Normalization 3NF and BCNF

CS 4750 Database Systems

[A. Silberschatz, H. F. Korth, S. Sudarshan, Database System Concepts, Ch.7]
[Ricardo and Urban, Database Illuminated, Ch.6]
[https://www.w3schools.in/dbms/database-normalization/]

3NF and Decomposition

- **Lossless-join**
- Always **dependency preserving**
- Possible to have extra data (there may be redundancy)

Questions:
Is the relation in 3NF?
Is any refinement needed?

To calculate 3NF

- Identify PK of the original table
- Take Canonical Cover (Fc)
- Turn (minimal set of) FDs into tables

Canonical Cover (Fc)

- A minimal set of functional dependencies that has the same closure as the original set F
- Extraneous attributes = attribute of FDs that we can removed without changing the closure of FDs

- F logically implies all dependencies in Fc
- Fc logically implies all dependencies in F
- No FD in Fc contains an extraneous attribute

F and F+ are logically equivalent

Minimal basis for a set of FDs: For any set of FDs, there is at least one minimal basis, which is a set of FDs equivalent to the original (each set implies the other set), with singleton right sides, no FD that can be eliminated while preserving equivalence, and no attribute in a left side that can be eliminated while preserving equivalence

Canonical Cover (Fc)

Compute the canonical cover of a set of functional dependencies F

Always start with F and use rules to minimize

Fc = *F*
repeat
    apply union rule to replace any dependencies *f:* $X_1 \rightarrow Y_1$
        and *f:* $X_1 \rightarrow Y_2$ with *f:* $X_1 \rightarrow Y_1Y_2$
    for each functional dependency f_i
        if f_i contains an extraneous attribute either in *X* or in *Y*
            then remove an extraneous attribute
until *Fc* does not change any further

Example 1: 3NF and Fc

Let's do this together

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

FDs = { A → B, AB → D, B → BDE, C → D, D → D }

Compute Fc and convert the relation into 3NF

Observation: AC is a minimal super key of the given R

(1) write all LHS

A →
B →
AB →
C →
D →

(2) copy FDs as is

A → B
B → B DE
AB → D
C → D
D → D

(3) remove reflexivity

A → B
B → ~~B~~ DE
AB → D
C → D
~~D → D~~

(4) remove extraneous attr

A → B
B → DE
~~AB → D~~
C → D

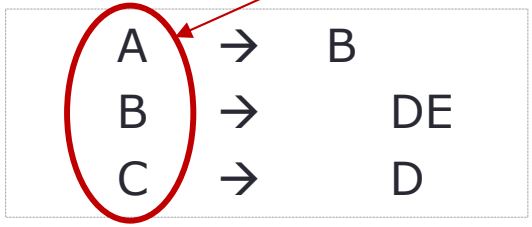
A → B and B → D.
Thus, remove AB → D

Example 1: 3NF and Fc

Let's do this together

(from previous page)

(4) Remove extraneous attr



Super key

Turn Fc into tables

AC (a minimal super key of the original R)

AB

BDE

CD

A relation R(A, B, C, D, E) is converted into 3NF by putting LHS and RHS of each FD in Fc together in one relation.

Dependency preserving

R(A, B, C, D, E) becomes R_1(A, C), R_2(A, B), R_3(B, D, E), R_4(C, D)

Or write it in another format: AC // AB // BDE // CD

Example 2: 3NF and Fc

Let's do this together

Given R(A,B,C)

FDs = { A → BC, B → C, A → B, AB → C }

Compute Fc and convert the relation into 3NF

(1) write all LHS

A →
B →
AB →

(2) copy FDs as is

A → BC
B → C
AB → C

Combine the given FDs A → BC and A → B

No reflexivity to remove

(3) remove extraneous attr

A → BC
B → C
~~AB → C~~

Consider AB → C and B → C,
A is an extraneous attr, remove A from AB → C (resulting in B → C)

(4) remove extraneous attr

A → B~~C~~
B → C

Apply decomposition to A → BC, thus, A → B and A → C.

A → C is logically equivalent to A → B and B → C (transitivity).

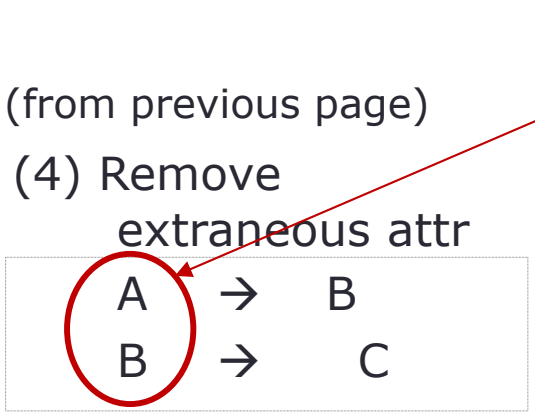
Thus, C is an extraneous attr, remove C from A → BC

Example 2: 3NF and Fc

Let's do this together

(from previous page)

(4) Remove extraneous attr



Turn Fc into tables

AB
BC

R(A, B, C) becomes R_1(A, B) and R_2(B, C)

Or write it in another format: AB // BC

BCNF and Decomposition

- **Lossless-join**
- Guarantee **redundancy free**
- May involve dependency across relations

Given a relation R,

for every nontrivial FD X → Y in R, X is a super key

- For all FDs, "key → everything"

Questions:
Is the relation in BCNF?
Is any refinement needed?

BCNF and Decomposition

To calculate BCNF

Compute $F+$
repeat given a relation R (or a decomposed R) and FDs F
    for each functional dependency f_i in a relation R
        if f_i violates $X \rightarrow Y$
            then decompose R into two relations:
                one with X U Y as its attributes (i.e., everything f)
                one with X U (attrs(R) – X – Y) as its attributes
until no violation

Example: BCNF and F+

Let's do this together

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

FDs = { A → B, AB → D, B → BDE, C → D, D → D }

Compute F+ and convert the relation into BCNF

Compute F+

(1) write all LHS & remaining

A	→
B	→
AB	→
C	→
D	→
E	→

(2) copy FDs as is

A	→	B
B	→	B DE
AB	→	D
C	→	D
D	→	D
E	→	

(3) apply reflexivity

A	→	AB
B	→	B DE
AB	→	AB D
C	→	CD
D	→	D
E	→	E

(4) apply transitivity

A	→	AB DE
B	→	B DE
AB	→	AB DE
C	→	CD
D	→	D
E	→	E

Example: BCNF and F+

Let's do this together

(from previous page)

(4) apply transitivity

F+

A	→	AB	DE
B	→	B	DE
AB	→	AB	DE
C	→		CD
D	→		D
E	→		E

Not trivial FDs. A, B, C are not super key

Trivial FDs (let's abbreviate to TR)

Based on F+, let's rewrite using the following format to help us calculate

R (A B C D E)

! ! ! TR TR

TR	– trivial
SK	– super key
X	– neither trivial nor super key
!	– (possibly) need to work on

Example: BCNF and F+

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

! ! ! TR TR

Let's do this together

F+

A	→	AB DE
B	→	B DE
AB	→	AB DE
C	→	CD
D	→	D
E	→	E

To choose which FD to work on, two ways:

- Choose the first FD, or
- Choose the longest FD (yield better solution)

Let's consider A:

A is not a super key, not trivial, thus A → ABDE violates BCNF, break a relation on A

Example: BCNF and F+

R (A B C D E)

! ! ! TR TR

Let's do this together

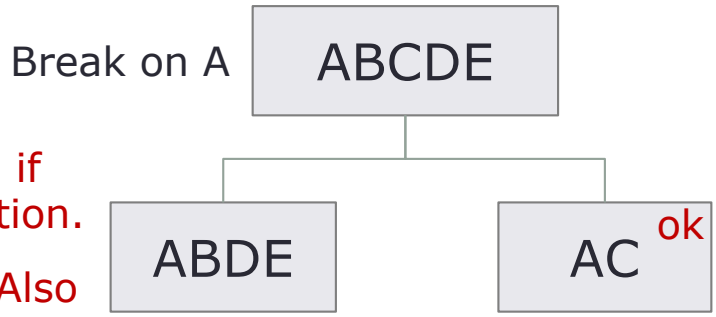
F+

A	→	AB DE
B	→	B DE
AB	→	AB DE
C	→	CD
D	→	D
E	→	E

A → ABDE

Take *RHS*, make a table: ABDE
Take *LHS*, make a table where A is a key
A plus (original – (*RHS*)) --- thus, AC

There are only 2 attrs, this relation is ok



Verify table ABDE if there is any violation.

A is a super key. Also can't break on A twice. Still need to work on B.

Restriction: Cannot break on A 2 times in a row

Next: consider B. B is neither trivial nor super key, break on B

Example: Calculate BCNF

R (A B D E)

X ! TR TR

Let's do this together

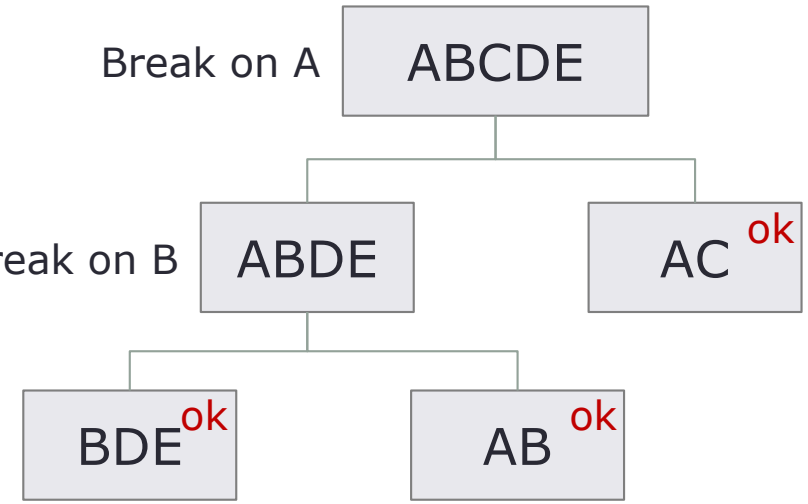
F+

A	→	AB DE
B	→	B DE
AB	→	AB DE
C	→	CD
D	→	D
E	→	E

B → BDE

Take *RHS*, make a table: BDE
Take *LHS*, make a table where B is a key
B plus (original – (*RHS*)) --- thus, AB

There are only 2 attrs, this relation is ok



R(ABCDE) becomes
AC // AB // BDE

Verify that B → BDE does not violate BCNF.

B is super key. Also, can't break on B twice.

D and E are trivial.

(notice: results are different 3NF)

Wrap-Up

- Compute F+ and Fc
- 3NF and decomposition
- BCNF and decomposition

What's next?

- SQL