Database – Slide 4

Normalization

Goal = BCNF = Boyce-Codd Normal Form =
all FD's follow from the fact "key $\rightarrow$ everything."

- Formally, R is in BCNF if for every nontrivial FD for R, say $X \rightarrow A$, then X is a superkey.
  - "Nontrivial" = right-side attribute not in left side.

Why?

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 of Problems

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

FD's:

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 Janeway gets transferred to the *Intrepid*, will we change `addr` in each of her tuples?
- Deletion anomalies: If nobody likes Bud, we lose track of Bud's manufacturer.

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

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

Another Example

`Beers(name, manf, manfAddr)`.

- FD's = `name` $\rightarrow$ `manf`, `manf` $\rightarrow$ `manfAddr`.
- Only key is `name`.
  - `Manf` $\rightarrow$ `manfAddr` violates BCNF with a left side unrelated to any key.

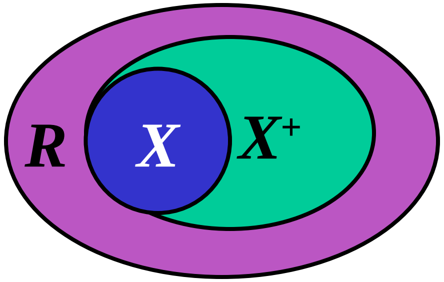
Decomposition to Reach BCNF

Setting: relation R, given FD's F.

Suppose relation R has BCNF violation $X \to B$.

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

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 $(R–X^+) \cup X$.

Example

R = `Drinkers(`<u>`name`</u>`, addr, `<u>`beersLiked`</u>`, manf, favoriteBeer)`

F =

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

Pick BCNF violation `name` $\rightarrow$ `addr`.

- Close the left side: `name` $^+$ = `name addr favoriteBeer`.
- Decomposed relations:

  `Drinkers1(`<u>`name`</u>`, addr, favoriteBeer)`

  `Drinkers2(`<u>`name`</u>`, `<u>`beersLiked`</u>`, manf)`
- Projected FD's (skipping a lot of work that leads nowhere interesting):
  - For `Drinkers1`: `name` $\rightarrow$ `addr` and `name` $\rightarrow$ `favoriteBeer`.
  - For `Drinkers2`: `beersLiked` $\rightarrow$ `manf`.

(Repeating)

- Decomposed relations:

  `Drinkers1(name, addr, favoriteBeer)`

  `Drinkers2(name, beersLiked, manf)`

- Projected FD's:
  - For `Drinkers1`: `name` → `addr` and `name` → `favoriteBeer`.
  - For `Drinkers2`: `beersLiked` → `manf`.

- BCNF violations?
  - For `Drinkers1`, `name` is key and all left sides of FD's are superkeys.
  - For `Drinkers2`, {`name`, `beersLiked`} is the key, and `beersLiked` → `manf` violates BCNF.

Decompose `Drinkers2`

- First set of decomposed relations:

  `Drinkers1(`<u>`name`</u>`, addr, favoriteBeer)`

  `Drinkers2(`<u>`name`</u>`, `<u>`beersLiked`</u>`, manf)`

- Close `beersLiked`$^+$ = `beersLiked`, `manf`.
- Decompose `Drinkers2` into:

  `Drinkers3(`<u>`beersLiked`</u>`, manf)`

  `Drinkers4(`<u>`name`</u>`, `<u>`beersLiked`</u>`)`

- Resulting relations are all in BCNF:

  `Drinkers1(`<u>`name`</u>`, addr, favoriteBeer)`

  `Drinkers3(`<u>`beersLiked`</u>`, manf)`

  `Drinkers4(`<u>`name`</u>`, `<u>`beersLiked`</u>`)`

3NF

One FD structure causes problems:

- If you decompose, you can't check all the FD's only in the decomposed relations.
- If you don't decompose, you violate BCNF.

Abstractly: $AB \to C$ and $C \to B$.

- Example 1: `title city` $\to$ `theatre` and `theatre` $\to$ `city`.
- Example 2: `street city` $\to$ `zip`, `zip` $\to$ `city`.

Keys: $\{A, B\}$ and $\{A, C\}$, but $C \to B$ has a left side that is not a superkey.

- Suggests decomposition into BC and AC.
  - But you can't check the FD $AB \to C$ in only these relations.

“Elegant” Workaround

Define the problem away.

- A relation *R* is in 3NF iff (if and only if) for every nontrivial FD $X \to A$, either:
  1. *X* is a superkey, or
  2. *A* is *prime* = member of at least one key.
- Thus, the canonical problem goes away: you don’t have to decompose because all attributes are prime.

What 3NF Gives You

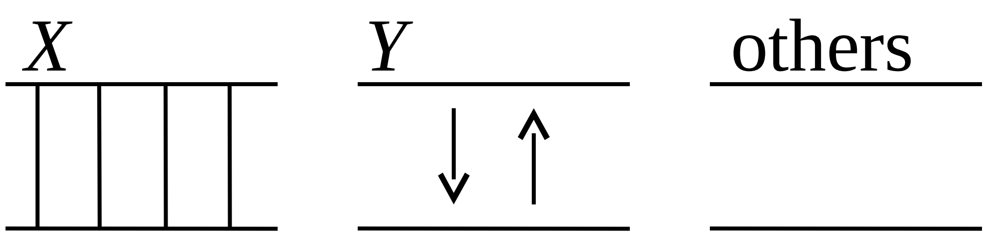
There are two important properties of a decomposition:

2. We should be able to recover from the decomposed relations the data of the original.
   - Recovery involves projection and join, which we shall defer until we've discussed relational algebra.
3. We should be able to check that the FD's for the original relation are satisfied by checking the projections of those FD's in the decomposed relations.

- Without proof, we assert that it is always possible to decompose into BCNF and satisfy (1).
- Also without proof, we can decompose into 3NF and satisfy both (1) and (2).
- But it is not possible to decompose into BNCF and get both (1) and (2).
  - Street-city-zip is an example of this point.

Multivalued Dependencies

The *multivalued dependency* $X \rightarrow\rightarrow Y$ holds in a relation *R* if whenever we have two tuples of *R* that agree in all the attributes of *X*, then we can swap their *Y* components and get two new tuples that are also in *R*.



Example

`Drinkers(name, addr, phones, beersLiked)` with MVD `Name` $\to\to$ `phones`. If `Drinkers` has the two tuples:

name	addr	phones	beersLiked
sue	*a*	*p*1	*b*1
sue	*a*	*p*2	*b*2

it must also have the same tuples with `phones` components swapped:

name	addr	phones	beersLiked
sue	*a*	*p*2	*b*1
sue	*a*	*p*1	*b*2

Note: we must check this condition for *all* pairs of tuples that agree on `name`, not just one pair.

MVD Rules

1. Every FD is an MVD.
   - Because if $X \to Y$, then swapping Y's between tuples that agree on X doesn't create new tuples.
   - Example, in `Drinkers`: `name` $\to\to$ `addr`.
2. *Complementation*: if $X \to\to Y$, then $X \to\to Z$, where Z is all attributes not in X or Y.
   - Example: since `name` $\to\to$ `phones` holds in `Drinkers`, so does `name` $\to\to$ `addr beersLiked`.

Splitting Doesn't Hold

Sometimes you need to have several attributes on the right of an MVD. For example:

`Drinkers(name, areaCode, phones, beersLiked, beerManf)`

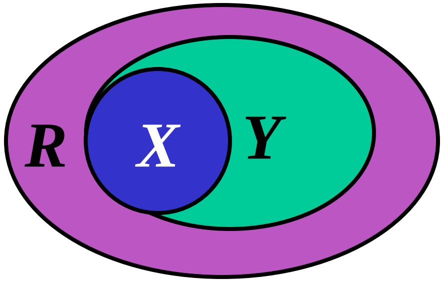
name	areaCode	phones	beersLiked	beerManf
Sue	831	555-1111	Bud	A.B.
Sue	831	555-1111	Wicked Ale	Pete's
Sue	408	555-9999	Bud	A.B.
Sue	408	555-9999	Wicked Ale	Pete's

- `name` $\rightarrow\rightarrow$ `areaCode phones` holds, but neither `name` $\rightarrow\rightarrow$ `areaCode` nor `name` $\rightarrow\rightarrow$ `phones` do.

4NF

Eliminate redundancy due to multiplicative effect of MVD's.

- Roughly: treat MVD's as FD's for decomposition, but not for finding keys.
- Formally: R is in Fourth Normal Form if whenever MVD $X \rightarrow\rightarrow Y$ is *nontrivial* (Y is not a subset of X, and $X \cup Y$ is not all attributes), then X is a superkey.
  - Remember, $X \rightarrow Y$ implies $X \rightarrow\rightarrow Y$, so 4NF is more stringent than BCNF.
- Decompose R, using 4NF violation $X \rightarrow\rightarrow Y$, into XY and $X \cup (R—Y)$.



Example

`Drinkers(name, addr, phones, beersLiked)`

- FD: `name` $\rightarrow$ `addr`
- Nontrivial MVD's: `name` $\rightarrow\rightarrow$ `phones` and `name` $\rightarrow\rightarrow$ `beersLiked`.
- Only key: {`name, phones, beersLiked`}
- All three dependencies above violate 4NF.
- Successive decomposition yields 4NF relations:

  D1(<u>name</u>, addr)

  D2(<u>name</u>, <u>phones</u>)

  D3(<u>name</u>, <u>beersLiked</u>)