Fundamentals of Database Systems

[Normalization – II]

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

October, 2019

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

First normal form

Definition (First normal form (1NF))

A relational schema R is in 1NF iff the domains of all attributes in R are *atomic*.

The advantages of 1NF are as follows:

- It eliminates redundancy
- It eliminates repeating groups.

Note: In practice, 1NF includes a few more practical constraints like each attribute must be unique, no tuples are duplicated, and no columns are duplicated.

First normal form

The following relation is not in 1NF because the attribute `Model` is not atomic.

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

We can convert this relation into 1NF in two ways!!!

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	WagonR	LXI	Bhalla
Maruti	India	Ertiga	VXI	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
WagonR	LXI	Bhalla
Ertiga	VXI	Bhalla
City	SV	Bhalla
RAV4	EV	CarTrade
RAV4	EV	CarTrade
X1	Expedition	CarTrade

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.

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

Second normal form

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

The following relation is in 1NF but not in 2NF because `Country` is a non-prime attribute that partially depends on `Company`, which is a proper subset of the candidate key {`Company`, `Make`, `Model`, `Distributor`}.

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

We can convert this relation into 2NF!!!

Second normal form

Approach: Decompose the relation into multiple relations.

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

Company	**Make**	**Model**	**Distributor**
Maruti	WagonR	LXI	Carwala
Maruti	WagonR	VXI	Carwala
Maruti	WagonR	LXI	Bhalla
Maruti	Ertiga	VXI	Bhalla
Honda	City	SV	Bhalla
Tesla	RAV4	EV	CarTrade
Toyota	RAV4	EV	CarTrade
BMW	X1	Expedition	CarTrade

Note: Each attribute in the left relation is a part of the candidate key {`Company`, `Country`} and in the right relation is a part of the candidate key {`Company`, `Make`, `Model`, `Distributor`}.

Functional dependency

The notion of functional dependency generalizes the notion of superkey. Consider a relation schema R, and let $X \subseteq R$ and $Y \subseteq R$. The functional dependency $X \rightarrow Y$ holds on schema R if

$$t1[X] = t2[X],$$

in any legal relation $r(R)$, for all pairs of tuples $t1$ and $t2$ in r, then

$$t1[Y] = t2[Y].$$

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.

Functional dependency

Armstrong's axioms:

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

Other properties:

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

Closure of functional dependencies (FDs) – An example

Consider a relation R = <UVWXYZ> and the set of FDs = {U $\rightarrow$ V, U $\rightarrow$ W, WX $\rightarrow$ Y, WX $\rightarrow$ Z, V $\rightarrow$ Y}. Let us compute some non-trivial FDs that can be obtained from this.

- By applying the augmentation property, we obtain
  1. UX $\rightarrow$ WX (from U $\rightarrow$ W)
  2. WX $\rightarrow$ WXZ (from WX $\rightarrow$ Z)
  3. WXZ $\rightarrow$ YZ (from WX $\rightarrow$ Y)
- By applying the transitivity property, we obtain
  1. U $\rightarrow$ Y (from U $\rightarrow$ V and V $\rightarrow$ Y)
  2. UX $\rightarrow$ Z (from UX $\rightarrow$ WX and WX $\rightarrow$ Z)
  3. WX $\rightarrow$ YZ (from WX $\rightarrow$ WXZ and WXZ $\rightarrow$ YZ)

Closure of attribute sets

We can find A^+, the closure of a set of attributes A, as follows:

Initialize A^+ with A
repeat
  for each functional dependency $f = X \rightarrow Y \in F^+$ **do**
    if $X \subseteq A^+$ **then**
      $A^+ \leftarrow A^+ \cup Y$
    end if
  end for
until A^+ does not further change

Note: The closure is defined as the set of attributes that are functionally determined by A under a set of FDs F.

Closure of attribute sets

The usefulness of finding attribute closure is as follows:

- Testing for superkey
  – Compute A^+ and check if $R \subseteq \mathrm{A}^+$
- Testing functional dependencies
  – To check if an FD $\mathrm{X} \rightarrow \mathrm{Y}$ holds, just check if $\mathrm{Y} \subseteq \mathrm{X}^+$
  – Same for checking if $\mathrm{X} \rightarrow \mathrm{Y}$ 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 $A \rightarrow S$

Closure of attribute sets – An example

Consider a relation $R = \langle UVWXYZ \rangle$ and the set of FDs = $\{U \rightarrow V, U \rightarrow W, WX \rightarrow Y, WX \rightarrow Z, V \rightarrow Y\}$. Let us compute UX^+, i.e., the closure of UX.

- Initially $UX^+ = UX$
- Then we have $UX^+ = UVX$ (as $U \rightarrow V$ and $U \subseteq UX$)
- Then we have $UX^+ = UVWX$ (as $U \rightarrow W$ and $U \subseteq UVX$)
- Then we have $UX^+ = UVWXY$ (as $WX \rightarrow Y$ and $WX \subseteq UVWX$)
- Finally, we have $UX^+ = UVWXYZ$ (as $WX \rightarrow Z$ and $WX \subseteq UVWXY$)

Note: The closure of UX covers all the attributes in R.

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

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

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

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_2

X	Y
x_1	y_1
x_1	y_2

X	Z
x_1	z_1
x_1	z_2

X	Y	Z
x_1	y_1	z_1
x_1	y_1	z_2
x_1	y_2	z_1
x_1	y_2	z_2

Lossy-join decomposition

X	Z
x_1	z_1
x_1	z_2

Y	Z
y_1	z_1
y_2	z_2

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_2

Lossless-join decomposition

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 $R = <$UVWXY$>$ and the set of FDs $= \{$U $\rightarrow$ VW, WX $\rightarrow$ Y, V $\rightarrow$ X, Y $\rightarrow$ U$\}$.

Note that, the decomposition $R_1 = <$UVW$>$ and $R_2 = <$WXY$>$ 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 $R_1 = <$UVW$>$ and $R_2 = <$UXY$>$ 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 $\{R_1, \ldots, R_m\}$ with FDs $\{F_1, \ldots, F_m\}$ 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 $R = <XYZ>$, having the key X, and the set of FDs $= \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$.

Note that, the decomposition $R_1 = <XY>$ and $R_2 = <XZ>$ 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 $R_1 = <XY>$ and $R_2 = <YZ>$ 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

Definition (Third normal form (3NF))

A relational schema R is in 3NF if for every non-trivial functional dependency $X \rightarrow A$, one of the following statements is true:

1. X is a superkey of R.
2. A is a part of some key for R.

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

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 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	WagonR	LXI	Bhalla
P04	Maruti	India	Ertiga	VXI	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

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	WagonR	LXI	Bhalla
P04	Maruti	Ertiga	VXI	Bhalla
P05	Honda	City	SV	Bhalla
P06	Tesla	RAV4	EV	CarTrade
P07	Toyota	RAV4	EV	CarTrade
P08	BMW	X1	Expedition	CarTrade

Note: Each attribute in the left relation is a part of the superkey $\{\texttt{Company}, \texttt{Country}\}$ and in the right relation is a part of the candidate key $\{\texttt{PID}\}$.

Boyce-Codd normal form

Definition (Boyce-Codd normal form (BCNF))

A relational schema R is in BCNF if for every non-trivial functional dependency $X \to A$, 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 `Price` depends on non-superkey attributes.

PID	Company	Make	Model	Distributor	Price
P01	Maruti	WagonR	LXI	Carwala	415K
P02	Maruti	WagonR	VXI	Carwala	470K
P03	Maruti	WagonR	LXI	Bhalla	410K
P04	Maruti	Ertiga	VXI	Bhalla	820K
P05	Honda	City	SV	Bhalla	990K
P06	Tesla	RAV4	EV	CarTrade	1700K
P07	Toyota	RAV4	EV	CarTrade	1700K
P08	BMW	X1	Expedition	CarTrade	3520K

We can convert this relation into BCNF!!!

Decomposition into BCNF – An algorithm

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

Decomposition into BCNF – An algorithm

Result := $\{R\}$ and *flag* := 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 – An example

Given a relation R = <ABCDPQVZ>, which is not in BCNF, having the key A and functional dependencies {CP $\rightarrow$ A, BD $\rightarrow$ P, C $\rightarrow$ B}.

Decomposition into BCNF – An example

Given a relation R = $<$ABCDPQVZ$>$, which is not in BCNF, having the key A and functional dependencies $\{$CP $\rightarrow$ A, BD $\rightarrow$ P, C $\rightarrow$ B$\}$.

Solution: Let us start with BD $\rightarrow$ P. Based on this, we decompose R and obtain $<$ABCDQVZ$>$ and $<$BDP$>$. Now $<$BDP$>$ is in BCNF (BD is the key).

Decomposition into BCNF – An example

Given a relation $R = <\mathrm{ABCDPQVZ}>$, which is not in BCNF, having the key A and functional dependencies $\{\mathrm{CP} \rightarrow \mathrm{A}, \mathrm{BD} \rightarrow \mathrm{P}, \mathrm{C} \rightarrow \mathrm{B}\}$.

Solution: Let us start with BD $\rightarrow$ P. Based on this, we decompose R and obtain $<$ABCDQVZ$>$ and $<$BDP$>$. Now $<$BDP$>$ is in BCNF (BD is the key).
For C $\rightarrow$ B, $<$ABCDQVZ$>$ is not in BCNF. Therefore, we have further decomposition into $<$ACDQVZ$>$ and $<$CB$>$.

Decomposition into BCNF – An example

Given a relation R = $<$ABCDPQVZ$>$, which is not in BCNF, having the key A and functional dependencies $\{$CP $\rightarrow$ A, BD $\rightarrow$ P, C $\rightarrow$ B$\}$.

Solution: Let us start with BD $\rightarrow$ P. Based on this, we decompose R and obtain $<$ABCDQVZ$>$ and $<$BDP$>$. Now $<$BDP$>$ is in BCNF (BD is the key).
For C $\rightarrow$ B, $<$ABCDQVZ$>$ is not in BCNF. Therefore, we have further decomposition into $<$ACDQVZ$>$ and $<$CB$>$.
Thus, the decomposition $<$ACDQVZ$>$, $<$CB$>$ and $<$BDP$>$ is a lossless-join decomposition of R into BCNF.

Decomposition into BCNF – An example

Given a relation R = <ABCDPQVZ>, which is not in BCNF, having the key A and functional dependencies {CP → A, BD → P, C → B}.

<u>**Solution**</u>**:** Let us start with BD → P. Based on this, we decompose R and obtain <ABCDQVZ> and <BDP>. Now <BDP> is in BCNF (BD is the key).
For C → B, <ABCDQVZ> is not in BCNF. Therefore, we have further decomposition into <ACDQVZ> and <CB>.
Thus, the decomposition <ACDQVZ>, <CB> and <BDP> is a lossless-join decomposition of R into BCNF.
<u>**Alternate solution**</u>**:** Suppose, we start with C → B. Then the relation R would be decomposed into <ACDPQVZ> and <CB>. The only dependencies that hold over <ACDPQVZ> are CP → A and the key dependency A → ACDPQVZ. CP is a key. Hence the decomposed relations are in BCNF.

Comments

Note that

- BCNF is stronger than 3NF – if a schema R is in BCNF then it is also in 3NF.
- 3NF is stronger than 2NF – if a schema R is in 3NF then it is also in 2NF.
- 2NF is stronger than 1NF – if a schema R is in 2NF then it is also in 1NF.