More Practice
3NF and BCNF

CS 4750
Database Systems

Practice 1: 3NF and BCNF

Consider a relation Stocks(B, O, I, S, Q, D), whose attributes may be thought of informally as broker, office (of the broker), investor, stock, quantity (of the stock owned by the investor), and dividend (of the stock). Let the set of FDs for Stocks be

FDs = { $S \rightarrow D$, $I \rightarrow B$, $IS \rightarrow Q$, $B \rightarrow O$ }

Minimal basis ~ Fc

1. Verify that the given set of FDs is a minimal basis

To verify that the given FDs are their own minimal basis, we need to check:

1. Can any of the FDs be removed without losing the dependencies?

   No. If we remove any one of the four FDs, the remaining FDs do not imply the removed FD.

2. Can any attributes be removed from LHS and/or RHS without losing the dependencies?

   No. If we remove any attribute, we lose the dependency

We can conclude that the given set of FDs is the minimal basis.

Practice 1: 3NF and BCNF (2)

Stocks(B, O, I, S, Q, D)
FDs = { S → D, I → B, IS → Q, B → O }

2. Use 3NF, decompose the given Stocks relation into proper relations

The first step is to compute Canonical Cover (Fc).

```
S  -> D
I  -> B
IS -> Q
B  -> O
```

There is no trivial FD, no reflexivity, no extraneous attr. Thus, Fc = the given set of FDs. (note: Fc is the minimal basis of the given set of FDs).

To convert the given relation into 3NF, put the left-hand-side (LHS) and the right-hand-side (RHS) of each FD in Fc together in one relation. Thus, `Stocks(B,O,I,S,Q,D)` becomes R_1`(S,D)`, R_2`(I,B)`, R_3`(I,S,Q)`, and R_4`(B,O)`. Alternatively, we can write in another format: `SD // IB // ISQ // BO`

Practice 1: 3NF and BCNF (3)

Stocks(B, O, I, S, Q, D)
FDs = { S → D, I → B, IS → Q, B → O }

Decomposed relations:
SD // IB // ISQ // BO

3. Show that the decomposed relations are in 3NF

1. **Lossless join**
   Start with a relation of the decomposition with a superkey; thus consider `ISQ` where `IS` is a key. Compute attribute closure of `IS`. `IS+ = BOISQD` (or `IS -> BOISQD`), thus can reconstruct the original relation.

   Check if (R1 intersect R2) != { } and (R1 intersect R2) is a superkey of either R1 or R2 (or both), where R1 and R2 are any two decomposed relations.

```
SD and ISQ: S -> D  (S is a superkey of SD)
IB and ISQ: I -> B  (I is a superkey of IB)
IB and BO:  B -> O  (B is a superkey of BO)
```

2. **Dependency preserving**
   Consider all decomposed relations. Check if the given set of FDs can be verified within the decomposed relations.

```
S  -> D     can be verified within SD
I  -> B     can be verified within IB
IS -> Q     can be verified within ISQ
B  -> O     can be verified within BO
```

   No transitive dependency. Thus, satisfy dependency preserving property

Practice 1: 3NF and BCNF (4)

Stocks(B, O, I, S, Q, D)
FDs = { S → D, I → B, IS → Q, B → O }

Decomposed relations:
SD // IB // ISQ // BO

4. Are the decomposed relations in BCNF?

In BCNF. No violation.

In addition to checking for lossless join property, we need to verify that there is no non-key dependency in each decomposed relation.

```
For every non-trivial FD, X -> A, X is a superkey
```

Consider each decomposed relation (above), all LHS of the given FD is a superkey of the relation. Thus, no non-key dependency

Ready for the next one ??

Go!!

Practice 2: 3NF

Consider the following relation and functional dependencies

R(A, B, C, D)

FDs = { C → A, C → D, C → C, AB → C }

1. Decompose the given relation R(A, B, C, D) using 3NF
2. Discuss to show that the decomposed relations are in 3NF

Fc:

C → AD

AB → C

Decomposed relations:

ACD // ABC or written as R1(ACD) and R2(ABC)

Practice 2: 3NF (2)

2. Discuss to show that the decomposed relations are in 3NF

We need to show that the decomposed relations ACD // ABC satisfy lossless join and dependency preserving. Fc = { C → AD, AB → C }

Lossless join -- check if (R1 intersect R2) != { } and (R1 intersect R2) is a superkey of either R1 or R2 (or both)

- Let R1 = ACD and R2 = ABC, (R1 intersect R2) = AC
- Consider ACD and check if (R1 intersect R2) is a superkey of ACD.
- Since we know that C → AD
  - C → C, thus C → ACD (reflexive)
  - A → A, we can augment the above FD by adding attribute A to the left hand side, thus AC → ACD
- Therefore, we can conclude that AC (which is R1 intersect R2) is a superkey of ACD (R1)

Practice 2: 3NF (3)

2. Discuss to show that the decomposed relations are in 3NF

Decomposed relations ACD // ABC, Fc = { C → AD, AB → C }

Lossless join -- check no loose, no gain

- First, identify the superkey of the given relation R(ABCD).
  - Based on the given FDs, compute attribute closure.
  - Any combination of attributes that contains AB is a superkey. Thus, the only minimal superkey of R(ABCD) is AB.
- Then, start with a relation of the decomposition with a superkey AB, which is ABC (refer to Fc: AB → C).
- Compute attribute closure of AB.
- To help us envision, from the decomposed relation ABC,
  - Apply AB → C (Fc), AB → AB (reflexive), and C → AD (transitive), thus AB → ABCD. That is, ABC will be expanded to ABCD (~ reconstruct the original relation from this decomposition).
- Since AB+ = ABCD (i.e., AB → ABCD), the decomposition satisfies lossless join property (no loose, no gain).

Practice 2: 3NF (4)

2. Discuss to show that the decomposed relations are in 3NF

Decomposed relations ACD // ABC, Fc = { C → AD, AB → C }

Dependency preserving

- Check if the given FDs = { C → A, C → D, C → C, AB → C } can be verified within a single decomposed relation.
- Given FDs:
  - C → A can be verified within ACD
  - C → D can be verified within ACD
  - C → C can be verified within ACD
    - Always true in any relation that contains C
- Given FD: AB → C, we can verify it within ABC
- No transitive dependency.
- Thus, the decomposition satisfies dependency preserving property.

No 3NF violation

Ready for the next one ??

Go!!

Practice 3: BCNF

Consider the following relation and functional dependencies

R(A, B, C, D)

FDs = { C → A, C → D, C → C, AB → C }

1. Decompose the given relation R(A, B, C, D) using BCNF
2. Discuss to show that the decomposed relations are in BCNF

ABCD
tr tr ! tr

Break on c

ACD ok
tr x tr

BC ok

Verify table ACD if there is any violation.

Just broke on C, cannot break on C twice in a row.

F+

A → A		tr
B → B		tr
C → A CD		
D → D		tr
AB → ABCD		sk

Alternatively, we can verify through "for all non-trivial FDs, X→A, X must be a superkey"

For ACD, we know that only C has a non-trivial FD and we know that C → AD (given FD), we can conclude that C is a superkey of ACD, cannot break using the superkey (i.e., no violation).

Decomposed relations: ACD // BC

Practice 3: BCNF (2)

2. Discuss to show that the decomposed relations are in BCNF

We need to show that the decomposed relations ACD // BC satisfy lossless join and For every non-trivial FD, X → Attribute(s), X is a superkey.

F+		
A → A		tr
B → B		tr
C → A CD		
D → D		tr
AB → ABCD		sk

Lossless join -- check if (R1 intersect R2) != { } and (R1 intersect R2) is a superkey of either R1 or R2 (or both)

- Let R1 = ACD and R2 = BC, (R1 intersect R2) = C
- Consider ACD and check if (R1 intersect R2) is a superkey of ACD.
- From F+, we can conclude that (R1 intersect R2) is a superkey for ACD (R1)
- Thus, the decomposition satisfies lossless join property.

Practice 3: BCNF (3)

2. Discuss to show that the decomposed relations are in BCNF

Decomposed relations ACD // BC, FDs = { C → A, C → D, C → C, AB → C }

F+

A →	A	tr
B →	B	tr
C →	A CD	
D →	D	tr
AB →	ABCD	sk

Lossless join -- check no loose, no gain

- First, identify the superkey of the given relation R(ABCD).
  - Based on the given FDs, compute attribute closure.
  - Any combination of attributes that contains AB is a superkey. Thus, the only minimal superkey of R(ABCD) is AB.
- Then, start with a relation of the decomposition with a superkey AB, which is ABC (AB → C).
- Compute attribute closure of AB.
- To help us envision, from the decomposed relation ABC,
  - Apply AB → C, AB → AB (reflexive), and C → AD (transitive), thus AB → ABCD. That is, ABC will be expanded to ABCD (~ reconstruct the original relation from this decomposition).
- Since AB+ = ABCD (i.e., AB → ABCD), the decomposition satisfies lossless join property (no loose, no gain).

Practice 3: BCNF (4)

2. Discuss to show that the decomposed relations are in BCNF

To verify, we need to show that the decomposed relations ACD // BC satisfy lossless join and For every non-trivial FD, X → Attribute(s), X is a superkey.

Given FDs = { C → A, C → D, C → C, AB → C }

F+

A →	A	tr
B →	B	tr
C →	A CD	
D →	D	tr
AB →	ABCD	sk

For every non-trivial FD, X → Attribute(s), X is a superkey

- Consider ACD, we know that C → ACD (A and D have trivial FDs, so ignore)
  - C is the only non-trivial FD and C is a superkey of ACD.
  - No non-key dependency in this decomposed relation.
- Consider BC, no non-key dependency.
  - By default, a relation with 2 attributes is always in BCNF (even if there is no dependency at all).

No BCNF violation

Ready for the next one ??

Go!!

Practice 4: 3NF

Consider the following relation and functional dependencies

R(A, B, C, D)

FDs = { A → ABC, C → D, A → C, D → D }

1. Decompose the given relation R(A, B, C, D) using 3NF
2. Discuss to show that the decomposed relations are in 3NF

Fc:

A → BC

C → D

Decomposed relations:

ABC // CD or written as R1(ABC) and R2(CD)

Practice 4: 3NF (2)

2. Discuss to show that the decomposed relations are in 3NF

We need to show that the decomposed relations ABC // CD satisfy lossless join and dependency preserving. Fc = { A → BC, C → D }

Lossless join -- check if (R1 intersect R2) != { } and (R1 intersect R2) is a superkey of either R1 or R2 (or both)

- Let R1 = ABC and R2 = CD, (R1 intersect R2) = C
- From Fc, we can conclude that (R1 intersect R2) is a superkey of CD (R2)

Practice 4: 3NF (3)

2. Discuss to show that the decomposed relations are in 3NF

Decomposed relations ABC // CD, Fc = { A → BC, C → D }

Lossless join -- check no loose, no gain

- First, identify the superkey of the given relation R(ABCD).
  - Based on the given FDs, compute attribute closure.
  - Any combination of attributes that contains A is a superkey. Thus, the only minimal superkey of R(ABCD) is A.
- Then, start with a relation of the decomposition with a superkey A, which is ABC (refer to Fc: A → BC).
- Compute attribute closure of A.
- To help us envision, from the decomposed relation ABC,
  - Apply A → BC (Fc), A → A (reflexive), and C → D (transitive), thus A → ABCD. That is, ABC will be expanded to ABCD (~ reconstruct the original relation from this decomposition).
- Since A+ = ABCD (i.e., A → ABCD), the decomposition satisfies lossless join property (no loose, no gain).

Practice 4: 3NF (4)

2. Discuss to show that the decomposed relations are in 3NF

Decomposed relations ABC // CD, Fc = { A → BC, C → D }

Dependency preserving

- Check if the given FDs = { A → ABC, C → D, A → C, D → D } can be verified within a single decomposed relation.
- Given FDs:
  - A → ABC can be verified within ABC
  - C → D can be verified within CD
  - A → C can be verified within ABC
  - D → D can be verified within CD
    - Always true in any relation that contains D
- No transitive dependency.
- Thus, the decomposition satisfies dependency preserving property.

No 3NF violation

Ready for the next one ??

Go!!

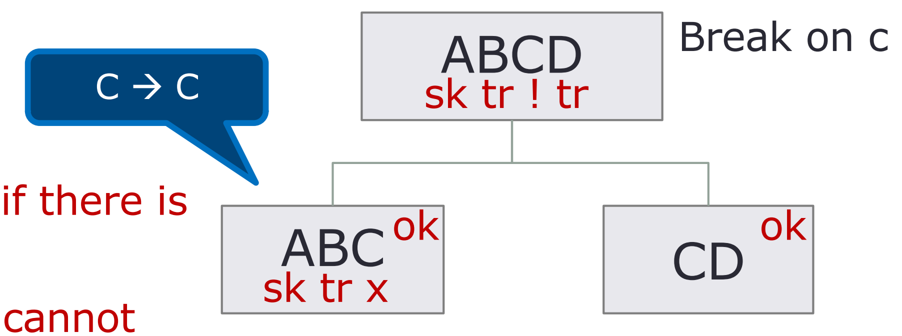
Practice 5: BCNF

Consider the following relation and functional dependencies

R(A, B, C, D)

FDs = { A → ABC, C → D, A → C, D → D }

1. Decompose the given relation R(A, B, C, D) using BCNF
2. Discuss to show that the decomposed relations are in BCNF



Verify table ABC if there is any violation.

Just broke on C, cannot break on C twice in a row.

F+

A	→	ABCD	sk
B	→	B	tr
C	→	CD	
D	→	D	tr

Alternatively, we can verify through "for all non-trivial FDs, X→A, X must be a superkey"

For ABC, we know that only A has a non-trivial FD and we know that A → BC (given FD), we can conclude that A is a superkey of ABC, cannot break using the superkey (i.e., no violation).

Decomposed relations: ABC // CD

Practice 5: BCNF (2)

2. Discuss to show that the decomposed relations are in BCNF

To verify, we need to show that the decomposed relations ABC // CD satisfy lossless join and For every non-trivial FD, X → Attribute(s), X is a superkey.

F+

A →	ABCD		sk
B →	B		tr
C →	CD		
D →	D		tr

Lossless join -- check if (R1 intersect R2) != { } and (R1 intersect R2) is a superkey of either R1 or R2 (or both)

- Let R1 = ABC and R2 = CD, (R1 intersect R2) = C
- From F+, we can conclude that (R1 intersect R2) is a superkey for CD (R2)
- Thus, no violation

Practice 5: BCNF (3)

2. Discuss to show that the decomposed relations are in BCNF

Decomposed relations ABC // CD, FDs = { A → ABC, C → D, A → C, D → D }

F+

A →	ABCD	sk
B →	B	tr
C →	CD	
D →	D	tr

Lossless join -- check no loose, no gain

- First, identify the superkey of the given relation R(ABCD).
  - Based on the given FDs, compute attribute closure.
  - Any combination of attributes that contains A is a superkey. Thus, the only minimal superkey of R(ABCD) is A.
- Then, start with a relation of the decomposition with a superkey A (ABC).
- Compute attribute closure of A.
- To help us envision, from the decomposed relation ABC,
  - Apply A → ABC (given FD) and C → D (transitive), thus A → ABCD. That is, ABC will be expanded to ABCD (~ reconstruct the original relation from this decomposition).
- Since A+ = ABCD (i.e., A → ABCD), the decomposition satisfies lossless join property (no loose, no gain).

Practice 5: BCNF (4)

2. Discuss to show that the decomposed relations are in BCNF

To verify, we need to show that the decomposed relations ABC // CD satisfy lossless join and For every non-trivial FD, X → Attribute(s), X is a superkey.

Given FDs = { A → ABC, C → D, A → C, D → D }

F+

A →	ABCD	sk
B →	B	tr
C →	CD	
D →	D	tr

For every non-trivial FD, X → Attribute(s), X is a superkey

- Consider ABC, we know that A → ABC. (B and C have trivial FDs here, ignore)
  - No non-key dependency in this decomposed relation.
- Consider CD, no non-key dependency.
  - By default, a relation with 2 attributes is always in BCNF (even if there is no dependency at all).

No BCNF violation

Ready for the next one ??

Go!!

Practice 6: BCNF

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

FDs = { B → DE, C → A, A → BC, D → E }

Convert the relation into BCNF

Compute F+

(1) write all LHS & remaining	(2) copy FDs as is	(3) apply reflexivity	(4) apply transitivity
A →	A → BC	A → ABC	A → ABCDE
B →	B → DE	B → B DE	B → B DE
C →	C → A	C → A C	C → ABCDE
D →	D → E	D → DE	D → DE
E →	E →	E → E	E → E

F+ = { A → ABCDE, B → BDE, C → ABCDE, D → DE, E → E }

Practice 6: BCNF (2)

(from previous page)

(4) apply transitivity

F+	A	→	ABCDE
	B	→	B DE
	C	→	ABCDE
	D	→	DE
	E	→	E

Non-trivial FDs. B, D 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)
SK ! SK ! TR

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

Practice 6: BCNF (3)

R (A $\boxed{B}$ C D E)
SK ! SK ! TR

F+

A	→	ABCDE
B	→	B DE
C	→	ABCDE
D	→	DE
E	→	E

Consider non-trivial FDs and attributes that are not SK

Let's consider B:

B is not a super key, not trivial, thus B → BDE violates BCNF, thus we can break a relation on B

Let's consider D:

D is not a super key, not trivial, thus D → DE violates BCNF, thus we can break a relation on D

To choose which FD to work on, two ways:

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

Break on B

Practice 6: BCNF (4)

R (A $\boxed{B}$ C D E)

SK ! SK ! TR

F+

A	→	ABCDE
B	→	B DE
C	→	ABCDE
D	→	DE
E	→	E

B → BDE

RHS, make a table: BDE
LHS, make a table with B plus (original – (*RHS*)) – thus, ABC

Break on B

ABCDE

Verify table ABC if there is any violation or if we can break any further.

A and C are super keys. – already satisfy, don't break on them.

Can't break on B twice.

ABC ok

SK x SK

BDE

x ! TR

Verify table BDE if there is any violation.

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

E is trivial, can't break.

Break on D

Practice 6: BCNF (5)

R (~~B~~ $\boxed{D}$ ~~E~~)

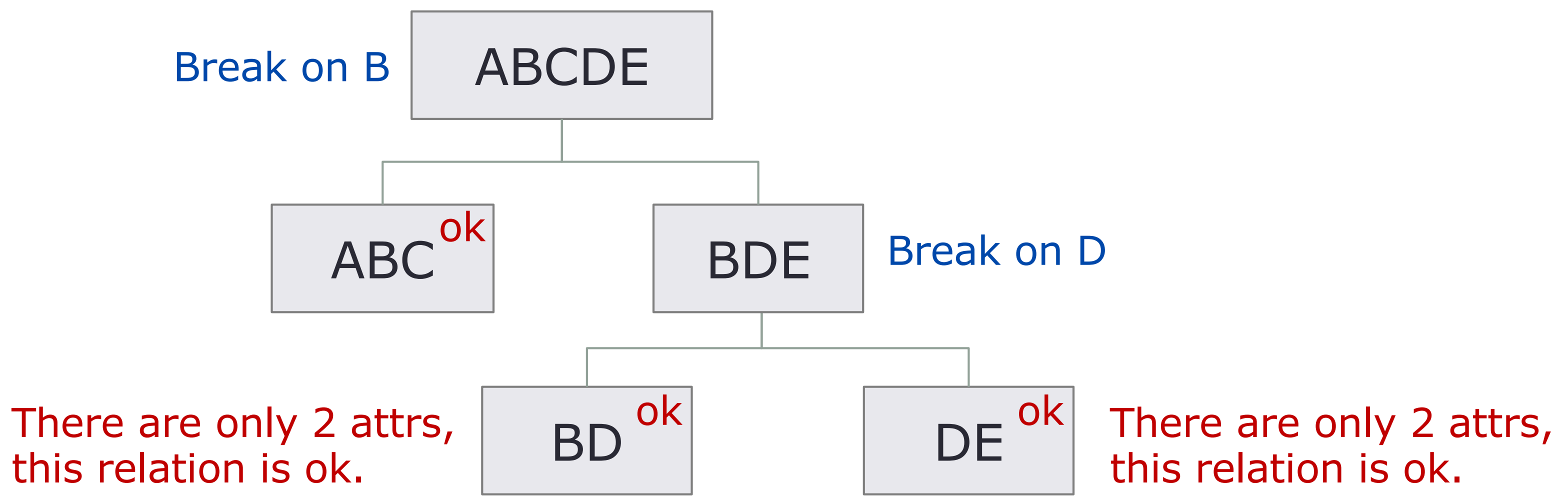
x ! TR

F+

A	→	ABCDE
B	→	B DE
C	→	ABCDE
D	→	DE
E	→	E

D → DE

RHS, make a table: DE
LHS, make a table with D plus (original – (*RHS*)) – thus, BD



Decomposed tables: R1(ABC), R2(BD), R3(DE)

Ready for the next one ??

Go!!

Practice 7: BCNF

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

FDs = { A → CE, A → B, B → D, D → CD, C → E }

Convert the relation into BCNF

Compute F+

(1) write all LHS & remaining	(2) copy FDs as is	(3) apply reflexivity	(4) apply transitivity
A →	A → BC E	A → ABC E	A → ABCDE
B →	B → D	B → B D	B → BCDE
C →	C → E	C → C E	C → C E
D →	D → CD	D → CD	D → CDE
E →	E →	E → E	E → E

F+ = { A → ABCDE, B → BCDE, C → CE, D → CDE, E → E }

Practice 7: BCNF (2)

(from previous page)

(4) apply transitivity

F+

A	→	ABCDE
B	→	BCDE
C	→	C E
D	→	CDE
E	→	E

Non-trivial FDs. B, C, D 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)
SK ! ! ! TR

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

Practice 7: BCNF (3)

R (~~A~~ $\boxed{B}$ C D ~~E~~)

SK ! ! ! TR

F+

A	→	ABCDE
B	→	BCDE
C	→	C E
D	→	CDE
E	→	E

Consider non-trivial FDs and attributes that are not SK

B is not a super key, not trivial, thus B → BCDE violates BCNF, thus we can break a relation on B

B is not a super key, not trivial, thus C → CE violates BCNF, thus we can break a relation on C

D is not a super key, not trivial, thus D → CDE violates BCNF, thus we can break a relation on D

To choose which FD to work on, two ways:

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

Break on B

Practice 7: BCNF (4)

F+

A	→	ABCDE
B	→	BCDE
C	→	C E
D	→	CDE
E	→	E

R (~~A~~ $\boxed{B}$ C D ~~E~~)
SK ! ! ! TR

Break on B

B → BCDE

RHS, make a table: BCDE
LHS, make a table with B plus (original – (*RHS*)) – thus, AB

There are only 2 attrs, this relation is ok.

R (~~B~~ C $\boxed{D}$ ~~E~~)
x ! ! TR

Break on D

D → CDE

RHS, make a table: CDE
LHS, make a table with D plus (original – (*RHS*)) – thus, BD

There are only 2 attrs, this relation is ok.

R ($\boxed{C}$ ~~D~~ ~~E~~)
! x TR

Break on C

C → CE

RHS, make a table: CE
LHS, make a table with C plus (original – (*RHS*)) – thus, CD

There are only 2 attrs, this relation is ok.

Decomposed tables: R1(AB), R2(BD), R3(CE), R4(CD)

Ready for the next one ??

Go!!

Practice 8: 3NF

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

FDs = { A → CE, A → B, B → D, D → CD, C → E }

Convert the relation into BCNF

Compute Fc

Compute Fc without using reflexivity

(1) write all LHS & remaining	(2) copy FDs as is	(3) remove reflexivity	(4) remove extraneous attr
A →	A → BC E	A → BC E	A → B~~C~~ ~~E~~
B →	B → D	B → D	B → D
C →	C → E	C → E	C → E
D →	D → CD	D → C~~D~~	D → C

A → B, B → D, D → C, C → E, thus A does not have to directly imply E.
A → B, B → D, D → C, thus A does not have to directly imply C.
Alternatively, if we apply A → C, C → E, thus A does not need to directly imply E. If this was done, C would be kept (i.e., A → BC); then, A → B, B → D, D → C, remove C

Fc = { A → B, B → D, C → E, D → C }

Decomposed tables: R1(AB), R2(BD), R3(CE), R4(CD)