VirginiaTech

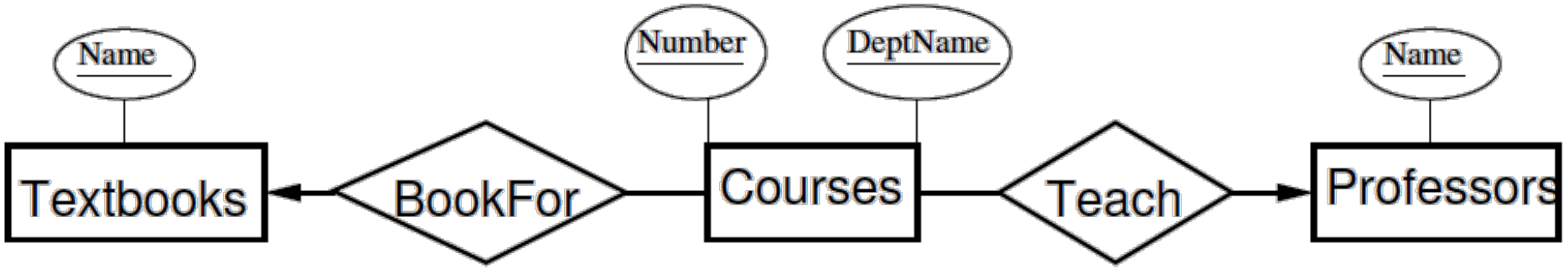
CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #17: Multivalued Dependencies and 4NF

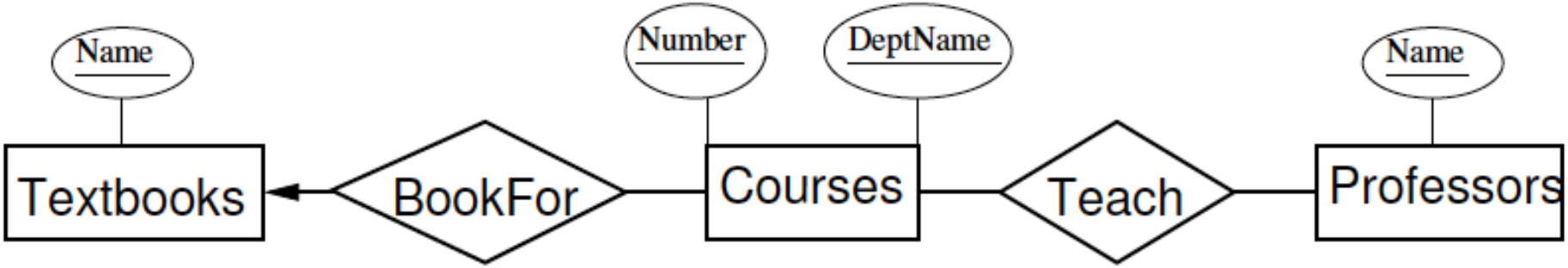
- Homework 4 is out. Due next Wed, April 10, in class.

Attribute Independence in BCNF Schemas



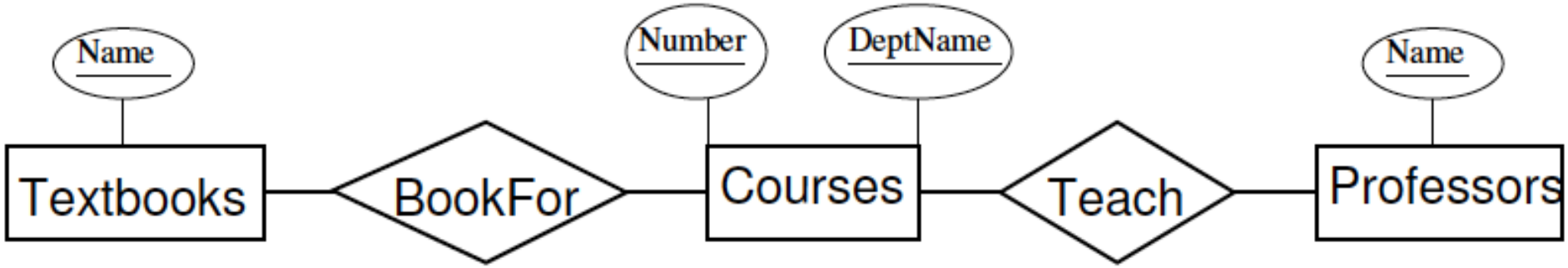
- Relation is Courses(Number, DeptName, Textbook, Professor)

Attribute Independence in BCNF Schemas



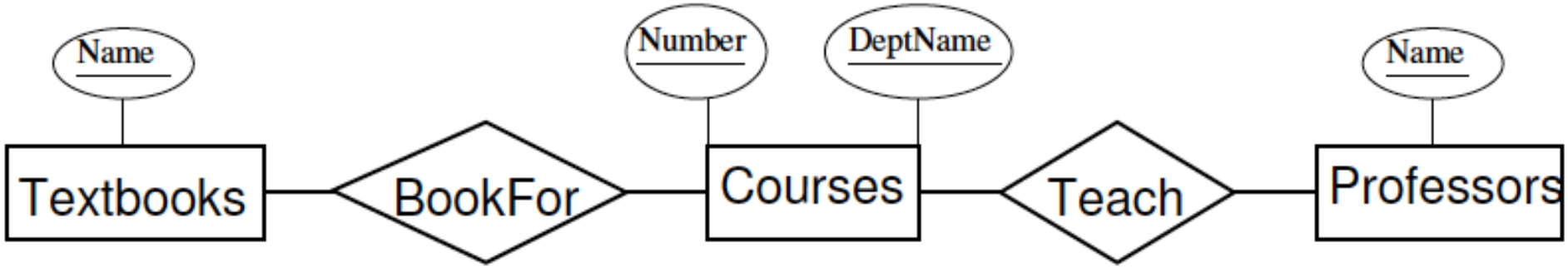
- Relation is Courses(Number, DeptName, Textbook, Professor)
- Allow more than professor to teach a course. Keep the same relation. Is the relation in BCNF?
  - No

Attribute Independence in BCNF Schemas



- Relation is Courses(Number, DeptName, Textbook, Professor)
- Allow more than one textbook for the same course. Keep the same relation. Each professor uses every textbook in the course. Is the relation in BCNF?
  - Yes!

Attribute Independence in BCNF Schemas



- Relation is Courses(Number, DeptName, Textbook, Professor)
- Is there any redundancy in the relation

Attribute Independence in BCNF Schemas

- BCNF schemas can have redundancy, e.g., when we force two or more many-many relationships in a single relation
- The relation is Courses(Number, DeptName, Textbook, Professor)
  - Each Course can have multiple required Textbooks
  - Each course can have multiple Professors
  - A Professor uses every required textbook while teaching a Course

VirginiaTech

Attribute Independence in BCNF Schemas

- The relation is Courses(Number, DeptName, Textbook, Professor)
  - Each Course can have multiple required Textbooks
  - Each course can have multiple Professors
  - A Professor uses every required textbook while teaching a Course

Number	DeptName	Textbook	Professor
4604	CS	FCDB	Ullman
4604	CS	SQL Made Easy	Ullman
4604	CS	FCDB	Widom
4604	CS	SQL Made Easy	Widom

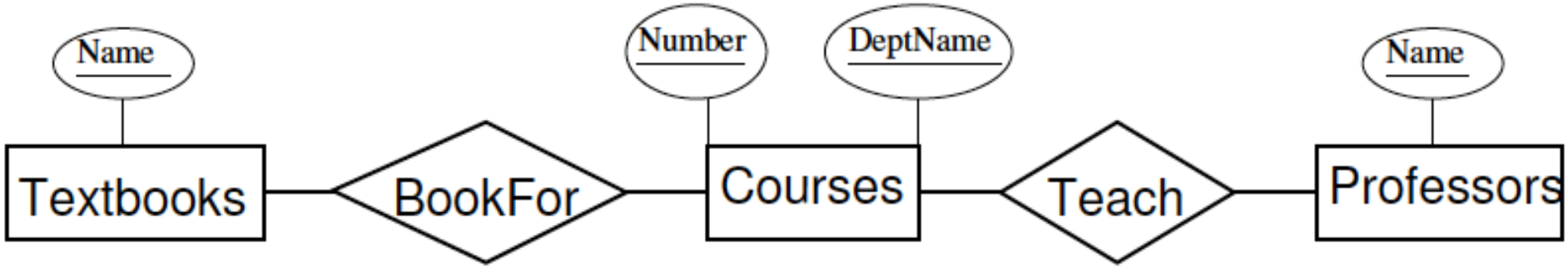
  - The relation is in BCNF, since there are no non-trivial FDs.

Is there any redundancy?

Yes---in the Textbook and Professor attributes

VirginiaTech

Removing redundancy from Courses



Number	DeptName	Textbook	Professor
4604	CS	FCDB	Ullman
4604	CS	SQL Made Easy	Ullman
4604	CS	FCDB	Widom
4604	CS	SQL Made Easy	Widom

- We can remove the redundancy by decomposing Courses into:

Courses1 (Number, DeptName, Textbook)

Courses2 (Number, DeptName, Professor)

FDs and BCNF are not rich enough to express these types of redundancies

Multi-valued Dependencies

- A multi-valued dependency (MVD or MD) is an assertion that two sets of attributes are independent of each other
- The multi-valued dependency A1 A2 … An→→ B1 B2 … Bm holds in a relation R if in every instance of R,

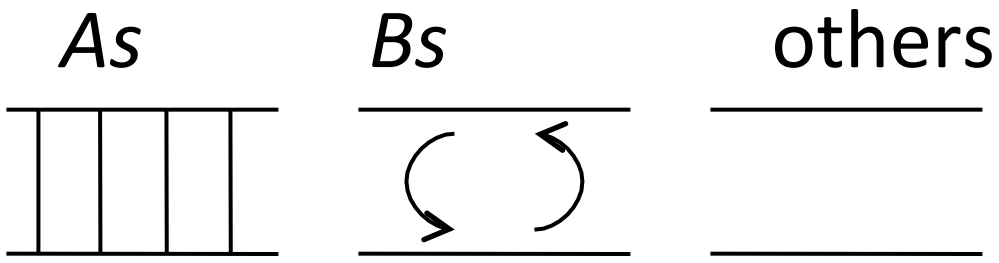
  for every pair of tuples t and u in R that agree on all the A's, we can find a tuple v in R that agrees

  1. With both t and u on the A's
  2. With t on the B's
  3. With u on all those attributes of R that are not A's or B's

i.e., for each value of As, the values of Bs are independent of the values of R-As-Bs

Multi-valued Dependencies: Another Equivalent View

- The *multivalued dependency As →→ Bs* holds in a relation *R* if whenever we have two tuples of *R* that agree in all the attributes of *As*, then we can swap their *Bs* components and get two new tuples that are also in *R*.



Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	
4604	CS	SQL Made Easy	Ullman	
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Textbook is an MD

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	*u*
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Textbook is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Textbook , and with u on Professor

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t* *v*
4604	CS	SQL Made Easy	Ullman	*u*
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Textbook is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Textbook , and with u on Professor

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	*u*
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Textbook is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Textbook , and with u on Professor

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	
4604	CS	FCDB	Widom	*v*
4604	CS	SQL Made Easy	Widom	*u*
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Textbook is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Textbook , and with u on Professor

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	
4604	CS	SQL Made Easy	Ullman	
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Professor is an MD

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	*u*
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Professor is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Professor , and with u on Textbook

Example…

Number	DeptName	Textbook	Professor		
4604	CS	FCDB	Ullman	*t*	
4604	CS	SQL Made Easy	Ullman	*u*	*v*
4604	CS	FCDB	Widom		
4604	CS	SQL Made Easy	Widom		
2604	CS	Data Structures	Ullman		
2604	CS	Data Structures	Widom		

- Number DeptName →→ Professor is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Professor , and with u on Textbook

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	*u*
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Professor is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Professor , and with u on Textbook

Example…

Number	DeptName	Textbook	Professor	
4604	CS	FCDB	Ullman	*t*
4604	CS	SQL Made Easy	Ullman	*v*
4604	CS	FCDB	Widom	
4604	CS	SQL Made Easy	Widom	*u*
2604	CS	Data Structures	Ullman	
2604	CS	Data Structures	Widom	

- Number DeptName →→ Professor is an MD

For every pair of tuples t and u that agree on Number and DeptName, we can find a tuple v that agrees

1. with both t and u on Number and DeptName
2. with t on Professor , and with u on Textbook

Fun facts about MDs

- Given tuples t, u and v that satisfy an MD, we can infer the existence of another tuple w that agrees

1. With both t and u on A's
2. With u on the B's
3. With t on all those attributes of R that are not in A's or B's

Rules for Manipulating MDs

- **FD promotion:** Every FD A→B is an MD A→→B

(Proof:

Make u and v the same tuple.

Note that definition of keys depends on FDs and not on MDs)

Rules for Manipulating MDs

- **FD promotion:** Every FD A→B is an MD A→→B
- **Trivial MDs:**

1. If A→→B, then A→→AB
2. If A1, A2…, An and B1, B2, …, Bm make up *all* the attributes of a relation, then A1, A2, …An →→ B1, B2, …Bm holds in the relation

Rules for Manipulating MDs

- **FD promotion:** Every FD A→B is an MD A→→B
- **Trivial MDs:**

1. If A→→B, then A→→AB
2. If A1, A2…, An and B1, B2, …, Bm make up *all* the attributes of a relation, then A1, A2, …An →→ B1, B2, …Bm holds in the relation

- **Transitive rule:** A→→B and B→→C ➔ A→→C
- **Complementation rule:** if A→→B, then A→→C, where C is the set of attributes not in the MD

Rules for Manipulating MDs

- Note that **Splitting rule does not hold!** If A→→BC, then it is not true that A→→B and A→→C

Decomposition into 4NF

- Consider relation R with set of attributes X
- A1 A2 … An →→ B1 B2 … Bm violates 4NF
- Decompose R into two relations whose attributes are:

1. The As and Bs i.e. {A1 A2 … An, B1, B2, …, Bm}
2. All the attributes of R which are not Bs i.e. X – {B1, B2 …, Bm}
3. Recursively check if the new relations are in 4NF and repeat

Example…

`Drinkers(name, addr, phones, beersLiked)`

- FD: `name` → `addr`
- Nontrivial MVD's:

  `name` →→ `phones` and
  `name` →→ `beersLiked`.
- Only key: {`name, phones, beersLiked`}
- All three dependencies above violate 4NF.
- Successive decomposition yields 4NF relations:

  `D1(name, addr)`
  `D2(name, phones)`
  `D3(name, beersLiked)`

Decomposition into 4NF

- Projecting MDs: Need a method to discover new FDs!
- Date-Fagin Theorem: if a relation schema is in BCNF and has a key with one attribute, then it is in 4NF

Discovering new FDs and MDs

- Given a set of FDs and MDs, what new FDs and MDs follow?
- Algorithm is similar to the chase algorithm to determine lossless joins

Chase for Discovering New FDs

- If we are given only FDs as input, we can use the algorithm to compute the closure of FDs
- An alternative is to use the chase process
- R(A, B, C, D) satisfies FDs AD→C and DC→B
- Show that FD AD→B holds in R
- Start with a tableau containing two tuples that agree in A and in D but are different in B
- Apply FDs repeatedly to equate values of attributes
- Stop either when both tuples agree in B (the FD holds) or no more FDs can be applied (the FD does not hold)

A	B	C	D
a	b_1	c_1	d
a	b_2	c_2	d

- Since AD→C, c = c1
- Now since DC→B, b1 = b2
- Hence proved AD→B

Chase for Discovering New FDs (2)

- We can also use the chase to infer new FDs when given FDs and MDs as input
- R(A, B, C, D) satisfies the FD D→C and the MD A→→BC
- Show that the FD A→C holds in R
- Start with a tableau containing two tuples that agree in A but are different in C
- Apply the FDs to equate values of attributes
- Apply the MDs to **infer new tuples** in the relation
- Stop either when both tuples agree on C (the FD holds) or no more FDs/ MDs can be applied (the FD does not hold)

A	B	C	D	
a	b_1	c_1	d_1	t
a	b_2	c_2	d_2	u

- Since A→→BC, R must also contain tuples v = (a, b1, c1, d2) and w = (a, b2, c2, d1)
- Applying D→C to t and w, c = c1.
- Hence proved A→C

Chase for discovering new MDs

- Same process for MDs as well ☺
- Relation R(A, B, C, D) satisfies FD A→B, and MD B→→C. Which new MDs hold in R?
- Let us try to prove A→→C holds
- Create a tableau with two different tuples that agree in A and differ in C. If A→→C holds, it implies the existence of two other tuples in R
- Use the FDs and MDs to prove the existence of these tuples
- To simplify name the attributes in the initial two tuples such that one of the new tuples we will prove is (a, b, c, d)

A	B	C	D	
a	b_1	c	d_1	t
a	b	c_2	d	u

- Since A→B, b = b1.
- Since B→→C, R also contains tuples (a, b, c2, d1) and (a, b, c, d), proving A→→C

Another example

- Let's prove transitivity
- If A→→B and B→→C, then A→→C
  - Obvious from the complementation rule if the Schema is (A, B, C)
  - But it holds no matter what the schema; let's assume (A, B, C, D)

The Tableau for A→→C

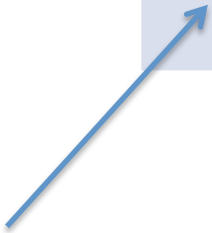
- Goal: derive tuple (a,b1,c2,d1)

A	B	C	D
a	b1	c1	d1
a	b2	c2	d2

The Tableau for A→→C

- Goal: derive tuple (a,b1,c2,d1)

A	B	C	D
a	b1	c1	d1
a	b2	c2	d2
a	b1	c2	d2



Use A→→B to swap B from the first row into the second

The Tableau for A→→C

- Goal: derive tuple (a,b1,c2,d1)

A	B	C	D
a	b1	c1	d1
a	b2	c2	d2
a	b1	c2	d2
a	b1	c2	d1

Use A→→B to swap B from the first row into the second

Use B→→C to swap B from the first row into the second

What if the chase fails?

- We have a counter-example relation!
  - The tableau satisfies all given dependencies
  - Original two rows violate target dependency

Projecting MDs into new relations

- Read Chapter 3.7.4 of the textbook

Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF

Property	3NF	BCNF	4NF
Eliminate redundancy due to FDs			
Eliminate redundancy due to MDs			
Preserves FDs			
Preserves MDs			

Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF

Property	3NF	BCNF	4NF
Eliminate redundancy due to FDs	Maybe	Yes	Yes
Eliminate redundancy due to MDs			
Preserves FDs			
Preserves MDs			

Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF

Property	3NF	BCNF	4NF
Eliminate redundancy due to FDs	Maybe	Yes	Yes
Eliminate redundancy due to MDs	No	No	Yes
Preserves FDs			
Preserves MDs			

Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF

Property	3NF	BCNF	4NF
Eliminate redundancy due to FDs	Maybe	Yes	Yes
Eliminate redundancy due to MDs	No	No	Yes
Preserves FDs	Yes	Maybe	Maybe
Preserves MDs			

Relationships Among Normal Forms

- 4NF implies BCNF, i.e., if a relation is in 4NF, it is also in BCNF
- BCNF implies 3NF, i.e., if a relation is in BCNF, it is also in 3NF

Property	3NF	BCNF	4NF
Eliminate redundancy due to FDs	Maybe	Yes	Yes
Eliminate redundancy due to MDs	No	No	Yes
Preserves FDs	Yes	Maybe	Maybe
Preserves MDs	Maybe	Maybe	Maybe

Normal Forms

- First Normal Form: each attribute is atomic
- Second Normal Form: No non-trivial FD has a left side that is a proper subset of a key
- Third Normal Form: just discussed it
- Fourth Normal Form: just discussed it
- Fifth Normal Form: outside the scope of CS4604
- Sixth Normal Form: different versions exist. One version developed for temporal databases
- Seventh Normal Form
  - just kidding ☺

Database Design Mantra

- "everything should depend on the key, the whole key, and nothing but the key"

- Announcement:
  - Make sure to bring a copy of Handout 3 to class on Monday
  - Go through the FD/MD lectures