

# Database Management Systems

## Database Normalization

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit  
and  
Centre for Artificial Intelligence and Machine Learning  
Indian Statistical Institute, Kolkata  
February, 2020



## 1 Data Redundancy

## 2 Normalization and Denormalization

## 3 Normal Forms

- 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 databases

## Redundancy in a database denotes the repetition of stored data

Redundancy might cause various anomalies and problems pertaining to storage requirements:

- 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!!!**

# Insertion 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 Tesla, a company from US, is now collaborating with Toyota to bring the make RAV4 in the Kolkata market with no distributor announced yet.

This insertion is not possible in the above table as the Distributor cannot be null.





# 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 / Non-key attribute)
3NF	Removing transitive dependency (Non-key attribute / Non-key attribute)
BCNF	Removing any kind of redundancy



# Denormalization

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



# Normalization versus denormalization



# Applications

## Normalization:

- 1 Use of normalization to minimize the impact of various anomalies created with database modification.
- 2 Use of normalization to reduce the data integrity problems.

## Denormalization:

- 1 Use of denormalization in case the data is not going to be updated after being created.
- 2 Use of denormalization results into the performance gain.

**Note:** There is no “ideal” normal form for a table or the data as a whole.

# 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

# 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

# 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!!!

# Partial dependency

The partial dependency  $X \not\rightarrow Y$  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  $A \not\rightarrow B$  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  $f$ 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

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

- $f(\text{Company, Make, Model, Distributor}) \rightarrow \text{Country}$
- $\text{Company} \rightarrow \text{Country}$  (Violating 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  $fCompany, Countryg$  and in the right relation is a part of the candidate key  $fCompany, Make, Model, Di stri butorg$ .

# 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  $X \twoheadrightarrow Y$  holds on schema  $R$  if

$$t_1[X] = t_2[X],$$

in any legal relation  $r(R)$ , for all pairs of tuples  $t_1$  and  $t_2$  in  $r$ , then

$$t_1[Y] = t_2[Y].$$

# Functional dependency

## Armstrong's axioms:

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

# Functional dependency

## Armstrong's axioms:

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

## Other properties:

- **Union property:** If  $X \twoheadrightarrow Y$  holds and  $X \twoheadrightarrow Z$  holds, then  $X \twoheadrightarrow YZ$  holds.
- **Decomposition property:** If  $X \twoheadrightarrow YZ$  holds, then both  $X \twoheadrightarrow Y$  and  $X \twoheadrightarrow Z$  holds.
- **Pseudotransitivity property:** If  $X \twoheadrightarrow Y$  and  $\gamma Y \twoheadrightarrow Z$  holds, then  $X\gamma \twoheadrightarrow Z$  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  $R = \langle UVWXYZ \rangle$  and the set of FDs =  $fU ! V, U ! W, WX ! Y, WX ! Z, V ! Yg$ . Let us compute some non-trivial FDs that can be obtained from this.

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

# 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  $X \twoheadrightarrow Y$  holds, just check if  $Y \subseteq X^+$
  - Same for checking if  $X \twoheadrightarrow Y$  is in  $F^+$  for a given  $F$
- Computing closure of  $F$ 
  - For each  $A \in A(R)$ , we find the closure  $A^+$ , and for each  $S \subseteq A^+$ , we output a functional dependency  $A \twoheadrightarrow S$

# Closure of attribute sets – An example

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

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

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

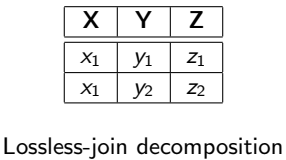
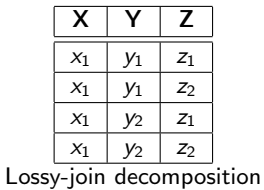
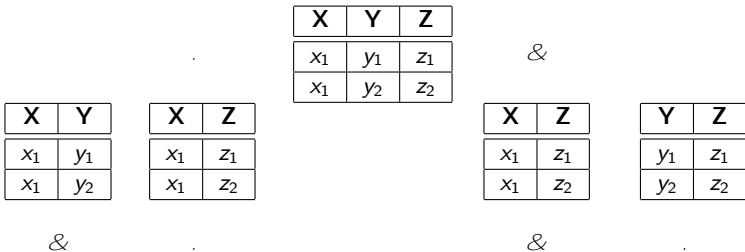
- $\forall i : A(R_i) \subseteq A(R)$
- $A(R_1) \cup \dots \cup A(R_m) = A(R)$

# 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



# Testing for lossless-join decomposition

A decomposition of  $R$  into  $\{R_1, R_2\}$  is *lossless-join*, iff  $A(R_1) \setminus A(R_2) \rightarrow A(R_1)$  or  $A(R_1) \setminus A(R_2) \rightarrow A(R_2)$  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  $R_1 \setminus R_2 = W$ , 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  $R_1 \setminus R_2 = U$ , 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 =  $fX \rightarrow Y, Y \rightarrow Z, X \rightarrow Zg$ .

Note that, the decomposition  $R_1 = \langle XY \rangle$  and  $R_2 = \langle XZ \rangle$  is lossless-join but not dependency preserving because  $F_1 = fX \rightarrow Yg$  and  $F_2 = fX \rightarrow Zg$  incur the loss of the FD  $fY \rightarrow Zg$ , resulting into  $(F_1 \cup F_2)^+ \not\subseteq F^+$ .

However, the decomposition  $R_1 = \langle XY \rangle$  and  $R_2 = \langle YZ \rangle$  is lossless-join and also dependency preserving because  $F_1 = fX \rightarrow Yg$  and  $F_2 = fY \rightarrow Zg$ , 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  $X \twoheadrightarrow A$ , 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 *PIDg*.

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

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

- PID ! fCompany, Country, Make, Model , Di stri butorg
- Company ! 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	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  $f\text{Company, Country}g$  and in the right relation is a part of the candidate key  $f\text{PID}g$ .

# Boyce-Codd normal form

## Definition (Boyce-Codd normal form (BCNF))

A relational schema  $R$  is in BCNF if for every non-trivial functional dependency  $X \twoheadrightarrow A$ ,  $X$  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.

# 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 key in this relation is  $f(\text{Company, Make, Model, Distributor}) \rightarrow \text{ShopID}$ .

Company	Make	Model	Distributor	ShopID
Maruti	WagonR	LXI	Carwala	S1
Maruti	WagonR	VXI	Carwala	S1
Maruti	WagonR	LXI	Bhalla	S2
Maruti	Ertiga	VXI	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	S7

We can convert this relation into BCNF!!!



# Boyce-Codd normal form

**Approach:** Decompose the relation into multiple relations.

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

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

**Note:** Each attribute in the left relation depends on the superkey ShopID.



# Decomposition into BCNF – An algorithm

*Result* := *fRg* and *ag* := FALSE

Compute  $F^+$

**while NOT** *ag* **do**

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

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

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

**else**

*ag* := TRUE

**end if**

**end while**

# Decomposition into BCNF – An algorithm

*Result* := *fRg* and *ag* := FALSE

Compute  $F^+$

**while NOT** *ag* **do**

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

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

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

**else**

*ag* := TRUE

**end if**

**end while**

**Note:** This decomposition process ensures lossless property

# Decomposition into BCNF – Example I

Consider a relation  $R = \langle ABCDE \rangle$  having the functional dependencies  $fA \mid BC, C \mid DEg$ .

# Decomposition into BCNF – Example I

Consider a relation  $R = \langle ABCDE \rangle$  having the functional dependencies  $fA \mid BC, C \mid DEg$ .

Solution: The attribute closures provide  $A^+ = ABCDE$ ,  $B^+ = B$ ,  $C^+ = CDE$ ,  $D^+ = D$ , and  $E^+ = E$ . Hence, A is the key of R.

## Decomposition into BCNF – Example I

Consider a relation  $R = \langle ABCDE \rangle$  having the functional dependencies  $fA \mid BC, C \mid DEg$ .

**Solution:** The attribute closures provide  $A^+ = ABCDE$ ,  $B^+ = B$ ,  $C^+ = CDE$ ,  $D^+ = D$ , and  $E^+ = E$ . Hence,  $A$  is the key of  $R$ .

Note that, the functional dependency  $A \mid BC$  does not violate BCNF but  $C \mid DE$  does violate. By applying  $C \mid DE$ , we decompose  $R$  and obtain  $\langle ABC \rangle$  and  $\langle CDE \rangle$ .



## Decomposition into BCNF – Example II

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

## Decomposition into BCNF – Example II

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

**Solution:** The attribute closures provide  $A^+ = A, B^+ = BD, C^+ = AC, D^+ = D, AB^+ = ABCD, \text{ and } BC^+ = ABCD$ . 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  $\langle BD \rangle$ .



## Decomposition into BCNF – Example II

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

**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  $fAB \rightarrow C, B \rightarrow D, C \rightarrow A$ .

**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.

# 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  $X \twoheadrightarrow A$ , 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  $X \twoheadrightarrow Y$  is an elementary dependency if there exist no partial dependency. A key  $K$  is elementary key if  $K \twoheadrightarrow Y$  is an elementary dependency.

# Multi-valued dependency

Consider a relation schema  $R$ , and let  $X \twoheadrightarrow R$  and  $Y \twoheadrightarrow R$ . 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  $R \setminus (X \cup Y)$ .

# Multi-valued dependency

Consider a relation schema  $R$ , and let  $X \twoheadrightarrow R$  and  $Y \twoheadrightarrow R$ . 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  $R \setminus (X \cup Y)$ .

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

# Visualizing multi-valued dependency

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

# Visualizing multi-valued dependency

	$X$	$Y$	$R (X \parallel Y)$
$t1$	$m_1 \dots m_j$	$m_{j+1} \dots m_j$	$m_{j+1} \dots m_k$
$t2$	$m_1 \dots m_j$	$n_{j+1} \dots n_j$	$n_{j+1} \dots n_k$
$t3$	$m_1 \dots m_j$	$m_{j+1} \dots m_j$	$n_{j+1} \dots n_k$
$t4$	$m_1 \dots m_j$	$n_{j+1} \dots n_j$	$m_{j+1} \dots m_k$



# Visualizing multi-valued dependency

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

An example of  $X \twoheadrightarrow Y$

## Inference rules for multi-valued dependency

- If  $X \twoheadrightarrow Y$  holds, then  $X \twoheadrightarrow (R \ (X \parallel Y))$  holds.
- If  $X \twoheadrightarrow Y$  holds and  $W \twoheadrightarrow Z$ , then  $WX \twoheadrightarrow YZ$  holds.
- If  $X \twoheadrightarrow Y$  and  $Y \twoheadrightarrow Z$  both holds, then  $X \twoheadrightarrow (Z \parallel Y)$  holds.
- If  $X \not\parallel Y$  holds, then  $X \twoheadrightarrow Y$  holds.
- If  $X \twoheadrightarrow Y$  holds and there exists  $W$  such that (a)  $W \setminus Y = \phi$ , (b)  $W \not\parallel Z$  and (c)  $Y \twoheadrightarrow Z$ , then  $X \not\parallel Z$  holds.

# Fourth normal form

## Definition (Fourth 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.

# Fourth normal form

The following relation is not in 4NF because it satisfies the multi-valued dependency  $\text{Name} \twoheadrightarrow \text{Age}$  in which  $\text{Name}$  is not a superkey.

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

**We can convert this relation into 4NF!!!**

# Fourth normal form

**Approach:** Decompose the relation into multiple relations.

Name	Age
Irfan	28
Irfan	40
Imran	42

Name	Codeword	Media
Irfan	abc	News
Irfan	xyz	Radio
Imran	abc	News

**Note:** No multi-valued dependency exists in the decomposed relations.

# Decomposition into 4NF – An algorithm

```

Result := fRg and ag := FALSE
Compute D+ // Given schema Ri, let Di denote the restriction
of D+ to Ri
while NOT ag do
  if There is a schema Ri ⊆ Result that is not in 4NF w.r.t. Di
  then
    Let X → Y be a non-trivial functional dependency that
    holds on Ri such that (X → Ri) ∉ Di and X ∩ Y = ∅.
    Result := (Result - Ri) ∪ (Ri - Y) ∪ (X, Y) // Decompose
    R into R - Y and XY provided X → Y in R violates 4NF
  else
    ag := TRUE
  end if
end while

```

# Decomposition into 4NF – An algorithm

```

Result := fRg and ag := FALSE
Compute D+ // Given schema R_i, let D_i denote the restriction
of D+ to R_i
while NOT ag do
  if There is a schema R_i ⊆ Result that is not in 4NF w.r.t. D_i
  then
    Let X → Y be a non-trivial functional dependency that
    holds on R_i such that (X → R_i) ∉ D_i and X ∩ Y = ∅.
    Result := (Result - R_i) ∪ (R_i - Y) ∪ (X, Y) // Decompose
    R into R - Y and XY provided X → Y in R violates 4NF
  else
    ag := 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  $n = 2$ .

# 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  $R$ .



# 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.

# Sixth normal form

## Definition (Sixth 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.