

Malay Bhattacharyya

Machine Intelligence Unit
and

March, 2023

1

2

3

- First Normal Form
- Second Normal Form
- Third Normal Form
- Boyce-Codd Normal Form
- Elementary Key Normal Form
- Fourth Normal Form
- Fifth Normal Form
- Domain Key Normal Form
- Sixth Normal Form

Redundancy in a database denotes the repetition of stored data

- Insertion anomalies: It may be impossible to store certain information without storing some other, unrelated information.
- Deletion anomalies: It may be impossible to delete certain information without losing some other, unrelated information.
- Update anomalies: If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- Increasing storage requirements: The storage requirements may increase over time.

These issues can be addressed by decomposing the database – **normalization forces this!!!**

Update anomaly – An example

Consider the following table (the attributes are not null) detailing some of the cars available in the Kolkata market.

Company	Country	Make	Distributor
Maruti	India	WagonR	Carwala
Maruti	India	WagonR	Bhalla
Toyota	Japan	RAV4	CarTrade
BMW	Germany	X1	CarTrade

Suppose Maruti is no more an Indian company due to its 100% procurement by Suzuki Motor Corporation, a company from Japan.

This update is to be made in multiple records in the above table resulting into atomicity challenges.

An overview of different normal forms in the literature

Normal Form	Details	Reference
1NF (Codd (1970), Date (2006))	Domains should be atomic/At least one candidate key	[1, 9]
2NF (Codd (1971))	No non-prime attribute is functionally dependent on a proper subset of any candidate key	[2]
3NF (Codd (1971), Zaniolo (1982))	Every non-prime attribute is non-transitively dependent on every candidate key	[2, 7]
BCNF (Codd (1974))	Every non-trivial functional dependency is a dependency on a superkey	[3]
EKNF (Zaniolo (1982))	Every non-trivial functional dependency is either the dependency of an elementary key attribute or a dependency on a superkey	[7]
4NF (Fagin (1977))	Every non-trivial multi-valued dependency is a dependency on a superkey	[4]
5NF (Fagin (1979))	Every non-trivial join dependency is implied by the superkeys	[5]
DKNF (Fagin (1981))	Every constraint on the table is a logical consequence of the domain and key constraints	[6]
6NF (Date <i>et al.</i> (2002))	No non-trivial join dependencies at all (w.r.t generalized join)	[8]

A close-up photograph of a woman with dark hair, her face partially obscured by her hand as she covers her eyes. She appears to be crying or in a state of deep distress. The lighting is soft, highlighting the texture of her skin and the intensity of her expression.

NORMALIZATION

"MADE UP BY DESIGNERS TO SLOW DOWN DATABASES"

Denormalization

Denormalization is the process of converting a normalized schema to a non-normalized one

Denormalization is the process of converting a normalized schema to a non-normalized one

Note: Designers use denormalization to tune performance of systems to support time-critical operations. They assess the cost, benefit, and risk to identify the right normalization level with respect to the data, its use and its quality requirements.

First normal form

The domain (or value set) of an attribute defines the set of values it might contain.

A domain is *atomic* if elements of the domain are considered to be indivisible units.

Company	Make
Maruti	WagonR, Ertiga
Honda	City
Tesla	RAV4
Toyota	RAV4
BMW	X1

Only Company has atomic domain

Company	Make
Maruti	WagonR, Ertiga
Honda	City
Tesla, Toyota	RAV4
BMW	X1

None of the attributes have atomic domains

Database Management Systems

Company	Country	Make	Model	Distributor
Maruti	India	WagonR	LXI, VXI	Carwala
Maruti	India	Ertiga	VXI	Carwala
Maruti	India	WagonR	LXI	Bhalla
Honda	Japan	City	SV	Bhalla
Tesla	USA	RAV4	EV	CarTrade
Toyota	Japan	RAV4	EV	CarTrade
BMW	Germany	X1	Expedition	CarTrade

First normal form

Approach 1: Break the tuples containing non-atomic values into multiple tuples.

Company	Country	Make	Model	Distributor
Maruti	India	WagonR	LXI	Carwala
Maruti	India	WagonR	VXI	Carwala
Maruti	India	Ertiga	VXI	Carwala
Maruti	India	WagonR	LXI	Bhalla
Honda	Japan	City	SV	Bhalla
Tesla	USA	RAV4	EV	CarTrade
Toyota	Japan	RAV4	EV	CarTrade
BMW	Germany	X1	Expedition	CarTrade

First normal form

Approach 2: Decompose the relation into multiple relations.

Company	Country	Make
Maruti	India	WagonR
Maruti	India	Ertiga
Honda	Japan	City
Tesla	USA	RAV4
Toyota	Japan	RAV4
BMW	Germany	X1

Make	Model	Distributor
WagonR	LXI	Carwala
WagonR	VXI	Carwala
Ertiga	VXI	Carwala
WagonR	LXI	Bhalla
City	SV	Bhalla
RAV4	EV	CarTrade
X1	Expedition	CarTrade

Why data dependencies are so important?

Choose the best keyset for the locks given below.

Locks	Keyset 1	Keyset 2	Keyset 3
⊙	☞	☞	☞
L1	K1	K1	K3
⊙	☞	☞	☞
L2	K1	K2	K4
⊙	☞	☞	☞
L3	K1	K3	K5
⊙	☞	☞	☞
L3	K1	K4	K5

- Keyset 1 is not appropriate because a single key can open multiple locks.
- Keyset 2 is not appropriate because the same lock can be opened with multiple keys.
- Keyset 3 is the best option!!!

Functional dependency

Consider a relation schema R . A subset $X \subset A(R)$, the attributes in R , is a *superkey* of R if $t1 \neq t2$, for all pairs of tuples $t1, t2 \in R$, implies $t1[X] \neq t2[X]$. This means no two tuples in R may have the same value on attribute set X .

1. 2. 3. 4.

TABLE 1

□ □ □ □

Superkey versus functional dependency

A	B	C
1	1	2
1	2	2
2	3	1
3	3	1

AB is a superkey

$AB \rightarrow C$ holds

A	B	C
1	1	2
1	1	2
2	3	1
3	3	1

AB is not a superkey

$AB \rightarrow C$ holds

NOT POSSIBLE

A	B	C
1	1	2
1	1	1
2	3	1
3	3	1

AB is a superkey

$AB \rightarrow C$ does not hold

AB is not a superkey

$AB \rightarrow C$ does not hold

Partial dependency

The partial dependency $X \rightarrow Y$ holds in schema R if there is a $Z \subset X$ such that $Z \rightarrow Y$.

We say Y is partially dependent on X if and only if there is a proper subset of X that satisfies the dependency.

X	Y	Z
10	10	20
10	20	20
20	30	10
30	30	10

$XY \rightarrow Z$ is a partial dependency

Note: The dependency $A \rightarrow B$ implies if the A values are same, then the B values are also same.

Definition (Second normal form (2NF))

A relational schema R is in 2NF if each attribute A in R satisfies one of the following criteria:

- 1 A is part of a candidate key.
- 2 A is not partially dependent on a candidate key.

In other words, no non-prime attribute (not a part of any candidate key) is dependent on a proper subset of any candidate key.

Note: A *candidate key* is a *superkey* for which no proper subset is a superkey, i.e. a minimal *superkey*.

Second normal form

Company	Country	Make	Model	Distributor
Maruti	India	WagonR	LXI	Carwala
Maruti	India	WagonR	VXI	Carwala
Maruti	India	Ertiga	VXI	Carwala
Maruti	India	WagonR	LXI	Bhalla
Honda	Japan	City	SV	Bhalla
Tesla	USA	RAV4	EV	CarTrade
Toyota	Japan	RAV4	EV	CarTrade
BMW	Germany	X1	Expedition	CarTrade

- $\{Company, Make, Model, Distributor\} \rightarrow Country$
- $Company \rightarrow Country$ (Violating 2NF)

Note: Country is partially dependent on $\{Company, Make, Model, Distributor\}$.

Functional dependency

Armstrong's axioms:

- **Reflexivity property:** If X is a set of attributes and $Y \subseteq X$, then $X \rightarrow Y$ holds. (known as trivial functional dependency)
- **Augmentation property:** If $X \rightarrow Y$ holds and γ is a set of attributes, then $\gamma X \rightarrow \gamma Y$ holds.
- **Transitivity property:** If both $X \rightarrow Y$ and $Y \rightarrow Z$ holds, then $X \rightarrow Z$ holds.

Other properties:

- **Union property:** If $X \rightarrow Y$ holds and $X \rightarrow Z$ holds, then $X \rightarrow YZ$ holds.
- **Decomposition property:** If $X \rightarrow YZ$ holds, then both $X \rightarrow Y$ and $X \rightarrow Z$ holds.
- **Pseudotransitivity property:** If $X \rightarrow Y$ and $\gamma Y \rightarrow Z$ holds, then $X\gamma \rightarrow Z$ holds.

Closure of functional dependencies (FDs)

We can find F^+ , the closure of a set of FDs F , as follows:

Initialize F^+ with F

repeat

for each functional dependency $f = X \rightarrow Y \in F^+$ **do**

Apply reflexivity and augmentation properties on f and include the resulting functional dependencies in F^+

end for

for each pair of functional dependencies $f_1, f_2 \in F^+$ **do**

if f_1 and f_2 can be combined together using the transitivity property **then**

Include the resulting functional dependency in F^+

end if

end for

until F^+ does not further change

repeat

if $X \subseteq A^+$ then

and if

until A^+ does not further change

Closure of attribute sets

The usefulness of finding attribute closure is as follows:

- Testing for superkey
 - Compute A^+ and check whether $\mathcal{A}(R) \subseteq A^+$ and all the tuples are distinct.
- Testing functional dependencies
 - To check if an FD $X \rightarrow Y$ holds, just check if $Y \subseteq X^+$
 - Same for checking if $X \rightarrow Y$ is in F^+ for a given F
- Computing closure of F
 - For each $A \subseteq \mathcal{A}(R)$, we find the closure A^+ , and for each $S \subseteq A^+$, we output a functional dependency $A \rightarrow S$

Decomposition of a relation

If a relation is not in a desired normal form, it can be decomposed into multiple relations such that each decomposed relation satisfies the required normal form.

Suppose a relation R consists of a set of attributes $\mathcal{A}(R) = \{A_1, A_2, \dots, A_n\}$. A *decomposition* of R replaces R by a set of (two or more) relations $\{R_1, \dots, R_m\}$ such that both the following conditions hold:

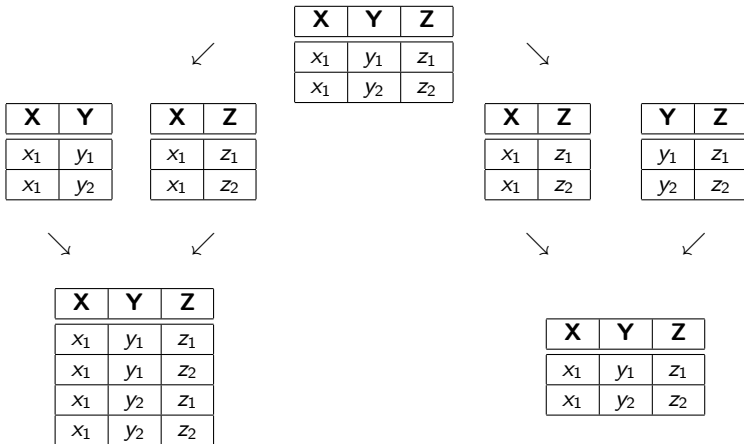
- $\forall i : \mathcal{A}(R_i) \subset \mathcal{A}(R)$
- $\mathcal{A}(R_1) \cup \dots \cup \mathcal{A}(R_m) = \mathcal{A}(R)$

Decomposition criteria

The decomposition of a relation might aim to satisfy different criteria as listed below:

- Preservation of the same relation through join (lossless-join)
- Dependency preservation
- Repetition of information

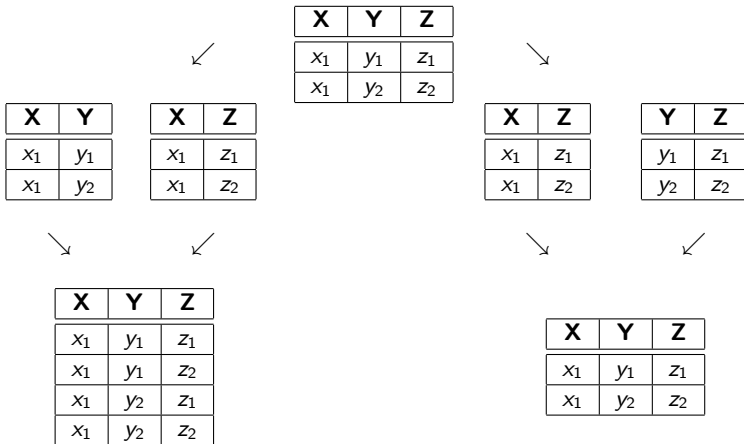
Preservation of the same relation through join



Lossy-join decomposition

Lossless-join decomposition

Preservation of the same relation through join



Lossy-join decomposition

Lossless-join decomposition

Is the decomposition into <XY> and <YZ> lossy or lossless?

Testing for lossless-join decomposition

A decomposition of R into $\{R_1, R_2\}$ is *lossless-join*, iff $\mathcal{A}(R_1) \cap \mathcal{A}(R_2) \rightarrow \mathcal{A}(R_1)$ or $\mathcal{A}(R_1) \cap \mathcal{A}(R_2) \rightarrow \mathcal{A}(R_2)$ in F^+ .

Consider the example of a relation $R = \langle UVWXY \rangle$ and the set of FDs = $\{U \rightarrow VW, WX \rightarrow Y, V \rightarrow X, Y \rightarrow U\}$.

Note that, the decomposition $R_1 = \langle UVW \rangle$ and $R_2 = \langle WXY \rangle$ is not lossless-join because $R_1 \cap R_2 = W$, and W is neither a key for R_1 nor for R_2 .

However, the decomposition $R_1 = \langle UVW \rangle$ and $R_2 = \langle UXY \rangle$ is lossless-join because $R_1 \cap R_2 = U$, and U is a key for R_1 .

Dependency preservation

The decomposition of a relation R with respect to a set of FDs F replaces R with a set of (two or more) relations $\{R_1, \dots, R_m\}$ with FDs $\{F_1, \dots, F_m\}$ such that F_i is the subset of dependencies in F^+ (the closure of F) that include only the attributes in R_i .

The decomposition is *dependency preserving* iff $(\cup_i F_i)^+ = F^+$.

Note: Through dependency preserving decomposition, we want to minimize the cost of global integrity constraints based on FDs' (i.e., avoid big joins in assertions).

Testing for dependency preserving decomposition

Consider the example of a relation $R = \langle XYZ \rangle$, having the key X , and the set of FDs = $\{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$.

Note that, the decomposition $R_1 = \langle XY \rangle$ and $R_2 = \langle XZ \rangle$ is lossless-join but not dependency preserving because $F_1 = \{X \rightarrow Y\}$ and $F_2 = \{X \rightarrow Z\}$ incur the loss of the FD $\{Y \rightarrow Z\}$, resulting into $(F_1 \cup F_2)^+ \neq F^+$.

However, the decomposition $R_1 = \langle XY \rangle$ and $R_2 = \langle YZ \rangle$ is lossless-join and also dependency preserving because $F_1 = \{X \rightarrow Y\}$ and $F_2 = \{Y \rightarrow Z\}$, satisfying $(F_1 \cup F_2)^+ = F^+$.

Third normal form

The following relation is in 2NF but not in 3NF because Country is a non-prime attribute that depends on Company, which is again a non-prime attribute. Notably, the candidate key in this relation is {PID}.

PID	Company	Country	Make	Model	Distributor
P01	Maruti	India	WagonR	LXI	Carwala
P02	Maruti	India	WagonR	VXI	Carwala
P03	Maruti	India	Ertiga	VXI	Carwala
P04	Maruti	India	WagonR	LXI	Bhalla
P05	Honda	Japan	City	SV	Bhalla
P06	Tesla	USA	RAV4	EV	CarTrade
P07	Toyota	Japan	RAV4	EV	CarTrade
P08	BMW	Germany	X1	Expedition	CarTrade

We can convert this relation into 3NF!!!

Third normal form

PID	Company	Country	Make	Model	Distributor
P01	Maruti	India	WagonR	LXI	Carwala
P02	Maruti	India	WagonR	VXI	Carwala
P03	Maruti	India	Ertiga	VXI	Carwala
P04	Maruti	India	WagonR	LXI	Bhalla
P05	Honda	Japan	City	SV	Bhalla
P06	Tesla	USA	RAV4	EV	CarTrade
P07	Toyota	Japan	RAV4	EV	CarTrade
P08	BMW	Germany	X1	Expedition	CarTrade

- $PID \rightarrow \{Company, Country, Make, Model, Distributor\}$
- $\{Company, Make, Model, Distributor\} \rightarrow Country$
(Violating 3NF)

Third normal form

Approach: Decompose the relation into multiple relations.

Company	Country
Maruti	India
Honda	Japan
Tesla	USA
Toyota	Japan
BMW	Germany

PID	Company	Make	Model	Distributor
P01	Maruti	WagonR	LXI	Carwala
P02	Maruti	WagonR	VXI	Carwala
P03	Maruti	Ertiga	VXI	Carwala
P04	Maruti	WagonR	LXI	Bhalla
P05	Honda	City	SV	Bhalla
P06	Tesla	RAV4	EV	CarTrade
P07	Toyota	RAV4	EV	CarTrade
P08	BMW	X1	Expedition	CarTrade

Note: Each non-trivial functional dependency in the left relation is on the superkey {Company}, and in the right relation is on the superkey {PID}.

Boyce-Codd normal form

Definition (Boyce-Codd normal form (BCNF))

A relational schema R is in BCNF if for every non-trivial functional dependency $X \rightarrow A$, where $X \cap A = \emptyset$, X is a superkey of R .

Note: A *superkey* is a set of one or more attributes that can uniquely identify an entity in the entity set.

Boyce-Codd normal form

The following relation is in 3NF but not in BCNF because the attribute Distributor, which depends on the non-key attribute ShopID, is a part of the key. Notably, the candidate key in this relation is {Company, Make, Model, Distributor}.

Company	Make	Model	Distributor	ShopID
Maruti	WagonR	LXI	Carwala	S1
Maruti	WagonR	VXI	Carwala	S1
Maruti	Ertiga	VXI	Carwala	S2
Maruti	WagonR	LXI	Bhalla	S3
Honda	City	SV	Bhalla	S4
Tesla	RAV4	EV	CarTrade	S5
Toyota	RAV4	EV	CarTrade	S5
BMW	X1	Expedition	CarTrade	S6
BMW	X1	Expedition	CarTrade	S6

We can convert this relation into BCNF!!!

Boyce-Codd normal form

Approach: Decompose the relation into multiple relations.

Distributor	ShopID
Carwala	S1
Carwala	S2
Bhalla	S3
Bhalla	S4
CarTrade	S5
CarTrade	S6

Company	Make	Model	ShopID
Maruti	WagonR	LXI	S1
Maruti	WagonR	VXI	S1
Maruti	Ertiga	VXI	S2
Maruti	WagonR	LXI	S3
Honda	City	SV	S4
Tesla	RAV4	EV	S5
Toyota	RAV4	EV	S5
BMW	X1	Expedition	S6

Note: Each non-trivial functional dependency in the left relation is on the superkey {ShopID}. What about the relation in the right?

Decomposition into BCNF – An algorithm

Result := {*R*} and *flag* := FALSE

Compute F^+

while NOT *flag* **do**

if There is a schema $R_i \in \text{Result}$ that is not in BCNF **then**

Let $X \rightarrow Y$ be a non-trivial functional dependency that holds on R_i such that $(X \rightarrow R_i) \notin F^+$ and $X \cap Y = \phi$.

Result := (*Result* – R_i) $\cup$ ($R_i - Y$) $\cup$ (X, Y) // This is simply decomposing R into $R - Y$ and XY provided $X \rightarrow Y$ in R violates BCNF

else

flag := TRUE

end if

end while

Decomposition into BCNF – An algorithm

$Result := \{R\}$ and $flag := \text{FALSE}$

Compute F^+

while NOT *flag* do

if There is a schema $R_i \in Result$ that is not in BCNF **then**

Let $X \rightarrow Y$ be a non-trivial functional dependency that holds on R_i such that $(X \rightarrow R_i) \notin F^+$ and $X \cap Y = \phi$.

$Result := (Result - R_i) \cup (R_i - Y) \cup (X, Y)$ // This is simply decomposing R into $R - Y$ and XY provided $X \rightarrow Y$ in R violates BCNF

else

```
flag := TRUE
```

end if

end while

Note: This decomposition process ensures lossless property.

Decomposition into BCNF – Example I

Consider a relation $R = \langle \text{ABCDE} \rangle$ having the functional dependencies $\{A \rightarrow BC, C \rightarrow DE\}$.

Decomposition into BCNF – Example II

Suppose a relation $R = \langle ABCD \rangle$ is given with the functional dependencies $\{AB \rightarrow C, B \rightarrow D, C \rightarrow A\}$.

Solution: The attribute closures provide $A^+ = A$, $B^+ = BD$, $C^+ = AC$, $D^+ = D$, $AB^+ = ABCD$, and $BC^+ = ABCD$. Hence, AB and BC are the keys of R . Note that, the functional dependency $AB \rightarrow C$ does not violate BCNF but $B \rightarrow D$ and $C \rightarrow A$ do violate. By applying $B \rightarrow D$, we decompose R and obtain $\langle ABC \rangle$ and $\langle BD \rangle$.

Now $\langle BD \rangle$ is in BCNF (B is the key) but not $\langle ABC \rangle$. The functional dependency $C \rightarrow A$ violates BCNF. By applying $C \rightarrow A$, we further decompose $\langle ABC \rangle$ and obtain $\langle BC \rangle$ and $\langle CA \rangle$. Now $\langle BD \rangle$, $\langle BC \rangle$ and $\langle CA \rangle$ are all in BCNF.

Decomposition into BCNF – Example II

Suppose a relation $R = \langle ABCD \rangle$ is given with the functional dependencies $\{AB \rightarrow C, B \rightarrow D, C \rightarrow A\}$.

Solution: The attribute closures provide $A^+ = A$, $B^+ = BD$, $C^+ = AC$, $D^+ = D$, $AB^+ = ABCD$, and $BC^+ = ABCD$. Hence, AB and BC are the keys of R . Note that, the functional dependency $AB \rightarrow C$ does not violate BCNF but $B \rightarrow D$ and $C \rightarrow A$ do violate. By applying $B \rightarrow D$, we decompose R and obtain $\langle ABC \rangle$ and $\langle BD \rangle$.

Now $\langle BD \rangle$ is in BCNF (B is the key) but not $\langle ABC \rangle$. The functional dependency $C \rightarrow A$ violates BCNF. By applying $C \rightarrow A$, we further decompose $\langle ABC \rangle$ and obtain $\langle BC \rangle$ and $\langle CA \rangle$. Now $\langle BD \rangle$, $\langle BC \rangle$ and $\langle CA \rangle$ are all in BCNF.

Note: This BCNF decomposition does not preserve dependencies.

Comparison

Decomposition	Preservation of the same relation through join	Functional dependency preservation
1NF	lossless	kept
2NF	lossless	kept
3NF	lossless	kept
BCNF	lossless	lost

	X	Y	$R - (X \cup Y)$
$t1$	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$m_{j+1} \dots m_k$
$t2$	$m_1 \dots m_i$	$n_{i+1} \dots n_i$	$n_{j+1} \dots n_k$

Visualizing multi-valued dependency

	X	Y	$R - (X \cup Y)$
t_1	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$m_{j+1} \dots m_k$
t_2	$m_1 \dots m_i$	$n_{i+1} \dots n_j$	$n_{j+1} \dots n_k$
t_3	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$n_{j+1} \dots n_k$
t_4	$m_1 \dots m_i$	$n_{i+1} \dots n_j$	$m_{j+1} \dots m_k$

Database Management Systems

○ ○

○ ○

○ ○

○ ○

Decomposition into 4NF – An algorithm

Result := {*R*} and *flag* := FALSE

Compute D^+ // Given schema R_i , let D_i denote the restriction of D^+ to R_i

while NOT *flag* **do**

if There is a schema $R_i \in \textit{Result}$ that is not in 4NF w.r.t. D_i
then

Let $X \twoheadrightarrow Y$ be a non-trivial functional dependency that holds on R_i such that $(X \rightarrow R_i) \notin D_i$ and $X \cap Y = \phi$.

Result := (*Result* – R_i) $\cup$ ($R_i - Y$) $\cup$ (X, Y) // Decompose R into $R - Y$ and XY provided $X \twoheadrightarrow Y$ in R violates 4NF

else

flag := TRUE

end if

end while

Decomposition into 4NF – An algorithm

$Result := \{R\}$ and $flag := \text{FALSE}$

Compute D^+ // Given schema R_i , let D_i denote the restriction of D^+ to R_i

while NOT *flag* do

if There is a schema $R_i \in Result$ that is not in 4NF w.r.t. D_i
then

Let $X \twoheadrightarrow Y$ be a non-trivial functional dependency that holds on R_i such that $(X \rightarrow R_i) \notin D_i$ and $X \cap Y = \phi$.

$Result := (Result - R_i) \cup (R_i - Y) \cup (X, Y)$ // Decompose R into $R - Y$ and XY provided $X \twoheadrightarrow Y$ in R violates 4NF

else

```
flag := TRUE
```

end if

end while

Note: The decomposition process ensures lossless property

() 1 () 2 () 3 () 4 () 5

Fifth normal form

Definition (Fifth normal form (5NF))

A relational schema R is in 5NF if for every non-trivial join dependency $JD(R_1, R_2, \dots, R_n)$ in F^+ , every R_i is a superkey of R .

Note: 5NF is also known as project-join normal form.

References

- C. J. Date (2006) *Date on Database: Writings 2000-2006*, Springer-Verlag.