Design Theory for Relational Databases

FUNCTIONAL DEPENDENCIES

DECOMPOSITIONS

NORMAL FORMS

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 ->C*, *C ->D*, and *D ->A*.

- Assume we decompose into *ABC*, *AD*. What FD's hold in *ABC* ?
- *AB ->C* holds but how about *C ->A*? Perform Inference Test OR Closure Test

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 ->C*, *C ->D*, and *D ->A*.

- Decompose into *ABC*, *AD*. What FD's hold in *ABC* ?
- Not only *AB ->C*, but also *C ->A*? as C+ = CDA Yes!

Basic Idea for decomposing table

1. Start with given FD’s and **find** all FD’s that follow from the given FD’s.
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** $\boldsymbol{X}^+$.
2. Finally, use only FD's involving projected attributes.

A Few Tricks

No need to compute the closure of the set of all attributes.

If we find $\boldsymbol{X^+}$ **= 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^+ or ABC^+.
- $B^+ = BC$; 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 $\{A, C\}$.

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

Boyce-Codd Normal Form

We say a relation *R* is in *BCNF* if whenever *X ->Y* is a **nontrivial FD** that holds in *R*, ***X*** is a ***superkey***.

- *nontrivial* means *Y* is not contained in *X*.
- Remember, a *superkey* is any superset of a key

Example

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

FD's: name->addr, favBeer, beersLiked->manf

In each FD, the left side is *not* a superkey.

Any one of these FD's shows *Drinkers* is not in BCNF

Another Example

Beers(name, manf, manfAddr)

FD's: name->manf,   manf->manfAddr

Only key is {name}.

name->manf does not violate BCNF because name is a superkey.

But manf->manfAddr does because manf is not a superkey.

Decomposition into BCNF

Given: relation *R* with FD's *F*.

Look among the given FD's for a **BCNF violation**: $X \rightarrow Y$.

- Compute X^+

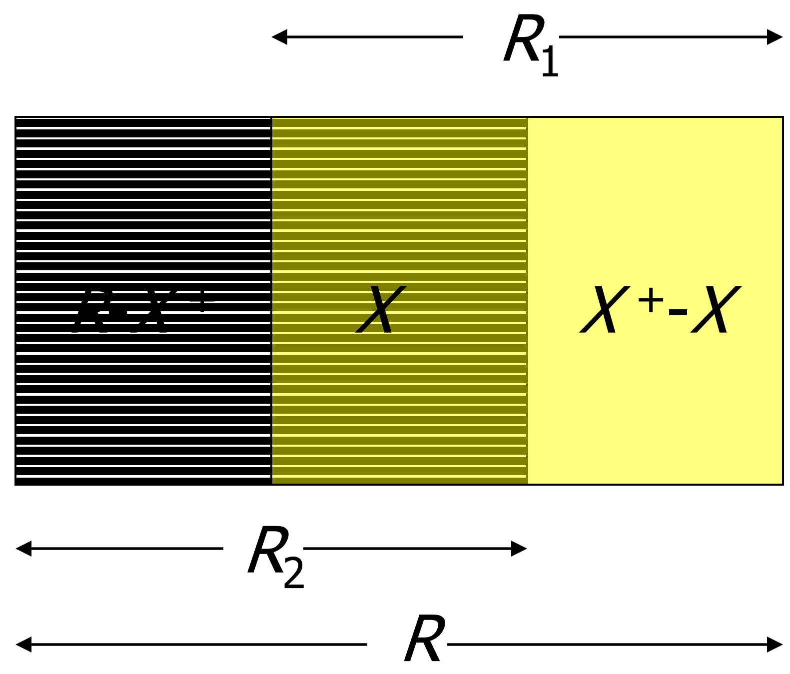
Decompose *R* Using *X* -> *Y*

Replace *R* by relations with schemas:

1. $R_1 = X^+$.
2. $R_2 = R - X^+ + X$.

Project given FD's *F* onto the <u>two new relations</u>.

Decomposition Picture



Example: BCNF Decomposition

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

F = name->addr, name -> favBeer, beersLiked->manf

Pick BCNF violation name->addr.

Closure of the left side: $\{\text{name}\}^+ = \{\text{name, addr, favBeer}\}$.

Decomposed relations:

1. Drinkers1(name, addr, favBeer)
2. Drinkers2(name, beersLiked, manf)

Example -- Continued

We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.

Projecting FD’s is easy here.

For Drinkers1(name, addr, favBeer), relevant FD’s are name->addr and name->favBeer.

- Thus, Drinkers1 is in BCNF.

Example -- Continued

For Drinkers2(<u>name</u>, <u>beersLiked</u>, manf), one of the FDs is beersLiked->manf but beersLiked is not a superkey.

- Violation of BCNF.

beersLiked$^+$ = {beersLiked, manf}, so we decompose *Drinkers2* into:

1. Drinkers3(<u>beersLiked</u>, manf)
2. Drinkers4(<u>name</u>, <u>beersLiked</u>)

Example -- Concluded

The resulting decomposition of *Drinkers* :

1. Drinkers1(<u>name</u>, addr, favBeer)
2. Drinkers3(<u>beersLiked</u>, manf)
3. Drinkers4(<u>name</u>, <u>beersLiked</u>)

Notice: *Drinkers1* tells us about **drinkers**, *Drinkers3* tells us about **beers**, and *Drinkers4* tells us the **relationship between drinkers and the beers** they like.

Actions

Review slides!

Read Chapters 3.1. – 3.5 (Design Theory for Relational Databases).