Functional Dependencies

Meaning of FD's

Keys and Superkeys

Inferring FD's

Functional Dependencies

- $X \text{->} A$ is an assertion about a relation R that whenever two tuples of R agree on all the attributes of X, then they must also agree on the attribute A.
  - Say "$X \text{->} A$ holds in R."
  - Convention: ..., X, Y, Z represent sets of attributes; A, B, C,... represent single attributes.
  - Convention: no set formers in sets of attributes, just ABC, rather than $\{A,B,C\}$.

Example

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

- Reasonable FD's to assert:
  1. name -> addr
  2. name -> favBeer
  3. beersLiked -> manf

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

Because name -> addr

Because name -> favBeer

Because beersLiked -> manf

FD's With Multiple Attributes

- No need for FD's with > 1 attribute on right.
  - But sometimes convenient to combine FD's as a shorthand.
  - Example: name -> addr and name -> favBeer become name -> addr favBeer
- \> 1 attribute on left may be essential.
  - Example: bar beer -> price

Keys of Relations

- *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.

Example

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

- {name, beersLiked} is a superkey because together these attributes determine all the other attributes.
  - name -> addr favBeer
  - beersLiked -> manf

Example, Cont.

- {name, beersLiked} is a key because neither {name} nor {beersLiked} is a superkey.
  - name doesn't -> manf; beersLiked doesn't -> addr.
- There are no other keys, but lots of superkeys.
  - Any superset of {name, beersLiked}.

E/R and Relational Keys

- Keys in E/R concern entities.
- Keys in relations concern tuples.
- Usually, one tuple corresponds to one entity, so the ideas are the same.
- But --- in poor relational designs, one entity can become several tuples, so E/R 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

Relational key = {name beersLiked}

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

Note: 2 tuples for Janeway entity and 2 tuples for Bud entity.

Where Do Keys Come From?

1. Just assert a key *K*.
   - The only FD's are K -> A for all attributes *A*.
2. Assert FD's and deduce the keys by systematic exploration.
   - E/R model gives us FD's from entity-set keys and from many-one relationships.

More FD's From "Physics"

- Example: "no two courses can meet in the same room at the same time" tells us: hour room -> course.

Inferring FD's

- We are given FD's X_1 -> A_1, X_2 -> A_2,…, X_n -> A_n , and we want to know whether an FD Y -> B must hold in any relation that satisfies the given FD's.
  - Example: If A -> B and B -> C hold, surely A -> C holds, even if we don't say so.
- Important for design of good relation schemas.

Inference Test

- To test if Y -> B, start by assuming two tuples agree in all attributes of Y.

$\leftarrow Y \rightarrow$

0000000. . . 0

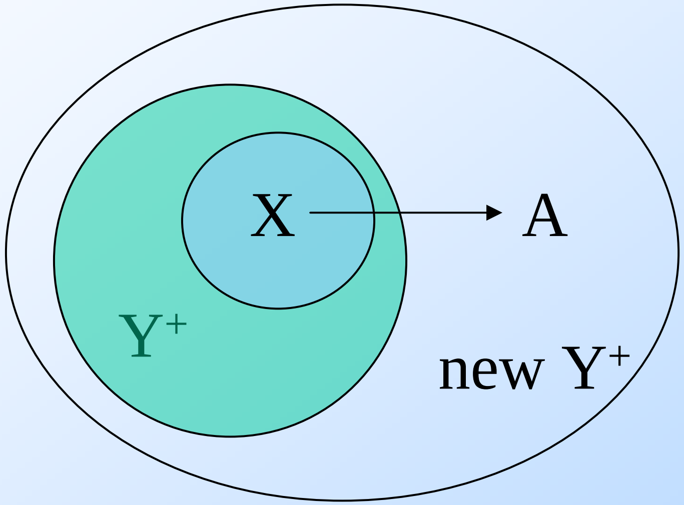
00000?? . . . ?

Inference Test – (2)

- Use the given FD's to infer that these tuples must also agree in certain other attributes.
  - If B is one of these attributes, then *Y* -> *B* is true.
  - Otherwise, the two tuples, with any forced equalities, form a two-tuple relation that proves *Y* -> *B* does not follow from the given FD's.

Closure Test

- An easier way to test is to compute the *closure* of Y, denoted Y^+.
- Basis: $Y^+ = Y$.
- Induction: Look for an FD's left side X that is a subset of the current Y^+. If the FD is $X \rightarrow A$, add A to Y^+.


X
A
Y+
new Y+

Finding All Implied FD's

- Motivation: "normalization," the process where we break a relation schema into two or more schemas.
- Example: *ABCD* with FD's $AB \to C$, $C \to D$, and $D \to A$.
  - Decompose into *ABC*, *AD*. What FD's hold in *ABC* ?
  - Not only $AB \to C$, but also $C \to A$!

Why?

ABCD $a_1b_1cd_1$ $a_2b_2cd_2$

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

comes from

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

ABC a_1b_1c a_2b_2c

Thus, tuples in the projection with equal C's have equal A's; C -> A.

Basic Idea

1. Start with given FD's and find all *nontrivial* FD's that follow from the given FD's.
   - Nontrivial = left and right sides disjoint.
2. Restrict to those FD's that involve only attributes of the projected schema.

Simple, Exponential Algorithm

1. For each set of attributes X, compute X^+.
2. Add $X \to A$ for all A in $X^+ - X$.
3. However, drop $XY \to A$ whenever we discover $X \to A$.
   - Because $XY \to A$ follows from $X \to A$ in any projection.
4. Finally, use only FD's involving projected attributes.

A Few Tricks

- No need to compute the closure of the empty set or of the set of all attributes.
- If we find X^+ = all attributes, so is the closure of any superset of X.

Example

- ABC with FD's $A \to B$ and $B \to C$. Project onto AC.
  - $A^+ = ABC$; yields $A \to B$, $A \to C$.
    - We do not need to compute AB^+ or AC^+.
  - $B^+ = BC$; yields $B \to C$.
  - $C^+ = C$; yields nothing.
  - $BC^+ = BC$; yields nothing.

Example --- Continued

- Resulting FD's: $A \rightarrow B$, $A \rightarrow C$, and $B \rightarrow C$.
- Projection onto AC : $A \rightarrow C$.
  - Only FD that involves a subset of $\{A,C\}$.

A Geometric View of FD's

- Imagine the set of all *instances* of a particular relation.
- 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)}

{}

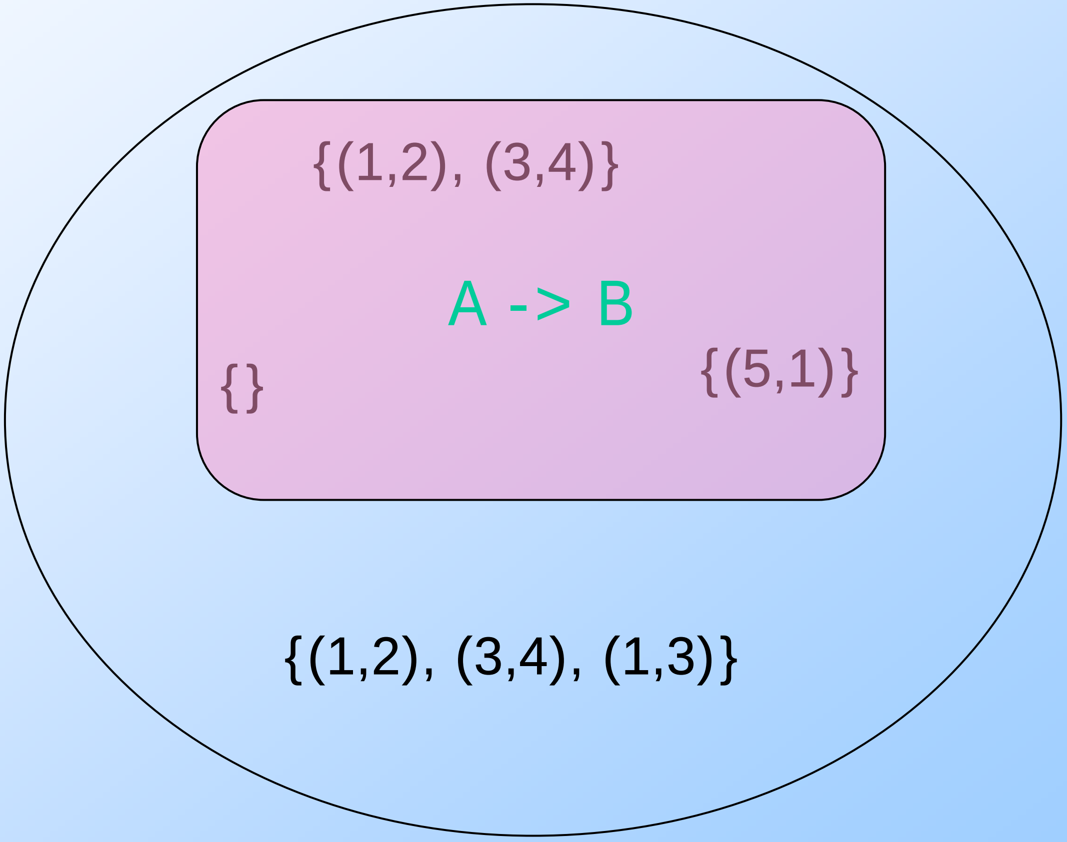
{(5,1)}

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

An FD is a Subset of Instances

- For each FD $X \text{ -> } A$ there is a subset of all instances that satisfy the FD.
- We can represent an FD by a region in the space.
- Trivial FD = an FD that is represented by the entire space.
  - Example: $A \text{ -> } A$.

Example: A -> B for R(A,B)

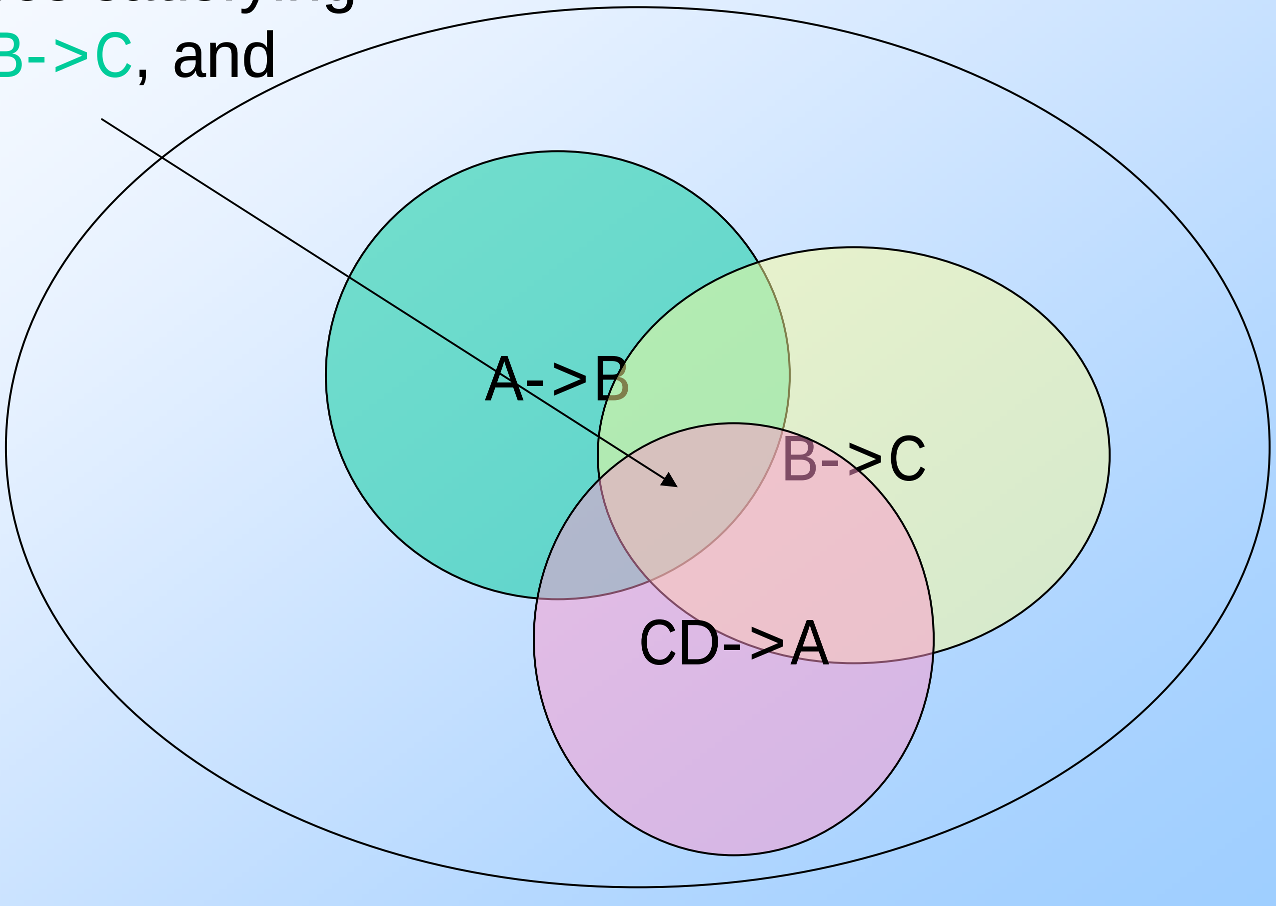


Representing Sets of FD's

- If each FD is a set of relation instances, then a collection of FD's corresponds to the intersection of those sets.
  - Intersection = all instances that satisfy all of the FD's.

Example

Instances satisfying
A->B, B->C, and
CD->A



Implication of FD's

- If an FD $Y \rightarrow B$ follows from FD's $X_1 \rightarrow A_1, \ldots, X_n \rightarrow A_n$, then the region in the space of instances for $Y \rightarrow B$ must include the intersection of the regions for the FD's $X_i \rightarrow A_i$.
  - That is, every instance satisfying all the FD's $X_i \rightarrow A_i$ surely satisfies $Y \rightarrow B$.
  - But an instance could satisfy $Y \rightarrow B$, yet not be in this intersection.

Example