

# Database Management System 16 Normalization

## Normalization

- Lossless Decomposition
- Lossless - Join Algorithm
- Dependency Preservation

Guidelines followed in  
Designing Good  
Database

Chittaranjan Pradhan  
School of Computer Engineering,  
KIIT University

## Normalization

**Normalization** is the process of decomposing or breaking a relation schema  $R$  into fragments (i.e. smaller schemas)  $R_1, R_2, \dots, R_n$  such that the following conditions hold:

- **Lossless decomposition:** The fragments should contain the same information as the original relation
- **Dependency preservation:** All the functional dependencies should be preserved within each fragment  $R_i$
- **Good form:** Each fragment  $R_i$  should be free from any type of redundancy

In other words, normalization is the process of refining the relational data model. It is used because of the following reasons:

- It improves database design
- It ensures minimum redundancy of data
- It removes anomalies for database activities

## Normalization

Lossless Decomposition  
Lossless - Join Algorithm  
Dependency Preservation

[Guidelines followed in Designing Good Database](#)

# Lossless Decomposition

## Lossless Decomposition

The decomposition of a base relation is said to be **lossless** if the original relation can be recovered back by joining the fragment relations

Let  $R$  be the base relation, which is decomposed into  $R_1, R_2, \dots, R_n$ . This decomposition is lossless iff  $R = R_1 \bowtie R_2 \bowtie \dots \bowtie R_n$ . In other words, the consecutive fragments are interrelated by the primary key and foreign key relationships

A	B
a	1
a	2
b	1
b	2

$R_1$	$R_2$
A	B
a	1
b	2

$R_1 \bowtie R_2$	A	B
	a	1
	a	2
	b	1
	b	2

As  $R_1 \bowtie R_2 = R$ , the decomposition is lossless

# Lossless Decomposition...

## Normalization

## Lossless Decomposition

Lossless - Join Algorithm

Dependency Preservation

Guidelines followed in  
Designing Good  
Database

$R$

A	B
a	1
a	2
b	1

$R_1$

A
a
b

$R_2$

B
1
2

$R_1 \bowtie R_2$

A	B
a	1
a	2
b	1
b	2

As  $R_1 \bowtie R_2 \neq R$ , the decomposition is lossy

## Lossless Decomposition...

### Lossless Decomposition...

When the base relation schema is decomposed into the fragmented relation schemas, the consecutive relations should be related by primary key - foreign key pair on the common column; so that natural join be possible on the common column. Thus, we have to check whether the common column is the key of any relation or not:

- If the common column is the key, the decomposition is lossless
- If the common column is not the key, the decomposition is lossy

Ex:  $R(A, B, C, D)$  with  $F=\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$  is decomposed to  $R_1(A, B, C)$  and  $R_2(C, D)$  with  $F_1=\{A \rightarrow B, B \rightarrow C\}$  and  $F_2=\{C \rightarrow D\}$  respectively

Ex:  $R(A, B, C, D)$  with  $F=\{A \rightarrow B, B \rightarrow C, D \rightarrow C\}$  is decomposed to  $R_1(A, B, C)$  and  $R_2(C, D)$  with  $F_1=\{A \rightarrow B, B \rightarrow C\}$  and  $F_2=\{D \rightarrow C\}$  respectively

## Lossless - Join Algorithm

This algorithm is used to check whether the decomposition of a relation is lossless or lossy type. The steps are:

- Construct a table with  $n$  columns, where  $n$  is the number of attributes in the original relation and  $k$  rows, where  $k$  is the number of decomposed relations. Label the columns as  $A_1, A_2, \dots, A_n$
- Fill the entries in this table as follows: for each attribute  $A_i$ , check if this attribute is one of the attributes of the relation schema  $R_j$ 
  - If attribute  $A_i$  is in the Relation  $R_j$ , then the entry  $(A_i, R_j)$  of the table will be  $a_i$
  - If attribute  $A_i$  is not an attribute in the relation  $R_j$ , then the entry  $(A_i, R_j)$  of the table will be  $b_{ij}$
- For each of FD  $X \rightarrow Y$  of  $F$ , do the following until it is not possible to make any more changes to the table. When the table no longer changes, continue with step4

### Normalization

[Lossless Decomposition](#)[Lossless - Join Algorithm](#)[Dependency Preservation](#)[Guidelines followed in Designing Good Database](#)

## Lossless - Join Algorithm...

- .
- If there are two or more rows with the same value under the attribute or attributes of the determinant X, then make equal their entries under attribute Y (i.e. RHS of FD)
  - When making equal two or more entries under any column, if one of them is  $a_i$ , then make all of them  $a_i$
  - If none of them is  $a_i$ , then make all the entries uniform by considering one b term. If they are  $b_{ij}$  and  $b_{kl}$ , chose one of these two values as the representative value and make the other values equal to it. That means, make all the entries as either  $b_{ij}$  or  $b_{kl}$ . Continue with step3
- If there are no two rows with the same value under the attribute or attributes of the determinant X, continue with step3
- Check the rows of the table. If there is a row with its entries equal to  $a_1, a_2 \dots a_n$ , then the decomposition is lossless. Otherwise, the decomposition is lossy

### Normalization

Lossless Decomposition

Lossless - Join Algorithm

Dependency Preservation

Guidelines followed in  
Designing Good  
Database

## Lossless - Join Algorithm...

### Lossless - Join Algorithm...

$R=(X, Y, Z)$ ,  $F=\{X \rightarrow Y, Z \rightarrow Y\}$ ,  $R_1=(X, Y)$ ,  $R_2=(Y, Z)$

	$A_1(X)$	$A_2(Y)$	$A_3(Z)$
$R_1$			
$R_2$			

	$A_1(X)$	$A_2(Y)$	$A_3(Z)$
$R_1$	$a_1$	$a_2$	$b_{31}$
$R_2$	$b_{12}$	$a_2$	$a_3$

For  $X \rightarrow Y$ , the table remains unchanged

For  $Z \rightarrow Y$ , the table remains unchanged

Decomposition is lossy type



# Lossless - Join Algorithm...

## Lossless - Join Algorithm...

$R=(X, Y, Z)$ ,  $F=\{X \rightarrow Y, Y \rightarrow Z\}$ ,  $R_1=(X, Y)$ ,  $R_2=(Y, Z)$

	$A_1(X)$	$A_2(Y)$	$A_3(Z)$
$R_1$			
$R_2$			

	$A_1(X)$	$A_2(Y)$	$A_3(Z)$
$R_1$	$a_1$	$a_2$	$b_{31}$
$R_2$	$b_{12}$	$a_2$	$a_3$

For  $X \rightarrow Y$ , the table remains unchanged

For  $Y \rightarrow Z$ , the table will be modified

	$A_1(X)$	$A_2(Y)$	$A_3(Z)$
$R_1$	$a_1$	$a_2$	$a_3$
$R_2$	$b_{12}$	$a_2$	$a_3$

Decomposition is lossless type

## Dependency Preservation

The decomposition of a relation schema  $R$  with FDs  $F$  is a set of fragment relations  $R_i$  with FDs  $F_i$ , where  $F_i$  is the subset of dependencies in  $F^+$  that include only attributes in  $R_i$ . The decomposition is dependency preserving if and only if

$$(\cup_i F_i)^+ = F^+$$

$R = (A, B, C)$  and  $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

R

A	B	C
1	2	3
2	2	3
3	2	3
4	3	4

Key=A

## Dependency Preservation...

## Normalization

Lossless Decomposition

Lossless - Join Algorithm

Dependency Preservation

Guidelines followed in  
Designing Good  
Database $R_1 = (A, B) \text{ and } F=\{A \rightarrow B\}$ 

A	B
1	2
2	2
3	2
4	3

 $R_2 = (A, C) \text{ and } F=\{A \rightarrow C\}$ 

A	C
1	3
2	3
3	3
4	4

The decomposition is lossless because  $R_1 \bowtie R_2 = R$

But, the decomposition is not dependency preserving because  $(F_1 \cup F_2)^+ = \{A \rightarrow B, A \rightarrow C\}$

Thus,  $(F_1 \cup F_2)^+ \neq F^+$  as we lost the FD  $B \rightarrow C$

## Dependency Preservation...

## Normalization

Lossless Decomposition

Lossless - Join Algorithm

Dependency Preservation

Guidelines followed in  
Designing Good  
Database

$R_1 = (A, B)$  and  $F = \{A \rightarrow B\}$

A	B
1	2
2	2
3	2
4	3

$R_2 = (B, C)$  and  $F = \{B \rightarrow C\}$

B	C
2	3
3	4

The decomposition is lossless because  $R_1 \bowtie R_2 = R$

Similarly, the decomposition is dependency preserving because  $(F_1 \cup F_2)^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

Thus,  $(F_1 \cup F_2)^+ = F^+$

# Guidelines followed in Designing Good Database

## Guideline 1

*Design a relation schema so that it is easy to explain its meaning. Do not combine attributes from multiple entity sets and relationship sets into a single relation*

If a relation schema corresponds to one entity set or one relationship set, then the meaning tends to be simple and clear. Otherwise, the relation corresponds to a mixture of multiple entities and relationships and hence becomes complex and unclear

Only foreign keys should be used to refer to other entities. Entity and relationship attributes should be kept apart as much as possible

**Bottom Line:** *design the schemas that can be explained easily relation by relation. In such cases, the semantics of attributes should be easy to interpret*

### Normalization

Lossless Decomposition  
Lossless - Join Algorithm  
Dependency Preservation

### Guidelines followed in Designing Good Database

## Guidelines followed in Designing Good Database...

### Guideline 2

*Design the base relation schemas in such a way that the anomalies such as insertion, deletion, or updation anomalies are removed from the relations*

If any anomalies are present, note them clearly and make sure that the programs that modify (update) the database will operate correctly

### Guideline 3

*Avoid placing attributes in a base relation whose values may frequently be NULL*

If NULLs are unavoidable, make sure that they apply in exceptional cases only and do not apply to a majority of tuples in the relation

Attributes that are NULL frequently could be placed in separate relations (with the primary key)

# Guidelines followed in Designing Good Database...

## Guideline 4

*Design the relation schemas so that they can be joined in a such a way that no spurious tuples are generated*

Avoid relations that contain matching attributes that are not (foreign key and primary key) combinations, because joining on such attributes may produce spurious tuples