### Relational Design Theory CSE462 Database Concepts

Demian Lessa/Jan Chomicki

Department of Computer Science and Engineering State University of New York, Buffalo

Fall 2013

### Overview

How does one design a relational schema?

- From requirements, high-level notations, etc.
- Initial schemas tend to combine too much into one relation.
  - This creates redundancy in the data.
  - Redundancies can be eliminated systematically.
- Relational theory.
  - Provides the foundation for designing good relational schemas.
  - Involves the study of dependencies invoving data elements.
  - Permits us to distinguish good from bad schemas.
  - Gives us the means to fix flawed designs.

### Overview

- Certain dependencies in relation schemas may cause problems.
  - Often referred to as anomalies.
- Data dependencies.
  - Functional dependencies generalize the notion of key.
  - Multivalued dependencies identify sets of independent attributes.
- Dependencies are used to define normal forms for relation schemas.
- Normal forms are used in a process called normalization.
  - Decompose relations into two or more relations to remove anomalies.
  - Each decomposition targets a particular normal form.

### **Functional Dependencies**

A functional dependency (FD) on a relation R is a statement of the form:

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

where  $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$  are attributes of R. We say that  $A_1, A_2, \ldots, A_n$  functionally determines  $B_1, B_2, \ldots, B_m$ .

- Let t, u be tuples in some R instance. If t and u agree on their  $A_1, A_2, \ldots, A_n$  values, they must agree on their  $B_1, B_2, \ldots, B_m$  values.
- If some FD *F* is true for every instance of *R*, then we say *R* satisfies *F*. You can think of *F* as a constraint on *R*.

## Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers
Table : Movies1(ti	tle, ye	ar, leng	gth, genre	e, studioName	e, starName).
Intuitively, what s	eems to	be wron	<b>a with</b> Mov	ies1 <b>?</b>	

- What does the FD below mean? Does it hold? title year  $\rightarrow$  length genre studioName
- How about this other FD?

title year  $\rightarrow$  starName

Example: Functional Dependencies
Observation
Even if we had one tuple for each of the movies, the FD
title year $\rightarrow$ starName
would still be false: by definition, a FD is a property of all possible instances of the relation, not just this particular one.

\_\_\_\_



## Example: Functional Dependencies

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Mayna's Morld	1992	95	Comedy	Paramount	Dana Carvey
waynes wonu	1002	00	Comedy	raramount	Dana Garvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers
Wayne's World Table : Movies1 (tit	<b>1992</b>	95 ar, leng	Comedy Comedy gth, genre	Paramount , studioName	Mike Meyers

- is {title, year} a key? assume t<sub>1</sub> ≠ t<sub>2</sub> agree on {title, year}; they must agree on {length, genre, studioName}; but then, t<sub>1</sub> and t<sub>2</sub> must have distinct starName components.
- {year, starName}? how about {title, starName}?
- The key is {title, year, starName}.

## Required

- Read section 3.1 from chapter #3.
- Go over problems 3.1.1 and 3.1.3.

### Example: Reasoning about FDs

Assume that relation R(A, B, C) satisfies FDs A  $\rightarrow$  B and B  $\rightarrow$  C. Does it satisfy A  $\rightarrow$  C? Show that if two tuples agree on A and B, they must agree on C. If the time transformed by the time trans

## Reasoning About FDs

FDs may be presented in different yet equivalent ways.

- Two sets of FDs S and T are equivalent if the sets of relation instances satisfying S is exactly the same as those satisfying T.
- A set of FDs S follows from a set of FDs T if every relation instance that satisfies all FDs in T also satisfies all FDs in S.
- S and T are equivalent if and only if S and T follow from each other.

## Splitting Rule

### Splitting Rule

We can replace an FD

$$A_1 A_2 \cdots A_n o B_1 B_2 \cdots B_m$$

by the set of FDs

$$A_1 A_2 \cdots A_n \rightarrow B_1$$
  
 $\cdots$   
 $A_1 A_2 \cdots A_n \rightarrow B_m$ 

We may replace a FD with a set of FDs having the same lhs and only one of the attributes on the rhs.

## Combining Rule

### Combining Rule

We can replace a set of FDs

$$A_1 A_2 \cdots A_n \rightarrow B_1$$

$$A_1 A_2 \cdots A_n \rightarrow B_m$$

. . .

by the single FD

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

We may replace a set of FDs having a common lhs by one FD with the same lhs and all rhs attributes combined into one set.

## Splitting and Combining Rules

In both rules, the input and output FD sets are equivalent. But why do these rules hold?

- Assume two tuples agree in  $A_1, A_2, \ldots, A_n$ .
- Single FD: "then, the tuples agree in all  $B_1, B_2, \ldots, B_m$ ."
- Set of FDs: "then, the tuples agree in  $B_1$  and they agree in  $B_2, \ldots$ , and they agree in  $B_m$ ."
- These conclusions are exactly the same thing!

Why do you suppose splitting cannot be applied to the left side?

## Exmaple: Splitting and Combining Rules

#### The FD:

```
title year \rightarrow length genre studioName
```

#### is equivalent to the set of FDs:

```
title year \rightarrow length
title year \rightarrow genre
title year \rightarrow studioName
```

## Exmaple: Splitting and Combining Rules

#### The FD:

title year  $\rightarrow \, \text{length}$ 

### is not equivalent to the set of FDs:

 $\begin{array}{c} \text{title} \rightarrow \text{length} \\ \text{year} \rightarrow \text{length} \end{array}$ 

### Trivial FDs

#### Trivial FDs

A constraint of any kind on a relation is said to be trivial if it holds for every instance of the relation. An FD

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

is trivial if

$$\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$$

That is, the attributes on the rhs form a subset of the ones on the lhs. Every trivial FD holds in every relation. Thus, trivial FDs may be assumed without further justification.

### Trivial-Dependency Rule

#### Trivial-Dependency Rule

If some (but not all) of the attributes on the rhs of an FD also appear on the lhs, those that appear on both sides may be dropped from the rhs. That is,

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$

is equivalent to

$$A_1 A_2 \cdots A_n \rightarrow C_1 C_2 \dots C_k$$

where  $\{C_1, C_2, ..., C_k\} = \{B_1, B_2, ..., B_m\} \setminus \{A_1, A_2, ..., A_n\}.$ 

It should be clear that we can prove this rule by first splitting the FD and then combining the resulting non-trivial FDs.

## Exmaple: Trivial FDs

The FDs below are trivial:

title year  $\rightarrow$  title title year  $\rightarrow$  year

However, the FD below is not trivial:

title year  $\rightarrow$  title length

but may be simplified using the trivial-dependency rule:

title year  $\rightarrow$  length

### Closure of Attributes

### Set Closure (Mathematics)

In set theory, we say that a set is closed under some operation if performing that operation on members of the set always produces a member of the set.

### **Closure of Attributes**

#### Definition: Closure of Attributes

Given a set of attributes  $\{A_1, \ldots, A_n\}$  and a set of FDs S, the closure of  $\{A_1, \ldots, A_n\}$  under the FDs in S is the (maximal) set of attributes **B** such that every relation that satisfies all the FDs in S also satisfies  $\{A_1, \ldots, A_n\} \rightarrow \mathbf{B}$ .

The closure of  $\{A_1, \ldots, A_n\}$  is denoted  $\{A_1, \ldots, A_n\}^+$ . Note that every  $A_i \in \{A_1, \ldots, A_n\}$  is trivially a member of  $\{A_1, \ldots, A_n\}^+$ .

Applications:

- $A_1 \cdots A_n \rightarrow \mathbf{B}$  if and only if  $\mathbf{B} \subseteq \{A_1, \dots, A_n\}^+$ .
- {A<sub>1</sub>,...,A<sub>n</sub>} is a superkey for a relation if and only if {A<sub>1</sub>,...,A<sub>n</sub>}<sup>+</sup> is the set of all relation attributes. (Why not a key?)

### **Closure of Attributes**

#### Algorithm

**INPUT:** Set of attributes  $\{A_1, \ldots, A_n\}$  and set of FDs S.

**OUTPUT:** The closure  $\{A_1, \ldots, A_n\}^+$ .

- Split the FDs of S so that each FD has a single attribute on their rhs.
- 2 Let X be the output set and initialize it to  $\{A_1, \ldots, A_n\}$ .
- Search for some FD  $B_1 \cdots B_m \rightarrow C$  where each  $B_i \in X$  but  $C \notin X$ . Add *C* to *X* and repeat until no FD is found.
- Output *X*.

Note: X grows every time a new FD is found in step (3). Eventually, step (3) does not find a new FD so the algorithm must terminate and output X.

## Required

- Go over examples 3.8 and 3.9.
- Go over the soundness and correctness proofs for the closure algorithm in 3.2.5.

### Transitivity Rule

#### **Transitivity Rule**

If  $A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m$  and  $B_1 B_2 \cdots B_m \to C_1 C_2 \cdots C_k$  hold in relation R, then  $A_1 A_2 \cdots A_n \to C_1 C_2 \cdots C_k$  also holds in R.

Sketch.

- Compute  $\{A_1, A_2, \dots, A_n\}^+$  with respect to the given FDs.
- $A_1A_2 \cdots A_n \rightarrow B_1B_2 \cdots B_m$  implies  $\{B_1, B_2, \dots, B_m\} \subseteq \{A_1, A_2, \dots, A_n\}^+$ .
- Thus, we can use  $B_1 B_2 \cdots B_m \rightarrow C_1 C_2 \cdots C_k$ .
- $B_1B_2\cdots B_m \to C_1C_2\cdots C_k$  implies  $\{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+$ .
- It follows that  $A_1 A_2 \cdots A_n \rightarrow C_1 C_2 \cdots C_k$ .

# Example: Transitivity Rule

title	year	length	genre	studioName	studioAddr			
Star Wars	1977	124	SciFi	Fox	Hollywood			
Eight Below	2005	120	Drama	DIsney	Buena Vista			
Wayne's World	1992	95	Comedy	Paramount	Hollywood			
Table:Movies2(tit	tle, ye	ar, leng	gth, genre	e, studioName	e, studioAddr).			
Assume that the	e followii	ng FDs h	old:					
title year —	→ genre	e lengtl	h studioN	Iame				
studioName —	> stud:	ioAddr						
By transitivity, the second	By transitivity, the following FD is derived:							
title year —	> stud:	ioAddr						

OC		COOL
	••••	

• Go over example 3.11.

### Armstrong's Axioms

#### Armstrong's Axioms.

- 1. Reflexivity
  - If  $\{B_1, B_2, \ldots, B_m\} \subseteq \{A_1, A_2, \ldots, A_n\}$  then  $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$ .
- 2. Augmentation
  - If  $A_1A_2 \cdots A_n \rightarrow B_1B_2 \cdots B_m$  then  $A_1A_2 \cdots A_nC_1 \cdots C_k \rightarrow B_1B_2 \cdots B_mC_1 \cdots C_k$ for any set of attributes  $\{C_1, \ldots, C_k\}$ . Some of the *C*'s may appear among the *A*'s or *B*'s, so we eliminate duplicate attributes from both the lhs and rhs.
- 3. Transitivity
  - If  $A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$  and  $B_1 B_2 \cdots B_m \rightarrow C_1 \cdots C_k$  then  $A_1 A_2 \cdots A_n \rightarrow C_1 \cdots C_k$ .
- Observations.
  - Sound and complete set of rules for deriving FDs.
  - Thus, derives any FD that follows from a given set of FDs.
  - But not systematic as the closure computation we just saw.

## Projecting FDs

Let R be a relation with FDs S and  $R_1 = \pi_L(R)$  for some attributes L from R.

- What FDs hold in R<sub>1</sub>?
- Compute the projection of functional dependencies  $\mathcal{S}$ .
- This is the set of all FDs that
  - Follow from  $\mathcal{S}$ , and
  - Involve only attributes of R<sub>1</sub>.
- In general, this computation is exponential in the size of L.

### **Projecting FDs**

#### Algorithm

**INPUT:** Relations R and  $R_1 = \pi_L(R)$ , and a set of FDs S that hold in R.

**OUTPUT:** Set of FDs that hold in  $R_1$ .

• Let *T* be the output and initialize it as the empty set.

- ② For each subset *X* of attributes of  $R_1$ , compute *X*<sup>+</sup> w.r.t. *S*. Add to *T* all nontrivial FDs X → A such that *A* is both in *X*<sup>+</sup> and an attribute of  $R_1$ .
- **(**) *T* is a basis for the FDs that hold in  $R_1$ . If *T* is not minimal:
  - If some FD *F* in *T* follows from the other FDs in *T*, remove *F* from *T*.
  - Let Y → B be an FD in T, with at least two attributes in Y, and Z a subset of Y with one attribute removed. If Z → B follows from the FDs in T (including Y → B), replace Y → B by Z → B.
  - Repeat these steps in all possible ways until *T* can no longer be changed.
- Output *T*.

## Required

- Go over example 3.13.
- Go over problems 3.2.2, 3.2.4, 3.2.5, 3.2.8.
- Read section 3.2 from chapter #3 and go over all remaining problems.

## Example: Schema Design

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers

Table: Movies1(title, year, length, genre, studioName, starName).

#### Problems:

- Redundancy.
  - Information repeated unnecessarily in several tuples.
- Update Anomalies.
  - Update data in one tuple and leave *the same data* unchanged in another.
- Deletion Anomalies.
  - If a set of values becomes empty, we may lose other information as side-effect.

## **Relation Decomposition**

#### Definition: Relation Decomposition

Given a relation  $R(A_1, A_2, ..., A_n)$ , we may decompose R into two new relations  $S(B_1, B_2, ..., B_m)$  and  $T(C_1, C_2, ..., C_k)$  such that:

•  $\{A_1, A_2, \dots, A_n\} = \{B_1, B_2, \dots, B_m\} \cup \{C_1, C_2, \dots, C_k\}.$ 

• 
$$S = \pi_{B_1, B_2, \dots, B_m}(R).$$

•  $T = \pi_{C_1, C_2, ..., C_k}(R).$ 

Observations:

- $\{B_1, B_2, \dots, B_m\}$  and  $\{C_1, C_2, \dots, C_k\}$  need not be disjoint.
- Decomposition is defined independently of its effectiveness in eliminating anomalies.

# Example: Schema Design

title	year	length	genre	studioName
Star Wars	Star Wars 1977		SciFi	Fox
Gone with the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount
Table : Movies2(titl	e, year	, lengt	h, genre,	studioName)
		I		
title		year	starName	
Star Wars	Star Wars		Carrie Fisl	her
Star Wars	Star Wars Star Wars		Mark Hamill	
Star Wars			Harrison Ford	
Gone with t	the Wind	1939	Vivien Leig	gh
Wayne's W	orld	1992	Dana Carvey	
Wayne's W	Wayne's World		Mike Meye	ers
Table : Mov: o any anomalies remain in	ies3(ti <sup>.</sup> <b>this nev</b>	tle, yea v design'	ar, starNa ?	ume).

Boyce-Codd Normal Form
Definition: Boyce-Codd Normal Form (BCNF)
<ul> <li>Consider a relation R and two sets of attributes A and B. Relation R is in Boyce-Codd Normal Form (BCNF) if and only if:</li> <li>Whenever there exists a non-trivial functional dependency A → B in R, A is a superkey of R.</li> <li>If a schema conforms to BCNF, the anomalies discussed are guaranteed not to exist</li> </ul>

## Example: Boyce-Codd Normal Form

title	year	length	genre	studioName	starName
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
Star Wars	1977	124	SciFi	Fox	Mark Hamill
Star Wars	1977	124	SciFi	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers
	1	1	1	1	1

Table: Movies1(title, year, length, genre, studioName, starName).

Relation Movies1 is not in BCNF.

- The only key of Movies1 is {title, year, starName}.
- $\bullet$  Consider the FD title year  $\rightarrow$  length genre studioName.
- The lhs of the FD above is not a superkey of Movies1.

# Example: Boyce-Codd Normal Form

title	year	length	genre	studioName	
Star Wars	1977	124	SciFi	Fox	
Gone with the Wind	1939	231	Drama	MGM	
Wayne's World	1992	95	Comedy	Paramount	
Table:Movies2(titl	.e, year	r, lengt	h, genre,	studioName).	
<b>Belation</b> Movies2 is in BCN	F				
	in (Lit	. ]	)		
Ine only key of Movies2	IS {tit	cie, yea	ar}.		
All other non-trivial FDs I	have ti	tle <b>and</b> :	year <b>on th</b>	eir lhs.	
• Hence, their lhs's are su	Iperkeys	s and Mox	vies2 <b>is in</b>	BCNF.	
## Boyce-Codd Normal Form

Decomposition of a relation  $\ensuremath{\mathbb{R}}$  into BCNF (Sketch).

- Chose subsets of the attributes in R such that:
  - Each subset becomes the schema of decomposed relation in BCNF.
  - Data in R is represented faithfully by the decomposed relations.
  - That is, we are able to recompose the original relation exactly.
- Intuition: use non-trivial FDs violating BCNF to drive the decomposition.

# Example: Boyce-Codd Normal Form

title	year	length	genre	studioName	starName	
Star Wars	1977	124	SciFi	Fox	Carrie Fisher	
Star Wars	1977	124	SciFi	Fox	Mark Hamill	
Star Wars	1977	124	SciFi	Fox	Harrison Ford	
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh	
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey	
Wayne's World	1992	95	Comedy	Paramount	Mike Meyers	
Table: Movies1(title, year, length, genre, studioName, starName).						
<b>Decompose using the FD</b> title year $\rightarrow$ length genre studioName.						
• Using both the and the of the FD:						

- Using both the and the of the FD: Movies2(title, year, length, genre, studioName).
- The lhs of the FD and all remaining attributes: Movies3(title, year, starName).
- This is the decomposition we have seen before!

### Boyce-Codd Normal Form

#### Algorithm

**INPUT:** Relation  $R_0$  with a set of FDs  $S_0$ .

**OUTPUT:** Decomposition of R<sub>0</sub> into a collection of relations in BCNF.

**PROCEDURE:** Run the steps below recursively for any relation R and set of FDs S that must be decomposed into BCNF. Start with  $R = R_0$  and  $S = S_0$ .

- If R already is in BCNF, return  $\{R\}$ .
- 2 Let X → Y be a FD violating BCNF. Choose R<sub>1</sub> = X<sup>+</sup> as one schema and let R<sub>2</sub> have attributes X and those of R not in X<sup>+</sup>.
- O Compute the FDs  $S_1$  and  $S_2$  for  $R_1$  and  $R_2$ , respectively.
- **O** Recursively decompose R<sub>1</sub> and R<sub>2</sub>.
- Seturn the union of all decompositions.

# Required

- Go over example 3.19.
- Go over problems 3.3.1, 3.3.2, 3.3.3, 3.3.4.
- Read section 3.3 from chapter #3 and go over the remaining problems.

### Decomposition

#### **Decomposition: Desirable Properties**

When decomposing, we strive to eliminate all anomalies from a schema. However, decomposition may may introduce problems of its own. Thus, we aim for decompositions having the following properties:

- Elimination of Anomalies.
  - We already know an algorithm that accomplishes this (BCNF).
- Recoverability of Information.
  - Is it possible to recover the original relation from tuples in the decomposition?
  - Is there a systematic way do accomplish this?
- Preservation of Dependencies.
  - If we reconstruct the original relation by joining, will the result satisfy the original FDs from the ones projected onto the decomposed relations?

### **Decomposition: Desirable Properties**

#### Lossless Joins

A decomposition of a relation R is a lossless-join decomposition if we can join the decomposed relations to obtain the instance of R back.

**Intuition:** Consider the relation R(A, B, C) and a FD  $B \rightarrow C$  that is a BCNF violation. Decomposing based on this FD yields  $R_1(A, B)$  and  $R_2(B, C)$ . Now, let t = (a, b, c) be a tuple of R. Note that (a, b) is a (projected) tuple in  $R_1$  and (b, c) one in  $R_2$ . When we perform the natural join of  $R_1$  and  $R_2$ , since the tuples agree on their B components, we obtain the original tuple *t*.

The BCNF decomposition algorithm guarantees that we can recover the original relation by performing a natural join on the decomposed relations.

R	00		ro	$\mathbf{\alpha}$
	сu	UI		U

• Go over example 3.21.

Consider relation R(A, B, C) on which neither  $B \rightarrow A$  nor  $B \rightarrow C$  holds. Say we decompose R into  $R_1(A, B)$  and  $R_2(B, C)$ . Then,



### The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple *t* in  $\pi_{S_1}(\mathbb{R}) \bowtie \pi_{S_2}(\mathbb{R}) \bowtie \ldots \bowtie \pi_{S_k}(\mathbb{R})$ , the natural join of the decomposed relations, is also a tuple in the original relation  $\mathbb{R}$ . That is, the decomposition does not produce spurious tuples.

Observation #1. The join is associative and commutative– join order does not matter. The join above is the set of tuples *t* such that *t* projected onto the attributes of  $S_i$  is a tuple in  $\pi_{S_i}(\mathbb{R})$ , for each  $i \in [1, k]$ .

Observation #2. Any tuple  $t \in \mathbb{R}$  surely is in the join. Simply note that the projection of *t* onto each  $S_i$  must be in  $\pi_{S_i}(\mathbb{R})$  for every *i*.

Observation #3. If *t* is in the join, there must exist tuples  $t_1, \ldots, t_k$  in R such that *t* is obtained by joining the projections of each  $t_i$  onto the attributes  $S_i$ , for  $i \in [1, k]$ . Further,  $t_i$  must agree with *t* on the attributes in  $S_i$ , but has unknown values otherwise.

### The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple *t* in  $\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_k}(R)$ , the natural join of the decomposed relations, is also a tuple in the original relation R.

Setup. Create a tableau with the same schema as R. Add one symbolic tuple for every decomposed relation  $\pi_{S_i}(R)$  as follows: use non-subscripted symbols for attributes in  $S_i$  and *i*-subscripted symbols for the unknowns.

### The Chase for Lossless Joins

The chase test for a lossless join is a systematic way to check whether a tuple *t* in  $\pi_{S_1}(\mathbb{R}) \bowtie \pi_{S_2}(\mathbb{R}) \bowtie \ldots \bowtie \pi_{S_k}(\mathbb{R})$ , the natural join of the decomposed relations, is also a tuple in the original relation  $\mathbb{R}$ .

Chasing. Apply the FDs of R to equate symbols in the tableau until some tuple t' contains only non-subscripted symbols (i.e., t' is equal tuple t above) or no FDs can be further applied. When equating symbols, change all occurrences of one with the other.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Setup the tableau for this decomposition and apply the chase to verify if this decomposition has the lossless join property.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Α	В	С	D
а	b <sub>1</sub>	<i>C</i> <sub>1</sub>	d
а	b <sub>2</sub>	С	$d_2$
$a_3$	b	С	d

Step #1 : Tableau setup based on the decomposition of R.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Α	В	С	D
а	<i>b</i> <sub>1</sub>	<i>C</i> <sub>1</sub>	d
а	<i>b</i> 1	С	<i>d</i> <sub>2</sub>
$a_3$	b	С	d

Step #2 : Apply FD  $A \rightarrow B$ .

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Α	В	С	D
а	<i>b</i> <sub>1</sub>	С	d
а	$b_1$	с	$d_2$
$a_3$	b	С	d

Step #3 : Apply FD  $B \rightarrow C$ .

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Α	В	С	D
а	b <sub>1</sub>	С	d
а	$b_1$	С	$d_2$
а	b	С	d

Step #4 : Apply FD  $CD \rightarrow A$ .

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, CD \rightarrow A\}$ . Say we decompose R into relations with sets of attributes  $S_1 = \{A, D\}$ ,  $S_2 = \{A, C\}$ , and  $S_3 = \{B, C, D\}$ .

Α	В	С	D
а	b <sub>1</sub>	С	d
а	$b_1$	С	d <sub>2</sub>
а	b	С	d

Conclusion : Last tuple is evidence of lossless join decomposition.

## Example: Dependency Preservation

```
Consider relation Bookings(title,theater,city) with FDs {theater \rightarrow city,title city \rightarrow theater}. Decompose into BCNF.
```

- Keys: {title,city} and {theater,title}.
- BCNF Violation: theater  $\rightarrow$  city.
- Decomposition:  $S_1 = \{\text{theater}, \text{city}\}, S_2 = \{\text{theater}, \text{title}\}.$
- Problem: title city  $\rightarrow$  theater.

theater	city	theater	title		theater	city	title	
Guild	NYC	Guild	Antz		Guild	NYC	Antz	-
Park	NYC	Park	Antz		Park	NYC	Antz	
$\pi_{ ext{S}_1}( ext{Book})$	kings)	$\pi_{ ext{S}_2}$ (Book	kings)	$\pi_{ ext{S}_1}$	Booking	s) $\bowtie \pi_{S_2}$	Booking	gs)

- $\pi_{S_1}$  (Bookings) and  $\pi_{S_2}$  (Bookings) are valid instances.
- But  $\pi_{S_1}(\text{Bookings}) \bowtie \pi_{S_2}(\text{Bookings})$  violates title city  $\rightarrow$  theater.

# Dependency Preservation

Unfortunately, it is not always possible to decompose into BCNF and preserve functional dependencies.

# Required

- Go over example 3.25.
- Go over problems 3.4.1, 3.4.2.
- Read section 3.4 from chapter #3 and go over the remaining problems.

## Third Normal Form

#### Definition: Third Normal Form (BCNF)

Consider a relation R and two sets of attributes A and B. Relation R is in Third Normal Form (3NF) if and only if:

- Whenever there exists a non-trivial functional dependency A → B in R, A is a superkey of R, OR
- Every attribute in  $B \setminus A$  is a member of some key (need not be the same).

An attribute that is a member of some key is called prime. Thus, 3NF states that the lhs of every non-trivial FD is a superkey, or all attributes on the rhs are prime.

### Closure of FDs

#### **Definition: Minimal Basis**

Given a set of FDs S, any set of FDs equivalent to S is a basis for S. A minimal basis for a relation is a basis  $\mathcal{B}$  that satisfies:

- Every FD in  $\mathcal{B}$  has a single attribute on the rhs.
- **2** If any FD is removed from  $\mathcal{B}$ , it is no longer a basis.
- If for any FD in  $\mathcal{B}$  we remove one or more attributes from the lhs of the FD, the result is no longer a basis.

Note: a minimal basis has no trivial FD, as they are removed in step (2).

## Third Normal Form

#### Algorithm: 3NF Synthesis

**INPUT:** Relation R with a set of FDs  $\mathcal{F}$ .

**OUTPUT:** Decomposition of R into a collection of relations in 3NF.

- Find a minimal basis for  $\mathcal{F}$  (call it  $\mathcal{G}$ ).
- ② For each FD X → A in G, use X∪A as attributes of one of the relations in the decomposition.
- If none of the relations obtained in (2) contain a superkey of R, add another relation whose schema is a key of R.

The 3NF synthesis algorithm gurantees that: a) all decomposed relations are in 3NF, b) the decomposition has a lossless join, and c) the decomposition is dependency-preserving.

# Required

- Go over example 3.27.
- Go over problems 3.5.1, 3.5.4.
- Read section 3.5 from chapter #3 and go over the remaining problems.

# Attribute Independence

In certain cases, a BCNF schema may still show some redundancy. A common source is having two or more set-valued properties of the key into a single relation.

Example: Attr	Example: Attribute Independence						
name	street	city	title	year			
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977			
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977			
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980			
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980			
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983			
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983			
Table : Sets of addresses independent from movies.   Contract in movies.							

- No reason to associate one address with one movie and not the other.
- This relation has no non-trivial FD, therefore, it is in BCNF!

### Multivalued Dependency

#### **Definition: Multivalued Dependency**

A multivalued dependency (MVD) on a relation R is a statement of the form:

$$A_1 A_2 \ldots A_n \twoheadrightarrow B_1 B_2 \ldots B_m$$

where  $A_1, A_2, ..., A_n, B_1, B_2, ..., B_m$  are attributes of R. This MVD holds if, for each pair of tuples *t* and *u* of R that agree on their  $A_1, A_2, ..., A_n$  values, there exists some tuple *v* in R that agrees:

• with the  $A_1, A_2, \ldots, A_n$  values of *t* and *u*,

2 with the  $B_1, B_2, \ldots, B_m$  values of t, and

(a) with u on the values of all attributes of R not among the A's or B's.

Observation: swapping t and u guarantees the existence of another tuple w. Thus, for any fixed values of the A's, the B's and the other attributes appear in all possible combinations in different tuples (hence their independence).

## Example: Multivalued Dependency

name	street	city	title	year
C. Fisher	123 Maple St.	Hollywood	Star Wars	1977
C. Fisher	5 Locust Ln.	Malibu	Star Wars	1977
C. Fisher	123 Maple St.	Hollywood	Empire Strikes Back	1980
C. Fisher	5 Locust Ln.	Malibu	Empire Strikes Back	1980
C. Fisher	123 Maple St.	Hollywood	Return of the Jedi	1983
C. Fisher	5 Locust Ln.	Malibu	Return of the Jedi	1983

	Table :	Sets	of	addresses	independer	nt from	movies.
--	---------	------	----	-----------	------------	---------	---------

- Tha is, each star name appears with their set of addresses.
- Let *t* and *u* be the first and second blue rows. Then, *v* is the red row.
- If we swap *t* and *u*, then *v* is the yellow row.

### Multivalued Dependency

Rules involving MVDs:

- Triviality. (analogous to FDs)
- Transitivity. (analogous to FDs)
- Promotion. Every FD is an MVD.
- Complementation. If A<sub>1</sub>A<sub>2</sub>...A<sub>n</sub> → B<sub>1</sub>B<sub>2</sub>...B<sub>m</sub> is an MVD for a relation R then R also satisfies A<sub>1</sub>A<sub>2</sub>...A<sub>n</sub> → C<sub>1</sub>C<sub>2</sub>...C<sub>k</sub>, where the Cs are the attributes of R not among the As and Bs (swapping Bs in a tuple that agrees on the As has the same effect as swapping the Cs).
- More Triviality. If  $\{A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_m\}$  are all the attributes of R, then  $A_1A_2 \dots A_n \twoheadrightarrow B_1B_2 \dots B_m$ .

Observation: the combining and splitting rules do not apply to MVDs.

## Fourth Normal Form

#### Definition: Fourth Normal Form

Consider a relation R and two sets of attributes A and B. Relation R is in Fourth Normal Form (4NF) if and only if:

 Whenever there exists a non-trivial multivalued dependency A → B in R, A is a superkey of R.

Observation: 4NF is a generalization of BCNF, thus, every relation in 4NF is also in BCNF and every BCNF violation is also a 4NF violation.

## Fourth Normal Form

#### Algorithm

**INPUT:** Relation  $R_0$  with a set of FDs+MVDs  $S_0$ .

**OUTPUT:** Decomposition of R<sub>0</sub> into a collection of relations in 4NF.

**PROCEDURE:** Run the steps below recursively for any relation R and set of FDs+MVDs S that must be decomposed into 4NF. Start with  $R = R_0$  and  $S = S_0$ .

- If  $\mathbb{R}$  already is in 4NF, return  $\{\mathbb{R}\}$ .
- 2 Let  $X \rightarrow Y$  be a 4NF (true MVD or FD) violation. Let  $R_1$  have schema  $X \cup Y$ and  $R_2$  schema  $X \cup (R - X - Y)$ .
- O Compute the FDs+MVDs  $S_1$  and  $S_2$  for  $R_1$  and  $R_2$ , respectively.
- **O** Recursively decompose R<sub>1</sub> and R<sub>2</sub>.
- Seturn the union of all decompositions.

# Required

- Go over example 3.34.
- Go over problems 3.6.2, 3.6.3.
- Read section 3.6 from chapter #3 and go over the remaining problems.

## The Chase: Checking if an FD holds

The Chase.

- First application: checking for lossless join decomposition.
- New application: checking if a FD holds in a relation.
- Consider a relation  ${\tt R}$  and a set of FDs  ${\mathcal F}$  that hold in  ${\tt R}.$
- To know whether the FD X  $\rightarrow$  Y follows from  $\mathcal{F}$ , proceed as follows:
  - Create a tableau with two rows agreeing on their X values only.
  - Chase the tableau using the FDs in  $\mathcal{F}$ .
  - If the final tableau agrees in all columns of  ${\tt Y}$  then  ${\tt X} \ \rightarrow \ {\tt Y}$  holds in  ${\tt R}.$

# Example: The Chase for FDs

Consider relation R(A, B, C, D, E, F) with FDs  $\mathcal{F} = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$  and say we wish to know whether  $\{AB \rightarrow D\}$  holds in R.

Α	В	С	D	Е	F
а	b	<i>C</i> <sub>1</sub>	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>1</sub>
а	b	<i>C</i> <sub>2</sub>	d <sub>2</sub>	e <sub>2</sub>	f <sub>2</sub>

Step #1 : Tableau setup based on the FD {AB  $\rightarrow$  D}.

# Example: The Chase for FDs

Consider relation R(A, B, C, D, E, F) with FDs  $\mathcal{F} = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$  and say we wish to know whether  $\{AB \rightarrow D\}$  holds in R.

Α	В	С	D	Е	F
а	b	<i>C</i> <sub>1</sub>	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>1</sub>
а	b	<i>C</i> <sub>1</sub>	$d_2$	e <sub>2</sub>	f <sub>2</sub>

Step #2 : Apply FD {AB  $\rightarrow$  C}.

# Example: The Chase for FDs

Consider relation R(A, B, C, D, E, F) with FDs  $\mathcal{F} = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$  and say we wish to know whether  $\{AB \rightarrow D\}$  holds in R.

Α	В	С	D	Е	F
а	b	<i>C</i> <sub>1</sub>	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>1</sub>
а	b	<i>C</i> 1	<i>d</i> <sub>1</sub>	e <sub>2</sub>	f <sub>2</sub>

Step #3 : Apply FD {BC  $\rightarrow$  AD}.
Consider relation R(A, B, C, D, E, F) with FDs  $\mathcal{F} = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$  and say we wish to know whether  $\{AB \rightarrow D\}$  holds in R.

Α	В	С	D	Е	F
а	b	<i>C</i> <sub>1</sub>	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>1</sub>
а	b	<i>C</i> 1	<i>d</i> <sub>1</sub>	<i>e</i> 1	f <sub>2</sub>

Step #4 : Apply FD  $\{D \rightarrow E\}$ .

Consider relation R(A, B, C, D, E, F) with FDs  $\mathcal{F} = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$  and say we wish to know whether  $\{AB \rightarrow D\}$  holds in R.

Α	В	С	D	Е	F
а	b	<i>C</i> <sub>1</sub>	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>1</sub>
а	b	<i>C</i> 1	<i>d</i> <sub>1</sub>	<i>e</i> 1	<i>f</i> <sub>2</sub>

Conclusion : The tuples agree on D thus  $\{AB \rightarrow D\}$  holds in R.

### The Chase: Checking if an MVD holds

Checking if a MVD holds in a relation.

- Consider a relation  ${\tt R}$  and a set of FDs+MVDs  ${\mathcal F}$  for  ${\tt R}.$
- To know whether the MVD X  $\rightarrow$  Y follows from  $\mathcal{F}$ , proceed as follows:
  - Create a tableau with two rows agreeing on the values of X only.
  - Chase the tableau using the FDs of  $\mathcal{F}$ .
  - Apply each MVD of *F* as follows: given a MVD A → B, if two tuples in the tableau agree on A, form two new tuples by swapping all their B values.
  - If we ever discover that one of the original tuples with its Y values replaced by those of the other original tuple is in the tableau, the MVD holds in R.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$  and say we wish to know whether  $\{A \rightarrow C\}$  holds in R.

Α	В	С	D
а	<i>b</i> 1	С	<i>d</i> <sub>1</sub>
а	b	<i>C</i> <sub>2</sub>	d

Step #1 : Tableau setup based on the MVD {A  $\twoheadrightarrow$  C}.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$  and say we wish to know whether  $\{A \rightarrow C\}$  holds in R.

Α	В	С	D
а	b	С	$d_1$
а	b	<i>C</i> 2	d

Step #2 : Apply FD  $A \rightarrow B$ .

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$  and say we wish to know whether  $\{A \rightarrow C\}$  holds in R.

Α	В	С	D
а	b	С	<i>d</i> <sub>1</sub>
а	b	<b>C</b> 2	d
а	b	<i>C</i> <sub>2</sub>	<i>d</i> <sub>1</sub>
а	b	С	d

Step #3 : Apply MVD B ----> C.

Consider relation R(A, B, C, D) with FDs  $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$  and say we wish to know whether  $\{A \rightarrow C\}$  holds in R.

Α	В	С	D
а	b	С	<i>d</i> <sub>1</sub>
а	b	<b>C</b> 2	d
а	b	<i>C</i> <sub>2</sub>	$d_1$
а	b	С	d

Conclusion : Last tuple is evidence that  $\{A \twoheadrightarrow C\}$  holds in R.

#### **Projecting MVDs**

Just like the projection of FDs is essential to the BCNF algorithm, the projection of MVDs is essential to the 4NF algorithm.

To verify whether an MVD holds in a projected relation (decomposition), we apply the chase on the full set of attributes of the original relation. The goal is to produce two rows in the tableau that satisfy the MVD condition for the decomposed relation.

As with FDs, there are shortcuts: do not check trivial FDs and MVDs; it suffices to look for FDs with a single attribute on the rhs; FDs or MVDs whose lhs does not contain the lhs of any MVD or FD can be ignored (the chase cannot start).

# Required

- Go over example 3.38 and problems 3.7.1, 3.7.4.
- Read section 3.7 from chapter #3.
- Go over the remaining problems of section 3.7.