Review of Last Lecture

- Problems with redundancy
  - Update anomaly and NULL
- Functional Dependency
  - How to use key and FD to decide whether a design is good
- How to convert bad into good: Decomposition
  - How to check whether a decomposition is lossless
  - How check for FD preserving
  - How to check for minimal redundancy
- Normalization and different normal forms
- BCNF: For every FD A -> B, A is a superkey

Today's Plan

- BCNF exercises
- How to decompose a table with redundancy into BCNF
  - Attribute closure
- BCNF decomposition exercises
- Problem with BCNF
  - Lossless and minimal redundancy
  - But not FD preserving
- The need for 3NF

BCNF: Example I

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$
  Key = $\{A\}$
- R is not in BCNF
- Decomposition $R_1 = (A, B)$, $R_2 = (B, C)$
  - R_1 and R_2 in BCNF
  - Lossless-join decomposition
  - Dependency preserving

BCNF: Example II

Drinkers(name, addr, beersLiked, 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

BCNF: Example III

Beers(name, manf, manfAddr)

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

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

Compute Attribute Closure

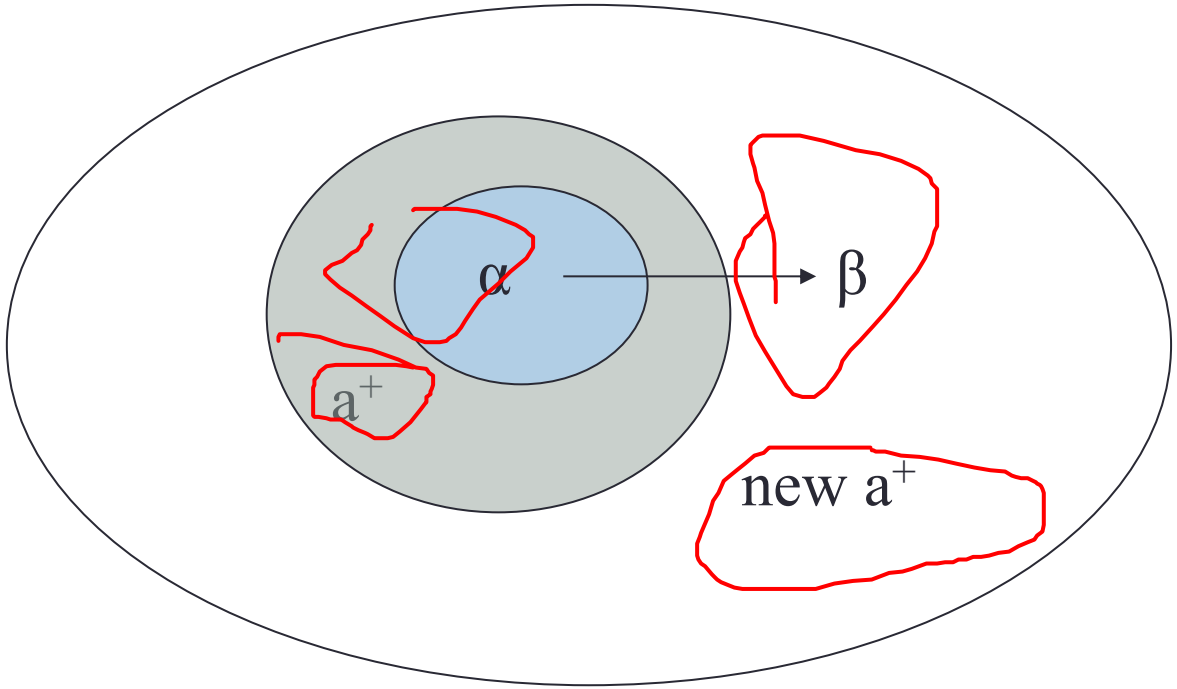
- Given a set of attributes α, define the *closure* of α under F (denoted by α^+) as the set of attributes that are functionally determined by α under F:

$$\alpha \rightarrow \beta \text{ is in } F^+ \Leftrightarrow \beta \subseteq \alpha^+$$

- Algorithm to compute α^+, the closure of α under F

```
result := α;
while (changes to result) do
        for each β → γ in F do
                begin
                        if β ⊆ result then result := result ∪ γ
                end
```

Compute Attribute Closure: Diagram



Attribute Closure: Example

- $\mathcal{F}$ includes:
  - $SID \rightarrow name, email$
  - $email \rightarrow SID$
  - $SID, CID \rightarrow grade$
- $\{ CID, email \}^+ = ?$

StudentGrade (*SID*, *name*, *email*, *CID*, *grade*)

{CID}$^+$=?

{CID, email}

- $email \rightarrow SID$
  - Add *SID*; closure is now { *CID*, *email*, *SID* }
- $SID \rightarrow name, email$
  - Add *name*, *email*; closure is now { *CID*, *email*, *SID*, *name* }
- $SID, CID \rightarrow grade$
  - Add *grade*; closure is now all the attributes in *StudentGrade*

Little Trick to Check Superkey

- If we know all the FD for a table, we can easily identify the SK
- Trick: if the closure of attribute(s) include all the attributes in the table, then it is a super key
- Example
  - (SID, CID)+ = (SID, name, email, CID, grade)
  - (email, CID)+ = (SID, name, email, CID, grade)

Closure Exercise

Let R = {ABCDE} and the set of FDs are:

FD1: AB → CD, FD2: C→ ED, FD3: AC → B

1. A^+ = ?
2. C^+ = ?
3. $(AC)^+$ = ?
4. What are all the candidate keys for R?
5. Is R in BCNF?

Lossless Decomposition into BCNF

- Given: relation R with FD's F.
- Find the candidate keys of R
  - A superkey is a superset of candidate keys
  - For example, if {A} is a CK, then {AB} is a SK
- Look among the given FD's for a BCNF violation $X \rightarrow B$.
- Compute X^+.
  - X^+ should not contain all attributes, or else X is a superkey.
- Replace R by relations with schemas:
  1. $R_1 = X^+$
  2. $R_2 = R - X^+ + (X)$
- *X is the PK in R_1 and FK in R2 between R_1 and R_2*
- *Project* given FD's F onto the two new relations.
- Review whether R1 and R2 are in BCNF based on the projected FDs.
- If not, repeat the steps for R1 and R2.

Example (1)

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

F = FD1: name → addr, FD2: name → favBeer,
FD3: beersLiked → manf

- CKs: (name, beersLiked)
- Pick BCNF violation name->addr.
- Closure of the left side: $(name)^+$ = (name, addr, favBeer)
- Decomposed relations:
  1. R1=Drinkers1(name, addr, favBeer)
  2. R2=Drinkers2(name, beersLiked, manf)

Example (2)

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

Example (3)

- 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(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)

- Notice: *Drinkers1* tells us about drinkers, *Drinkers3* tells us about beers, and *Drinkers4* tells us the relationship between drinkers and the beers they like.

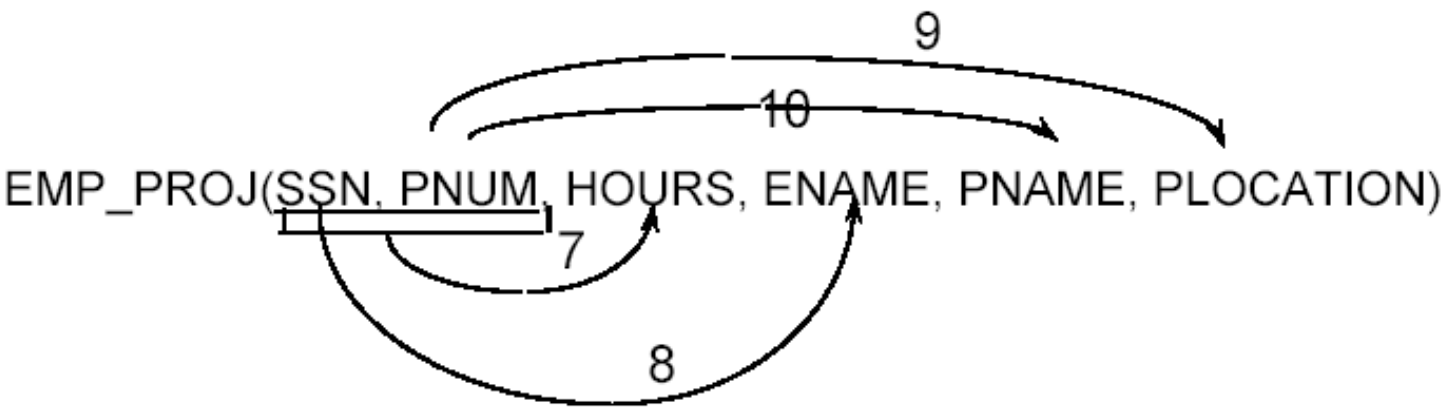
Question: lossless? FD preserving?

BCNF Decomposition: Another Example

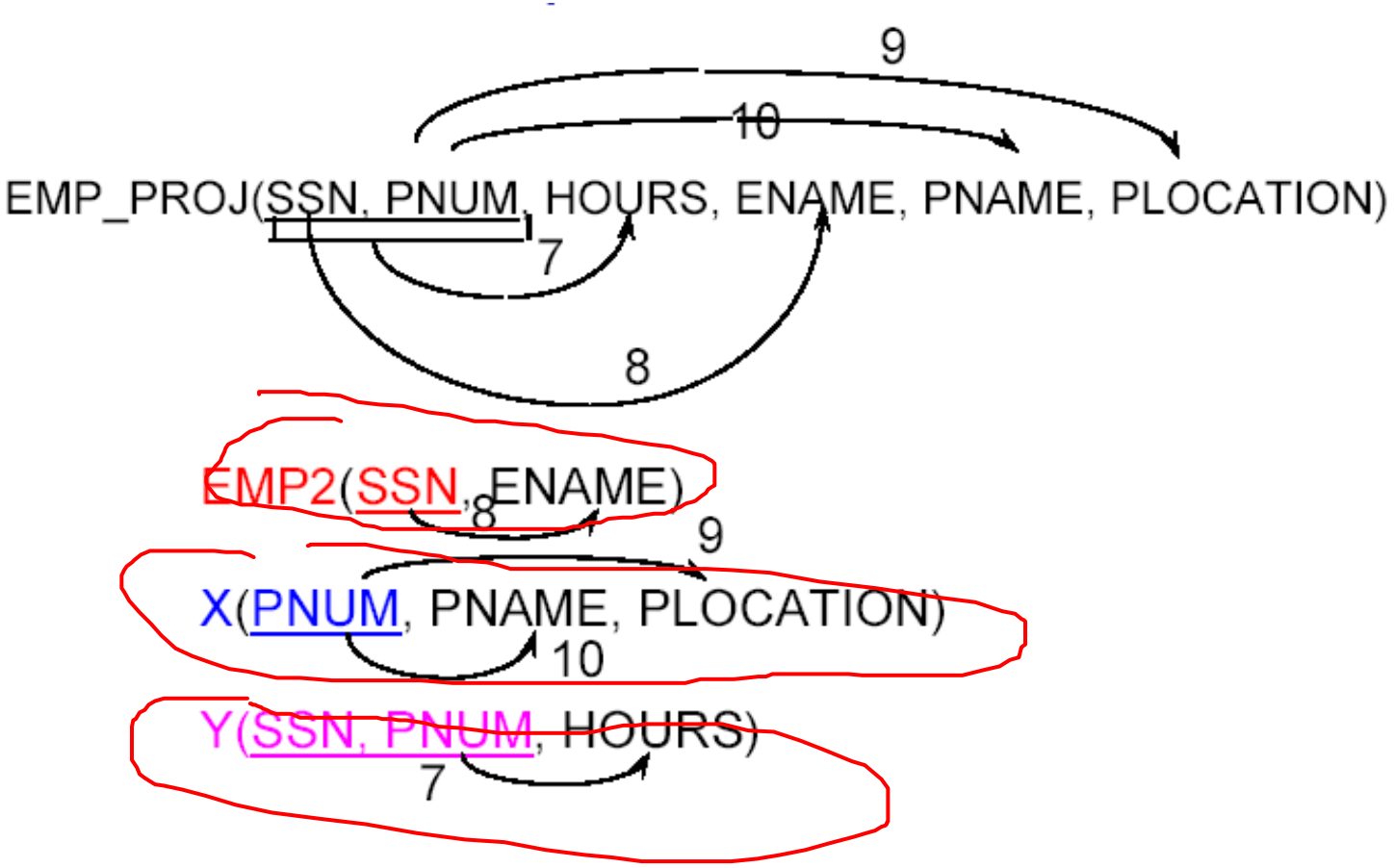
- *R* = *(branch-name, branch-city, assets,*
  customer-name, loan-number, amount)
  F = {branch-name → assets branch-city
  loan-number → amount branch-name}
  Key = *{loan-number, customer-name}*
- Decomposition
  - R_1 = *(branch-name, branch-city, assets)*
  - R_2 = *(branch-name, customer-name, loan-number, amount)*
  - R_3 = *(branch-name, loan-number, amount)*
  - R_4 = *(customer-name, loan-number)*
- Final decomposition
  R_1, R_3, R_4

Exercise: Decomposition Using FD

- Is the following relation in BCNF? If not, convert it into BCNF



Answer: Decomposition Using FD



What should we name these new tables?

Note

- BCNF decomposition is always lossless and minimizes redundancy
- But does not always preserve all FDs!!!

FD Preserving?

- There is one structure of FD's that causes trouble when we decompose.
- *R(SCZ)*, *SC* → *Z* and *Z* → *C*.
  - Example: *S* = street address, *C* = city, Z = zip code.
- There are two keys, {*S*,*C*} and {S,*Z* }.
- *Z* → *C* is a BCNF violation, so we must decompose R into R1(S*Z*), R2(*ZC*).

We Cannot Enforce FD's

- The problem is that if we use R1(*SZ*) and R2(*ZC*) as our database schema, we cannot enforce the FD $SC \rightarrow Z$ by simply checking FD's in these decomposed relations.
- Example with S = street address, C = city, and *Z* = zip Code on the next slide.

Example: 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 Us Avoid This Problem

- 3rd Normal Form (3NF) modifies the BCNF condition so we do not have to decompose in this problem situation.
- $X \rightarrow A$ violates 3NF if and only if X is not a superkey, and also A is not a key attribute.
  - In other words, for every FD in R: $X \rightarrow A$, if X is a either superkey or A is a key attribute, then R is in 3NF

This is fine in 3NF

- *R(SCZ),* SC $\rightarrow$ *Z* and *Z* $\rightarrow$ *C*.
  - Example: *S* = street address, *C* = city, *Z* = zip code.
- There are two keys, {*S*,C } and {*S*,*Z* }.
- *Z* $\rightarrow$ C is a NOT a 3NF violation, because C is a key attribute.
- So R is in 3NF, no need to do anything here.

Comparing 3NF and BCNF

- It is always possible to decompose a relation into relations in 3NF and
  - the decomposition is lossless
  - the dependencies are preserved

But not minimal redundancy!

- It is always possible to decompose a relation into relations in BCNF and
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

But minimal redundancy!

Which Form in Practice?

BCNF and Lossless and Minimal Redundancy

(first choice)

3NF and Lossless and Dependency-Preserving

(second choice)

because sometimes we can't preserve all dependencies

Assertion Exercise

- BCNF does not enforce (preserve) all FDs but we can enforce the missing FD at the application level using assertion.
- S(street, city), Z(city, zip)
- Write an assertion to enforce:

  street, city -> zip

General Steps of Normalization

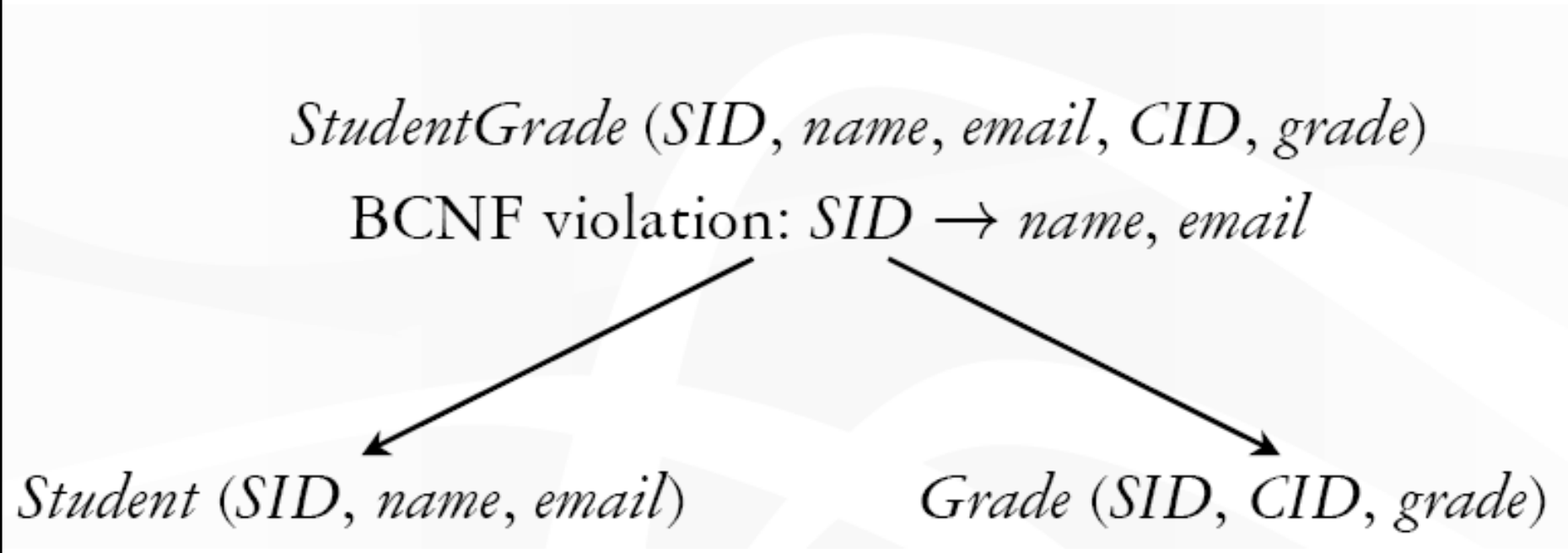
1. Come up with the attributes, tables, CKs and FKs
   - Use interviews, ERD, mapping
2. Figure out all the FDs for each table
   - User interviews
3. Determine what NF is needed
4. Check each FD and see whether it violates the desired NF
5. If no violation, hurrah!
6. If there is violation, normalize it using the given algorithm for the desired NF decomposition.

In-Class Exercise: BCNF Decomposition

StudentGrade(SID, name, email, CID, grade)

- CKs?
- FDs?
- BCNF?

BCNF Decomposition



In-Class Exercise: BCNF Decomposition

Let R = {ABCDE} and the set of FDs are:

FD1: AB → CD, FD2: C→ ED, FD3: AC → B

1. Is it in BCNF? Why or why not?
2. If not, decompose it into BCNF forms

Summary

- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Summary of Today's Lecture

- BCNF exercises
- How to decompose a table with redundancy into BCNF
  - Attribute closure
- BCNF decomposition exercises
- Problem with BCNF
  - Lossless and minimal redundancy
  - But not FD preserving
- The need for 3NF