

Database Management Systems

Database Normalization

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

January, 2019

1 Overview

2 1NF

3 2NF

4 3NF

5 BCNF

6 EKNF

7 4NF

8 5NF

9 DKNF

10 6NF

Why normalization?

Redundancy in a database denotes the repetition of stored data

Redundancy might cause various anomalies and problems in storage requirements:

- Update anomalies: If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- 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.
- Increasing storage requirements: The storage requirements may increase over time.

Why normalization?

Redundancy in a database denotes the repetition of stored data

Redundancy might cause various anomalies and problems in storage requirements:

- Update anomalies: If one copy of such repeated data is updated, all copies need to be updated to prevent inconsistency.
- 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.
- Increasing storage requirements: The storage requirements may increase over time.

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

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]

Motivations behind normalization

Normal Form	Basic Motivation
1NF	Removing non-atomicity
2NF	Removing partial dependency (Part of key attribute \rightarrow Non-key attribute)
3NF	Removing transitive dependency (Non-key attribute \rightarrow Non-key attribute)
BCNF	Removing any kind of redundancy

Denormalization

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

Denormalization

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.

1st 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

1st Normal Form

Definition (1st 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.

1st 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!!!

1st 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

1st 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.

2nd Normal Form

Definition (2nd 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*.

2nd 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!!!

2nd 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 $X \rightarrow Y$ 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 A^+ does not further change

Closure of functional dependencies (FDs) – 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 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 $R \subseteq A^+$
- 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 $A \rightarrow S$

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 $\mathcal{A}(R) = \{A_1, A_2, \dots, A_n\}$. 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:

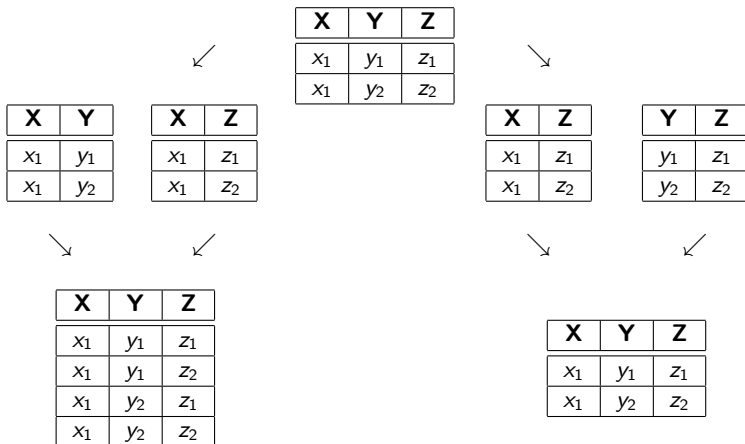
- $\forall i : \mathcal{A}(R_i) \subset \mathcal{A}(R)$
- $\mathcal{A}(R_1) \cup \dots \cup \mathcal{A}(R_m) = \mathcal{A}(R)$

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

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 $(F_1 \cup F_2)^+ = F^+$.

3rd Normal Form

Definition (3rd 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.

3rd 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!!!

3rd 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 {Company, Country} and in the right relation is a part of the candidate key {ProductID}.

Boyce-Codd Normal Form

Definition (Boyce-Codd Normal Form (BCNF))

A relational schema R is in BCNF if for every non-trivial functional dependency $X \rightarrow A$, 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 $R_i \in \textit{Result}$ 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 $R_i \in \textit{Result}$ 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 $R = \langle ABCDPQVZ \rangle$, 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 $R = \langle ABCDPQVZ \rangle$, 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 $\langle ABCDQVZ \rangle$ and $\langle BDP \rangle$. Now $\langle BDP \rangle$ is in BCNF (BD is the key).

Decomposition into BCNF – An example

Given a relation $R = \langle ABCDPQVZ \rangle$, 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 $\langle ABCDQVZ \rangle$ and $\langle BDP \rangle$. Now $\langle BDP \rangle$ is in BCNF (BD is the key).

For $C \rightarrow B$, $\langle ABCDQVZ \rangle$ is not in BCNF. Therefore, we have further decomposition into $\langle ACDQVZ \rangle$ and $\langle CB \rangle$.

Decomposition into BCNF – An example

Given a relation $R = \langle ABCDPQVZ \rangle$, 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 $\langle ABCDQVZ \rangle$ and $\langle BDP \rangle$. Now $\langle BDP \rangle$ is in BCNF (BD is the key).

For $C \rightarrow B$, $\langle ABCDQVZ \rangle$ is not in BCNF. Therefore, we have further decomposition into $\langle ACDQVZ \rangle$ and $\langle CB \rangle$.

Thus, the decomposition $\langle ACDQVZ \rangle$, $\langle CB \rangle$ and $\langle BDP \rangle$ is a lossless-join decomposition of R into BCNF.

Decomposition into BCNF – An example

Given a relation $R = \langle ABCDPQVZ \rangle$, 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 $\langle ABCDQVZ \rangle$ and $\langle BDP \rangle$. Now $\langle BDP \rangle$ is in BCNF (BD is the key).

For $C \rightarrow B$, $\langle ABCDQVZ \rangle$ is not in BCNF. Therefore, we have further decomposition into $\langle ACDQVZ \rangle$ and $\langle CB \rangle$.

Thus, the decomposition $\langle ACDQVZ \rangle$, $\langle CB \rangle$ and $\langle BDP \rangle$ is a lossless-join decomposition of R into BCNF.

Alternate solution: Suppose, we start with $C \rightarrow B$. Then the relation R would be decomposed into $\langle ACDPQVZ \rangle$ and $\langle CB \rangle$. The only dependencies that hold over $\langle ACDPQVZ \rangle$ are $CP \rightarrow A$ and the key dependency $A \rightarrow ACDPQVZ$. 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.

Elementary Key Normal Form

Definition (Elementary Key Normal Form (EKNF))

A relational schema R is in EKNF 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 X is an elementary key attribute

Note: A non-trivial functional dependency $X \rightarrow Y$ is an elementary dependency if there exist no partial dependency. A key K is elementary key if $K \rightarrow Y$ is an elementary dependency.

Multi-valued dependency

Consider a relation schema R , and let $X \subseteq R$ and $Y \subseteq R$. The functional dependency $X \twoheadrightarrow Y$ 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 , implies

- $t1[X] = t2[X] = t3[X] = t4[X]$
- $t1[Y] = t3[Y]$ and $t2[Y] = t4[Y]$
- $t1[Z] = t4[Z]$ and $t2[Z] = t3[Z]$

where the two tuples $t3$ and $t4$ are also in r and Z denotes $R - (X \cup Y)$.

Multi-valued dependency

Consider a relation schema R , and let $X \subseteq R$ and $Y \subseteq R$. The functional dependency $X \twoheadrightarrow Y$ 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 , implies

- $t1[X] = t2[X] = t3[X] = t4[X]$
- $t1[Y] = t3[Y]$ and $t2[Y] = t4[Y]$
- $t1[Z] = t4[Z]$ and $t2[Z] = t3[Z]$

where the two tuples $t3$ and $t4$ are also in r and Z denotes $R - (X \cup Y)$.

Note: The tuples $t1$, $t2$, $t3$ and $t4$ are not necessarily distinct.

Visualizing multi-valued dependency

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

Visualizing multi-valued dependency

	X	Y	$R - (X \cup Y)$
$t1$	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$m_{j+1} \dots m_k$
$t2$	$m_1 \dots m_i$	$n_{i+1} \dots n_i$	$n_{j+1} \dots n_k$
$t3$	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$n_{j+1} \dots n_k$
$t4$	$m_1 \dots m_i$	$n_{i+1} \dots n_i$	$m_{j+1} \dots m_k$

Visualizing multi-valued dependency

	X	Y	$R - (X \cup Y)$
$t1$	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$m_{j+1} \dots m_k$
$t2$	$m_1 \dots m_i$	$n_{i+1} \dots n_i$	$n_{j+1} \dots n_k$
$t3$	$m_1 \dots m_i$	$m_{i+1} \dots m_j$	$n_{j+1} \dots n_k$
$t4$	$m_1 \dots m_i$	$n_{i+1} \dots n_i$	$m_{j+1} \dots m_k$

An example of $X \twoheadrightarrow Y$

Inference rules for multi-valued dependency

- If $X \twoheadrightarrow Y$ holds, then $X \twoheadrightarrow (R - (X \cup Y))$ holds.
- If $X \twoheadrightarrow Y$ holds and $W \supseteq Z$, then $WX \twoheadrightarrow YZ$ holds.
- If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$ both holds, then $X \twoheadrightarrow (Z - Y)$ holds.
- If $X \rightarrow Y$ holds, then $X \twoheadrightarrow Y$ holds.
- If $X \twoheadrightarrow Y$ holds and there exists W such that (a) $W \cap Y = \phi$, (b) $W \rightarrow Z$ and (c) $Y \supseteq Z$, then $X \rightarrow Z$ holds.

4th Normal Form

Definition (4th Normal Form (4NF))

A relational schema R is in 4NF if for every non-trivial multi-valued dependency $X \twoheadrightarrow A$, 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.

4th Normal Form

The following relation is in 4NF.

Name	Age	Codeword	Media
Irfan	28	abc	News
Irfan	40	xyz	News
Irfan	40	abc	Radio
Irfan	28	xyz	Radio
Imran	42	abc	News

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 $X \cap Y = \emptyset$.

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* := 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 $X \cap Y = \emptyset$.

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

Join dependency

Given a relation schema R , a join dependency $JD(R_1, R_2, \dots, R_n)$ is defined by the constraint that every legal relation $r(R)$ should have a non-additive join decomposition into R_1, R_2, \dots, R_n , i.e. for every such r we have

$$(\pi_{R_1}(r), \pi_{R_2}(r), \dots, \pi_{R_n}(r)) = r.$$

Join dependency

Given a relation schema R , a join dependency $JD(R_1, R_2, \dots, R_n)$ is defined by the constraint that every legal relation $r(R)$ should have a non-additive join decomposition into R_1, R_2, \dots, R_n , i.e. for every such r we have

$$(\pi_{R_1}(r), \pi_{R_2}(r), \dots, \pi_{R_n}(r)) = r.$$

Note: Multi-valued dependency is a special case of join dependency where $n = 2$.

5th Normal Form

Definition (5th 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 .

5th Normal Form

Definition (5th 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.

Domain Key Normal Form

Definition (Domain Key Normal Form (DKNF))

A relational schema R is in DKNF if all the constraints and dependencies that should hold on the valid relation states is a logical consequence of the domain and key constraints on the relation.

6th Normal Form

Definition (6th Normal Form (6NF))

A relational schema R is in 6NF if there exists no non-trivial join dependencies at all (with reference to generalized join operator).

References



E. F. Codd (1970) *CACM*, 13(6):377-387.



E. F. Codd (1971) *IBM Research Report*, RJ909.



E. F. Codd (1974) *IBM Research Report*, RJ1385.



R. Fagin (1977) *ACM TDS*, 2(3), 262-278.



R. Fagin (1979) *IBM Research Report*, RJ2471.



R. Fagin (1981) *CACM*, 6, 387-415.



C. Zaniolo (1982) *ACM TDS*, 7(3), 489-499.



C. J. Date (2002) *Temporal Data and the Relational Model*, Morgan Kaufmann.



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