CS 235: Introduction to Databases

Svetlozar Nestorov

Lecture Notes #5

Outline

- Functional dependencies (FD)
- Properties of FD
- Inferring FD
- Normalization

Functional Dependencies

- $X \rightarrow A$
  - assertion about a relation *R* that whenever two tuples agree on all the attributes of *X*, then they must also agree on attribute *A*.
- Important as a constraint on the data that may appear within a relation.
  - Schema-level control of data.
- Mathematical tool for explaining the process of "normalization" – vital for redesigning database schemas when original design has certain flaws.

FD Conventions

- *X*, etc., represent sets of attributes; *A* etc., represent single attributes.
- No set formers ({}) in FD's, e.g., *ABC* instead of {*A*, *B*, *C*}.

Example

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

name	*addr*	*beersLiked*	*manf*	*favoriteBeer*
Mike	111 E Ohio	Bud	A.B.	Blonde Ale
Mike	111 E Ohio	Blonde Ale	G.I.	Blonde Ale
Anna	123 W Grand	BudLite	A.B.	BudLite

- Reasonable FD's to assert:
  1. …
  2. …
  3. …
- Note: FD's can give more detail than just assertion of a key.

Properties of FD's

- Key (in general) functionally determines all attributes. In our example:

name beersLiked $\rightarrow$ *addr favoriteBeer beerManf*

- Shorthand: combine FD's with common left side by concatenating their right sides.
- When FD's are *not* of the form Key $\rightarrow$ other attribute(s), then there is typically an attempt to "cram" too much into one relation.

Properties of FD's

- Sometimes, several attributes jointly determine another attribute, although neither does by itself.
- Example:

  beer bar → *price*

Formal Notion of Key

- *K* is a *key* for relation *R* if:
  1. *K* → all attributes of *R*.
  2. For no proper subset of *K* is (1) true.
- If *K* at least satisfies (1), then *K* is a *superkey*.

Example

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

- {*name, beersLiked*} FD's all attributes, as seen.
  - Shows {*name, beersLiked*} is a superkey.
- *name* → *beersLiked* is false, so *name* not a superkey.
- *beersLiked* → *name* also false, so *beersLiked* not a superkey.
- Thus, {*name, beersLiked*} is a key.
- No other keys in this example.
  - Neither *name* nor *beersLiked* is on the right of any observed FD, so they must be part of *any* superkey.

Who Determines Keys/FD's?

- We could define a relation schema by simply giving a single key *K*.
  - Then the only FD's asserted are that *K* → *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.
  - E/R diagram implies FD's by key declarations and many-one relationship declarations.
- Rule of thumb: FD's either come from keyness, many-1 relationship, or from physics.
  - E.g., "no two courses can meet in the same room at the same time" yields *room time* → *course*.

Inferring FD's

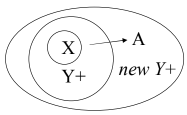
- When we talk about improving relational designs, we often need to ask "does this FD hold in this relation?"
- Given FD's *X1*→ *A1*, *X2* → *A2* ··· *Xn* → *An*, does FD *Y* → *B* necessarily hold in the same relation?
- Start by assuming two tuples agree in *Y*. Use given FD's to infer other attributes on which they must agree. If *B* is among them, then yes, else no.

Closure of Attributes

- Given a relation R with attributes X and a subset of the attributes Y.
- Find all A's such that Y → A.
- Define Y^+ = closure of Y = set of attributes functionally determined by Y (all the A's)

Closure Algorithm

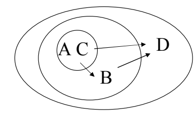
- Basis: Y^+:=Y.
- Induction: If $X \subseteq Y^+$, and $X \rightarrow A$ is a given FD, then add A to Y^+.



- End when Y^+ cannot be changed.

Example

- Relation R(A,B,C,D).
- FD's: $A \rightarrow B$, $BC \rightarrow D$.
- $A^+ = AB$.
- $C^+=C$.
- $(AC)^+ = ABCD$.



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*.

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

Algorithm

- For each set of attributes *X* compute X^+.
- Add $X \rightarrow A$ for each *A* in $X^+ – X$.
- Ignore or drop some "obvious" dependencies that follow from others:
- 1. *Trivial FD's*: right side is a subset of left side.
  - Consequence: no point in computing $\emptyset^+$ or closure of full set of attributes.
- 2. Drop $XY \rightarrow A$ if $X \rightarrow A$ holds.
  - Consequence: If X^+ is all attributes, then there is no point in computing closure of supersets of *X*.
- 3. Ignore FD's whose right sides are not single attributes.
- Notice that after we project the discovered FD's onto some relation, the FD's eliminated by rules 1, 2, and 3 can be inferred *in the projected relation*.

Example

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

- $A^+ = A$; $B^+=B$ (nothing).
- $C^+=ACD$ (add $C \rightarrow A$).
- $D^+=AD$ (nothing new).

…

Normalization

- Improve the schema by decomposing relations and removing anomalies.
- Boyce-Codd Normal Form (BCNF): all FD's follow from the fact *key* $\rightarrow$ *everything*.
- Formally, *R* is in BCNF if every nontrivial FD for *R*, say $X \rightarrow A$, has *X* a superkey.
  – "Nontrivial" = right-side attribute not in left side.

BCNF properties

1. Guarantees no redundancy due to FD's.
2. Guarantees no *update anomalies* = one occurrence of a fact is updated, not all.
3. Guarantees no *deletion anomalies* = valid fact is lost when tuple is deleted.

Example (1/2)

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

name	*addr*	*beersLiked*	*manf*	*favoriteBeer*
Mike	111 E Ohio	Bud	A.B.	Blonde Ale
Mike	???	Blonde Ale	G.I.	???
Anna	123 W Grand	Bud	???	BudLite

- FDs:

1. *name* $\rightarrow$ *addr*
2. *name* $\rightarrow$ *favoriteBeer*
3. *beersLiked* $\rightarrow$ *manf*

- ???'s are redundant, since we can figure them out from the FD's.
- Update anomalies: If Mike moves, we need to change *addr* in each of his tuples?
- Deletion anomalies: If nobody likes Bud, we lose track of Bud's manufacturer.

Example (2/2)

Each of the given FD's is a BCNF violation:

- Key = {*name, beersLiked*}
  – Each of the given FD's has a left side a proper subset of the key.