LAB 6 NORMALIZATION

Summarized by Taylor Qin

Don’t distribute with others outside the tutorial!

Two properties

- Lossless join – " capture the same data "

R -> R1 and R2
R1 join R2 = R

- Dependency preservation – " capture the same meta-data "

R -> R1 and R2
FDs in R should be preserced in either R1 or R2 or derivable from the combinations of FDs from R1 and R2

LOSSLESS JOIN

- **Example 2:** The following decomposition from R into R_3 and R_4 doesn't have the lossless join property. **It generates spurious tuples.**

R		
Name	StudentID	DoB
Mike	123456	20/09/1989
Mike	123458	25/01/1988

SELECT * FROM R_3 NATURAL JOIN R_4		
Name	StudentID	DoB
Mike	123456	20/09/1989
Mike	123456	25/01/1988
Mike	123458	20/09/1989
Mike	123458	25/01/1988

R_3	
Name	StudentID
Mike	123456
Mike	123458

R_4	
Name	DoB
Mike	20/09/1989
Mike	25/01/1988

DEPENDENCY PRESERVATION

- **Example 2:** Given a FD {StudentID} $\rightarrow$ {Name} defined on R

R		
Name	StudentID	CourseNo
Mike	123456	COMP2400
Mike	123458	COMP2600

R_1	
Name	CourseNo
Mike	COMP2400
Mike	COMP2600

R_2	
StudentID	CourseNo
123456	COMP2400
123458	COMP2600

- Does the above decomposition preserves {StudentID} $\rightarrow$ {Name}? **No**, because {StudentID} and {Name} are not in a same relation after decomposition.

Exercise: Consider R={A, B, C} with the set of FDs $\Sigma = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$. Does the decomposition of R into $R1 = \{A, B\}$ and $R2 = \{A, C\}$ fulfill lossless join and dependency preserving? $\Sigma 1 = \{A \rightarrow B\}$ and $\Sigma 2 = \{A \rightarrow C\}$

Answer:

Lossless join? Yes! because A is a superkey for R1 and R2.

Dependency preserving? No! because $(\Sigma 1 \cup \Sigma 2)* \mathrel{!=} \Sigma*$ from the fact that $\{A \rightarrow B, A \rightarrow C\}$ cannot derive $B \rightarrow C$.

Lossless join? Yes

The common attributes of R1 and R2 are {A}. A is a superkey for R1 and R2.

Dependency preservation?

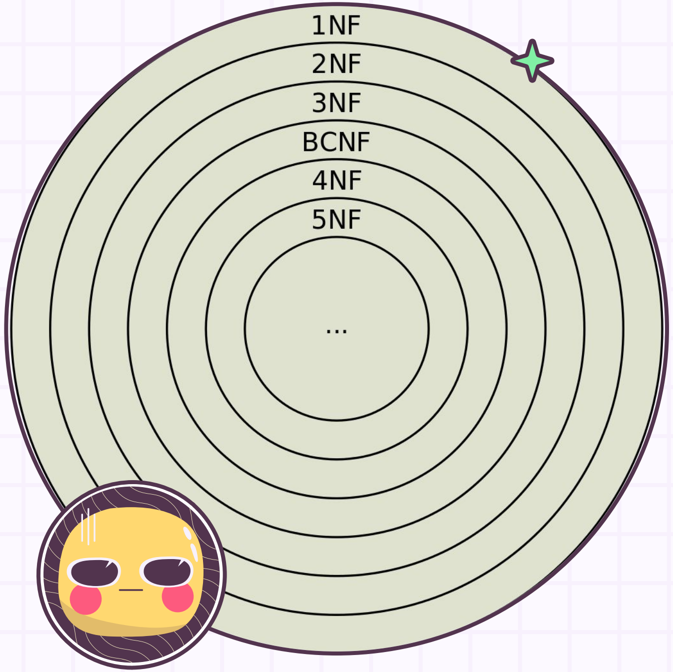
{A->B, A->C} -> {A->B, B->C, A->C} B->C cannot be inferred from the LHS B*=B

- If R with a set Σ of FDs is decomposed into R_1 with Σ_1 and R_2 with Σ_2,
  - **Lossless join** if and only if the common attributes of R_1 and R_2 are a superkey for R_1 or R_2;
  - **Dependency preserving** if and only if $(\Sigma_1 \cup \Sigma_2)^* = \Sigma^*$ holds.

Exercise: Consider R={A, B, C} with the set of FDs $\Sigma = \{A \to B, B \to C, A \to C\}$. Does the decomposition of R into R1 = {A, B} and R2 = {A, C} fulfill lossless join and dependency preserving? $\Sigma1 = \{A \to B\}$ and $\Sigma2 = \{A \to C\}$

NORMAL FORMS

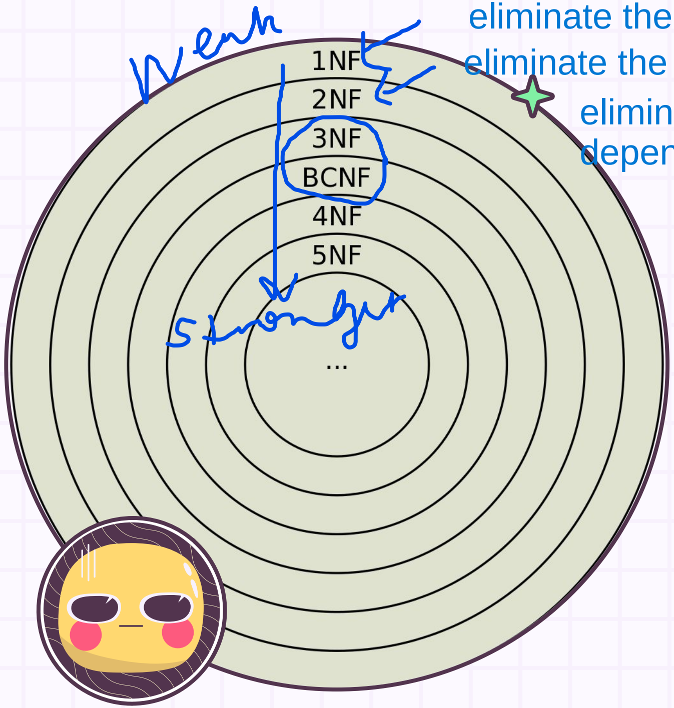
1NF –> BCNF: ?



NORMAL FORMS

1NF -> BCNF: Weak -> Strong

4NF, 5NF will not be covered in the course.

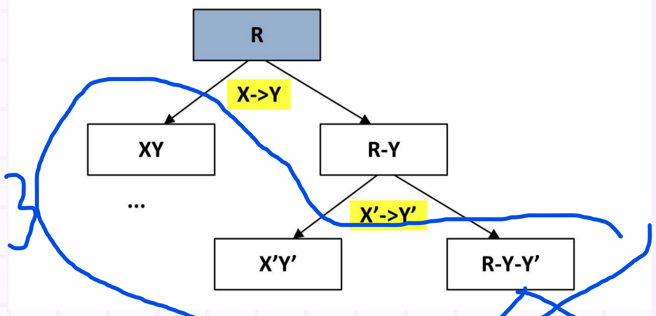


eliminate the duplicates
eliminate the partial FDs
eliminate transition dependency

BCNF

BCNF: A relation schema R is in BCNF if whenever a non-trivial FD X → A holds in R, then X is a superkey.

- Non-trivial: A is not a subset of X   trivial AB->A
- When a relation schema is in BCNF, all data redundancy based on functional dependency are removed.
- The order in which the FDs are applied may lead to different results.
- Lossless join! Why? X is the superkey of XY, same for X'/X'Y'.
- May not be dependency-preserving -> 3NF (less restrictive – lossless and dependency-preserving)



{XY, X'Y', R-Y-Y'}

3NF

3NF: A relation schema R is in 3NF if whenever a non-trivial FD X → A holds in R, then X is a superkey or A is a prime attribute .

- 3NF preserves all the functional dependencies at the cost of allowing some data redundancy
- Steps:
  - Find a minimal cover
  - Group FDs in the minimal cover
  - Remove redundant ones
  - Add a key (if necessary)
  - Project FDs

BCNF decomposition
{PropertyNo, Address, StaffNo, StaffName, Date, Time, CameraID}
-> {{ProperNo, Address}, {PropertyNo, StaffNo, StaffName, Date, Time, CameraID}}
-> {{ProperNo, Address}, {StaffNo, StaffName}, {PropertyNo, StaffNo, Date, Time, CameraID}}
->

BCNF EXAMPLE

For each relation, please (a) determine the candidate keys, and (b) in 3NF? In BCNF? and (c) if a relation is not in BCNF then decompose it into a collection of BCNF relations.

R6(A,B,C,D,E) with functional dependencies A→E, BC→A, DE→B.

(a)

C, D do not appear on the RHS of any FD, so C, D must be in the candidate keys.

CD+ = CD

Keys: {ACD, BCD, CDE}

(b)

All attributes are prime attributes, thus the schema is in 3NF.

Not in BCNF! Because A, BC, DF are not superkeys.

(c)

{ABCDE} –(A–>E)–> {AE,**ABCD**} (Not BC is not the key) –(BC–>A)–> {AE, ABC, BCD}

PRACTICE

For each relation, please (a) determine the candidate keys, and (b) if a relation is not in BCNF then decompose it into a collection of BCNF relations.

a. R1(A,C,B,D,E), A → B, C→ D

b. R2(A,B,F), AB→ F, B→ F

PRACTICE

For each relation, please (a) determine the candidate keys, and (b) if a relation is not in BCNF then decompose it into a collection of BCNF relations.

a. R1(A,C,B,D,E), A → B, C→ D

b. R2(A,B,F), AB→ F, B→ F

a.

(ACE)+=ABCDE

First compute the keys for R1. The attributes A, C, E do not appear on right hand side of any functional dependency therefore they must be part of a key. So we start from {A, C, E} and find out that this set can determine all features. So the key is {A, C, E}

We have dependencies A → B and C → D so the table is not BCNF. Applying the BCNF decomposition algorithm, the non-BCNF dependency is A → B, therefore create two relations (A, C, D, E) and (A, B). The first relation is still not in BCNF since we have a non-BCNF dependency C→ D. Therefore decompose further into (A, C, E) and (C, D). Now all relations are in BCNF and the final BCNF scheme is (A, C, E), (C, D), (A, B).

{ABCDE} -(A->B)> {AB, ACDE} -(C->D)> {AB, CD, ACE}

{ABCDE} -(A->B)> {AB, ACDE} -(C->D)> {AB, CD, ACE}

PRACTICE

For each relation, please (a) determine the candidate keys, and (b) if a relation is not in BCNF then decompose it into a collection of BCNF relations.

a. R1(A,C,B,D,E), A → B, C→ D

b. R2(A,B,F), AB→ F, B→ F

b.

First compute keys for R2. Note that AB -> F is totally redundancy since we already have B-> F. A,B do not appear on right side of any dependency, so start by computing attribute set closure of {AB}. Since AB→ F, we have {AB}+ = {ABF} and therefore {AB} is the key.

Since we have B → F, i.e., F is partially dependent on the key, the relation is not in BCNF. During BCNF decomposition, we have B → F as the non-BCNF relation therefore create new schema (A,B) (B,F). Both are in BCNF.

{ABF} -(B->F)> {BF, AB}

3NF EXAMPLE

R =(A, B, C, D). F = {C→D, C→A, B→C}.
Does it satisfy any normal forms?

3NF EXAMPLE

R =(A, B, C, D). F = {C→D, C→A, B→C}.
Does it satisfy any normal forms?
B+ = BC (B→C)
= BCD (C→D)
= ABCD (C→A) so the candidate key is B.
B is the ONLY candidate key, because nothing determines B: There is no rule that can produce B, except B →B.

R is not 3NF, because:
C→D causes a violation,

- C→D is non-trivial ($\{D\} \not\subset \{C\}$).
- C is not a superkey.
- D is not part of any candidate key.

C→A causes a violation
Similar to above
B→C causes no violation
Since R is not 3NF, it is not BCNF either.

3NF EXAMPLE

R =(A, B, C, D). F = $\{C \to D, C \to A, B \to C\}$.

Decompose it into 3NF:

- Find a minimal cover
  Already minimal cover
- Group FDs in the minimal cover
  {CD, AC, BC}
- Remove redundant ones
  No redundant ones
- Add a key (if necessary)
  B already in BC. No need.
- Project FDs
  Correct! Done.

MOVE ONTO YOUR LAB EXERCISE

- Ask in the channel if you have any doubts
- A2 already out on wattle