Functional Dependencies

Functional Dependencies (FD's)

$X \rightarrow A$ = assertion about a relation R that whenever two tuples agree on all the attributes of X, then they must also agree on attribute A.

- Say " $X \rightarrow A$ holds in R"
- Reads "X functionally determines A"

Example

Movies(*title*, *year*, *length*, *film-type*, *studio-Name*, *star-Name*)

title	*year*	*length*	*film-type*	*studio-Name*	*star-Name*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	2002	142	color	Fox	Natalie Portman
Wayne's	1992	104	color	Paramount	Dana Carvey
Wayne's	1992	104	color	Paramount	Mike Meyers

Reasonable FD's to assert:

1. *title*, *year* → *length*
2. *title*, *year* → *film-type*
3. *title*, *year* → *studio-Name*

What about the FD: *title*, *year* → *star-Name* !?!??

Functional Dependencies (cont.)

- Shorthand: combine FD's with common left side by concatenating their right sides.
  Example:
  Given *title*, *year* → *length* and *title*, *year* → *film-type*
  we can have *title*, *year* → *length*, *film-type*
- Sometimes, several attributes jointly determine another attribute, although neither does by itself.
  Example:
  We know that *title*, *year* → *length* holds…
  It is easy to see that *title* → *length* does not hold

Keys of Relations

Define key K for relation R w.r.t. FD's. Set K of attrs is key if:

1. $K \rightarrow$ all attributes of R. **(Uniqueness)**
2. For no proper subset of K is (1) true. **(Minimality)**
   If K at least satisfies (1), then K is a *superkey*.

A few words about notation…

- Pick one key; underline key attributes in the relation schema.
- A, B, C etc., represent single attributes;
  X, Y, Z etc., represent sets of attributes
- No set formers in FD's… use ABC instead of $\{A, B, C\}$

Example

Movies(*title*, *year*, *length*, *film-type*, *studio-Name*, *star-Name*)

- {*title*, *year*, *star-Name*} → all attributes.
  Thus, it 's a superkey.
- *title* → *year* is false, so *title* is not a superkey.
- *star-Name* → *title* also false, so *star-Name* not a superkey.
- Thus, {*title*, *year*, *star-Name*} is a key.
- No other keys in this example.

Note: Neither *title* nor *year* nor *star-Name* is on the right of any observed FD, so they must be part of *any* superkey.

Who Determines Keys/FD's?

- We could assert a key K.
  - Then the only FD's asserted are that $K \to A$ for every attribute A.
  - No surprise: K is then the only key for those FD's, according to the formal definition of "key."
- Or, we could assert some FD's and *deduce* one or more keys by the formal definition.
- Rule of thumb: FD's either come from keyness, or from physics.
  - *E.g.*, "no two courses can meet in the same room at the same time" yields *rooms*, *times* $\to$ *courses*.

7

Inferring FD's

And this is important because when we talk about improving relational designs, we often need to ask:

"does this FD hold in this relation?"

More formally, given FD's

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

does FD $Y \to B$ necessarily hold in the same relation?

Try the following: Start by assuming two tuples agree in Y. Use given FD's to infer other attributes on which they must agree. If B is among them, then yes, else no.

8

Example

Consider $R(A, B, C)$ and FD's $A \to B$ and $B \to C$

Prove that $A \to C$

Proof: Consider two tuples (a, b_1, c_1), (a, b_2, c_2)

Then, due to $A \to B$, we have (a, b, c_1), (a, b, c_2)

Since $B \to C$ we have (a, b, c), (a, b, c)

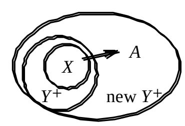
Thus, $A \to C$

9

Algorithm

The *closure* Y^+ of attribute set Y is the set of attributes functionally determined by Y

- Basis: $Y^+ := Y$.
- Induction: If $X \subseteq Y^+$, and $X \to A$ is a given FD, then add A to Y^+.



- Terminate: when Y^+ cannot be changed.

10

Example

Initial set of FDs: $A \to B$, $BC \to D$.

- $A^+ = AB$
- $C^+ = C$
- $(AC)^+ = ABCD$
- $(BC)^+ = BCD$

11

Another Example

- Set of FD's in *ABCGHI*:

  $A \to B$
  $A \to C$
  $CG \to H$
  $CG \to I$
  $B \to H$
- Compute $(CG)^+$, $(BG)^+$, $(AG)^+$
  - $(CG)^+ = CGHI$
  - $(BG)^+ = BGH$
  - $(AG)^+ = ABCGHI$

12

Armstrong's axioms

- Armstrong's Axioms:
  - *Reflexivity:* If $Y \subseteq X$, then $X \to Y$ (trivial FD's)
  - *Augmentation:* If $X \to Y$, then $XZ \to YZ$ for any Z
  - *Transitivity:* If $X \to Y$ and $Y \to Z$, then $X \to Z$
- These are *sound* and *complete* inference rules for FDs!

Example:
Initial set of FDs: $A \to B$, $BC \to D$. Show that $AC \to D$
Solution:
$A \to B$, then $AC \to BC$ (Augmentation).
$AC \to BC$ and $BC \to D$, then $AC \to D$ (Transitivity)

13

Given Versus Implied FD's

Typically, we state a few FD's that are known to hold for a relation *R*.

- Other FD's may follow logically from the given FD's; these are *implied* FD's.
- We are free to choose any *basis* for the FD's of *R* – a set of FD's that imply all the FD's that hold for *R*.

14

Finding All Implied FD's

Motivation: Suppose we have a relation *ABCD* with some FD's *F*. If we decide to decompose *ABCD* into *ABC* and *AD*, what are the FD's for *ABC*, *AD*?

- Example: $F = AB \to C, C \to D, D \to A$. It looks like just $AB \to C$ holds in *ABC*, but in fact $C \to A$ follows from *F* and applies to relation *ABC*.
- Problem is exponential in worst case.

15

Algorithm

Assume relation *R*, set *F* of FD's and set $S \subseteq R$

- Compute X^+ for each $X \subseteq S$
  - But skip the cases $X = \emptyset$, X = all attributes.
  - Add $X \to A$ for each A in $X^+ – X$.
- Drop $XY \to A$ if $X \to A$ holds.
  - Consequence: If X^+ is all attributes, then there is no point in computing closure of supersets of *X*.
- Finally, project the FD's by selecting only those FD's that involve only attributes of *S*

16

Example

$F = AB \to C, C \to D, D \to A$. What FD's follow?

- $A^+ = A$; $B^+ = B$ (nothing).
- $C^+ = ACD$ (add $C \to A$).
- $D^+ = AD$ (nothing new).
- $(AB)^+ = ABCD$ (add $AB \to D$; skip all supersets of *AB*).
- $(BC)^+ = ABCD$ (nothing new; skip all supersets of *BC*).
- $(BD)^+ = ABCD$ (add $BD \to C$; skip all supersets of *BD*).
- $(AC)^+ = ACD$; $(AD)^+ = AD$; $(CD)^+ = ACD$ (nothing new).
- $(ACD)^+ = ACD$ (nothing new).
- All other sets contain *AB*, *BC*, or *BD*, so skip.
- Thus, the only interesting FD's that follow from *F* are: $C \to A$, $AB \to D$, $BD \to C$.

17

Another Example

In *ABC* with FD's $A \to B$, $B \to C$, project onto *AC*

1. $A^+ = ABC$; yields $A \to B$, $A \to C$.
2. $B^+ = BC$; yields $B \to C$.
3. $C^+ = C$ and $BC^+ = BC$; adds nothing.

- Resulting FD's: $A \to B$, $A \to C$, $B \to C$.
- Projection onto *AC*: $A \to C$.

18

Design Anomalies and Normalization

19

Design Anomalies

Movies(*title*, *year*, *length*, *film-type*, *studio-Name*, *star-Name*)

title	*year*	*length*	*film-type*	*studio-Name*	*star-Name*
Star Wars	1977	124	color	???	Carrie Fisher
Star Wars	1977	???	color	Fox	Mark Hamill
Star Wars	1977	124	???	Fox	Harrison Ford
Wayne's	1992	104	color	Paramount	Mike Meyers

1. *title, year* → *length*
2. *title, year* → *film-type*
3. *title, year* → *studio-Name*

- ???'s are redundant, since we can figure them out from the FD's.
- Update anomalies: If Star Wars is actually 125 minutes long, will we change *length* in each of its tuples?
- Deletion anomalies: If we delete Mike Meyers, we lose the info for Wayne's World.

20

Decomposition of a Relation Scheme

- Suppose that relation R contains attributes $A_1 \ldots A_n$
  A *decomposition* of R consists of replacing R by two (or more) relations such that:
  - Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
  - Every attribute of R appears as an attribute in at least one of the new relations.
- Intuitively, decomposing R means we will store instances of the relation schemes produced by the decomposition, instead of instances of R.

21

Normal Forms

- Given relation R, the first question to ask is whether any refinement/decomposition is needed!
- If a relation is in a certain *normal form* (BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimized. This can be used to help us decide whether decomposing the relation will help.
- Role of FD's in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC.
    - No FD's hold: There is no redundancy here.
    - Given A → B: Several tuples could have the same A value, and if so, they'll all have the same B value!

22

Boyce-Codd Normal Form (BCNF)

Formally, *R* is in BCNF if for every nontrivial FD for *R*, say $X \rightarrow A$, then *X* is a superkey.

Advantages of BCNF:
1. Guarantees no redundancy due to FD's.
2. Guarantees no update anomalies
3. Guarantees no deletion anomalies

23

Example

Movies(*title*, *year*, *length*, *film-type*, *studio-Name*, *star-Name*)

title	*year*	*length*	*film-type*	*studio-Name*	*star-Name*
Star Wars	1977	124	color	Fox	Carrie Fisher
Star Wars	1977	124	color	Fox	Mark Hamill
Star Wars	1977	124	color	Fox	Harrison Ford
Wayne's	1992	104	color	Paramount	Mike Meyers

Functional dependencies:
1. *title, year* → *length*
2. *title, year* → *film-type*
3. *title, year* → *studio-Name*

Each of the given FD's is a BCNF violation… Why?
Remember that the key is {*title*, *year*, *star-Name*}

24

Decomposition to Reach BCNF

The setting: relation R, set F of FD's.

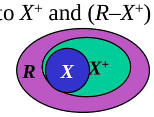
Suppose relation R has BCNF violation $X \rightarrow B$

- We need only look among FD's of F for a BCNF violation, not those that follow from F.
- Proof: If $Y \rightarrow A$ is a BCNF violation and follows from F, then the computation of Y^+ used at least one FD $X \rightarrow B$ from F.
  – X must be a subset of Y.
  – Thus, if Y is not a superkey, X cannot be a superkey either, and $X \rightarrow B$ is also a BCNF violation.

25

Decomposition to Reach BCNF (cont.)

1. Compute X^+.
   – Cannot be all attributes – why?
2. Decompose R into X^+ and $(R–X^+) \cup X$.



3. Find the FD's for the decomposed relations. Project the FD's from F calculate all consequents of F that involve only attributes from X^+ or only from $(R–X^+) \cup X$.

26

Example

R = *Movies*(*title*, *year*, *length*, *film-type*, *studio-Name*, *star-Name*)

F =

1. *title*, *year* $\rightarrow$ *length*
2. *title*, *year* $\rightarrow$ *film-type*
3. *title*, *year* $\rightarrow$ *studio-Name*

Pick BCNF violation, e.g.,

title, *year* $\rightarrow$ *length*, *film-type*, *studio-Name*

- Find the closure $(title, year)^+$… = {*title*, *year* , *length*, *film-type*, *studio-Name*}
- Create decomposed relations:
  $Movies_1$(<u>*title*</u>, <u>*year*</u>, *length*, *film-type*, *studio-Name*)
  $Movies_2$(<u>*title*</u>, <u>*year*</u>, <u>*star-Name*</u>)

27

Another Example

R = *Movies*(*title*, *year*, *studio-Name*, *president*, *presAddr*)

F =

1. *title*, *year* $\rightarrow$ *studio-Name*
2. *studio-Name* $\rightarrow$ *president*
3. *president* $\rightarrow$ *presAddr*

Pick BCNF violation, e.g., *studio-Name* $\rightarrow$ *president*

- Find the closure $(studio\text{-}Name)^+$… = {*studio-Name*, *president*, *presAddr*}
- Create decomposed relations:
  $Movies_1$(<u>*title*</u>, <u>*year*</u>, *studio-Name*)
  $Movies_2$(<u>*studio-Name*</u>, *president*, *presAddr*)
- Projected FD's (skipping a lot of work that leads nowhere interesting):
  – For $Movies_1$: *title*, *year* $\rightarrow$ *studio-Name*
  – For $Movies_2$: *studio-Name* $\rightarrow$ *president* and *president* $\rightarrow$ *presAddr*

28

Another Example (cont.)

(Continuing…)

- Decomposed relations:
  $Movies_1$(<u>*title*</u>, <u>*year*</u>, *studio-Name*)
  $Movies_2$(<u>*studio-Name*</u>, *president*, *presAddr*)
  Projected FD's:
  – For $Movies_1$: *title*, *year* $\rightarrow$ *studio-Name*
  – For $Movies_2$: *studio-Name* $\rightarrow$ *president* and *president* $\rightarrow$ *presAddr*
- BCNF violations?
  – For $Movies_1$, {<u>*title*</u>, <u>*year*</u>} is key and all left sides of FD's are superkeys.
  – For $Movies_2$, {<u>*studio-Name*</u>} is the key… but *president* $\rightarrow$ *presAddr* violates BCNF.

29

Decomposing $Movies_{2…}$

- First set of decomposed relations:
  $Movies_1$(<u>*title*</u>, <u>*year*</u>, *studio-Name*)
  $Movies_2$(<u>*studio-Name*</u>, *president*, *presAddr*)
- Close *president* $^+$… = {*president*, *presAddr*}
- Decompose $Movies_2$ into:
  $Movies_{2\text{-}1}$(<u>*studio-Name*</u>, *president*)
  $Movies_{2\text{-}2}$(<u>*president*</u>, *presAddr*)
- Resulting relations are all in BCNF:
  $Movies_1$(<u>*title*</u>, <u>*year*</u>, *studio-Name*)
  $Movies_{2\text{-}1}$(<u>*studio-Name*</u>, *president*)
  $Movies_{2\text{-}2}$(<u>*president*</u>, *presAddr*)

30