CPSC 304 – Administrative notes September 27 & October 1, 2024

- October 1: Milestone 1 due
- October 15: Milestone 2 due
- October 22: Midterm @ 6PM
  - Do you have a conflict? If so fill out the midterm conflict form that has been posted on Piazza by the end of Monday, October 7
  - More information on Piazza ~1 week before the midterm.

Now where were we...

- We'd been discussing normalization
- Why redundancy is bad
- What are some ways we know our data is redundant
- And how knowing all the Functional Dependencies that are relevant to a relation is helpful
- In particular, we'd talked about how to take the closure of a set of functional dependencies
  - e.g., given a relation S(A,B,C) and a set of FDs
    $A \rightarrow B$
    $B \rightarrow C$
    The interesting closures are
  - $A+ = \{A,B,C\}$
    $B+ = \{B,C\}$

Let's take a minute to appreciate what closures give us

- Consider relation R(A,B,C,D) with FDs
  $AB \rightarrow C$
  $C \rightarrow D$
- Closures give us:
  $AB^+ = \{A,B,C,D\}$
  $C^+ = \{C,D\}$
- This tells us all the FDs, both explicit and implied:
  $AB \rightarrow A$
  $AB \rightarrow B$
  $AB \rightarrow C$
  $AB \rightarrow D$
  $C \rightarrow C$
  $C \rightarrow D$
- But how can we be sure that we've found all the keys (I promise you we will need this later) without resorting to Armstrong's axioms?

Finding All the (minimal) Keys

Find all the (minimal) keys for R(ABCD) where
AB → C and C → BD.

Step 1: List all the attributes that **only** appear on the RHS of the FDs (i.e., things that can only be derived). Put on right:

Left	Middle	Right
		D

Step 2: List all the attributes that **only** ever appear on the LHS of a FD (or do not appear in a FD at all (i.e., things that have to be part of any key). Put on left:

Left	Middle	Right
A		D

Finding All the (minimal) Keys

Find all the (minimal) keys for R(ABCD) where
AB → C and C → BD.

Step 3: List all the attributes that appear on the LHS and RHS of the FDs (i.e., things that are not obviously required and may help you derive new things). Put in middle.

Left	Middle	Right
A	BC	D

Step 4: Take the closure of the attributes in the left column.

$\{A\}^+ = \{A\}$

Are all of the attributes there? If so, you have found a (minimal) key. If not, start adding in attributes from the middle column to see if you can determine all the attributes of the relation.

Finding All the (minimal) Keys

Find all the (minimal) keys for R(ABCD) where
AB → C and C → BD.

Left	Middle	Right
A	BC	D
Need	Maybe need	Don't need (can derive)

$\{AB\}^+ = \{ABCD\}$ ← (minimal) key

$\{AC\}^+ = \{ACBD\}$ ← (minimal) key

- What if the relation schema had attribute E? R(ABCDE)

Exercise: Find all Keys

Find all the (minimal) keys of the Relation R(ABCDE) with FD's:

$D \rightarrow C$

$CE \rightarrow A$,

$D \rightarrow A$

$AE \rightarrow D$.

Clicker Exercise: Finding Keys

Which of the following is a (minimal) key of the Relation R(ABCDE) with FD's:

$D \rightarrow C$

$CE \rightarrow A$,

$D \rightarrow A$

$AE \rightarrow D$.

A. ABDE
B. BCE
C. CDE
D. All of these are keys
E. None of these are keys

Clicker Exercise: Finding Keys

Which of the following is a (minimal) key of the Relation R(ABCDE) with FD's:

$D \rightarrow C$

$CE \rightarrow A$,

$D \rightarrow A$

$AE \rightarrow D$.

A. ABDE Superkey, since D→A

B. BCE Key

C. CDE CDE+=CDEA

D. All of these are keys

E. None of these are keys

Exercise: Find all Keys

Find all the (minimal) keys of the Relation R(ABCDE) with FD's:

$D \rightarrow C$

$CE \rightarrow A$,

$D \rightarrow A$

$AE \rightarrow D$.

Left	Middle	Right
BE	ACD	

$BE^+ = BE$

$ABE^+ = ABEDC$

$BCE^+ = BCEAD$

$BDE^+ = BDECA$

Popping back up to our original question…

- Is this really a good design for a table?

Name	Department	Mailing Location
Jessica Wong	Computer Science	201-2366 Main Mall
Rachel Pottinger	Computer Science	201-2366 Main Mall
Laks Lakshmanan	Computer Science	201-2366 Main Mall
Joel Friedman	Computer Science	201-2366 Main Mall
Joel Friedman	Math	121-1984 Mathematics Rd
Mark MacLean	Math	121-1984 Mathematics Rd

- Wouldn’t it be nice if there was some rule that said if the amount of redundancy that we had was good?

Approaching Normality

- Role of FDs in detecting redundancy:
  - Consider a relation R with 3 attributes, A B C.
    - No FDs hold: There is no redundancy here.
    - Given A → B: Several tuples could have the same A value, and if so, they'll all have the same B value!
- ***Normalization***: the process of removing redundancy from data

Normal Forms: Why have one rule when you can have four?

- Provide guidance for table refinement/reducing redundancy.
- Four important normal forms:
  - ***First normal form(1NF)***
  - ***Second normal form (2NF)***
  - ***Boyce-Codd Normal Form (BCNF)***
  - ***Third normal form (3NF)***
- If a relation is in a certain *normal form*, certain problems are avoided/minimized.
- Normal forms can help decide whether decomposition (i.e., splitting tables) will help.

1NF

- Each attribute in a tuple has only one value
  - E.g., for “postal code” you can’t have both V6T 1Z4 and V6S 1W6
- Why do we need 1NF?
  - because Codd’s original vision of the relational model allowed multi-valued attributes…

1NF

Client ID	Postal Code
1	V6T 1Z4, V6S 1W6

Can’t have multiple values in a single attribute

Normalize to 1NF

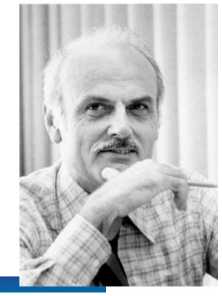
Client ID	Postal Code
1	V6T 1Z4
1	V6S 1W6

2NF

- No partial key dependency
- A relation is in 2NF, if it is in 1NF and for every FD X→Y where X is a (minimal) key and Y is a non-key attribute, then no proper subset of X determines Y
- e.g., the address relation is not in 2NF:
  - House#, street, postal_code is a (minimal) key
  - $\overbrace{\text{House\#, street, postal\_code}}^{X} \rightarrow \overbrace{\text{Province}}^{Y}$
  - $\overbrace{\text{Postal\_code}}^{\text{Subset of } X} \rightarrow \overbrace{\text{Province}}^{Y}$

Not in 2NF

Boyce-Codd Normal Form (BCNF)



A relation R is in BCNF if:

If X → b is a non-trivial dependency in R,

then X is a superkey for R

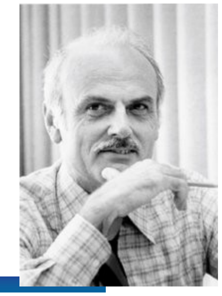
(Must be true for every such dependency)

Recall: A dependency is trivial if the LHS contains the RHS, e.g., City, Province→ City is a trivial dependency

In English (though a bit vague):

Whenever a set of attributes of *R* determine another attribute, it should determine ***<u>all</u>*** the attributes of *R*.

Boyce-Codd Normal Form (BCNF)



Ex: Address(House#, Street, City, Province, PostalCode), so

- House#, Street, PostalCode → City
- House#, Street, PostalCode → Province
- PostalCode → City
- PostalCode → Province

Is Address in BCNF? Why or why not?

A. Yes
B. No

What do we want? Guaranteed freedom from redundancy!

- A relation may be BCNF already!
- Helpful fact: all two attribute relations are in BCNF. Consider relation S(A,B). There are four cases:
  - A→B and B→A
  - Only A→B
  - Only B→A
  - No FDs
- Lets look at each one

Is the two attribute relation $S_1(A,B)$ where $A \rightarrow B$ and $B \rightarrow A$ in BCNF?

- What are the keys?
  A. {A}
  B. {B}
  C. {AB}
  D. {A} and {B}
- BCNF rule: It must be that if $X \rightarrow b$ is a non-trivial dependency, then X is a superkey
- Is A a superkey of S_1?
  - Yes! $A \rightarrow B$ doesn't violate BCNF
- Is B a super key of S_1?
  - Yes! $B \rightarrow A$ doesn't violate BCNF
- Are there any other non-trivial dependencies? No! Therefore S_1 is in BNCF

Is the two attribute relation $S_2(A,B)$ where (only) $A \rightarrow B$ in BCNF?

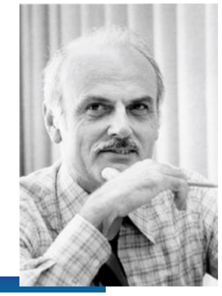
- What are the keys?
  A. A
  B. B
  C. AB
  D. A and B
- BCNF rule: It must be that if $X \rightarrow b$ is a non-trivial dependency, then X is a superkey
- Is A a superkey of S_2?
  - Yes! $A \rightarrow B$ doesn't violate BCNF
- Are there any other non-trivial dependencies?
  - No! Therefore S_2 is in BNCF

The same reasoning holds if only $B \rightarrow A$

The final case: $S_3(A,B)$, and no non-trivial functional dependencies

- If there are no non-trivial functional dependencies
- S_3 is in BCNF
- There are no non-trivial functional dependencies to violate BCNF

Reminder Boyce-Codd Normal Form (BCNF)



A relation R is in BCNF if:

If X → b is a non-trivial dependency in R,

then X is a superkey for R

(Must be true for every such dependency)

Recall: A dependency is trivial if the LHS contains the RHS, e.g., City, Province→ City is a trivial dependency

In English (though a bit vague):

Whenever a set of attributes of *R* determine another attribute, it should determine ***all*** the attributes of *R*.

What if a relation is not in BCNF? Decomposing a Relation

"Shhhhh!... the Maestro is decomposing!"

- A *decomposition* of R replaces R by two or more relations s.t.:
  - Each new relation contains a subset of the attributes of R (and no attributes not appearing in R), and
  - Every attribute of R appears in at least one new relation.
- Intuitively, decomposing R means storing instances of the relations produced by the decomposition, instead of instances of R.
- E.g., Address(House#,Street,City,Province,Postal Code)
  - Address(House#,Street#,PostalCode),
  - PC(City, Province, PostalCode)

How can we decompose correctly?

A sneak preview: The join

- Definition: $R_1 \bowtie R_2$ is the (natural) join of the two relations; i.e., each tuple of R_1 is concatenated with every tuple in R_2 having the same values on the common attributes.

R_1

A	B
1	2
4	5
7	2

R_2

B	C
2	3
5	6
2	8

$R_1 \bowtie R_2$ →

A	B	C
1	2	3
1	2	8
4	5	6
7	2	3
7	2	8

Lossless-Join Decompositions: Definition

Informally: If we break a relation, R, into bits, when we put the bits back together, we should get exactly R back again

Formally: Decomposition of R into X and Y is lossless-join w.r.t. a set of FDs F if, for every instance r that satisfies F:

  - If we JOIN the X-part of r with the Y-part of r the result is exactly r

- It is *always* true that r is a subset of the JOIN of its X-part and Y-part, even if the join isn't lossless
- In general, the other direction does not hold – you may get back additional information! If it does hold, the decomposition is a lossless-join.
- Note: The word loss in lossless refers to loss of information, not to loss of tuples. In fact, for "loss of information" a better way to refer to it might be "addition of spurious information".

Example Lossy-Join Decomposition

A	B	C
1	2	3
4	5	6
7	2	8

decompose

A	B
1	2
4	5
7	2

B	C
2	3
5	6
2	8

join

A	B	C
1	2	3
4	5	6
7	2	8
1	2	8
7	2	3

Last two rows are not in the original.

All decompositions used to resolve redundancy *must* be lossless!

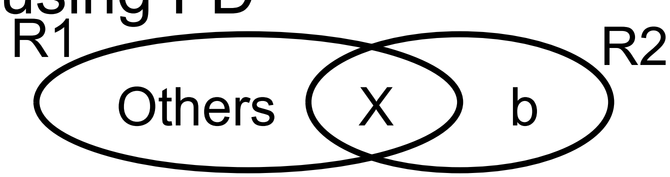
How do we decompose into BCNF losslessly?

- Let R be a relation with attributes A, and FD be a set of FDs on R s.t. all FDs determine a single attribute

1. Pick any $f \in$ FD that violates BCNF of the form $X \rightarrow b$
2. Decompose R into two relations: $R_1(A\text{-}b)$ & $R_2(X \cup b)$

- Recurse on R_1 and R_2 using FD

Pictorally:



Note: answer may vary depending on order you choose. That's okay

BCNF Example (Let's do the first one together)

Remember BCNF def: For all non-trivial functional dependencies $X \rightarrow b$, X must be a superkey for a relation to be in BCNF

Relation: R(ABCD)   FD: $B \rightarrow C$, $D \rightarrow A$

Keys?

- $B^+ = \{B,C\}$ [$B \rightarrow B$, $B \rightarrow C$] ← What the closure tells us
- $D^+ = \{A,D\}$ [$D \rightarrow A$, $D \rightarrow D$] ← What the closure tells us
- $BD^+ = \{B,D,C,A\}$
- BD is the only key

BCNF Example (Let's do the first one together)

Remember BCNF def: For all non-trivial functional dependencies $X \rightarrow b$, X must be a superkey for a relation to be in BCNF

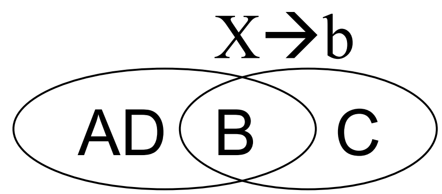
Relation: R(ABCD) FD: $B \rightarrow C$, $D \rightarrow A$

Keys?

- $A^+ = \{A\}$
- $B^+ = \{B,C\}$
- $C^+ = \{C\}$
- $D^+ = \{A,D\}$
- $BD^+ = \{B,D,C,A\}$
- BD is the only key

Look at FD $B \rightarrow C$. Is B a superkey?

- No. Decompose
- R1(B,C), R2(A,B,D)



BCNF Example (Let's do the first one together)

Remember BCNF def: For all non-trivial functional dependencies X→b, X must be a superkey for a relation to be in BCNF

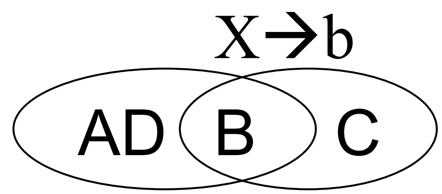
Relation: R(ABCD)   FD: B→C, D→A

Keys?

- $A^+ = \{A\}$
- $B^+ = \{B,C\}$
- $C^+ = \{C\}$
- $D^+ = \{A,D\}$
- $BD^+ = \{B,D,C,A\}$
- BD is the only key

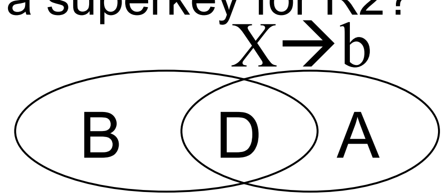
Look at FD B→ C. Is B a superkey?

- No. Decompose
- R1(B,C), R2(A,B,D)



Are R1 and R2 in BCNF? Look at FD D→ A. Is D a superkey for R2?

- No. Decompose
- R3(D,A), R4(D,B)



Final answer: R1(B,C), R3(D,A), R4(D,B)

What does this answer mean?

Another BCNF Example

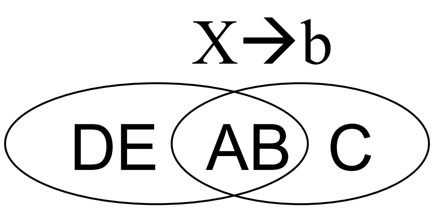
- R(ABCDE)
- FD: $AB \rightarrow C$, $D \rightarrow E$
- Find the closures

$AB^+ = \{A,B,C\}$ [$AB \rightarrow A$, $AB \rightarrow B$, $AB \rightarrow C$]

$D^+ = \{D,E\}$ [$D \rightarrow D$, $D \rightarrow E$]

Another BCNF Example

- R(ABCDE)
- FD: AB→C, D→E
- Find the closures
  - $AB^+ = \{A,B,C\}$
  - $D^+ = \{D,E\}$
- So the relation is not in BCNF, so we need to decompose
  - R_1(ABC), R_2(ABDE)
    - AB is a key for R_1 so it is in BCNF
    - D is not a key for R_2 so not in BCNF



A dependency for decomposing the second time

- When you first decide to decompose, it's easy to see which FDs apply: all of them. Returning to the previous example:
  R(ABCDE) FD: AB→C,   D→E
  Clearly both FDs apply to R.
- But when you decompose to R_1(ABC), R_2(ABDE), how do we know which FDs apply to R_2?

Determining relevant FDs after decomposing

- Take the closure of the attributes using *all* FDs
- For an FD X→ b that holds in the original relation, if the decomposed relation S contains {X U b}, then the FD holds for S:
- For example. Consider the relation R(A, B, C, D, E) with functional dependencies AB→C, CD→E, C→D. Assume that you have decomposed to T(A,B,C) and S(A,B,D,E)
- Does AB→E hold on S?
  - First check if A, B and E are all in S? They are
  - Find AB^+= ABCDE ← This includes E
  - Then yes AB→ E does hold in S.
- Does AB→C hold? No ← S does not include C

Determining relevant FDs after decomposing The clicker version

- Take the closure of the attributes using *all* FDs
- For an FD X→b that holds in the original relation, if the decomposed relation S contains {X U b}, then the FD holds for S
  - For example: consider relation R(A,B,C,D,E) with functional dependencies AB → C, BC → D, CD → E, DE → A, and AE → B.
    Project these FD's onto the relation S(A,B,C,D).
- Which of the following hold in S?

A. A→B
B. AB→E
C. AE→B
D. BCD→A
E. None of the above

Determining relevant FDs after decomposing The clicker version

- Take the closure of the attributes using *all* FDs
- For an FD X→b, if the decomposed relation S contains {X U b}, then the FD holds for S:
- For example. Consider relation R(A,B,C,D,E) with functional dependencies AB → C, BC → D, CD → E, DE → A, and AE → B.

  Project these FD's onto the relation S(A,B,C,D).
- Which of the following hold in S?

A. A→B — A+=A

B. AB→E — AB+=ABCDE, but E is not in S

C. AE→B — AE+=ABCDE, but E is not in S

D. BCD→A — Yes. BCD+=ABCDE; all in S

E. None of the above

Keys after a decomposition

Now that we know what FDs hold, we can determine keys

Previous example:

R(A,B,C,D,E)

FD: AB→C, D→E

- We decomposed on AB→ C:

  R_1(<u>A,B,</u>C), R_2(A,B,D,E) (AB+ = {A,B,C}, therefore AB is a key of R_1)

- Then we decomposed on D→E

  R_3(<u>A,B,D</u>), R_4(<u>D</u>E) (D+={DE}, so D is a key of R_4. No other interesting closures, so key of R_3 is ABD)

- Final relations: R_1(<u>A,B</u>,C), R_3,(<u>A,B,D</u>), R_4(<u>D</u>, E)

What about foreign keys?

Previous example: R(A,B,C,D,E) FD: AB→C,   D→E

- Final relations were R_1(<u>A,B</u>,C), R_3,(<u>A,B,D</u>), R_4(<u>D</u>, E). What are the foreign keys?
- Basically, any time you decompose on a FD X→b, the "**others**" relation **references** X in the (X U b) relation. Assuming all attributes are ints:

```
CREATE TABLE R3 (
    A       INT,
    B       INT,
    D       INT,
    PRIMARY KEY (A, B, D)
    FOREIGN KEY (A,B) REFERENCES R1(A,B),
    FOREIGN KEY (D) REFERENCES R4(D)
```

Note: sometimes things get hard to follow when you decompose multiple times. Just do what makes sense.