Introduction to Database Systems

Functional Dependencies

Werner Nutt

Functional Dependencies

1. Meaning of FDs

1. **Meaning of FDs**
2. Keys and Superkeys
3. Inferring FDs

Functional Dependencies

- A FD is written

$$X \to A \quad \text{or} \quad X \to Y$$

- Notation: X, Y, Z represent sets of attributes; A, B, C,… represent single attributes

- $X \to A$ ("X determines A") is an assertion about a relation R: whenever two tuples of R agree on all the attributes of X, then they must also agree on the attribute A, or

  t1[X] = t2[X] implies t1[A] = t2[A]   for all t1, t2 in R

  (analogously for $X \to Y$)

- Convention: We say "$X \to A$ holds in R"

- Notation: No set braces in sets of attributes: just ABC, rather than $\{A,B,C\}$.

3

Example

Table   Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Reasonable FD's to assert:

1. name $\to$ addr
2. name $\to$ favBeer
3. beersLiked $\to$ manf

4

Example FDs

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Because of name → addr

Because of name → favBeer

Because of beersLiked → manf

FD's With Multiple Attributes

FDs with more than one attribute on the right don't increase expressivity …

… but allow for convenient shorthands that combine FDs

Example: name → addr and name → favBeer
become name → addr favBeer

More than one attribute on the left may be essential.

Example: bar beer → price

Functional Dependencies

2. Keys and Superkeys

1. Meaning of FDs
2. **Keys and Superkeys**
3. Inferring FDs

Keys of Relations

R relation, *K* a set of attributes of *R*

- *K* is a *superkey* for relation *R* if
  K functionally determines all of *R*

- *K* is a *key* for *R* if *K* is a superkey,
  but no proper subset of *K* is a superkey

  (that is, *K* is a *minimal superkey*)

*Sometimes we call "keys" also "candidate keys",
to indicate they are candidates for choosing the primary key*

Example

Drinkers(name, addr, beersLiked, manf, favBeer)

We have
    name $\rightarrow$ addr favBeer
    beersLiked $\rightarrow$ manf

Therefore, {name, beersLiked} determine all the other attributes

Hence, {name, beersLiked} is a superkey

Example (cntd.)

Neither {name} nor {beersLiked} is a superkey:
– name $\rightarrow$ manf doesn't hold
– beersLiked $\rightarrow$ addr doesn't hold

Hence: {name, beersLiked} is a key

There are no other keys, but lots of superkeys:
Any superset of {name, beersLiked} is a superkey

ER and Relational Keys

- Keys in ER concern entities
- Keys in relations concern tuples
- Usually, one tuple corresponds to one entity, so the ideas are similar
- But — in poor relational designs, one tuple may represent several entities, … so ER keys and relational keys are different

Example Data

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

The relational key is {name, beersLiked}

But in E/R,
- name is a key for Drinkers
- beersLiked is a key for Beers.

The relation contains
- 2 tuples for Janeway entity
- 2 tuples for Bud entity.

Where Do Keys Come From?

1. Just assert one key K
   - The only FDs are $K \rightarrow A$ for all attributes A

2. Assert FDs and
   deduce the keys by systematic exploration
   - ER model gives us FDs from entity-set keys and from many-one relationships
   - Other FDs we may know from our domain knowledge

     ("no two courses take place in a room at the same time")

13

Functional Dependencies

3. Inferring FDs

1. Meaning of FDs
2. Keys and Superkeys
3. **Inferring FDs**

14

Inferring FDs

We are given FD's

$$X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_n \rightarrow A_n,$$

and we want to know whether an FD

$$Y \rightarrow B$$

must hold in any relation that satisfies the given FDs

Example: If $A \rightarrow B$ and $B \rightarrow C$ hold,

then surely $A \rightarrow C$ holds

Important for design of good relation schemas

15

Inference Test

We are given a set of FDs and want to know whether

$$Y \rightarrow B$$

follows from the given FDs.

Test: We consider two tuples
and assume they agree in all attributes of Y:

Y

0000000. . . 0

00000?? . . .?

16

Inference Test (cntd.)

Apply the given FDs to infer that these tuples must also agree on certain other attributes

- If B is one of these attributes, then $Y \rightarrow B$ is true

- Otherwise, the two tuples, with any forced equalities, form a two-tuple relation
  - that satisfies the given FDs
  - but does not satisfy $Y \rightarrow B.$

  This would show that $Y \rightarrow B$ does not follow from the given FDs.

Inference Test: Example

Drinkers(name, addr, beersLiked, manf, favBeer)
with
1. name $\rightarrow$ addr
2. name $\rightarrow$ favBeer
3. beersLiked $\rightarrow$ manf

Which of the FDs below follow from the given FDs?
4. addr $\rightarrow$ favBeer
5. name beersLiked $\rightarrow$ favBeer

Closure Test

An easier test is based on the concept of attribute closure

- Let R be a relation,
  $\mathcal{F}$ a set of FDs over R,
  Y a set of attributes of R.

- The closure of Y with respect to $\mathcal{F}$, written Y^+,
  consists of all attributes that are determined by Y, given $\mathcal{F}$.

- Observation:
  $Y \rightarrow B$ follows from $\mathcal{F}$
  if and only if
  $B \in Y^+$

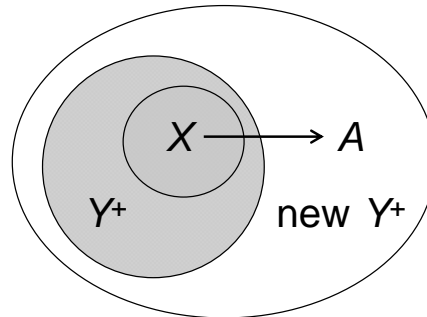# The Closure Algorithm

- Initialization: $Y^+ = Y$
- Loop:
  - Look for an FD $X \rightarrow A$ in $\mathcal{F}$ such that $X \subseteq Y^+$ and $A \notin Y^+$
  - Add A to Y^+
- Until: there is no applicable FD left in $\mathcal{F}$
- Output: Y^+

This is a typical example of a fixpoint algorithm

Closure Algorithm: The Idea

Example

Contracts(cno, supplno, projno, depno, partno, qty, value)

Short: *CSPrDPaQV*

A designer has found the following set of FDs:
- *C* is a key, i.e., $C \rightarrow SPrDPaQV$
- A project purchases each part using a single contract, $PrPa \rightarrow C$
- A department purchases at most one part from a supplier, $SD \rightarrow Pa$

His colleague has come up with a slightly different set:
- A project purchases each part using a single contract, $PrPa \rightarrow C$
- A contract determines project, supplier and department, $C \rightarrow PrSD$
- *SPrD* is a key, $SDPr \rightarrow CPaQV$

Are the findings of the second designer different from those of the first?

Finding All Implied FDs

We know how we can determine whether one FD follows
from a set of FDs $\mathcal{F} = \{\ X_1 \rightarrow A_1,\ldots, X_n \rightarrow A_n\ \}$

Question: How can we find all such FDs?

Motivation: To get a better schema, we "normalize", i.e.,
   we break one relation schema
   into two or more schemas.

Finding All Implied FD's: Example

Relation R with attributes *ABCD*, with FD's
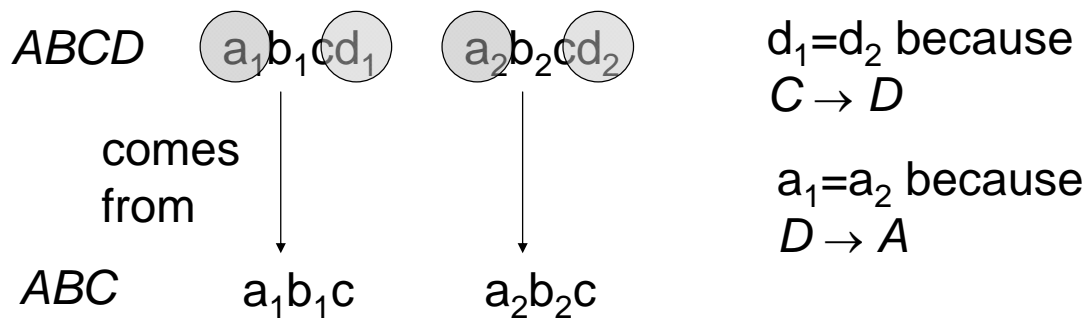$$AB \rightarrow C,\ \ C \rightarrow D,\ \ \ D \rightarrow A.$$

Decompose R into *ABC*, *AD*,
   (i.e., project onto *ABC* and *AD*)

Question: What FDs hold in *ABC* ?

Answer: Not only $AB \rightarrow C$, but also $C \rightarrow A$!

Why?

$ABCD$ $a_1b_1cd_1$ $a_2b_2cd_2$

comes
from

ABC a_1b_1c a_2b_2c

$d_1=d_2$ because
$C \rightarrow D$

$a_1=a_2$ because
$D \rightarrow A$

Thus, tuples in the projection
with equal Cs have equal As:
$C \rightarrow A$.

Projecting FDs

How can we find the FDs that hold on the projection of R?

Basic Idea:

1. Start with the given FDs

2. Find all *nontrivial* FDs that follow from the given FDs
   (nontrivial = left and right sides disjoint)

3. Restrict to those FDs that involve only attributes
   of the projected schema

A Simple, Exponential Algorithm

1. For each set of attributes X, compute X^+

2. Add $X \rightarrow A$ for all $A \in X^+ - X$

3. However, drop $XY \rightarrow A$ whenever we discover $X \rightarrow A$

   (Because $XY \rightarrow A$ follows from $X \rightarrow A$ in any projection)

4. Finally, return only FDs involving projected attributes

Optimizations

Suppose that Z is the set of all attributes of R. Then:

- $\emptyset^+ = \emptyset$

- $Z^+ = Z$

- If $X \subseteq Y$ and $X^+ = Z$ then $Y^+ = Z$

This can be used for optimizations!

Example

Relation ABC with FDs $A \rightarrow B$ and $B \rightarrow C$

Project onto AC

- $A^+ = ABC \implies A \rightarrow B, A \rightarrow C$
  (Optimization: We do not need to compute AB^+ or AC^+)
- $B^+ = BC \implies B \rightarrow C$
- $C^+ = C \implies$ nothing
- $BC^+ = BC \implies$ nothing

Resulting FDs: $A \rightarrow B$, $A \rightarrow C$, and $B \rightarrow C$.

Projection onto AC: $A \rightarrow C$.
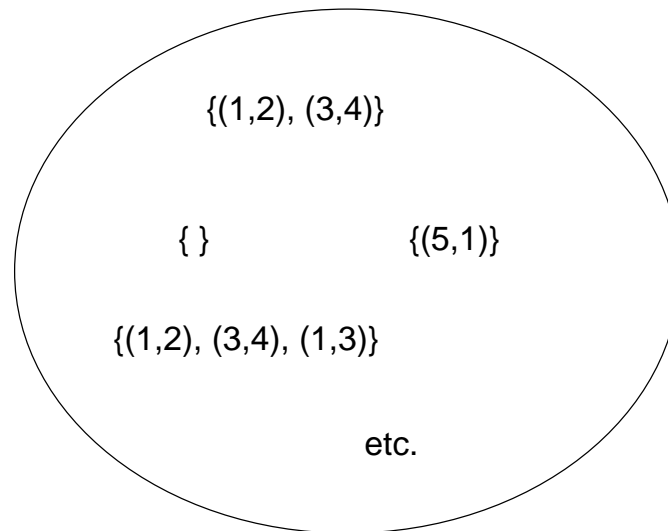 (The only FD that involves a subset of { A,C })

A Geometric View of FDs

- We consider the set of all *instances*
  of a particular relation R
- That is,
  - all finite sets of tuples
  - that have the proper number of components.
- Each instance is a point in this space

Example: R(A,B)

{(1,2), (3,4)}

{ }   {(5,1)}

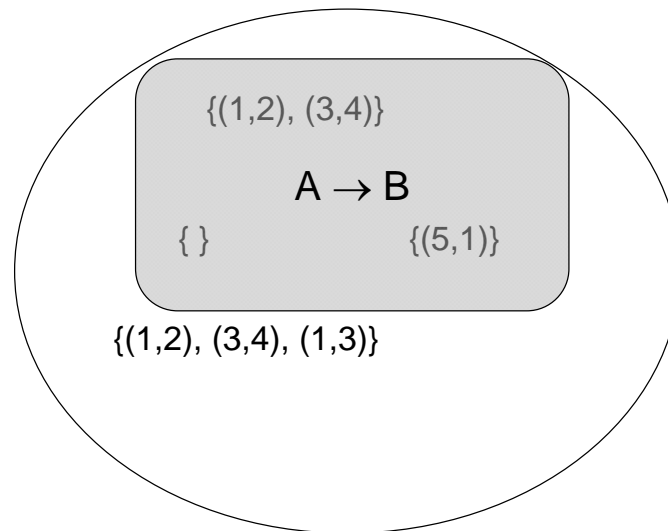{(1,2), (3,4), (1,3)}

etc.

An FD is a Subset of Instances

- For each FD $X \rightarrow A$, there is the subset of all instances that satisfy the FD

- Thus, we can identify an FD
  with a region in the space

- An FD is trivial
  if and only if
  it is represented by the entire space

Example: $A \rightarrow A$.

Example: A → B for R(A,B)

$$\{(1,2), (3,4)\}$$

$$A \rightarrow B$$
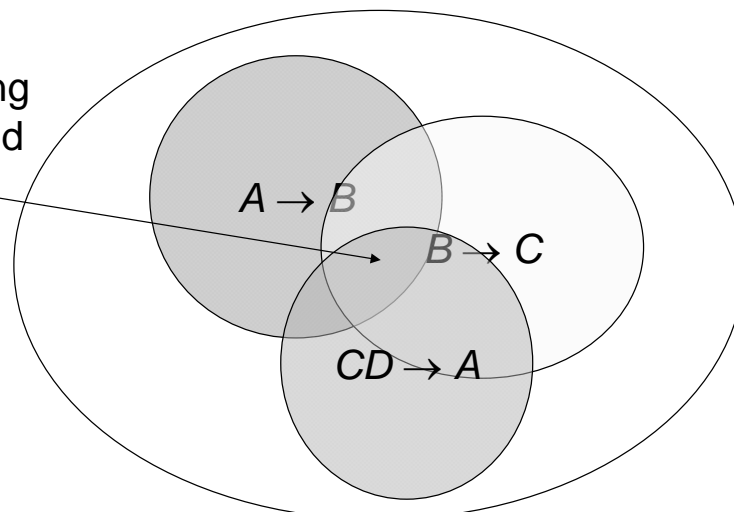
$$\{ \} \qquad \{(5,1)\}$$

$$\{(1,2), (3,4), (1,3)\}$$

Representing Sets of FDs

If each FD is a set of relation instances, then a collection of
FDs corresponds to the intersection of those sets

(Intersection = all instances that satisfy all of the FDs)

Instances satisfying
$A \rightarrow B$, $B \rightarrow C$, and
$CD \rightarrow A$

$A \rightarrow B$

$B \rightarrow C$

$CD \rightarrow A$

Entailment of FDs

$\mathcal{F} = \{ X_1 \rightarrow A_1,\ldots,X_n \rightarrow A_n \}$ set of FDs

- An FD $Y \rightarrow B$ follows from $\mathcal{F}$ or is entailed by $\mathcal{F}$ if
  every instance that satisfies all FDs in $\mathcal{F}$ also satisfies $Y \rightarrow B$
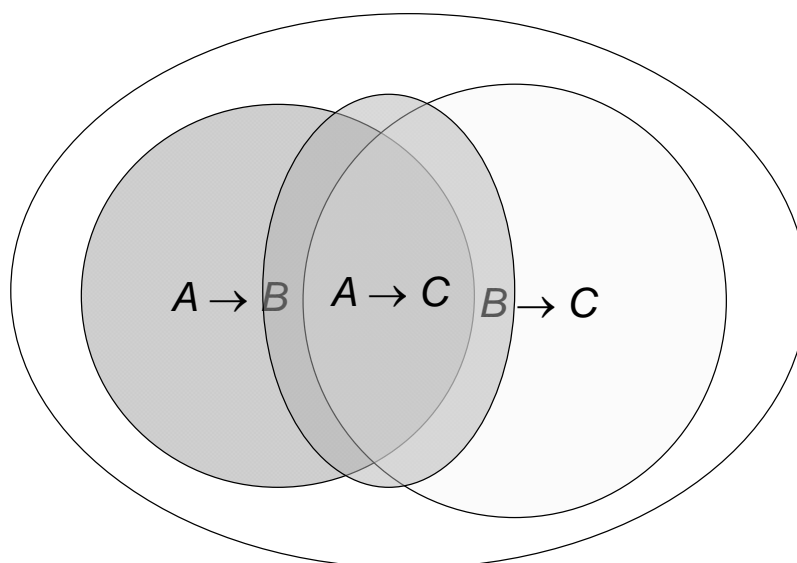
This can be visualized:

- If $Y \rightarrow B$ follows from the set $\mathcal{F} = \{ X_1 \rightarrow A_1,\ldots,X_n \rightarrow A_n \}$,
  then in the space of instances the *region for $Y \rightarrow B$* must include the
  intersection of the regions for the FDs $X_i \rightarrow A_i$.

  That is:
  - Every instance satisfying all the $X_i \rightarrow A_i$ surely satisfies $Y \rightarrow B$.
  - But an instance could satisfy $Y \rightarrow B$, yet not be in this intersection.

35

Example

$A \rightarrow B$ $A \rightarrow C$ $B \rightarrow C$

36

References

In preparing the lectures I have used several sources.
The main ones are the following:

Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe

Slides:

- The slides of this chapter are based on slides prepared by Jeff Ullman for his introductory course on database systems at Stanford University

37