Functional Dependencies and Schema Refinement II

Alvin Cheung
Aditya Parameswaran
R&G 19

Berkeley cs186

Thanks for that…

- So we know a lot about FDs
- So what?

- Can they help with removing redundancy, update and deletion anomalies?
- Yes! We use **normalization** to cast schemas into **Normal Forms** (aka good schemas)

Normal Forms

First Normal Form = all attributes are atomic
Second Normal Form (2NF) = old and obsolete
Third Normal Form (3NF) = rarely preferred over BCNF
Fourth Normal Form (4NF) = unnecessary/complex

Boyce Codd Normal Form (BCNF)

Boyce-Codd Normal Form

A simple condition for removing redundancy/anomalies from relations:

A relation R is in BCNF if and only if:

Whenever there is a nontrivial FD: $A_1A_2\ldots A_n \rightarrow B$,
then $A_1A_2\ldots A_n$ is a super-key for R.

- Non-trivial means RHS is not a subset of LHS
- "Whenever a set of attributes of R is determining another attribute, it should determine all attributes of R."

Why does this make sense?

Say R(A, B, C) with AB as the key has an FD: $A \rightarrow C$.
Then C is being repeated for multiple Bs

Example

Name	SSN	Phone Number
Jia	123-32-9931	(201) 555-1234
Jia	123-32-9931	(206) 572-4312
Marco	909-43-4486	(908) 464-0028
Marco	909-43-4486	(212) 555-4000

What are the dependencies?

SSN ➔ Name

Is the left side a superkey?

No

Is it in BCNF?

No.

Decompose it into BCNF

SSN	Name
123-32-9931	Jia
909-43-4486	Marco

SSN	Phone Number
123-32-9931	(201) 555-1234
123-32-9931	(206) 572-4312
909-43-4486	(908) 464-0028
909-43-4486	(212) 555-4000

SSN $\rightarrow$ Name

Now is it in BCNF?

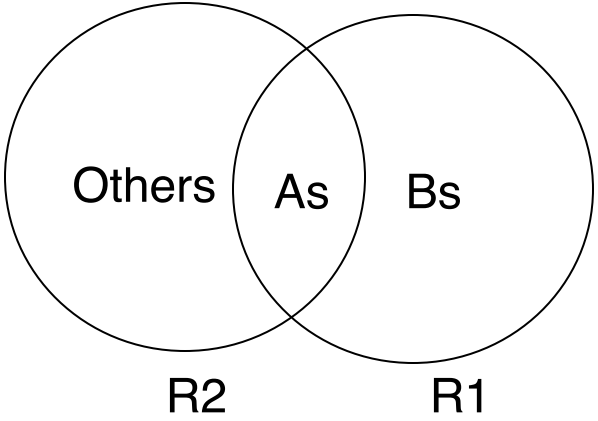
BCNF Decomposition

Find a dependency that violates the BCNF condition:

$$A_1, A_2 \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$$

Heuristic : choose $B_1, B_2, \ldots B_m$ "as large as possible" helps avoid unnecessarily fine-grained decomposition

Decompose:



Continue until there are no BCNF violations left.

Example Decomposition

Person:

Name	SSN	Age	EyeColor	PhoneNum

Functional dependencies:

SSN ➔ Name, Age, Eye Color

BCNF: Person1 (SSN, Name, Age, EyeColor),
Person2 (SSN, PhoneNum)

Example

Same example, slightly more complex.

Person (Name, SSN, Age, EyeColor, PhoneNum, Draftworthy)

- FD 1: SSN → Name, Age, EyeColor
- FD 2: Age → Draftworthy

Example

- Person (Name, SSN, Age, EyeColor, PhoneNum, Draftworthy)
- FD 1: SSN → Name, Age, EyeColor
- FD 2: Age →Draftworthy

- FD 1 and 2 imply SSN → Name, Age, EyeColor, Draftworthy
- Split based on this
  - (SSN, Name, Age, EyeColor, Draftworthy)
  - (SSN, PhoneNum)
- Split based on Age → Draftworthy
  - (SSN, Name, Age, EyeColor)
  - (Age, Draftworthy)
  - (SSN, Phone Number)
- Will get same result if you apply in a different order (but not always!)

Example

- Movie (Title, Yr, StudioName, President, PresAddr)
- FD: Title, Yr ➔ StudioName
- FD: StudioName ➔ President
- FD: President ➔ PresAddr

Example

- Movie (Title, Yr, StudioName, President, PresAddr)
- FD: Title, Yr ➔ StudioName
- FD: StudioName ➔ President
- FD: President ➔ PresAddr

(Title, Yr, StudioName, President)

(President, PresAddr)

➔

(Title, Yr, StudioName)

(StudioName, President)

(President, PresAddr)

Two-attribute relations

- Let A and B be the only two attributes of R
- Claim: R is in BCNF.
- Symmetric cases:
  - If A ➔ B is true, B ➔ A is not:
  - If B ➔ A is true, A ➔ B is not:
- If A ➔ B is true, B ➔ A is true:

Two-attribute relations

- Let A and B be the only two attributes of R
- Claim: R is in BCNF.
- If A ➔ B is true, B ➔ A is not:
  - A → B does not violate BCNF
- If B ➔ A is true, A ➔ B is not:
  - Symmetric
- If A ➔ B is true, B ➔ A is true:
  - Both are keys, therefore neither violate BCNF

BCNF Decomposition: The Algorithm

Input: relation R, set S of FDs over R

1) Check if R is in BCNF, if not:

    a) pick a violation FD f: A ➔ B

    b) compute A+

    c) create R1 = A+, R2 = A union (R – A+)

    d) compute all FDs over R1 and R2, using R and S.

    e) repeat Step 1 for R1 and R2

2) Stop when all relations are BCNF or are two attributes

(Remember, two attribute relations are always in BCNF)

Q: Is BCNF Decomposition unique?

- R(SSN, netid, phone).
- FD1: SSN -> netid
- FD2: netid -> SSN
- Each of these two FDs violates BCNF.

Can you tell me two different BCNF decomp for R?

- Pick FD1 and decompose, you get:
  - (SSN, netid); (SSN, phone).
- Pick FD2 and you get
  - (netid, SSN); (netid, phone).

Properties of BCNF

- BCNF removes certain types of redundancies
  - for examples of redundancy that it cannot remove, see "multi-valued redundancy” (Addressed by 4NF, see textbook)
- BCNF decomposition **avoids information loss**
  - You can construct the original relation instance from the decomposed relations’ instances.
  - How? What would the relational algebra exp look like?
    - R(A, B, C) from R(A, B), R(B, C)
  - Ans: Natural join
  - Proof: in the textbook

Can we cheat?

- We saw that two-attribute relations are in BCNF.
- Why don’t we break any R(A,B,C,D,E) into R1(A,B); R2(B,C); R3(C,D); R4(D,E)? Why bother with finding BCNF violations etc.?
- Turns out, this leads to information loss …

Example of the "easy decomposition"

- R = (A,B,C); decomposed into R1(A,B); R2(B,C)

A	B	C
1	2	3
4	2	6

A	B
1	2
4	2

B	C
2	3
2	6

Example of the "easy decomposition"

- R = (A,B,C); decomposed into R1(A,B); R2(B,C)

A	B	C
1	2	3
4	2	6

A	B
1	2
4	2

B	C
2	3
2	6

Nat.Join

A	B	C
1	2	3
4	2	6
1	2	6
4	2	3

We get back some "bogus tuples"!
Lossless decompositions (like BCNF) don't give bogus tuples.

Summary of Schema Refinement

- BCNF: each field contains data that cannot be inferred via FDs.
  - ensuring BCNF is a good heuristic.
- Not in BCNF? Try decomposing into BCNF relations.
- Downside of BCNF: not all dependencies are preserved (some are split across relations)
  - TINSTAFL! If you want to preserve dependencies, you will have redundancy
  - Take a look at the textbook for these tradeoffs