

Lecture 17

# Normal Forms

Week 11

# Summary of what you must know by now

(also to be exercised on tutorials)

- Functional dependencies & inference rules for FDs
- The closure of a functional dependency set
- Functional dependencies and keys (superkeys, candidate keys) & how to determine all candidate keys.
- Closure of set of attributes under an FD set
- Minimal FD set (do tutorial examples!)
- Equivalence of FD sets (do tutorial examples!)

# Overview

- Nontrivial functional dependencies, prime attributes
- Normal forms
  - 1<sup>st</sup> Normal Form (1NF)
  - 2<sup>nd</sup> Normal Form (2NF)
  - 3<sup>rd</sup> Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
- Lossless join decompositions
- Dependency preserving decompositions

# Trivial/non-trivial Functional Dependencies

- A functional dependency  $X \rightarrow Y$  is *trivial* if  $X \supseteq Y$
- *Trivial functional dependencies* do not capture any property of the universe of discourse
- *Non-trivial FDs* capture important constraints about the universe of discourse. They define the meaning (semantics) of the attributes

# Trivial FDs

- E.g.

name, ss<sub>n</sub>, address  $\rightarrow$  name, ss<sub>n</sub>

is a trivial FD - it does not matter what name, ss<sub>n</sub> or address mean, this FD holds irrespective of the universe of discourse

# Non-trivial FDs

- E.g.

$ssn \rightarrow name$

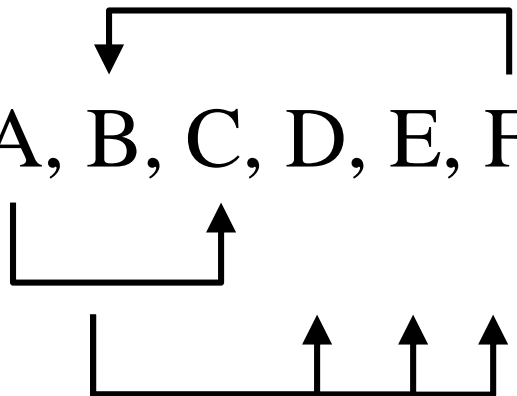
is a non-trivial FD, expressing that in the particular universe of discourse the value of the *ssn* attribute uniquely determines the value of the *name* attribute

# Prime attributes

## Definition

*Prime attributes* of a relation R are those attributes that are part of *any* candidate key of R

# Prime attributes (example)

- Take a relation R ( A, B, C, D, E, F )  
with FDs as shown
- 

determine all candidate keys

{ AB } { AF }

A,B,F are prime attributes of R

C,D,E are non-prime attributes of R



# Normal forms

- In relational database design we would like to implement such tables that avoid the possibility of update anomalies
- We intend to design our tables in such a way that if the DBMS enforces the uniqueness of primary keys then update anomalies can not occur

(this is good, because other checks before updating a table may be expensive)

# Normal forms (cont'd)

- A successively more restrictive set of normal forms are defined, each avoiding a typical update anomaly
- Normal forms are defined based on FDs

# First Normal Form (1NF)

## Definition

A relation is in First Normal form if all attributes are atomic, i.e. there are no set-valued attributes or nested relations.

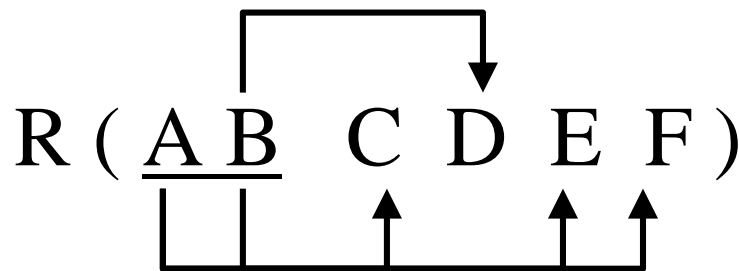
Although this is trivially true of every relation we have seen so far, extensions of relational databases to ‘object-relational’ databases may not satisfy this criterion (this is not discussed further in these lectures – refer Chapter 13 in the textbook)

# What does this definition have to do with FDs?

If there was a set-valued attribute in a relation, then the primary key would not functionally determine the value of that attribute (i.e. for a given pk value there could be many values for that attribute)

# Second Normal Form (2NF)

- Motivating example

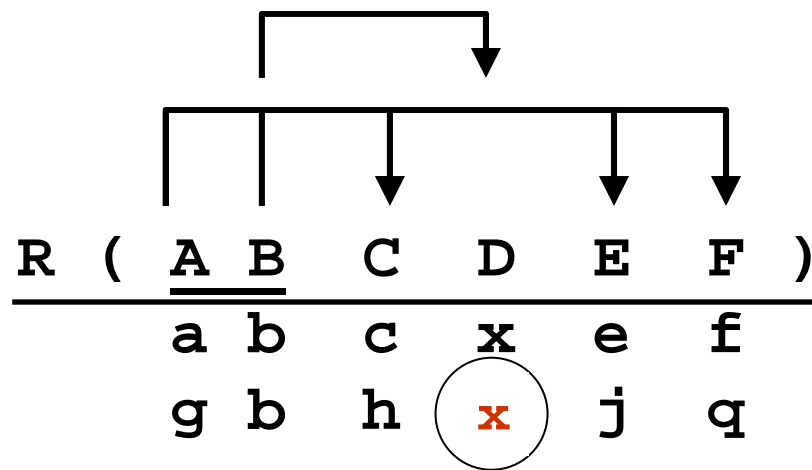


In this example R has  
only one candidate key,  
 $\{AB\}$

Relation R has a ‘**partial dependency**’  $B \twoheadrightarrow D$   
because D is functionally dependent on B and  
B is part of a key of R  
i.e.  $\{B\} \subset \{A, B\}$

# 2NF

- A partial dependency is not desirable, consider




$AB \rightarrow CEF$   
 $B \rightarrow D$

- the two tuples agree on the value of B, and
- $B \rightarrow D$ , therefore this MUST be 'x' -- hence this design is redundant

## 2NF (cont.)

Conversely, even though the two tuples below *violate*  $B \rightarrow D$ , a DBMS *would accept* them for insertion because they do not agree on AB (since their primary keys, ab and gb, are different)



R	(	<u>A</u>	<u>B</u>	C	D	E	F	)
		a	b	c	x	e	f	
		g	b	h	y	j	q	

Example: update anomaly  
caused by R not being  
in 2NF

# 2NF Definition

## Definition

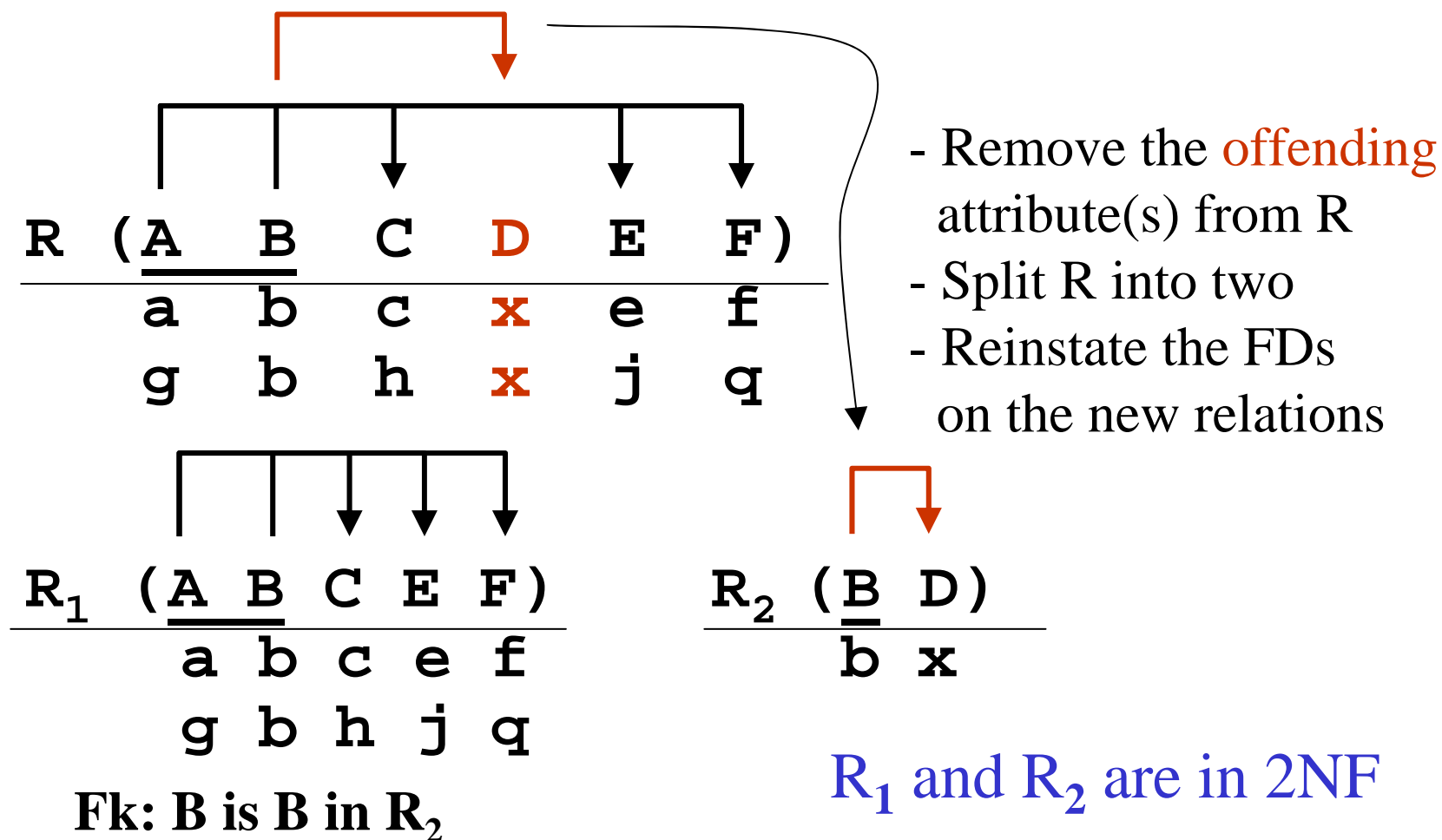
A relation R is in 2NF if there is no such non-prime attribute in R which is functionally dependent on *a part of* a candidate key of R

Note: this definition actually allows prime attributes to be functionally dependent on part of a key. (Even though this is anomalous, the cure is not simple. Only BCNF will eliminate such anomaly)



# 2NF

To fix the problem we could e.g. decompose R:



# Note

We do not learn a 2NF decomposition algorithm, because 3NF is always achievable, so we better use that algorithm straight away (see later)

# Before we continue...

- To improve the database design we may have to split relations into portions, as we did on the previous slide. There are two properties that must be observed when doing this. The decompositions should:
  - be *dependency preserving*;
  - have the *lossless join* property.

# Dependency Preserving Decomposition

## Definition

If we decompose  $R$  into a set of relations  $R_1, R_2, \dots, R_n$  and  $R$  has a set of FDs  $F$ ,  
and there is a set of FDs  $G$  equivalent to  $F$  such that  
for each  $X \rightarrow Y$  both  $X$  and  $Y$  are part of the same  $R_i$   
then the decomposition is *dependency preserving*

In the simple case the decomposition allows  
each FDs in  $F$  to be represented in one of  $R_i$

Informally: we do not want any FD to be lost

# Example

- Given  $R(A\ B\ C\ D)$  with the FD set  
 $A \rightarrow B\ A \rightarrow C\ B \rightarrow D\ C \rightarrow D$
- $R_1(ABC)\ A \rightarrow B\ A \rightarrow C$   
 $R_2(BCD)\ C \rightarrow D\ B \rightarrow D$  is dependency preserving
- $R_1(AB)\ A \rightarrow B$   
 $R_2(BCD)\ C \rightarrow D\ B \rightarrow D$  is not dependency preserving ( $A \rightarrow C$  is lost)

# Lossless Join Property

## Definition

A decomposition  $R_1, R_2, \dots, R_n$  of a relation  $R$  has the *lossless join* property if  $R$  can be reconstructed through joining  $R_1, R_2, \dots, R_n$

(In other words the decomposition is such that no spurious tuples are generated through joining  $R_1, R_2, \dots, R_n$  [refer Lecture 15])

# Lossless Join Property for Two Relations

A decomposition into  $R_1$  and  $R_2$  of  $R$  has the lossless join property if  $R_1 \cap R_2$  is a superkey in either  $R_1$  or in  $R_2$  or both  
[refer **LJ1** 15.1.3, in the textbook]

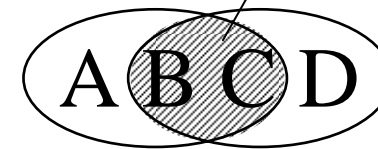
(The above definition is simpler but equivalent to the one in the book)

**Remember this!**

# Example

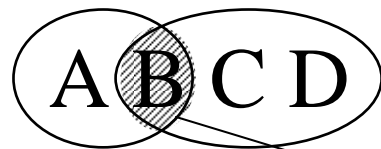
- Given  $R (A \ B \ C \ D)$  with the FD set  
 $A \rightarrow B \ A \rightarrow C \ B \rightarrow D \ C \rightarrow D$

Superkey in  
BCD ( $R_2$ )



- $R_1(A\mathbf{BC}) \ A \rightarrow B \ A \rightarrow C$   
 $R_2(\mathbf{BCD}) \ C \rightarrow D \ B \rightarrow D$  is lossless, since **BC** is a superkey in  $R_2$

- $R_1(A\mathbf{B}) \ A \rightarrow B$   
 $R_2(\mathbf{BCD}) \ C \rightarrow D \ B \rightarrow D$  is not lossless, since **B** is not a key in either  $R_1$  or in  $R_2$

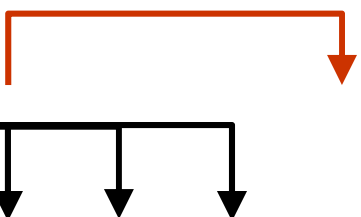


Not a Superkey in AB ( $R_1$ ) or BCD ( $R_2$ )



# Third Normal Form (3NF)

- Motivating example: R is in 2NF but it is still redundant



R	(	<u>A</u>	B	C	D	E	F	)
		a	b	c	x	e	f	
		g	b	c	y	j	f	

(Note:  $A \rightarrow C$  and  $C \rightarrow F$  is called a *transitive dependency*)

- The left hand side of  $C \rightarrow F$  is not a superkey, *therefore* there can be two tuples in R which have the same value for C, therefore
- if these tuples disagree on the value of F then they violate  $C \rightarrow F$
- if they agree on C and F then there is redundancy

# 3NF Definition

## Definition

A relation R is in 3NF, if for *every* non-trivial functional dependency  $X \rightarrow Y$  in R

- either X is a superkey, or
- Y is prime attribute

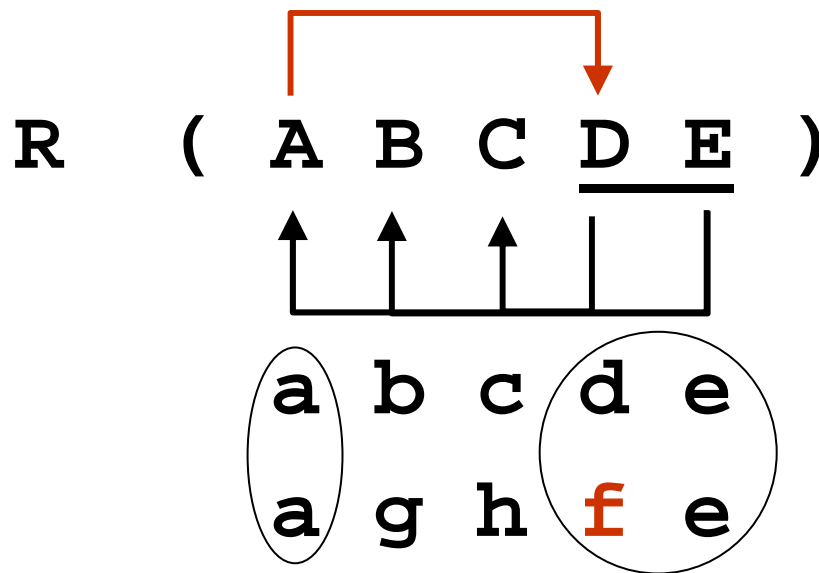
**Note:** this definition actually allows prime attributes to be functionally dependent on a set of attributes that do not form a superkey. (Even though this is anomalous, the cure is not simple. Only BCNF will eliminate such anomaly)

# 3NF algorithm

- It is always possible to produce an FD preserving lossless join decomposition of a relation  $R$  such that the result is in 3NF
- Algorithm [15.4] will be presented in a subsequent lecture

# Boyce-Codd Normal Form (BCNF)

Motivating example: even though R is in 3NF  
still there is redundancy  $DE \rightarrow ABC, A \rightarrow D$



DE is the primary key of  
relation R

If the two tuples do not  
agree on DE the DBMS  
admits them and the DB  
population may **violate**  
 $A \rightarrow D$

# BCNF definition

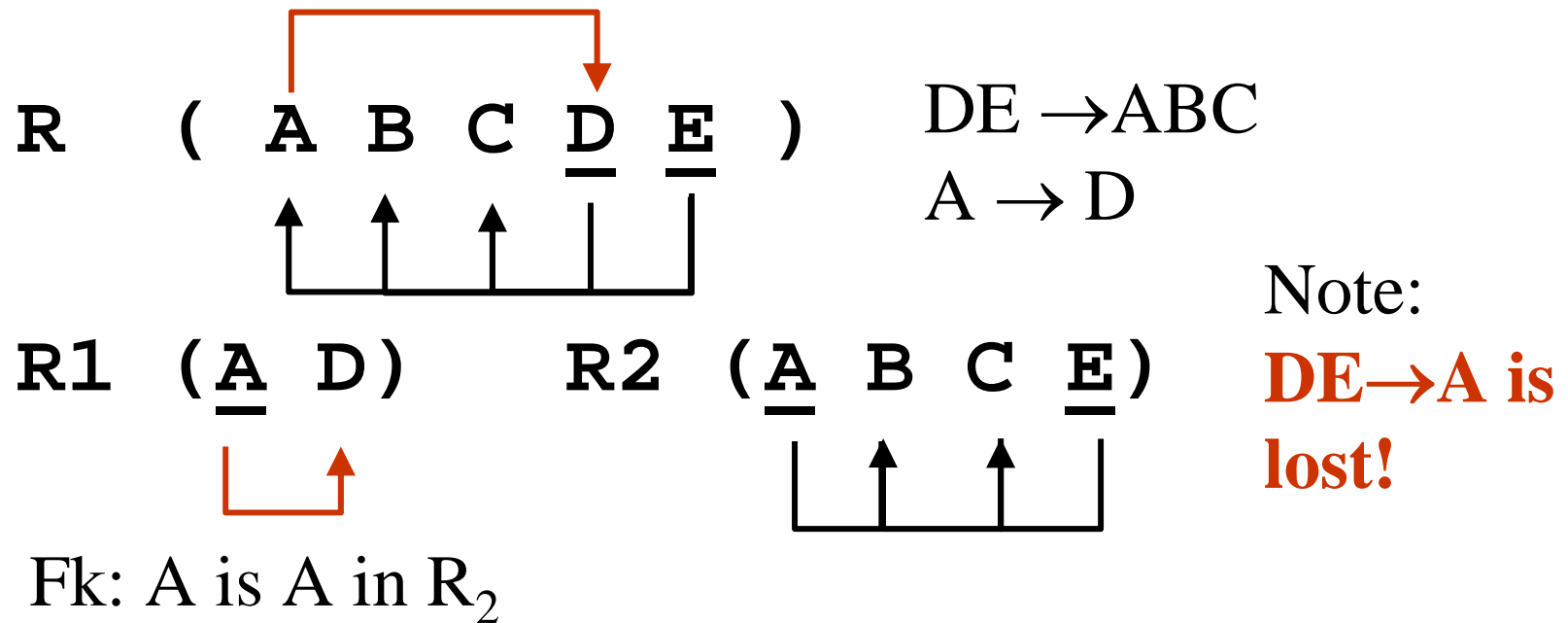
## Definition

A relation R is in BCNF, if for *every* non-trivial functional dependency  $X \rightarrow Y$  in R  
X is a superkey

Note that BCNF is a further restriction on 3NF.

## BCNF (cont.)

A remedy for the previous example:



If the cost of redundancy in R is high we might opt for BCNF but if e.g.  $DE \rightarrow A$  is lost, the DBMS must always check when inserting a new tuple that  $DE \rightarrow A$  still holds

# BCNF algorithm

- It is always possible to produce a lossless join decomposition of a relation  $R$  such that the result is in BCNF
- It is often not possible to produce a dependency preserving decomposition into BCNF
- Algorithm [15.3] will be presented in a subsequent lecture.

$$1NF < 2NF < 3NF < BCNF$$

- If a relation is in BCNF then it is in 3NF, 2NF and 1NF
- If a relation is in 3NF then it is in 2NF and 1NF, etc...
- 3NF is always achievable and should be attained
- To produce BCNF is worth if no FDs are lost, otherwise generally not worth it



# Stronger normal forms

- There are stronger normal forms (4NF, 5NF, etc) which we do not study in this subject. [Refer to textbook]

# Summary

- Trivial dependencies ( $X \rightarrow Y$  trivial if  $X \supseteq Y$ )
- Prime attributes (part of any ck)
- 1NF: no set valued attributes  
Fix: have separate relations
- 2NF: no **partial** dependencies (non-prime attribute(s) determined by **part of** ck)  
Fix: decompose relation, move offending attribute(s) to other relations

# Summary (cont.)

- Decomposition: lossless join, dependency preserving.
- 3NF:  $X \rightarrow Y$  in  $F$  then:  $X$  must be superkey OR  
Y prime attribute
- BCNF:  $X \rightarrow Y$  in  $R$  then  $X$  must be superkey  
Problem: some FD may be lost;  
Fix: Decompose – lossless join, not always FD preserving.

The end