Databases 1

Functional Dependencies

Relational Schema Design

- Goal of relational schema design is to avoid anomalies and redundancy.
  - *Update anomaly* : one occurrence of a fact is changed, but not all occurrences.
  - *Deletion anomaly* : valid fact is lost when a tuple is deleted.

Example of Bad Design

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

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

Data is redundant, because each of the ???'s can be figured out by using the FD's name -> addr favBeer and beersLiked -> manf.

This Bad Design Also Exhibits Anomalies

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

- Update anomaly: if Janeway is transferred to *Intrepid*, will we remember to change each of her tuples?
- Deletion anomaly: If nobody likes Bud, we lose track of the fact that Anheuser-Busch manufactures Bud.

Decomposition

- Definition

  $d=\{R_1,...,R_k\}$ decomposition, if $R_1 \cup ... \cup R_k = R$.

- Example:

  R=ABCDE, d={AD,BCE,ABE}

  R_1=AD, R_2=BCE, R_3=ABE

Functional Dependencies

- *X* -> *Y* 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 all attributes in set *Y*.
  - Say "*X* -> *Y* 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* }.

Splitting Right Sides of FD's

- $X \to A_1A_2\ldots A_n$ holds for R exactly when each of $X \to A_1$, $X \to A_2$,…, $X \to A_n$ hold for R.
- Example: $A \to BC$ is equivalent to $A \to B$ and $A \to C$.
- There is no splitting rule for left sides.
- We'll generally express FD's with singleton right sides.

Example: FD's

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

- Reasonable FD's to assert:
  1. name -> addr favBeer
     - Note this FD is the same as
       name -> addr and
       name -> favBeer.
  2. beersLiked -> manf

Example: Possible 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

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: Superkey

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: Key

- {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}.

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.

More FD's From "Physics"

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

- *ABC* relational schemas *AB* ->*C* and *C* ->*B*
  - *A* = street, *B* = city, *C* = zip code.
- Keys: {*A*,*B* } and {*A*,*C* }, too.

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.

Armstrong axioms

Let R(U) relation schema and X,Y $\subseteq$ U, and denote

XY is the union of attribute-sets X and Y

Let F functional dependencies

Armstrong axioms:

- A1 (reflexivity or trivial fd): if Y$\subseteq$X then X$\rightarrow$Y.
- A2 (augmentation): if X$\rightarrow$Y then XZ$\rightarrow$YZ.
- A3 (tranzitivity): if X$\rightarrow$Y and Y$\rightarrow$Z then X$\rightarrow$Z.

More rules about functional dependencies

4. Splitting (decomposition) rule
   X→Y and Z⊆ Y then X→Z.
5. Combining (union) rule
   X→Y and X→Z then X→YZ.
6. Pseudotranzitivity
   X→Y and WY→Z then XW→Z.

Proof (4): Reflexivity axiom Y→Z, and
tranzitivity axiom X→Z.

Proof (5): Augmentation axioms XX→YX and YX→YZ,
XX=X, tranzitivity axioms X→YZ.

Proof (6): Augmentation axioms XW→YW and YW=WY,
tranzitivity axiom XW→Z.

Closure of set of attributes

- Definition: $X^{+(F)} := \{A \mid F \vdash X \rightarrow A\}$
- The *closure* of X under the FD's in *S* is the set of attributes *A* such that every relation that satisfies all the FD's in set *S* also satisfies $X \rightarrow A$, that is $X \rightarrow A$ follows from the FD's of *S.*
- Lemma: $F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$.

Proof: $(\Rightarrow)$ if $\forall A \in Y$ reflexivity and transitivity rule $F \vdash X \rightarrow A$, so $Y \subseteq X^+$.

$(\Leftarrow)$ if $\forall A \in Y \subseteq X^+$ then $F \vdash X \rightarrow A$, union rule $F \vdash X \rightarrow Y$.

Closure Test for Y^+

- Input: Y set of attributes, F funct.dependencies
- Output: Y^+
- Algorithm Y^+ :

loop

$Y(0):=Y$

$Y(n+1):= Y(n) \cup \{A \mid \underline{X \rightarrow Z \in F}, A \in Z, X \subseteq Y(n)\}$

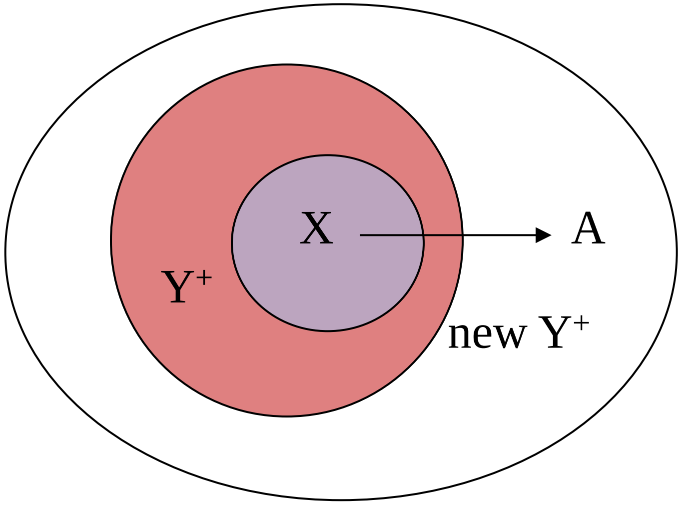
if $Y(v+1)=Y(v)$, then leave the loop

end of the loop

Output: $Y(v)=Y^+$.

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^+.



Example: Closure Test

$R=ABCDEFG, \{AB\rightarrow C, B\rightarrow G, CD\rightarrow EG, BG \rightarrow E\}$

$X=ABF, X^{+}=?$

$X(0):=ABF$

$X(1):=ABF\cup\{C,G\}=ABCFG$

$X(2):=ABCFG \cup\{C,G,E\}=ABCEFG$

$X(3):=ABCEFG$

$X^{+}= ABCEFG$

Exercises (Book 3.5.2.)

Consider the relation Courses(*C ,T ,H ,R ,S ,G),*

F= {C→T, HR→C, HT→R, HS→R, CS→G}

whose attributes may be thought of informally as course, teacher, hour, room, student, and grade.

Let the set of FD's for Courses be C → *T, HR → C, HT → R, HS → R,* and *CS → G.*

Intuitively, the first says that a course has a unique teacher, and the second says that only one course can meet in a given room at a given hour. The third says that a teacher can be in only one room at a given hour, and the fourth says the same about students. The last says that students get only one grade in a course.

What are all the keys for Courses?

Finding All Implied FD's - Projecting FD's

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

$a_1=a_2$ because *D -> A*

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 = right side not contained in the left.
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* ->*A* for all *A* in X^+ - *X*.
3. However, drop *XY* ->*A* whenever we discover *X* ->*A*.
   - Because *XY* ->*A* follows from *X* ->*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: Projecting FD's

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