalization:

How Redundancy can occur in any given FD $X \longrightarrow Y$?

- If non-trivial FD $X \longrightarrow Y$ with X is not Super Key in R then $X \longrightarrow Y$ form Redundancy in R.

$$\text{NOT SUPERKEY} \longrightarrow X \rightarrow \textbf{Y : Non-trivial FD} \;\Big\}\; \text{Forms redundancy in Relation}$$

Normalization:

How Redundancy can occur in any given FD X → Y?

•For example, consider below relation:

Table 1.8 : Redundancy in database table

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S1	A	C3

Redundancy

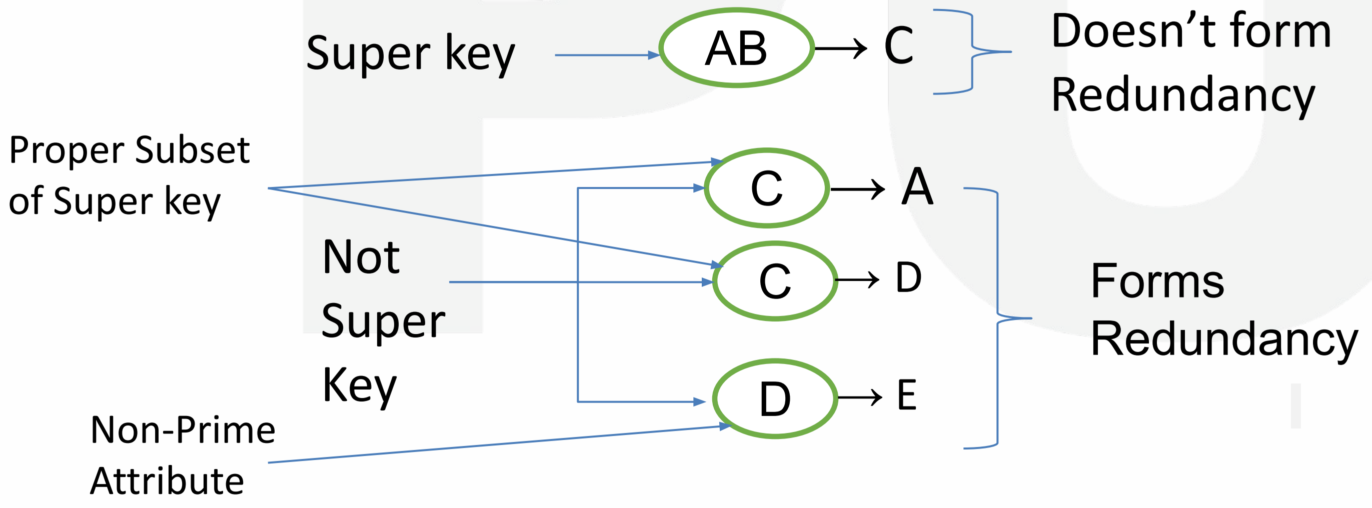
•Super key for this relation is {sidcid}.

•Consider FD, Sid → Sname. Sid is not Super Key for this relation. Hence we can see redundancy is the all the tuples in the relation.

Normalization:

How many FD's of F form Redundancy?

- R(ABCDE), F = { AB ⟶ C, C ⟶ A, C ⟶ D, D ⟶ E }
- **Super key/Candidate key: { AB,BC}**



Parul® University

DIGITAL LEARNING CONTENT

Normalization:

X ⟶ Y, non-trivial FD in R with X not Super key can be,

Case 1: Proper Subset of CK (X) → Non-Prime attribute (Y) E.g. C ⟶ D

Case 2: Non-Prime attribute (X) → Non-Prime attribute (Y) E.g. D ⟶ E

Case 3: Proper Subset of CK (X) → Proper Subset of CK (Y) E.g. C ⟶ A

Normalization:

Summary:

Table 1.9 : Summary of Normal forms

	1 NF	2 NF	3 NF	BCNF
Case 1	Yes	yes	No	No
Case 2	Yes	Yes	No	No
Case 3	Yes	Yes	Yes	no

Parul® University

DIGITAL LEARNING CONTENT

Normalization:

2NF:

A relation schema R is in second normal form (2NF) if and only if it is in 1NF

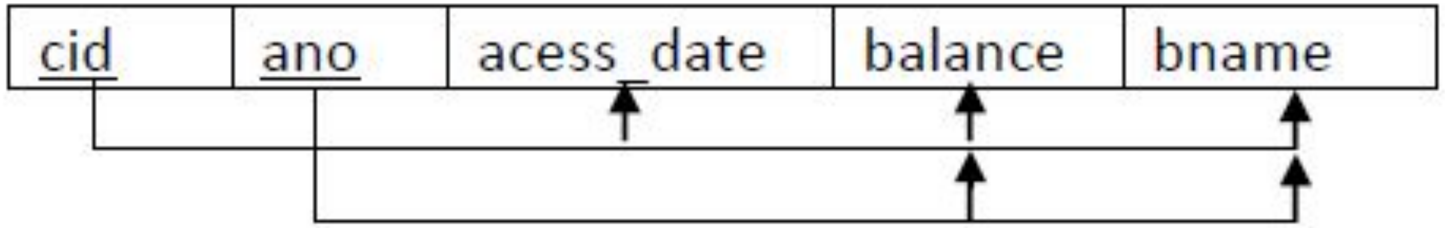
and

no any non-key attribute is **partially dependent on the primary key**.

PU

Normalization:

2NF Example:



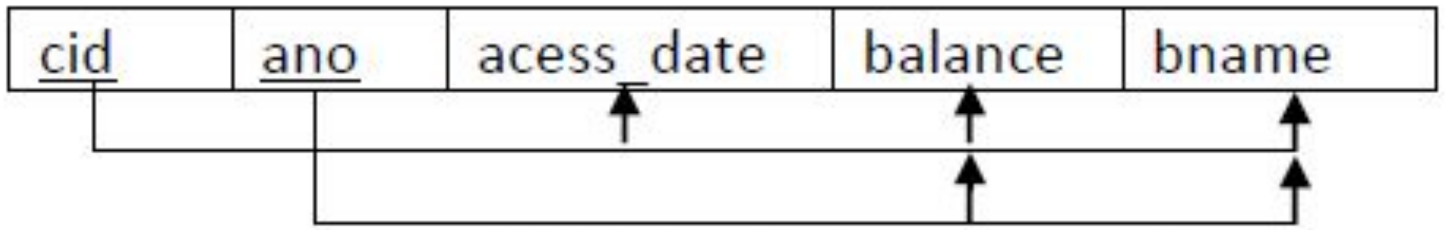
- Above relation has five attributes cid, ano, acess_date, balance, bname and two FDS

  FD1 : {cid,ano} -> {acess_date,balance,bname} and

  FD2 : {ano} -> {balance,bname}

Normalization:

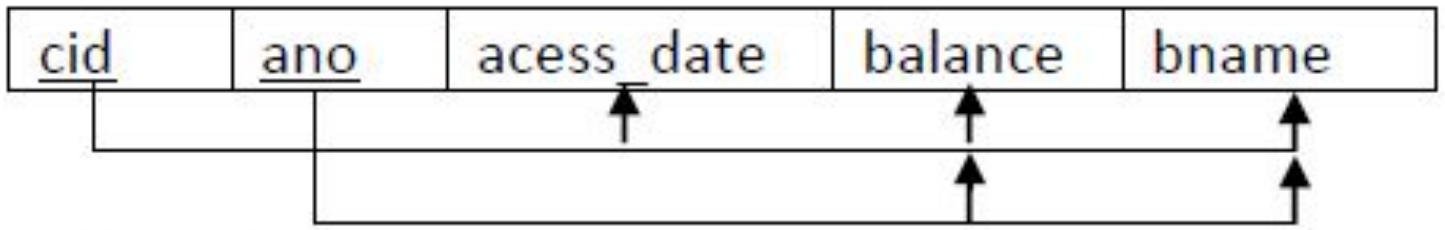
2NF Example:



- We have cid and ano as primary key.
- As per FD2 **balace and bname are only depend on ano not cid.**
- In above table **balance and bname are partial dependent on primary key**.
- So above relation is **not in 2NF**.

Normalization:

2NF Example:

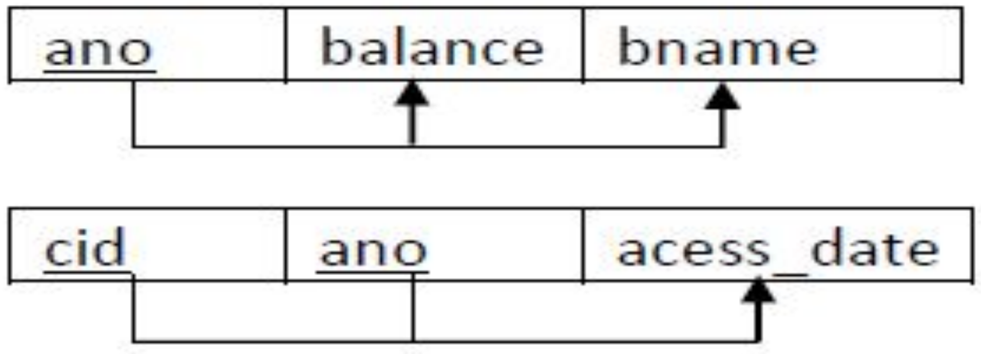


Problem:

- For example in case of joint account multiple customers have common accounts.
- If some account says ‘A02’ is jointly by two customers says ‘C02’ and ‘C04’ then data values for attributes balance and bname will be duplicated in two different tuples of customers ‘C02’ and ‘C04’.

Normalization:

2NF Example:



Solution:

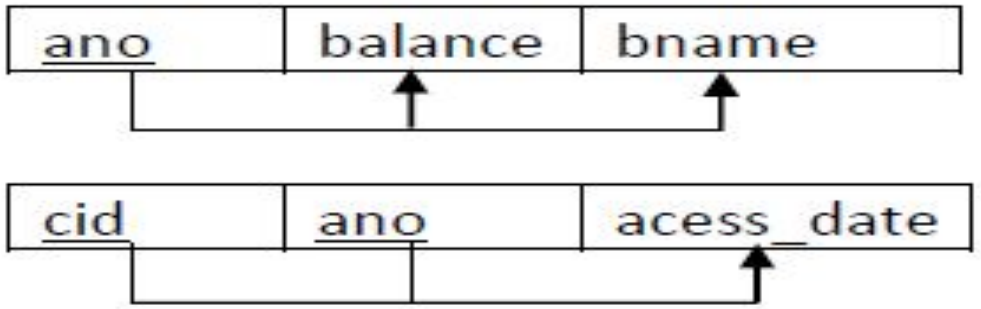
- Decompose relation in such a way that resultant relation does not have any partial FD.
- For this purpose **remove partial dependent attribute that violets 2NF from relation**.

Parul® University

DIGITAL LEARNING CONTENT

Normalization:

2NF Example:



Solution:

- **Place them in separate new relation along with the prime attribute** on which they are full dependent.
- The primary key of new relation will be the attribute on which it if fully dependent.
- Keep other attribute same as in that table with same primary key.

Normalization:

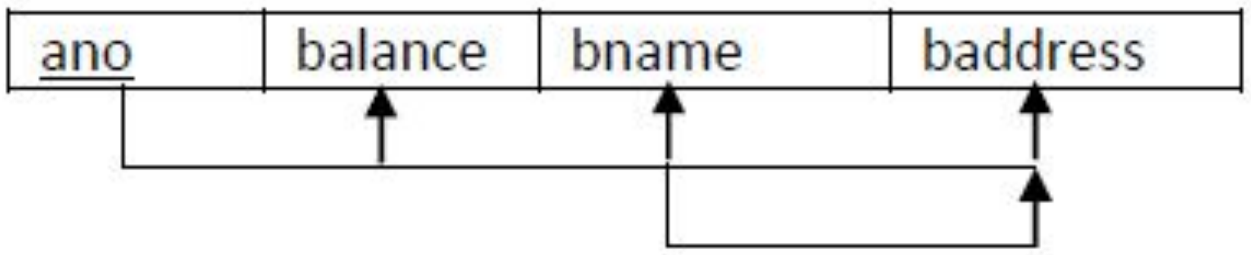
3NF:

A relation schema R is in third normal form (3NF) if and only if it is in 2NF and no transitive dependencies in R.

PU

Normalization:

3NF Example:



- Above relation has five attributes ano, balance, bname and baddress and two FDS

  FD1 : ano -> {balance, bname, baddress} and

  FD2 : bname -> baddress

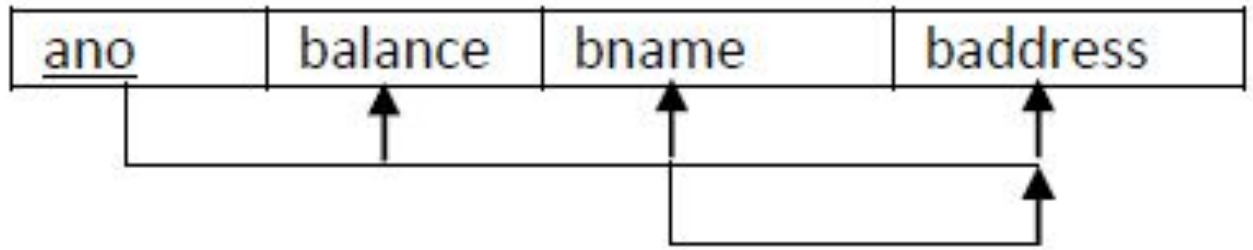
- Using transitivity rule, ano -> baddress

Parul® University

DIGITAL LEARNING CONTENT

Normalization:

3NF Example:



- So there is a **non-prime attribute baddress which is transitively dependent on primary key ano.**
- So above relation is **not in 3NF.**

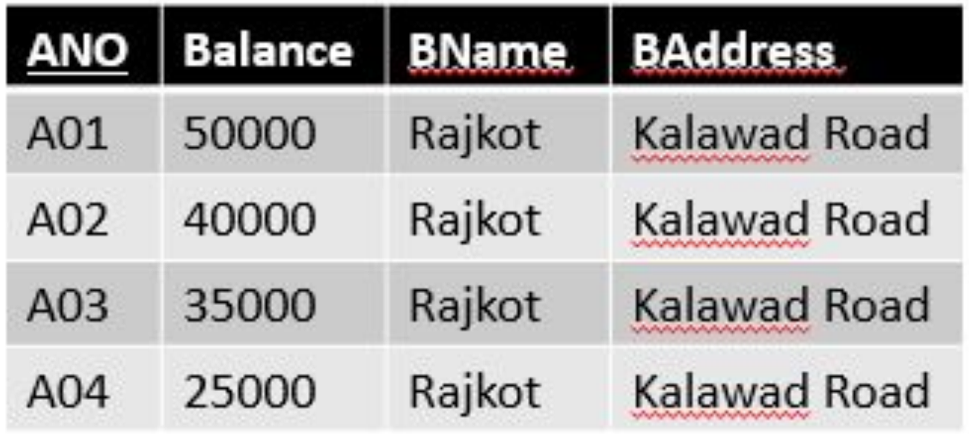
Parul® University

DIGITAL LEARNING CONTENT

Normalization:

3NF Example:

Table 1.10 : 3NF Example



ANO	Balance	BName	BAddress
A01	50000	Rajkot	Kalawad Road
A02	40000	Rajkot	Kalawad Road
A03	35000	Rajkot	Kalawad Road
A04	25000	Rajkot	Kalawad Road

Problem:

- Transitively dependency results in data redundancy.
- In this relation branch address will be stored repeatedly from each account of same branch which occupy more space.

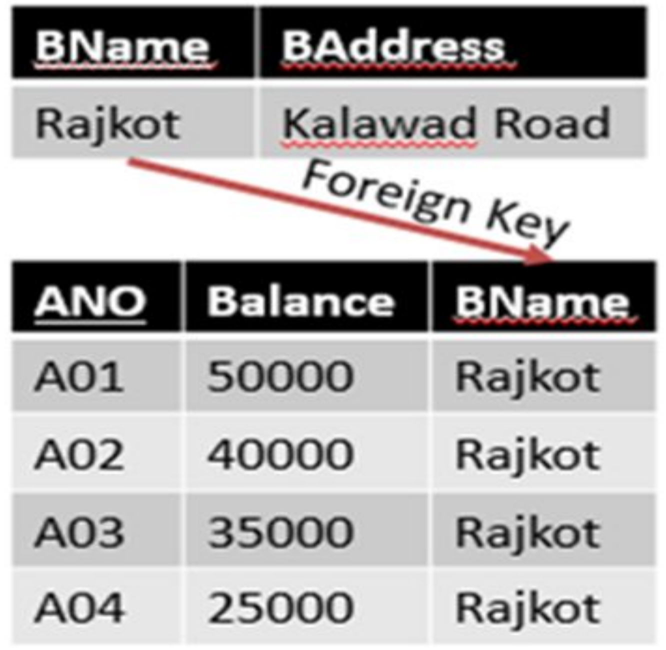
Normalization:

3NF Example:

Solution:

- Decompose relation in such a way that resultant relation does not have any non-prime attribute that are transitively dependent on primary key.
- For this purpose remove transitively dependent attribute that violets 3NF from relation.

Table 1.11 : 3NF Solution



BName	BAddress
Rajkot	Kalawad Road

Foreign Key

ANO	Balance	BName
A01	50000	Rajkot
A02	40000	Rajkot
A03	35000	Rajkot
A04	25000	Rajkot

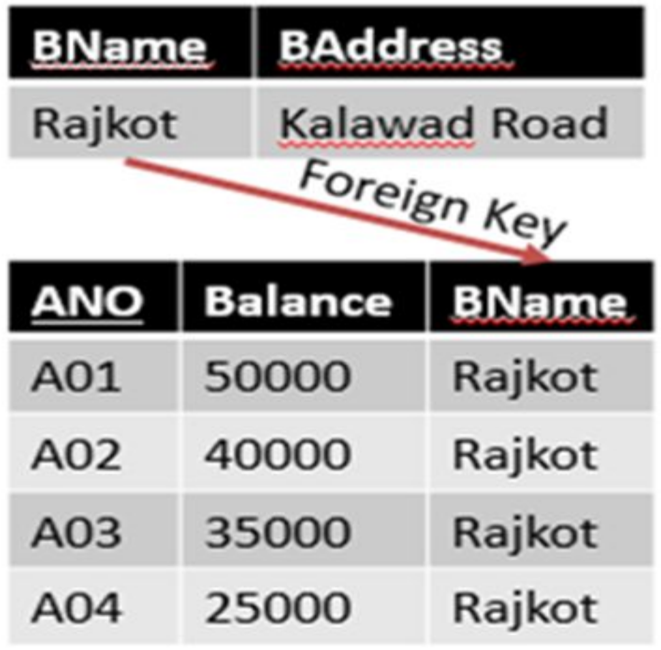
Normalization:

3NF Example:

Solution:

- Place them in separate new relation along with the non-prime attribute due to which transitive dependency occurred. primary key of new relation will be this non-prime attribute.
- Keep other attributes same as in that table with same primary key.

Table 1.11 : 3NF Solution



BName	BAddress
Rajkot	Kalawad Road

Foreign Key

ANO	Balance	BName
A01	50000	Rajkot
A02	40000	Rajkot
A03	35000	Rajkot
A04	25000	Rajkot

Normalization:

The 4NF :-comes after 1NF, 2NF, 3NF, and Boyce-Codd Normal Form. It was introduced by Ronald Fagin in 1977.

To be in 4NF, a relation should be in Bouce-Codd Normal Form and may not contain more than one multi-valued attribute.

Example

Let us see an example − <Movie>

Movie_Name	Shooting_Location	Listing
MovieOne	UK	Comedy
MovieOne	UK	Thriller
MovieTwo	Australia	Action
MovieTwo	Australia	Crime
MovieThree	India	Drama

Normalization:

The above is not in 4NF, since

- More than one movie can have the same listing
- Many shooting locations can have the same movie

Let us convert the above table in 4NF –

<Movie_Shooting>

Movie_Name	Shooting_Location
MovieOne	UK
MovieOne	UK
MovieTwo	Australia
MovieTwo	Australia
MovieThree	India

<Movie_Listing>

Movie_Name Listing

Normalization:

<Movie_Listing>

Movie_Name	Listing
MovieOne	Comedy
MovieOne	Thriller
MovieTwo	Action
MovieTwo	Crime
MovieThree	Drama

Now the violation is removed and the tables are in 4NF.

Normalization:

5NF: It is also called as project join normal form. A relation R is in 5NF if it is in 4NF and there is no join dependency. No join dependency means lossless-join decomposition.
Decomposition is lossless-join decomposition if it preserves all the data in original relation and does not result in additional tuples.
A relation R satisfies join dependency if R is equal to the join of R1,R2,.....Rn where Ri are a subset of the set of attributes of R.
Relation R

Dept	Subject	Name
CSE	C	Ammu
CSE	C	Amar
CSE	Java	Amar
IT	C	Bhanu

Normalization:

Therefore the relation can be decomposed into following three relations –
R1(dept, subject)
R2(dept, name) and
R3(subject, name) and it can be shown that decomposition is lossless.

R1

Dept	Subject
CSE	C
CSE	Java
IT	C

Normalization:

R2

Dept	Name
CSE	Ammu
CSE	Amar
IT	Bhanu

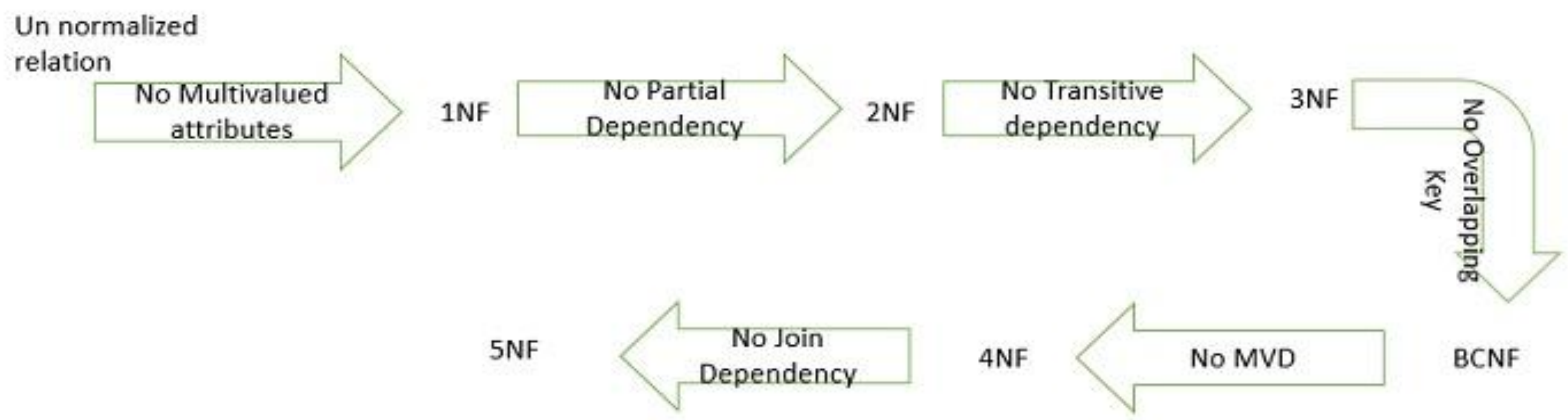
R3

Subject	Name
C	Ammu
C	Amar
Java	Amar
C	Bhanu

In above decomposition of table reduction of redundancy is not apparent, but it can be realized when tables contain large amounts of data.

Normalization:

The above relation is in 4NF. Anomalies can occur in relation in 4NF if the primary key has three or more fields. The primary key is (dept,subject, name). Sometimes decomposition of a relation into two smaller relations does not remove redundancy. In such cases it may be possible to decompose the relation in three or more relations using 5NF.



Normalization:

Boyce Codd Normal Form (BCNF):

A relation R is in BCNF if and only if it is in 3NF and **for every functional dependency X ⟶ Y, X should be the primary key of the table**.

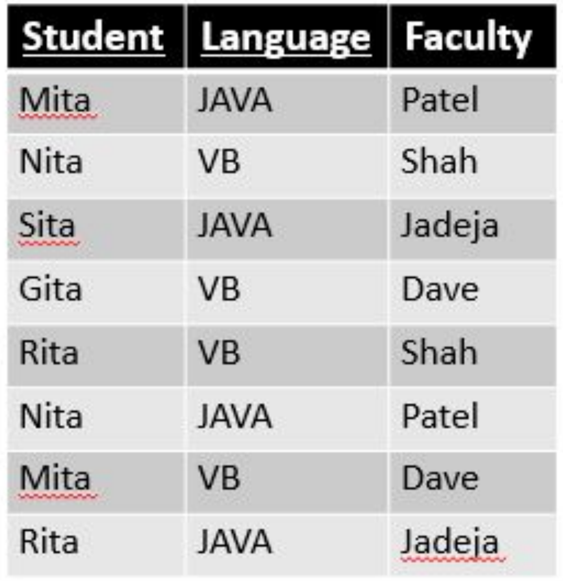
Parul® University

Normalization:

BCNF Example:

- Let R be a relation with following three attributes: student, language, Faculty
- FD1 : **{student, language}** → **Faculty**
- FD2 : **Faculty** → **language**
- Using transitivity rule, **student** → **language**
- So above relation is **not in BCNF.**

Table 1.12: BCNF Example



Student	Language	Faculty
Mita	JAVA	Patel
Nita	VB	Shah
Sita	JAVA	Jadeja
Gita	VB	Dave
Rita	VB	Shah
Nita	JAVA	Patel
Mita	VB	Dave
Rita	JAVA	Jadeja

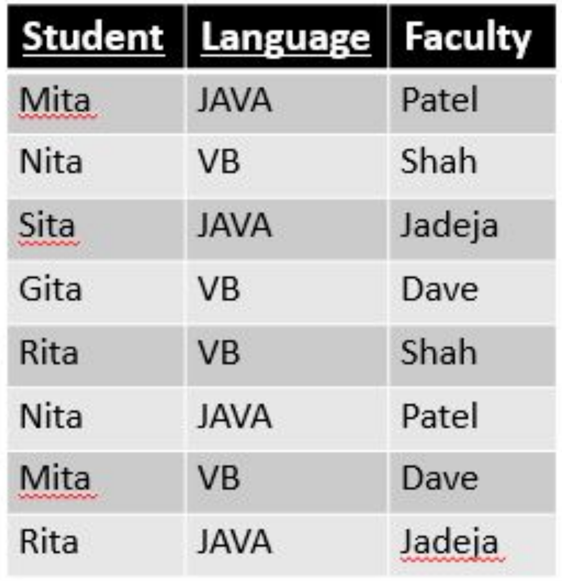
Normalization:

BCNF Example:

Problem:

- Transitively dependency results in data redundancy.
- In this relation one student have more than one language with different faculty then records will be stored repeatedly from each student and language and faculties combination which occupies more space.

Table 1.12: BCNF Example



Student	Language	Faculty
Mita	JAVA	Patel
Nita	VB	Shah
Sita	JAVA	Jadeja
Gita	VB	Dave
Rita	VB	Shah
Nita	JAVA	Patel
Mita	VB	Dave
Rita	JAVA	Jadeja

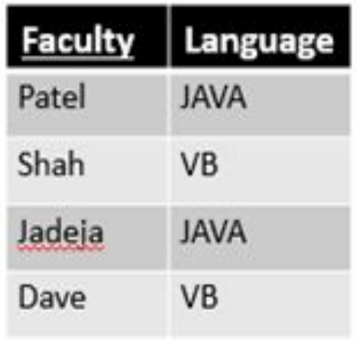
Normalization:

BCNF Example:

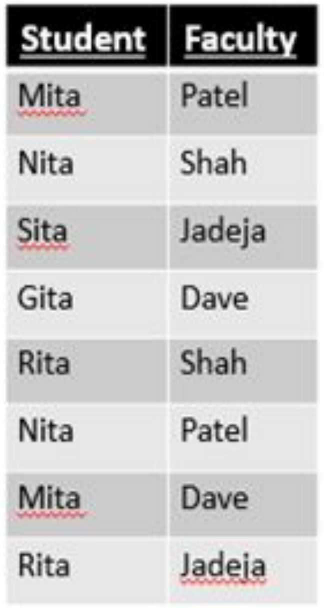
Solution:

- Decompose relation in such a way that for X → Y, X should be the primary key of the table.
- For this purpose remove prime attribute, place them in separate new relation along with the non prime attribute on which it is dependent. The primary key of new relation will be this nonprime attribute.

Table 1.13: BCNF Solution



Faculty	Language
Patel	JAVA
Shah	VB
Jadeja	JAVA
Dave	VB



Student	Faculty
Mita	Patel
Nita	Shah
Sita	Jadeja
Gita	Dave
Rita	Shah
Nita	Patel
Mita	Dave
Rita	Jadeja

Normalization:

Examples: Find out Highest NF of given Relation.

1.R(ABCDE), F = { ABC ⟶ D, D ⟶ A, BD ⟶ E}

Find out Candidate key: Candidate keys : { ABC, BCD}
Prime attributes: A, B, C, D. Non-Prime attributes: E

	BCNF	3NF	2NF
ABC ⟶ D	Yes	Yes	Yes
D ⟶ A	No	Yes	Yes
BD ⟶ E	No	No	yes

Highest NF of given Relation : **2 NF**

Normalization:

Examples: Find out Highest NF of given Relation.

2.R(ABCDE), F = { A ⟶ B, B ⟶ AC, C ⟶ DE}

Find out Candidate key: Candidate keys : {A, B}
Prime attributes: A, B. Non-Prime attributes: C, D, E

	BCNF	3NF	2NF
A ⟶ B	Yes	Yes	Yes
B ⟶ AC	Yes	Yes	Yes
C ⟶ DE	No	No	Yes

Highest NF of given Relation : **2 NF**

Normalization:

Examples: Find out Highest NF of given Relation.

3.R(ABCDE), F = { A ⟶ BC, CD ⟶ E, B ⟶ D, E ⟶ A}

Find out Candidate key: Candidate keys : {A, E, CD, BC}

Prime attributes: A, B, C, D, E. Non-Prime attributes: none

	BCNF	3NF	2NF
A ⟶ BC	Yes	Yes	Yes
CD ⟶ E	Yes	Yes	Yes
E ⟶ A	Yes	Yes	Yes
B ⟶ D	No	Yes	Yes

Highest NF of given Relation : **3 NF**

Normalization:

Question:

- **A college maintains details of its lecturers' subject area skills. These details comprise: Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name, Subject Code, Subject Name and Subject Level. Assume that each lecturer may teach many subjects but may not belong to more than one department. Subject Code, Subject Name and Subject Level are repeating fields. Normalize this data to Third Normal Form.**

Parul® University

DIGITAL LEARNING CONTENT

Normalization:

Solution:

- UNF

    Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name, Subject Code, Subject Name, Subject Level

- After removing multi valued attributes we get **1NF**
    - Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name
    - Lecturer Number, **Subject Code, Subject Name, Subject Level** (Partial Dependency)

Normalization:

Solution:

- After removing partial dependency we get **2NF**
  - Lecturer Number, Lecturer Name, Lecturer Grade, **Department Code, Department Name (**Transitive Dependency)
  - Lecturer Number, Subject Code
  - Subject Code, Subject Name, Subject Level
- After removing transitive dependency we get **3NF**
  - Lecturer Number, Lecturer Name, Lecturer Grade, Department Code
  - Department Code, Department Name
  - Lecturer Number, Subject Code
  - Subject Code, Subject Name, Subject Level

Decomposition:

- **Properties of Decomposition:**

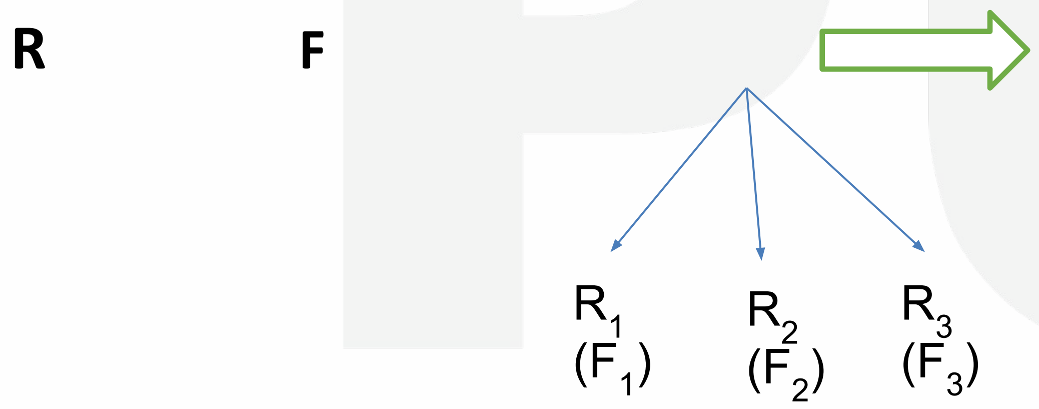
1) Dependency Preserving Decomposition
2) Lossless join Decomposition
3) Lossy Decomposition

Parul® University
DIGITAL LEARNING CONTENT

Decomposition:

Dependency Preserving Decomposition:

- Let R be the Relational schema with FD set F Decomposed into R_1, R_2, R_3…… R_n sub relations with FD set F_1, F_2, F_3,…….. F_n respectively.



Decomposition:

Dependency Preserving Decomposition:

In general,

$[F_1 \cup F_2 \cup F_3 \cup \ldots\ldots F_n] \subseteq F$

If $[F_1 \cup F_2 \cup F_3 \cup \ldots\ldots F_n] = F$ then it is DP Decomposition.

If $[F_1 \cup F_2 \cup F_3 \cup \ldots\ldots F_n] \subset F$ then it is NOT DP Decomposition.

Decomposition:

Example 1:

R(ABCDE) is decomposed into D = { AB, BC, CD, DE }
F = { $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$, $D \rightarrow B$ }

Step 1 : Find FD's of Sub relations:
R_1 (AB) : (F_1)

$[A]^+$ = { A, B, C, D, E}   [We can derive B from A. So $A \rightarrow B$. C,D,E are not in R_1]
$[B]^+$ = { B, C, D, E}   [We cannot derive A from B.]

Parul® University

DIGITAL LEARNING CONTENT

Decomposition:

Example 1:

R_2 (BC) : (F_2)
$[B]^+$ = { B, C, D, E} [We can derive C from B. So B $\longrightarrow$ C. A,D,E are not in R_2]
$[C]^+$ = { B, C, D, E} [We cannot derive B from C. So C $\longrightarrow$ B. A,D,E are not in R_2]
R_3 (CD) : (F_3)
$[C]^+$ = { B, C, D, E} [We can derive D from C. So C $\longrightarrow$ D. A,B,E are not in R_2]
$[D]^+$ = { B, C, D, E} [We cannot derive C from D. So D $\longrightarrow$ C. A,B,E are not in R_2]
R_4 (DE) : (F_4)
$[D]^+$ = { B, C, D, E} [We can derive E from D. So D $\longrightarrow$ E. A,B,E are not in R_2]
$[E]^+$ = {E} [We cannot derive E from D.]

Decomposition:

Example 1:

$\{F_1 \cup F_2 \cup F_3 \cup F_4\} = \{A \longrightarrow B, B \longrightarrow C, C \longrightarrow B, C \longrightarrow D, D \longrightarrow C, D \longrightarrow E\}$

Now, $F = \{ A \longrightarrow B, B \longrightarrow C, C \longrightarrow D, D \longrightarrow E, D \longrightarrow B \}$

We can say that If $[F_1 \cup F_2 \cup F_3 \cup \ldots\ldots F_n] = F$.

This Decomposition is **Dependency Preserving Decomposition.**

Decomposition:

Example 2:

R(ABCD) is decomposed into D = { ABC, AD, BCD }
F = { AB $\longrightarrow$ CD, D $\longrightarrow$ A}

Step 1 : Find FD's of Sub relations:

R_1 (ABC) : (F_1)
$[AB]^+$ = { A, B, C, D} [We can derive C from A and B. So AB $\longrightarrow$ C. D is not in R_1]

Decomposition:

Example 2:

R_2 (AD) : (F_2)

$[A]^+ = \{A\}$ [We cannot derive D from A.]

$[D]^+ = \{ A,D\}$ [We can derive A from d. So D $\rightarrow$ A. B,C are not in R_2]

R_3 (BCD) : (F_3)

$[B]^+ = \{B\}$ [We cannot derive C and D from B. A,E are not in R_3]

$[C]^+ = \{C\}$ [We cannot derive B and D from C. A,E are not in R_3]

$[D]^+ = \{DA\}$ [We cannot derive B and C from D. A,E are not in R_3]

$[BD]^+ = \{ A, B, C, D\}$ [We can derive C from BD. A ,E is not in R_3]

Decomposition:

Example 2:

$\{F_1 \cup F_2 \cup F_3\} = \{AB \longrightarrow C, D \longrightarrow A, BD \longrightarrow C\}$

Now, $F = \{AB \longrightarrow CD, D \longrightarrow A\}$

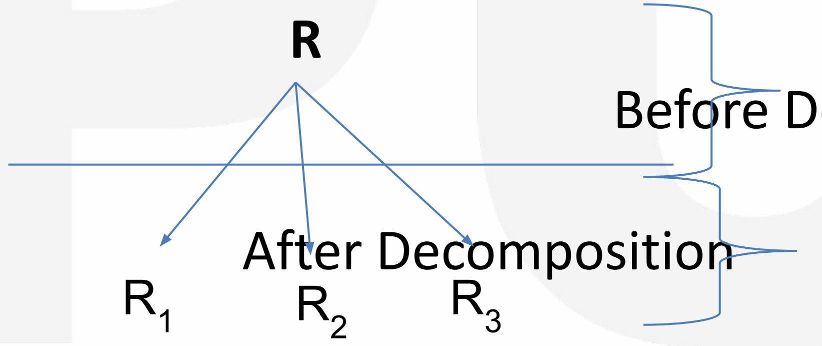
We can say that If $[F_1 \cup F_2 \cup F_3 \cup \ldots\ldots F_n] \subset F$.

This Decomposition is **NOT Dependency Preserving Decomposition.**

Decomposition:

Lossless join Decomposition:

- Let R be the Relational schema with FD set F Decomposed into R_1, R_2, R_3…… R_n sub relations.



Decomposition:

Lossless join Decomposition:

In general,

$[R_1 \bowtie R_2 \bowtie R_3 \bowtie \ldots\ldots \bowtie R_n] \supseteq R$

If $[R_1 \bowtie R_2 \bowtie R_3 \bowtie \ldots\ldots \bowtie R_n] = F$ then it is lossless join Decomposition.

If $[R_1 \bowtie R_2 \bowtie R_3 \bowtie \ldots\ldots \bowtie R_n] \supset F$ then it is lossy Decomposition.

Decomposition:

Lossless join Decomposition:

- Consider the below Relation,

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S3	B	C3

{ Sid → Sname}

Candidate key: {SidCid}

Parul® University

DIGITAL LEARNING CONTENT

Decomposition:

Lossless join Decomposition:

- This relation is decomposed into R_1 (SidSname) and R_2 (SidCid)

R_1 (SidSname)

Sid	Sname
S1	A
S2	B
S3	B

{Sid}: key

R_2 (SidCid)

Sid	Cid
S1	C1
S1	C2
S2	C2
S3	C3

$R_1 \bowtie R_2$

{SidCid}: key

R(SidSnameCid)

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S3	B	C3

$R_1 \bowtie R_2$ = R, so it is LOSSLESS JOIN DECOMPOSTION.

Parul® University

DIGITAL LEARNING CONTENT

Decomposition:

Lossless join Decomposition:

- This relation is decomposed into R_1 (SidSname) and R_2 (SnameCid)

R_1 (SidSname)

Sid	Sname
S1	A
S2	B
S3	B

{Sid}: key

R_2 (SnameCid)

Sname	Cid
A	C1
A	C2
B	C2
B	C3

{SnameCid}: key

$R_1 \bowtie R_2$

R(SidSnameCid)

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S2	B	C3
S3	B	C3
S3	B	C2
S3	B	C3

$R_1 \bowtie R_2 \supset$ R, so LOSSY JOIN DECOMPOSTION.

Decomposition:

Lossless join Decomposition:

- Common attribute should be the candidate key of one of the table.

- Relational schema R with FD set F decomposed into R_1 & R_2 Decomposition is lossless if and only if,

a) $R_1 \cap R_2 \rightarrow R_1$ $(R_1 \cap R_2)$: Candidate key for R_1 **OR**

b) $R_1 \cap R_2 \rightarrow R_2$ $(R_1 \cap R_2)$: Candidate key for R_2

- Union of attributes of R_1 and R_2 must be equal to the attributes of R.

- Intersection of attributes in R_1 and R_2 must not be NULL.

Decomposition:

Example:

R(ABCDE), F = { AB $\longrightarrow$ C, C $\longrightarrow$ D, B $\longrightarrow$ E}

Relation R is decomposed into,

1)D = { ABC, CD }

$R_1(ABC) \cap R_2(CD) = C$

$[C]^+ = \{ C,D \}$

Here, union of attributes of R_1 and R_2 is not equal to attributes of R.

Hence, this decomposition is lossy join decomposition.

Parul® University

DIGITAL LEARNING CONTENT

Decomposition:

Example:

R(ABCDE), F = { AB $\longrightarrow$ C, C $\longrightarrow$ D, B $\longrightarrow$ E}

Relation R is decomposed into,

2)D = { ABC, DE }

Here, Intersection of attributes in R_1 and R_2 is NULL.

Hence, this decomposition is lossy join decomposition.

Decomposition:

Example:

R(ABCDE), F = { AB ⟶ C, C ⟶ D, B ⟶ E}

Relation R is decomposed into,

3)D = { ABC, CDE }

$R_1(ABC) \cap R_2(CDE) = C$

$[C]^+ = \{ C,D \}$

C is NOT candidate key for either of R_1 & R_2.

Hence, this decomposition is lossy join decomposition.

Decomposition:

Example:

R(ABCDE), F = { AB ⟶ C, C ⟶ D, B ⟶ E}

Relation R is decomposed into,

4)D = { ABCD, BE }

$R_1(ABCD) \cap R_2(BE) = B$

$[B]^+ = \{ B,E \}$

B is candidate key for $R_2(BE)$.

Hence, this decomposition is lossy join decomposition.

Decomposition:

Example:

R(ABCDE), F = { $AB \rightarrow C$, $C \rightarrow D$, $B \rightarrow E$}

Relation R is decomposed into,

5)D = { ABC, ABDE }

$R_1(ABC) \cap R_2(ABDE) = AB$

$[AB]^+ = \{ A,B,C,D,E \}$

AB is candidate key for both tables R_1 (ABC) and R_2(ABDE).

Hence, this decomposition is lossless join decomposition.

Decomposition:

Example:

R(ABCDE), F = { AB $\longrightarrow$ C, C $\longrightarrow$ D, B $\longrightarrow$ E}

Relation R is decomposed into,

6)D = { ABC, CD, BE }

$R_1(ABC) \cap R_2(CD) = C$

$[C]^+ = \{ C,D \}$

CD is candidate key for either of $R_2(CD)$.

$R_1 \bowtie R_2\ (ABCD) \cap R_3(BE) = B$, $[B]^+ = \{ B,E \}$

B is candidate key for $R_3(BE)$.

Hence, this decomposition is lossless join decomposition.

Decomposition:

Example:

R(ABCDE), F = { AB ⟶ C, C ⟶ D, B ⟶ E}

Relation R is decomposed into,

7)D = { ABC, CD, DE }

$R_1(ABC) \cap R_2(CD) = C$

$[C]^+ = \{ C,D \}$

CD is candidate key for either of $R_2(CD)$.

$R_1 \bowtie R_2 (ABCD) \cap R_3(DE) = D$, $[D]^+ = \{ D \}$

D is NOT candidate key for any table.

Hence, this decomposition is lossy join decomposition.

Decomposition:

Lossy Decomposition

As the name suggests, when a relation is decomposed into two or more relational schemas, the loss of information is unavoidable when the original relation is retrieved.

Let us see an example –

<EmpInfo>

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Jacob	29	Alabama	Dpt1	Operations
E002	Henry	32	Alabama	Dpt2	HR
E003	Tom	22	Texas	Dpt3	Finance

Decomposition:

Decompose the above table into two tables −

<EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Jacob	29	Alabama
E002	Henry	32	Alabama
E003	Tom	22	Texas

<DeptDetails>

Dept_ID	Dept_Name
Dpt1	Operations
Dpt2	HR
Dpt3	Finance

Now, you won’t be able to join the above tables, since Emp_ID isn’t part of the DeptDetails relation. Therefore, the above relation has lossy decomposition.

Anamolies:

Anomalies in the relational model refer to inconsistencies or errors that can arise when working with relational databases, specifically in the context of data insertion, deletion, and modification. There are different types of anomalies that can occur in referencing and referenced relations which can be discussed as:

These anomalies can be categorized into three types:

1. Insertion Anomalies
2. Deletion Anomalies
3. Update Anomalies.

How Are Anomalies Caused in DBMS?

Database anomalies are the faults in the database caused due to poor management of storing everything in the flat database. It can be removed with the process of Normalization, which generally splits the database which results in reducing the anomalies in the database.

Parul® University

DIGITAL LEARNING CONTENT

Anomalies:

Anomalies

STUDENT Table

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD-COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajasthan	India	18
4	SURESH		Punjab	India	21

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Anomalies:

Insertion anomaly: If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow insertion in referencing relation.

Example: If we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow it.

Deletion and Updation anomaly: If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.

- **ON DELETE/UPDATE SET NULL:** If a tuple is deleted or updated from referenced relation and set the value of referencing attribute to NULL.
- **ON DELETE/UPDATE CASCADE:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and referencing relation as well.

Parul® University

References

[1] Abraham Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, McGraw-Hill Education (Asia), Seventh Edition, 2019.

[2] C. J. Date, A. Kannan and S. Swamynathan, An Introduction to Database Systems, Pearson Education, Eighth Edition, 2009.

[3] Database Management Systems, CSE, DIET, https://www.darshan.ac.in/DIET/CE/GTU-Computer-Engineering-Study-Material

[4] Database management systems by Raghu Ramakrishnan and Johannes Gehrke http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html

[5] Database management system tutorial, https://www.tutorialspoint.com/dbms/index.htm

[6] Vishal Hatmode, Sonali Rangdale, 2014, Query Optimization Based on Heuristic Rules, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 03, Issue 07 (July 2014).

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in