Introduction to Database Systems

Normalization

Werner Nutt

Normalization

1. Anomalies

1. **Anomalies**
2. Boyce-Codd Normal Form
3. 3rd Normal Form

Anomalies

The 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* : a valid fact is lost
  when a tuple is deleted

Example of Bad Design

Drinkers(name, addr, beersLiked, 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 FDs

- name $\rightarrow$ addr favBeer
- beersLiked $\rightarrow$ 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.

Normalization

2. Boyce-Codd Normal Form

1. Anomalies
2. **Boyce-Codd Normal Form**
3. 3rd Normal Form

Boyce-Codd Normal Form

A relation R is in Boyce-Codd Normal Form (*BCNF*) if whenever $X \rightarrow A$ is a *nontrivial* FD that holds in R, then X is a *superkey*

Remember:

- *nontrivial* means $A \notin X$
- a *superkey* is any superset of a key (not necessarily a strict superset)

"*Each attribute must describe the key, the whole key, and nothing but the key*"

Example

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

FDs: name $\rightarrow$ addr favBeer, beersLiked $\rightarrow$ manf

- The only key is {name, beersLiked}
- In each FD above, the left side is *not* a superkey

$\Rightarrow$ Any one of these FDs shows Drinkers is not in BCNF

Each of the above FDs is a partial dependency,
i.e., the right side depends only on a *part of the key*

Another Example

Beers(name, manf, manfAddr)

FD's: name $\rightarrow$ manf, manf $\rightarrow$ manfAddr

- The only key is {name}
- name $\rightarrow$ manf does not violate BCNF,
- … but manf $\rightarrow$ manfAddr does

The second FD is a transitive dependency, because manfAddr depend on manf, which is not part of any key

Decomposition into BCNF

Given: relation R with FDs $\mathcal{F}$

Goal: decompose R into relations $R_1,\ldots,R_m$ such that

- each R_i is a *projection* of R
- each R_i is in *BCNF*
  (wrt the projection of $\mathcal{F}$)
- R is the *natural join* of $R_1,\ldots,R_m$

Intuition: R is broken into pieces

- that contain the same information as R,
- but are free of redundancy

The Algorithm: Divide and Conquer

- Look in $\mathcal{F}$ for an FD $X \to B$ that violates BCNF

  (If any FD following from $\mathcal{F}$ violates BCNF, then there is surely an FD in $\mathcal{F}$ itself that violates BCNF)

- Compute X^+

  (X^+ does not contain all attributes of R,

  otherwise X would be superkey)

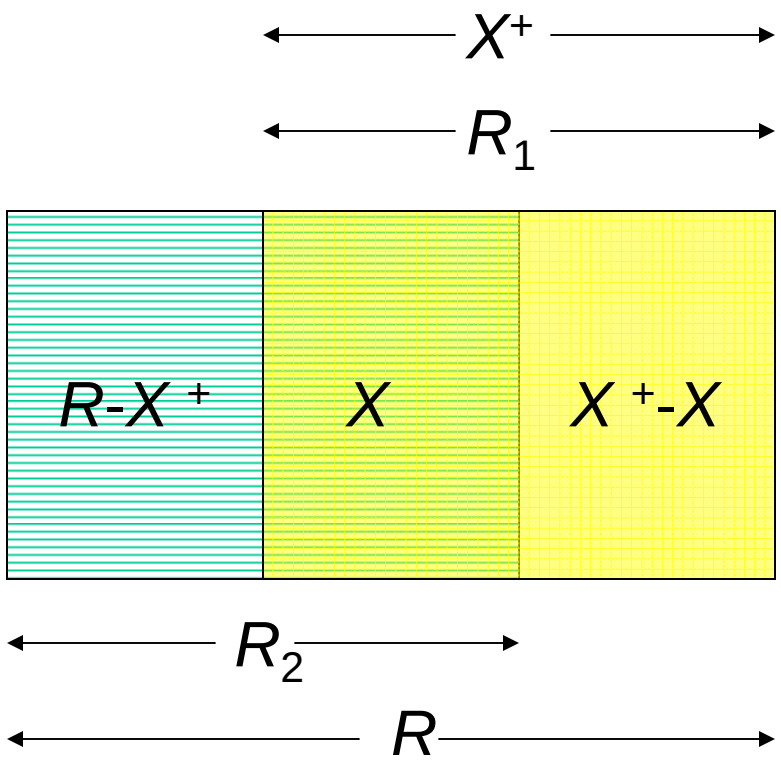
- Decompose R using $X \to B$, i.e., replace R by relations with schemas

  $R_1 = X^+$

  $R_2 = (R - X^+) \cup X$

- Compute the projections $\mathcal{F}_1$, $\mathcal{F}_2$ of $\mathcal{F}$ on R_1, R_2
- Continue with R_1, $\mathcal{F}_1$ and R_2, $\mathcal{F}_2$

Decomposition Picture



Example

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

$\mathcal{F}$ = {name $\rightarrow$ addr, name $\rightarrow$ favBeer, beersLiked $\rightarrow$ manf}

- Pick the BCNF violation name $\rightarrow$ addr
- Close the left side: {name}$^+$ = {name, addr, favBeer}
- Decomposed relations:

  Drinkers1(name, addr, favBeer)

  Drinkers2(name, beersLiked, manf)

Example (cntd)

Projecting FDs:

- For Drinkers1(name, addr, favBeer), the only relevant FDs are name $\rightarrow$ addr and name $\rightarrow$ favBeer:
  $\Rightarrow$ the only key is {name}
  $\Rightarrow$ Drinkers1 is in BCNF

- For Drinkers2(name, beersLiked, manf), the only relevant FD is beersLiked $\rightarrow$ manf:
  $\Rightarrow$ the only key is {name, beersLiked}
  $\Rightarrow$ violation of BCNF (because there is partial dependency)

- Continue with Drinkers2

Example (cntd)

Drinkers2(name, beersLiked, manf)

$\mathcal{F}_2$ = {beersLiked $\rightarrow$ manf}

- Pick the BCNF violation beersLiked $\rightarrow$ manf
- Close the left side: {beersLiked}$^+$ = {beersLiked, manf}
- Decomposed relations:

  Drinkers3(beersLiked, manf)

  Drinkers4(name, beersLiked)

Example (cntd)

Projecting FDs:

- For Drinkers3(beersLiked, manf), the only relevant FD is beersLiked $\rightarrow$ manf:
  $\Rightarrow$ the only key is {beersLiked}
  $\Rightarrow$ Drinkers3 is in BCNF

- For Drinkers4(name, beersLiked), there is no relevant FD:
  $\Rightarrow$ the only key is {name, beersLiked}
  $\Rightarrow$ Drinkers4 is in BCNF

Example (concluded)

We have decomposed

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

into

- Drinkers1(name, addr, favBeer)
- Drinkers3(beersLiked, manf)
- Drinkers4(name, beersLiked)

Notice:

- Drinkers1 is about drinkers
- Drinkers3 is about beers
- Drinkers4 is about the relationship between drinkers and the beers they like

Normalization

3. 3rd Normal Form

1. Anomalies
2. Boyce-Codd Normal Form
3. **3rd Normal Form**

Third Normal Form — Motivation

There is one structure of FDs that causes trouble when we decompose:

- $AB \rightarrow C$ and $C \rightarrow B$

  Examples:

  – A = street address, B = city, C = zip code

  – A = lecturer, B = hour, C = course

- There are two keys, $\{ A, B \}$ and $\{ A, C \}$
- $C \rightarrow B$ is a BCNF violation, so we must decompose into

  – AC

  – BC

We Cannot Enforce FDs

If we decompose *ABC* into *AC* and *BC*,
then we cannot enforce $AB \rightarrow C$
by checking FDs in the decomposed relations

Example with *A* = street,
B = city
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 FDs were violated in the decomposed relations, FD street city $\rightarrow$ zip is violated by the database as a whole

3NF Lets 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

- $X \rightarrow A$ violates 3NF if and only if
  - X is *not a superkey*
  - and also A is *not prime*

Back to the *ABC* Example

On ABC, we have FDs $AB \rightarrow C$ and $C \rightarrow B$

$\Rightarrow$ There are two keys: AB and AC

$\Rightarrow$ A, B, and C are each prime

$\Rightarrow$ Although $C \rightarrow B$ violates BCNF,
it does not violate 3NF

What 3NF and BCNF Give You

There are two important properties of a decomposition:

- *Losslessness*: It should be possible
  - to project the original relation onto the decomposed schema
  - and then reconstruct the original by a natural join
- *Dependency Preservation*: It should be possible to check in the projected relations whether all the given FDs 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

Exercise

Consider the relation PI (= passengerInfo) with the attributes

PI(FlightNo, Date, DepartureTime, SeatNo, TicketNo,
Name, Address, LuggageId, Weight)

and the FDs

- FlightNo, Date $\rightarrow$ DepartureTime
- FlightNo, Date, TicketNo $\rightarrow$ SeatNo
- TicketNo $\rightarrow$ Name Address
- LuggageId $\rightarrow$ Weight Date TicketNo

- Is the relation in Boyce-Codd normal form?
- If not, decompose into relation that are in BCNF. Is the resulting decomposition dependency preserving?

Exercise

Let R be a relation with attributes *ABCD*. Consider the following combinations of FDs on R:

- $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$
- $B \rightarrow C$, $B \rightarrow D$
- $AB \rightarrow C$, $BC \rightarrow D$, $CD \rightarrow A$, $AD \rightarrow B$
- $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow A$

For each collection of FDs do the following:

1. Indicate all the BCNF violations
2. Decompose the relations into collections of relations that are in BCNF
3. Are the decompositions dependency preserving?

References

In preparing the lectures I have used several sources.
The main ones are the following:

Books:

- A First Course in Database Systems, by J. Ullman and J. Widom
- Fundamentals of Database Systems, by R. Elmasri and S. Navathe

Slides:

- The slides of this chapter are based on slides prepared by Jeff Ullman for his introductory course on database systems at Stanford University