

Database Normalization

CSCI 220: Database Management and Systems Design

Slides adapted from
Simon Miner
Gordon College

Practice Quiz: Decomposition

- Check whether the decompositions have the lossless join property

Scheme (ABCD)

FDs: $A \rightarrow B$ and $B \rightarrow CD$

Decomposition	Lossless join?
(AB)(BCD)	
(AB)(BC)(BD)	
(AB)(ACD)	

Scheme (ABCD)

FDs: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow CD$

Decomposition	Lossless join?
(AB)(BCD)	
(ACD)(BD)	

Agenda

- Homework Review
- Functional Dependencies (continued)
- Database Normalization
- Introduce Homework

Today you will learn...

- How to test the "goodness" of a relational schema
- How to create a normalized relational schema

Database Normalization

Introduction

- Terminology review
 - Relation schema: set of attributes for a relation (R_1, R_2, \dots)
 - Relation: the actual data stored in some relational schema (r_1, r_2, \dots)
 - Tuple: a single actual row in the relation (t_1, t_2, \dots)

Database Design Goals (Updated)

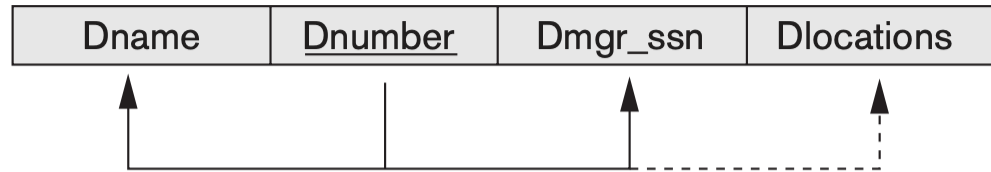
- Goals
 - Avoid redundancies and the resulting from insert, update, and delete anomalies by decomposing schemes as needed
 - Ensure that all decompositions are lossless-join
 - Ensure that all decompositions are dependency preserving
- Sometimes you cannot have all three
 - Allow for redundancy to preserve dependencies
 - Or give up dependency preservation to eliminate redundancy
 - **Never** give up lossless-join as doing so would remove the ability to connect tuples in different relations
- Database *normal forms* help eliminate redundancy and anomalies
 - Specify a set of decomposition rules to convert a database that is not in a given normal form into one that is

First Normal Form (1NF)

- A relation scheme R is in 1NF if the domains of all attributes in R are atomic
 - Single and non-composite
 - Guarantees that each non-key attribute in R is functionally dependent on the primary key

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 14.9

Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Second Normal Form (2NF)

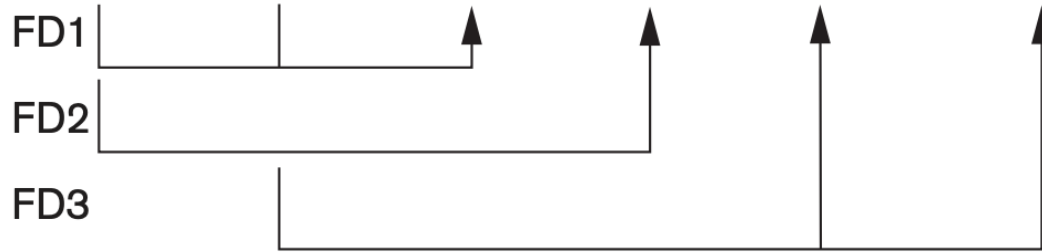
- A 1NF relationship scheme R is in 2NF if each non-key attribute is fully functionally dependent on each candidate key
 - Functionally dependent on the whole key, not just part of it
 - This restriction does not apply to key attributes
 - Avoids redundancy of information which is dependent on part of the primary key
- Any non-2NF scheme can be decomposed into 2NF schemes by factoring out
 - The non-key attributes dependent on a portion of a candidate key
 - The portion of the candidate key these attributes depend on
- Any 1NF scheme without a composite primary is in 2NF

2NF Example

(a)

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------



2NF Normalization

EP1

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------



EP2

<u>Ssn</u>	Ename
------------	-------



EP3

<u>Pnumber</u>	Pname	Plocation
----------------	-------	-----------



Third Normal Form (3NF)

- A 2NF relation scheme R is in 3NF if no non-key attribute of R is transitively dependent on a candidate key through some other non-key attribute(s)
 - This restriction does not apply to key attributes
 - Transitive dependencies on a candidate key lead to insert, update, and delete anomalies
- Any non-3NF scheme can be decomposed into 3NF schemes by factoring out
 - The transitively dependent attributes
 - The “transitional” attributes which connect these to the candidate key
- Any non-3NF relation can be decomposed into 3NF in a lossless-join and dependency preserving manner

3NF Example

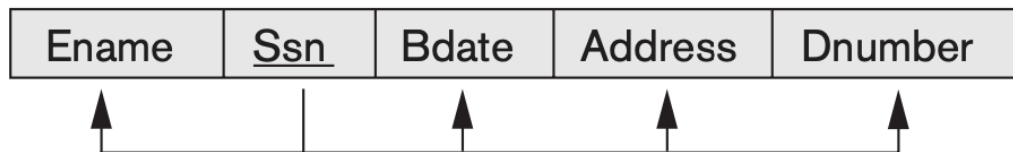
(b)

EMP_DEPT

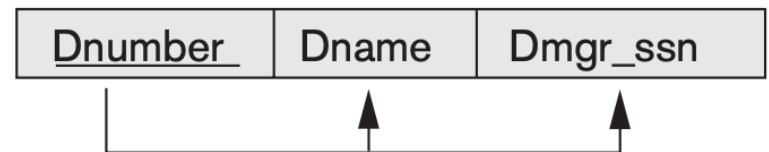


3NF Normalization

ED1



ED2



3NF Decomposition Algorithm

```
Let  $F_c$  be a canonical cover for  $F$ ;  
 $i := 0$ ;  
for each functional dependency  $\alpha \rightarrow \beta$  in  $F_c$  do  
  if none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains  $\alpha \beta$   
    then begin  
       $i := i + 1$ ;  
       $R_i := \alpha \beta$   
    end  
if none of the schemas  $R_j$ ,  $1 \leq j \leq i$  contains a candidate key for  $R$   
  then begin  
     $i := i + 1$ ;  
     $R_i :=$  any candidate key for  $R$ ;  
  end  
/* Optionally, remove redundant relations */  
  
repeat  
if any schema  $R_j$  is contained in another schema  $R_k$   
  then /* delete  $R_j$  */  
     $R_j = R_k$ ;  
     $i = i - 1$ ;  
return ( $R_1, R_2, \dots, R_i$ )
```

1NF, 2NF, 3NF Summary

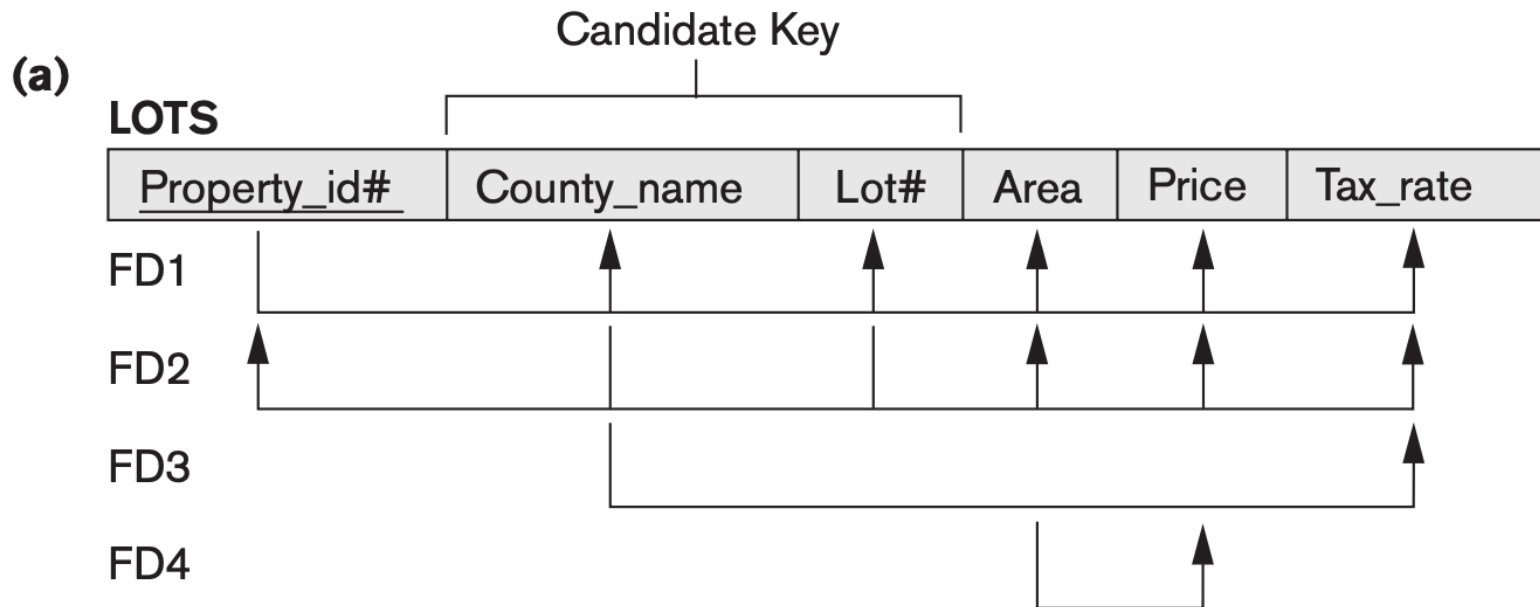
Table 14.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

LOTS Example

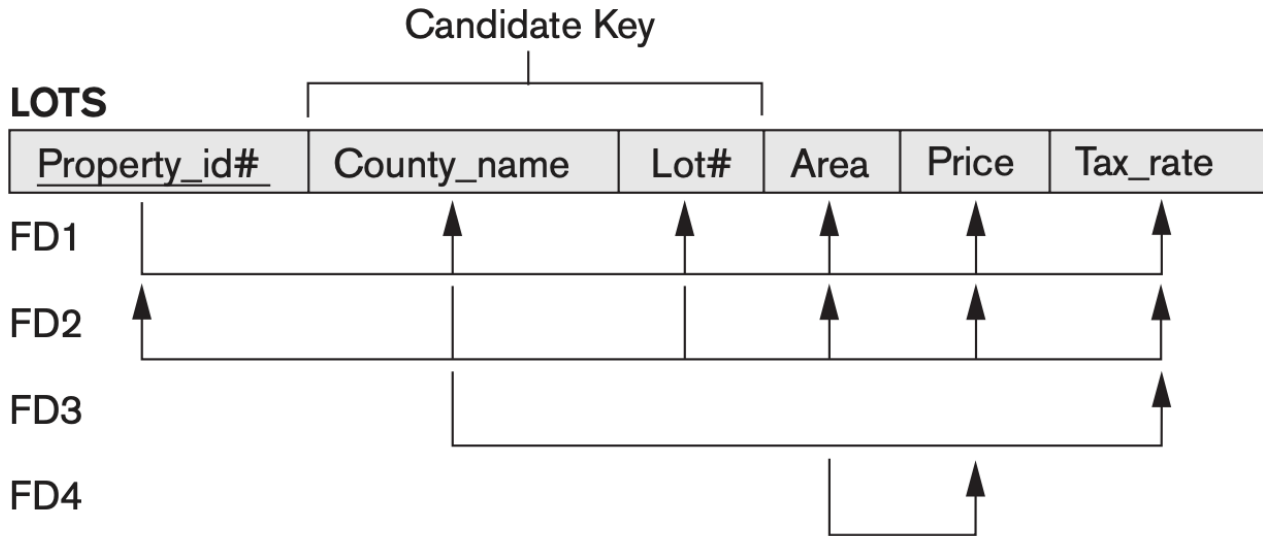
Figure 14.12

Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.

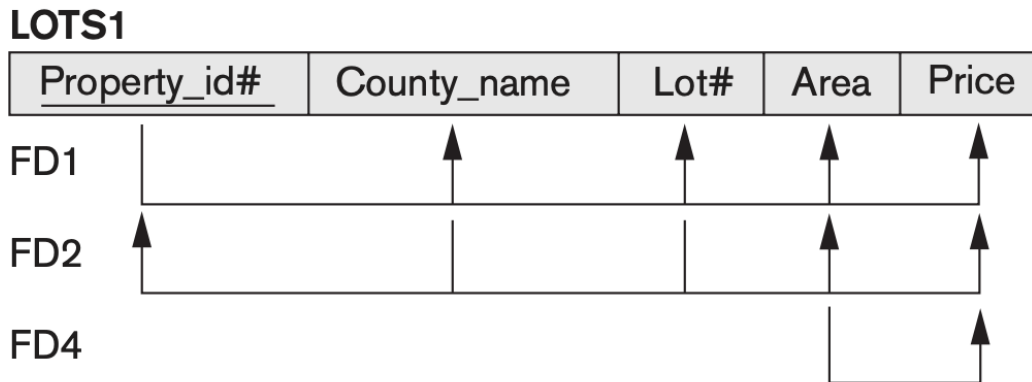


LOTS Example

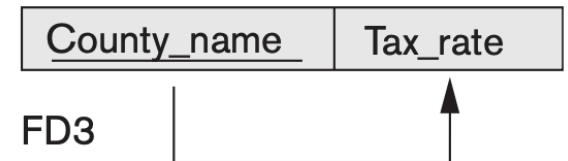
(a)



(b)



LOTS2

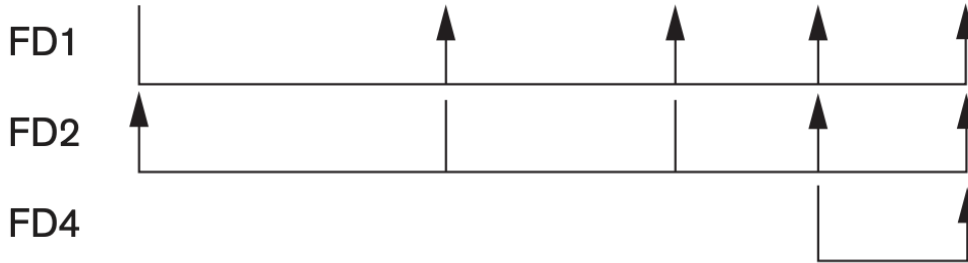


LOTS Example

(b)

LOTS1

<u>Property_id#</u>	County_name	Lot#	Area	Price
---------------------	-------------	------	------	-------



LOTS2

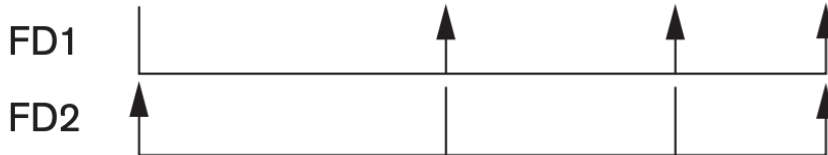
<u>County_name</u>	Tax_rate
--------------------	----------



(c)

LOTS1A

<u>Property_id#</u>	County_name	Lot#	Area
---------------------	-------------	------	------



LOTS1B

<u>Area</u>	Price
-------------	-------

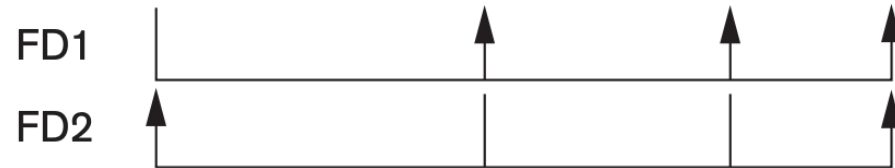


LOTS Example

(c)

LOTS1A

<u>Property_id#</u>	County_name	Lot#	Area
---------------------	-------------	------	------

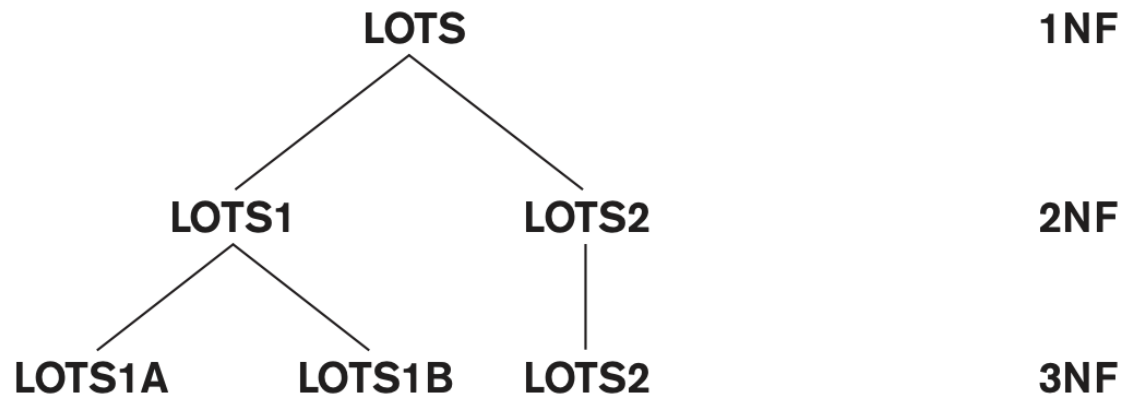


LOTS1B

<u>Area</u>	Price
-------------	-------



(d)



Boyce-Codd Normal Form (BCNF)

- 3NF did not take multiple candidate keys into account
 - BCNF developed to address this
- A normalized relation is in BCNF if every FD satisfied by R is of the form $A \rightarrow B$, where A is a superkey
 - BCNF is a stronger 3NF
 - Every BCNF schema is also in 3NF
 - Not every 3NF schema is in BCNF
- Some 3NF schemas cannot be decomposed into BCNF in a lossless-join and dependency preserving manner
- BCNF does not build on other normal forms

LOTS Example: Have we gone too far?

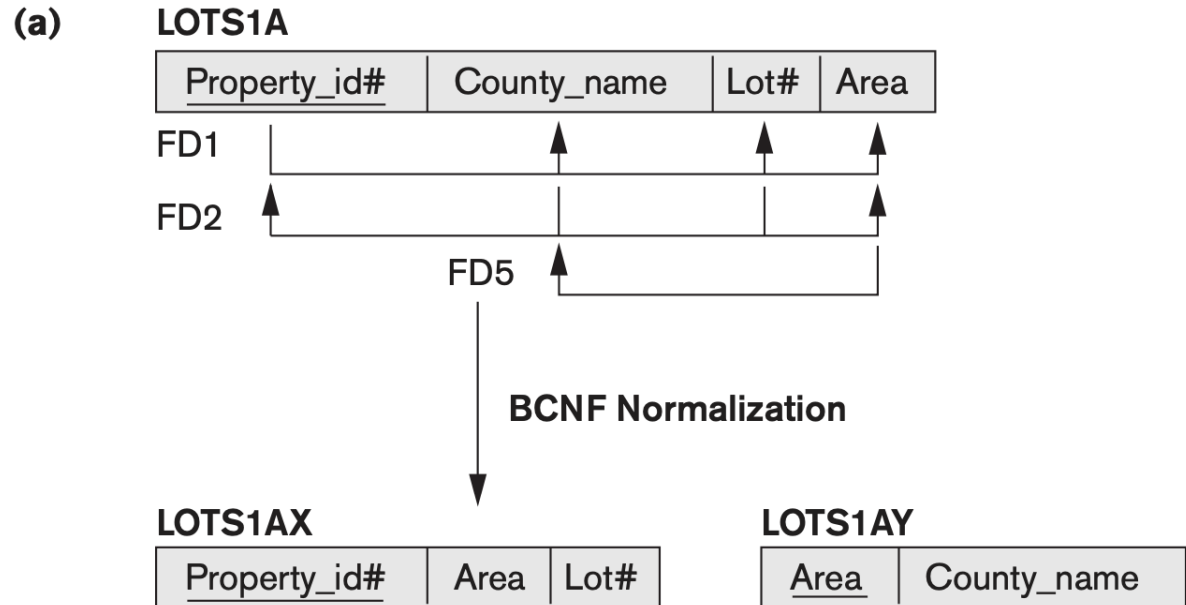
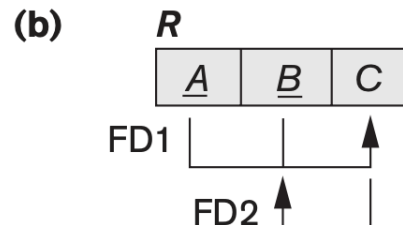


Figure 14.13

Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. $C \rightarrow B$.



3NF, but not BCNF


- FDs:
 - Court, Start time → End time, Rate Type
 - Court, End time → Start time, Rate Type
 - Rate Type → Court
- Rate Type:
 - SAVER, for member Court 1 bookings
 - STANDARD, for non-member Court 1 bookings
 - PREMIUM-A, for member Court 2 bookings
 - PREMIUM-B, for non-member Court 2 bookings

Today's court bookings

Court	Start time	End time	Rate type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	PREMIUM-B
2	11:30	13:30	PREMIUM-B
2	15:00	16:30	PREMIUM-A

https://en.wikipedia.org/wiki/Boyce-Codd_normal_form

Why isn't it BCNF?

- FDs:
 - Court, Start time → End time, Rate Type
 - Court, End time → Start time, Rate Type
 - Rate Type → Court
 - Candidate keys:
 - Court, Start time
 - Court, End time
 - Rate type, Start time
 - Rate type, End time
- Problem: left-hand side is not a superkey**
- 

Today's court bookings

Court	Start time	End time	Rate type
-------	------------	----------	-----------

https://en.wikipedia.org/wiki/Boyce-Codd_normal_form

Normalized to BCNF

Rate types

Rate type	Court	Member flag
SAVER	1	Yes
STANDARD	1	No
PREMIUM-A	2	Yes
PREMIUM-B	2	No

FDs:

- Rate type → Court, Member flag
- Court, Member flag → Rate type

Candidate keys: {Rate type},
{Court, Member flag}

Today's bookings

Member flag	Court	Start time	End time
Yes	1	09:30	10:30
Yes	1	11:00	12:00
No	1	14:00	15:30
No	2	10:00	11:30
No	2	11:30	13:30
Yes	2	15:00	16:30

FDs: Court, Start time → End time, Member flag
Court, End Time → Start time, Member flag

Candidate keys: {Court, Start time},
{Court, End time}

Multivalued Dependencies (MVDs)

- MVDs can occur when independent one-to-many relationships are included in the same relation
- The set of attributes *A* *multi-determines* the set of attributes *B* if in any relation including attributes *A* and *B*:
 - For any given value of *A* there is a (non-empty) set of values for *B* such that we expect to see all of those *B* values (and no others) associated with the given *A* along with the remaining attribute values
 - The number of *B* values associated with a given *A* value may vary between *A* values

MVD Example

- FDs:
 - team_name, owner_name → arena
 - arena → team_name
- "team_name" has one-to-many relationship with "owner_name"
 - team_name → owner_name

Team

<u>team_name</u>	<u>owner_name</u>	arena
Golden State Warriors	Peter Guber	Chase Center
Golden State Warriors	Joe Lacob	Chase Center

More MVDs

- Suppose that "team_name" has one-to-many relationships with "owner_name" and "team_color"
 - team_name \rightarrow owner_name
 - team_name \rightarrow team_color

Team

	<u>team_name</u>	<u>owner_name</u>	<u>team_color</u>	arena
t_1	Golden State Warriors	Peter Guber	royal blue	Chase Center
t_2	Golden State Warriors	Joe Lacob	yellow	Chase Center
t_3	Golden State Warriors	Joe Lacob	royal blue	Chase Center
t_4	Golden State Warriors	Peter Guber	yellow	Chase Center

Formal Definition of Multivalued Dependency

Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

$\alpha \twoheadrightarrow \beta$ e.g., team \twoheadrightarrow team_color

holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$

e.g., $t_1[\alpha] = \text{"Golden State Warriors"}$

$t_1[\beta] = t_3[\beta]$

e.g., $t_3[\beta] = \text{"royal blue"}$

$t_2[\beta] = t_4[\beta]$

e.g., $t_4[\beta] = \text{"yellow"}$

$t_1[R - (\alpha \cup \beta)] = t_4[R - (\alpha \cup \beta)]$

e.g., $t_1[R - \beta] = (\text{"Peter Guber"}, \text{"Chase Center"})$

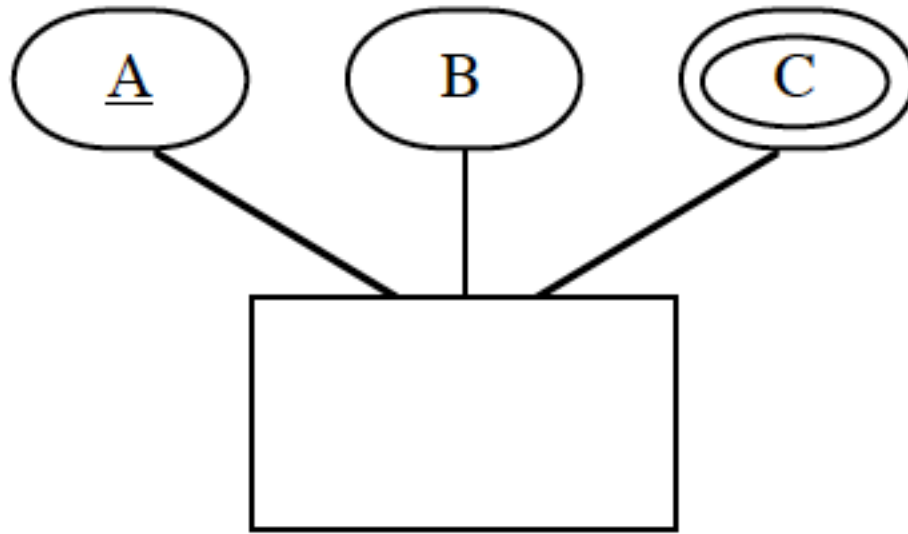
$t_2[R - (\alpha \cup \beta)] = t_3[R - (\alpha \cup \beta)]$

e.g., $t_2[R - (\alpha \cup \beta)] = (\text{"Joe Lacob"}, \text{"Chase Center"})$

Note: t_1, t_2, t_3, t_4 aren't necessarily distinct.

MVDs and E-R Diagrams

- MVDs correspond to multi-valued attributes



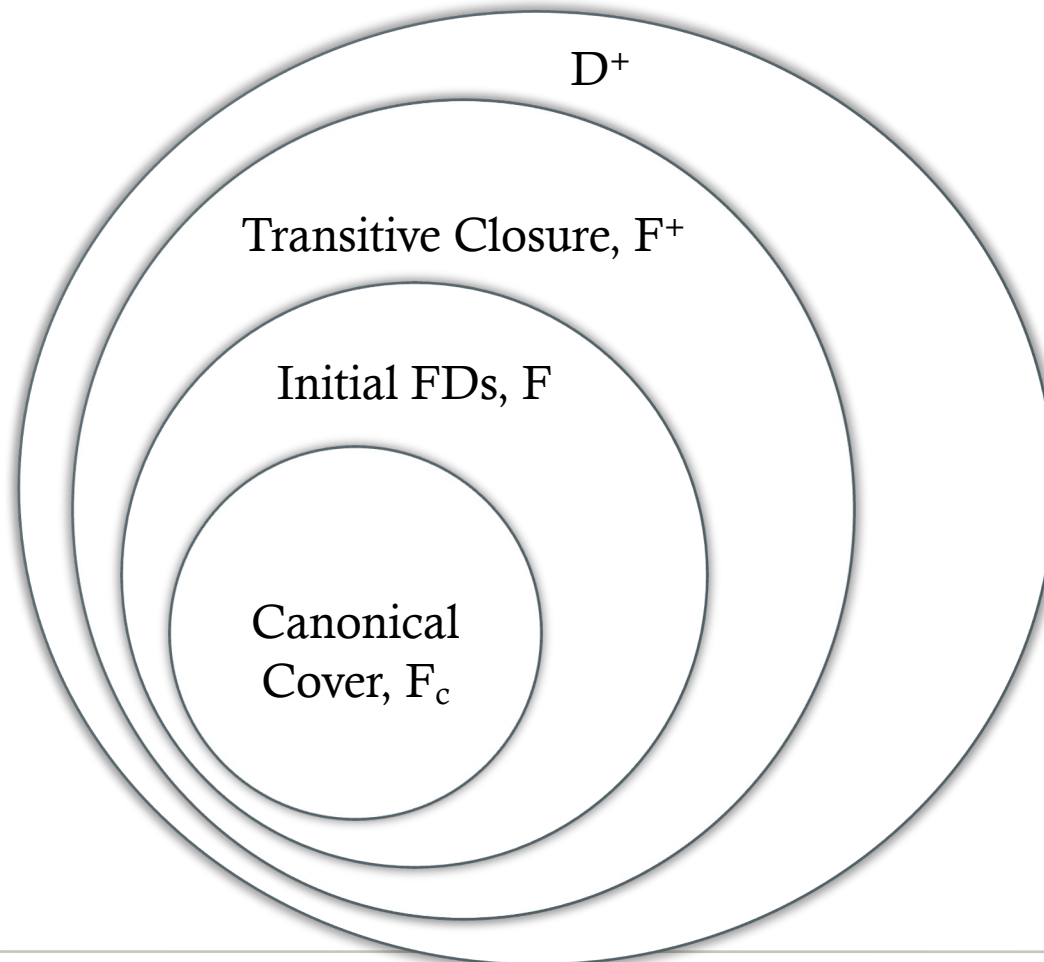
$A \rightarrow B$

$A \twoheadrightarrow C$

Properties of MVDs

- MVDs require the addition of certain tuples
 - Example: copies of a book with multiple authors
 - Opposite to FDs which prohibit certain tuples
- If $A \rightarrow B$, then $A \twoheadrightarrow B$
 - The reverse isn't true. FDs are a special case of MVDs.
- An MVD is trivial if either of the following is true
 - Its right-hand side is a subset of its left-hand side (just like FDs)
 - The union of its left- and right-hand sides is the entire scheme
- The closure D^+ of D is the set of all FDs and MVDs implied by D
 - D^+ can be computed from the formal definitions of FD and MVD

Closure D^+



Fourth Normal Form (4NF)

- A relation schema R is in **4NF** for all MVDs in D^+ of the form $\alpha \twoheadrightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
 - $\alpha \twoheadrightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
 - α is a superkey for schema R (in which case it is an FD)
- If a relation is in 4NF it is in BCNF
- 4NF avoids redundancies introduced by MVDs

Normalized to 4NF

Team

<u>team_name</u>	arena
Golden State Warriors	Chase Center

Team_Owner

<u>team_name</u>	<u>owner_name</u>
Golden State Warriors	Peter Guber
Golden State Warriors	Joe Lacob

Team_Color

<u>team_name</u>	<u>team_color</u>
Golden State Warriors	royal blue
Golden State Warriors	yellow

Database Design Guidelines

- Use the highest normal form possible
 - 4NF unless it is not dependency preserving
 - BCNF unless (in rare cases) it is not dependency preserving
 - 3NF otherwise – never need to compromise beyond this
 - Lower normal forms may be useful for efficiency purposes
- Use good keys
 - Every attribute should depend on the key, the whole key, and nothing but the key (BCNF)
 - Avoid composite keys (automatic 2NF)
 - Generate a unique single-attribute key if needed
- Factor out transitive dependencies (“sub-relations”) into their own schemes (3NF)
- Isolate MVDs to their own schema (4NF)

Approaches to Database Design

- Start with a universal relation and decompose it
 - The approach taken in this lecture
- Start with an E-R diagram
 - Modify it while you normalize it
 - Normalize it when converting it to a relational schema

Database Design Lab

CSCI 220: Database Management and Systems Design

Practice Quiz: Normalization

- With a partner, discuss the differences between:
 - 1NF
 - 2NF
 - 3NF
 - BCNF
 - 4NF

Normalization Summary

- 1NF: No multivalued attributes or nested relations (e.g., comma-separated values, JSON, etc. in an attribute)
- 2NF: 1NF, plus attributes cannot depend on a subset of any candidate key, unless they are part of a candidate key themselves
- 3NF: 2NF, plus attributes must depend **only** on the candidate key(s) (no transitive dependencies), unless they are part of a candidate key themselves
- BCNF: For each relation, the left-hand-side of all applicable FDs must be a superkey for the relation
- 4NF: Doesn't allow repetition from MVDs

Review

BCNF and MVDs

Big Picture

- Functional dependencies and normalization help developers reason about data consistency
- Sometimes trade-offs are necessary (e.g., eliminate redundancy, at the cost of efficient checking of consistency?)
- Often the only cost is *thinking carefully*
 - Taking the time to do this will save you from headaches down the road (e.g., Did we overcharge these customers? Were these orders fulfilled? Etc.)