Databases 1

Relational Schema Design

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.
  - Remember: *nontrivial* means *Y* is not contained in *X*.
  - Remember, a *superkey* is any superset of a key (not necessarily a proper superset).

Example

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

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

- Only key is {name, beersLiked}.
- 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(<u>name</u>, manf, manfAddr)

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

- Only key is {name} .
- name->manf does not violate BCNF, but manf->manfAddr does.

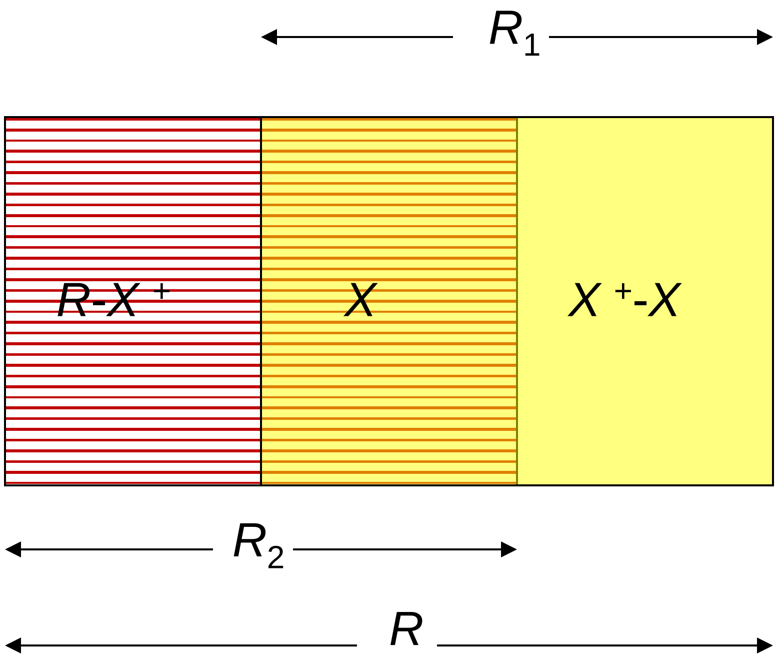
Decomposition into BCNF

- Given: relation *R* with FD's *F*.
- Look among the given FD's for a BCNF violation $X \text{->} Y$.
  - If any FD following from *F* violates BCNF, then there will surely be an FD in *F* itself that violates BCNF.
- Compute X^+.
  - Not all attributes, or else X is a superkey.

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 two new relations.

Decomposition Picture



Example: BCNF Decomposition

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

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

- Pick BCNF violation name->addr.
- Close the left side: $\{name\}^+$ = {name, addr, favBeer}.
- Decomposed relations:
  1. Drinkers1(<u>name</u>, addr, favBeer)
  2. Drinkers2(<u>name</u>, <u>beersLiked</u>, manf)

Example --- Continued

- We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.
- Projecting FD's is easy here.
- For Drinkers1(<u>name</u>, addr, favBeer), relevant FD's are name->addr and name->favBeer.
  - Thus, {name} is the only key and Drinkers1 is in BCNF.

Example --- Continued

- For Drinkers2(name, beersLiked, manf), the only FD is beersLiked->manf, and the only key is {name, beersLiked}.
  - Violation of BCNF.
- beersLiked^{+} = {beersLiked, manf}, so we decompose *Drinkers2* into:

1. Drinkers3(beersLiked, manf)
2. Drinkers4(name, beersLiked)

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.

Testing for a Lossless Join

- If we project R onto R_1, R_2,…, R_k , can we recover R by rejoining?
- Any tuple in R can be recovered from its projected fragments.
- So the only question is: when we rejoin, do we ever get back something we didn't have originally?

The Chase Test

- Suppose tuple *t* comes back in the join.
- Then *t* is the join of projections of some tuples of *R*, one for each R_i of the decomposition.
- Can we use the given FD's to show that one of these tuples must be *t* ?

The Chase – (2)

- Start by assuming $t = abc$… .
- For each i, there is a tuple s_i of R that has a, b, c,… in the attributes of R_i.
- s_i can have any values in other attributes.
- We'll use the same letter as in t, but with a subscript, for these components.

Example: The Chase

- Let *R* = *ABCD*, and the decomposition be *AB*, *BC*, and *CD*.
- Let the given FD's be *C->D* and *B ->A*.
- Suppose the tuple t = abcd is the join of tuples projected onto *AB*, *BC*, *CD*.

The *Tableau*

The tuples of R projected onto AB, BC, CD.

A	*B*	*C*	*D*
a	*b*	c_1	d_1
$\cancel{a_2}$ *a*	*b*	*c*	$\cancel{d_2}$ *d*
a_3	b_3	*c*	*d*

Use *B ->A*

Use *C->D*

We've proved the second tuple must be *t*.

Summary of the Chase

1. If two rows agree in the left side of a FD, make their right sides agree too.
2. Always replace a subscripted symbol by the corresponding unsubscripted one, if possible.
3. If we ever get an unsubscripted row, we know any tuple in the project-join is in the original (the join is lossless).
4. Otherwise, the final tableau is a counterexample.

Example: Lossy Join

- Same relation $R = ABCD$ and same decomposition.
- But with only the FD $C\text{->}D$.

The *Tableau*

These projections rejoin to form *abcd*.

A	B	C	D
a	b	c_1	d_1
a_2	b	c	d_2
a_3	b_3	c	$\cancel{d_3}$ d

Use *C->D*

These three tuples are an example *R* that shows the join lossy. *abcd* is not in *R*, but we can project and rejoin to get *abcd*.

Third Normal Form -- Motivation

- There is one structure of FD's that causes trouble when we decompose.
- *AB* ->*C* and *C* ->*B*.
  - Example: *A* = street address, *B* = city, *C* = zip code.
- There are two keys, {*A*,*B* } and {*A*,*C* }.
- *C* ->*B* is a BCNF violation, so we must decompose into *AC*, *BC*.

We Cannot Enforce FD's

- The problem is that if we use *AC* and *BC* as our database schema, we cannot enforce the FD *AB ->C* by checking FD's in these decomposed relations.
- Example with *A* = street, *B* = city, and *C* = zip on the next slide.

An Unenforceable FD

street	zip
545 Tech Sq.	02138
545 Tech Sq.	02139

city	zip
Cambridge	02138
Cambridge	02139

Join tuples with equal zip codes.

street	city	zip
545 Tech Sq.	Cambridge	02138
545 Tech Sq.	Cambridge	02139

Although no FD's were violated in the decomposed relations, FD street city -> zip is violated by the database as a whole.

3NF Let's Us Avoid This Problem

- 3rd Normal Form (3NF) modifies the BCNF condition so we do not have to decompose in this problem situation.
- An attribute is *prime* if it is a member of any key.
- $X \text{->} A$ violates 3NF if and only if X is not a superkey, and also A is not prime.

Example: 3NF

- In our problem situation with FD's *AB* ->*C* and *C* ->*B*, we have keys *AB* and *AC*.
- Thus *A*, *B*, and *C* are each prime.
- Although *C* ->*B* violates BCNF, it does not violate 3NF.

What 3NF and BCNF Give You

- There are two important properties of a decomposition:
  1. *Lossless Join* : it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original.
  2. *Dependency Preservation* : it should be possible to check in the projected relations whether all the given FD's are satisfied.

3NF and BCNF -- Continued

- We can get (1) with a BCNF decomposition.
- We can get both (1) and (2) with a 3NF decomposition.
- But we can't always get (1) and (2) with a BCNF decomposition.
  - street-city-zip is an example.

3NF Synthesis Algorithm

- We can always construct a decomposition into 3NF relations with a lossless join and dependency preservation.
- Need *minimal basis* for the FD's:
  1. Right sides are single attributes.
  2. No FD can be removed.
  3. No attribute can be removed from a left side.

Constructing a Minimal Basis

1. Split right sides.
2. Repeatedly try to remove an FD and see if the remaining FD’s are equivalent to the original.
3. Repeatedly try to remove an attribute from a left side and see if the resulting FD’s are equivalent to the original.

3NF Synthesis – (2)

- One relation for each FD in the minimal basis.
  - Schema is the union of the left and right sides.
- If no key is contained in an FD, then add one relation whose schema is some key.

Example: 3NF Synthesis

- Relation R = ABCD.
- FD's *A->B* and *A->C*.
- Decomposition: AB and AC from the FD's, plus AD for a key.

Why It Works

- Preserves dependencies: each FD from a minimal basis is contained in a relation, thus preserved.
- Lossless Join: use the chase to show that the row for the relation that contains a key can be made all-unsubscripted variables.
- 3NF: hard part – a property of minimal bases.