Normalisation

Thomas Heinis

t.heinis@imperial.ac.uk
wp.doc.ic.ac.uk/theinis

Imperial College
London

Database Design by Decomposition

Relational database schemas should be normalised. Normalised databases:

1. provide a good representation of the real world that is easy to understand and easy to evolve.
2. reduce data duplication, help avoid insert, update and deletion anomalies, and simplify the enforcement of database constraints.

One approach to designing a normalised schema, is to start with one or a few “big” relations and then to systematically decompose them into smaller “better relations that are in a suitable normal form.

We’ll look at the two most common normal forms:

Boyce-Codd Normal Form (BCNF) and

Third Normal Form (3NF)

Anomalies

title	year	length	genre	studio	actor
Star Wars	1977	124	SF	Fox	Carrie Fisher
Star Wars	1977	124	SF	Fox	Mark Hamill
Star Wars	1977	124	SF	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Myers

Redundancies - Unnecessary repetition of same information in several tuples, e.g. length
Updates - Changing length for Star Wars requires changing 3 tuples (error prone)
Inserts - Adding a new actor for a movie means we need to ensure other attributes are the same, also can't add a movie without an actor.
Deletes - Removing Vivien Leigh means we have no tuple for Gone with the Wind.

Boyce-Codd Normal Form (BCNF)

Such anomalies do not exist if the relation is in Boyce-Codd Normal Form (BCNF)

A relation R is in BCNF, if and only if, for all non-trivial FDs (incl. derived FDs) of the relation, the LHS of every FD is a superkey (i.e. contains a key).

Example: Is the movies relation below in BCNF if we have the following FD: title year → length genre studio?

title	year	length	genre	studio	actor
Star Wars	1977	124	SF	Fox	Carrie Fisher
Star Wars	1977	124	SF	Fox	Mark Hamill
Star Wars	1977	124	SF	Fox	Harrison Ford
Gone with the Wind	1939	231	Drama	MGM	Vivien Leigh
Wayne's World	1992	95	Comedy	Paramount	Dana Carvey
Wayne's World	1992	95	Comedy	Paramount	Mike Myers

Decomposition

Given a relation R(A1, A2, .. An) decompose it into two projected relations S and T such that attr(R)=attr(S) ∪ attr(T), S = $\pi_{attr(S)}$(R), T = $\pi_{attr(T)}$(R)

Movies1

title	year	length	genre	studio
Star Wars	1977	124	SF	Fox
Gone with the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Movies2

title	year	actor
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone with the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Myers

Decomposition Properties

When decomposing a relation it is also important to try ensure that we can recover the original relation by *joining* the decomposed relations **and** also preserving the FDs of the original relation. Note: Preserving FDs is not always possible with BCNF.

Lossless Decomposition

If a relation R is decomposed into 2 relations S and T, the decomposition is **lossless** if at least one of the following FDs holds in the closure of the FD set of R:

$$attr(S) \cap attr(T) \rightarrow attr(S) \text{ or}$$
$$attr(S) \cap attr(T) \rightarrow attr(T)$$

i.e. the common attributes of S and T form a superkey of either S or T.

Dependency Preserving Decomposition

If we can check the functional dependencies of R ***without joining*** S and T then the decomposition is said to be dependency preserving.

Example

Given R(A, B, C)
FDs A → B
B → C

Decomposition	Lossless Decomposition	Dependency Preserving
S(A,B) T(B,C)		
S(A,B) T(A,C)		

Example

Given R(A, B, C)
FDs A → B
B → C

Decomposition	Lossless Decomposition	Dependency Preserving
S(A,B) T(B,C)	B → A B doesn't hold B → B C holds therefore lossless	Yes. We can check A → B using S and B → C using T
S(A,B) T(A,C)	A → A B holds therefore lossless A → A C holds also	No. We cannot check B → C without joining S and T.

Exercise

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

FDs A → B C

C D → E

B → D

E → A

Decomposition	Lossless Decomposition	Dependency Preserving
S(A,B,C) T(A,D,E)		
S(A,B,C) T(C,D,E)		

Solution

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

FDs A → B C

C D → E

B → D

E → A

Decomposition	Lossless Decomposition	Dependency Preserving
S(A,B,C) T(A,D,E)	A → A B C holds therefore lossless	No. We cannot check CD → E, B → D
S(A,B,C) T(C,D,E)	C → A B C doesn't hold C → C D E doesn't hold	No. We cannot check B → D, E → A

Decomposition into BCNF

The following algorithm allows us to systematically decompose a relation into BCNF. The basic idea is to use violating FDs (where the LHS is not a superkey) to guide the decomposition strategy, replacing a relation with a violating FD with 2 new relations, one with all the attributes of the FD, and one with all the attributes of the relation minus the RHS of the FD, effectively allowing us to join the relations when required.

initialise decompositions with R
while ViolatingRelation V in decompositions do {
    let LHS → RHS be a nontrivialFD for V such that
        (i) LHS → attr(V) does not hold for FDset$^+$ of R
        (ii) LHS ∩ RHS={}
    remove V from decompositions
    add relation(LHS ∪ RHS) to decompositions
    add relation(attr(V)-RHS) to decompositions }

i.e. LHS not a superkey of R

Note + i.e. we may need to check a derived FD for decomposed relations

Alternatively we can check if a decomposed relation S is in BCNF, by checking that for every subset α of attr(S) - that {α}$^+$=attr(S) or has no attributes of attr(S)-α

Decomposition into BCNF

The previous slide is a bit “dense”.

Summarising, in BCNF decomposition we take one of the violating FDs, which could be a derived FD, and split the relation, into two child relations.

Then for each of the child relations we independently determine their FDs. A child FD set is the subset of the parent FD set that uses only the child's attributes.

Once this is done, we apply the BCNF procedure to each child, calculating keys for the child relation and checking whether any of the child FDs (including derived child FDs) violate the FDs for the child relation.

The decomposition proceeds recursively.

Note: Because there is sometimes a choice of which violating FD to use to split a relation, **there can be multiple BCNF decompositions that are valid**.

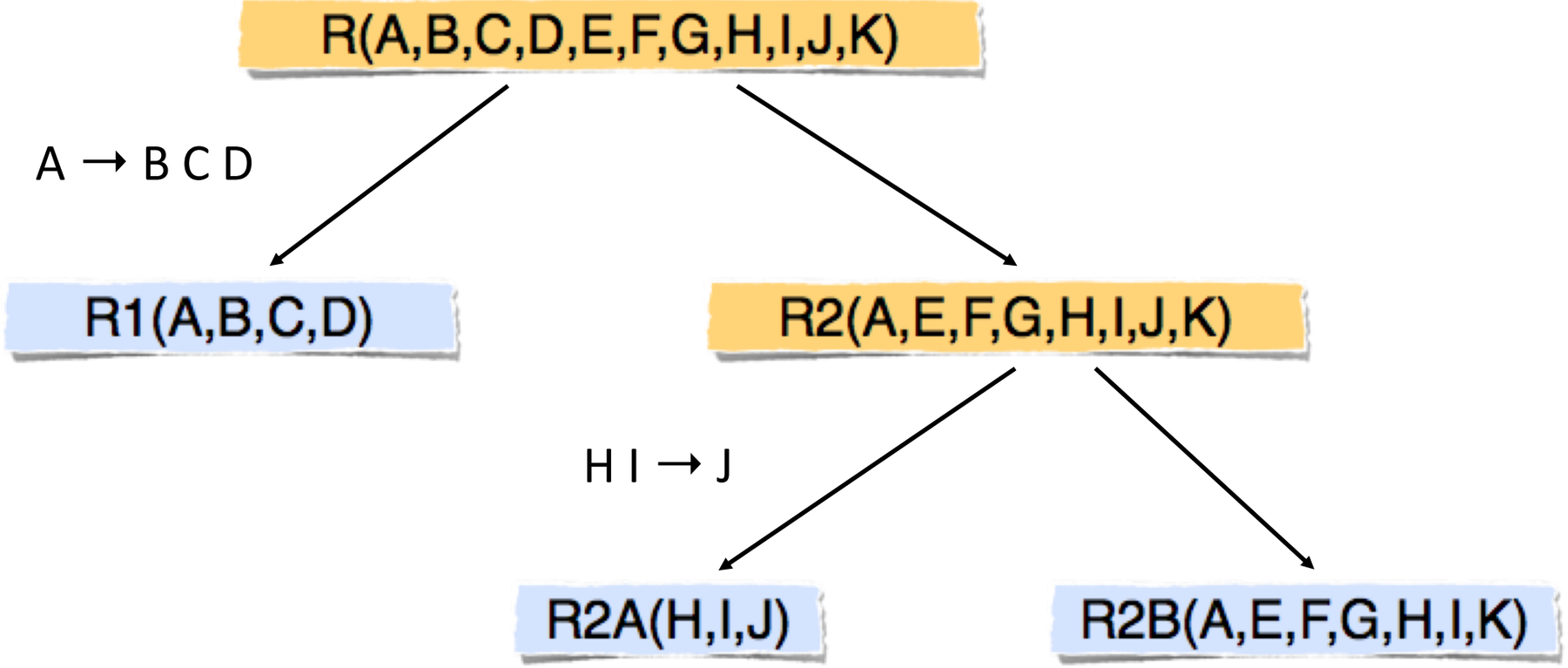
Example

Given R(A, B, C, D, E, F, G, H, I, J, K)

FDs A → B C D

H I → J

A E F G → H I K



Example Contd

Given R(A, B, C, D, E, F, G, H, I, J, K)

FDs $A \rightarrow B\ C\ D$

$H\ I \rightarrow J$

$A\ E\ F\ G \rightarrow H\ I\ K$

FD	Action	Relations
		R(A,B,C,D,E,F,G,H,I,J,K)
$A \rightarrow B\ C\ D$	A is not a superkey, replace R with R1 and R2.	R1(A,B,C,D) i.e. LHS ∪ RHS R2(A,E,F,G,H,I,J,K) i.e. R-RHS
	R1 is in BCNF (by checking all derivable FDs in F^+ that contain R1 attributes)	
$H\ I \rightarrow J$	FD holds but H I is not a superkey for R2. Replace with R2A, R2B	R1(A,B,C,D) R2A(H,I,J) i.e. LHS ∪ RHS R2B(A,E,F,G,H,I,K) i.e. R-RHS
	R2A and R2B are in BCNF	

Exercise

Given R(A, B, C, D)

FDs D → A
C → D
AB → C

Derived: C → A, AB → D, AC → D, BC → A,
BC → D, BD → A, BD → C, CD → A
ABC → D, ABD → C, BCD → A

Keys {AB}, {BC}, {BD}

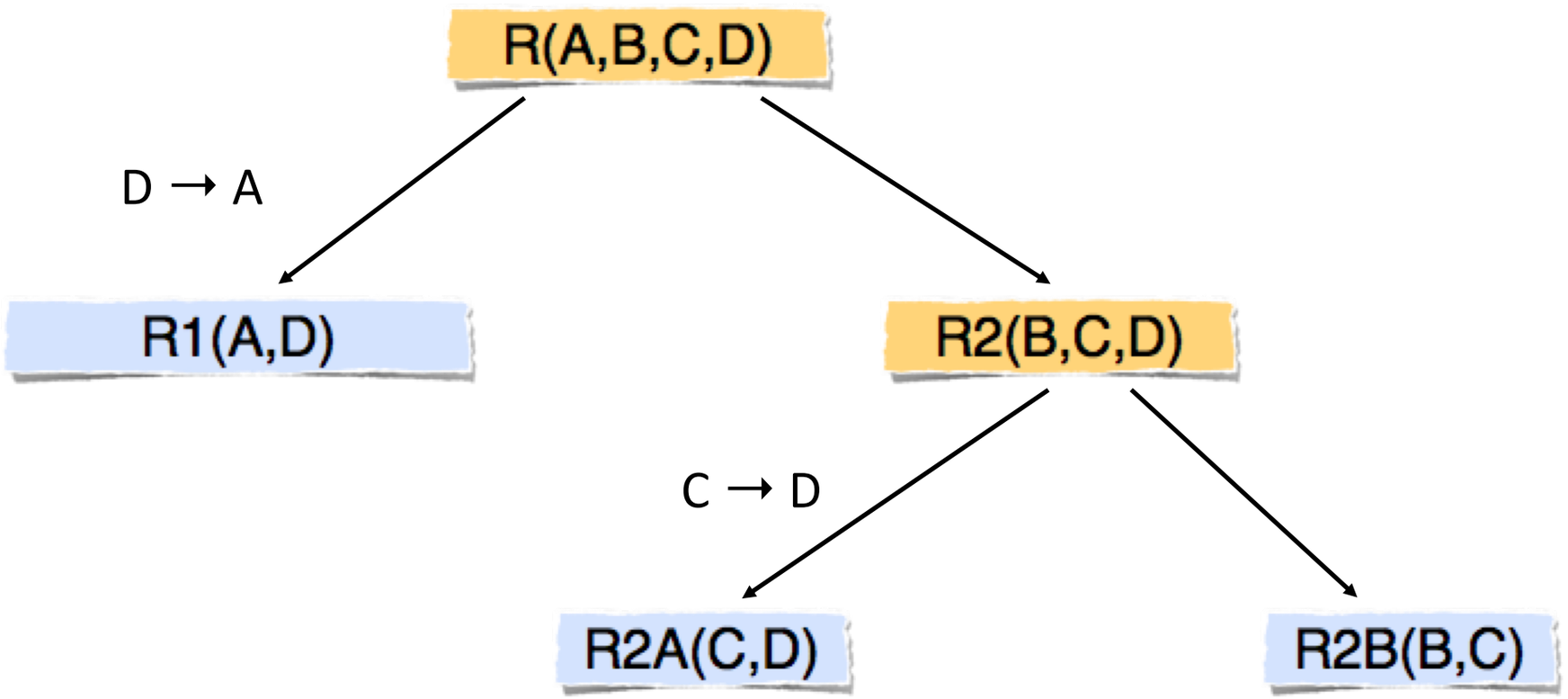
Solution

Given R(A, B, C, D)

FDs
- $D \rightarrow A$
- $C \rightarrow D$
- $AB \rightarrow C$

Derived: $C \rightarrow A$, $AB \rightarrow D$, $AC \rightarrow D$, $BC \rightarrow A$, $BC \rightarrow D$, $BD \rightarrow A$, $BD \rightarrow C$, $CD \rightarrow A$, $ABC \rightarrow D$, $ABD \rightarrow C$, $BCD \rightarrow A$

Keys {AB}, {BC}, {BD}



Solution

Given R(A, B, C, D)

FDs D → A, C → D, AB → C

Derived: C → A, AB → D, AC → D, BC → A, BC → D, BD → A, BD → C, CD → A, ABC → D, ABD → C, BCD → A

Keys{AB}, {BC}, {BD}

FD	Action	Relations
		R(A,B,C,D)
D → A	D is not a superkey for R with R1 and R2.	R1(A,D) i.e. LHS ∪ RHS R2(B,C,D) i.e. R-RHS
	Note: R1 is in BCNF, R2 isn't	
C → D	C is not a superkey for R2. Replace with R2A, R2B	R1(A,D) R2A(C,D) i.e. LHS ∪ RHS R2B(B,C) i.e. R-RHS
	R2A and R2B are in BCNF	

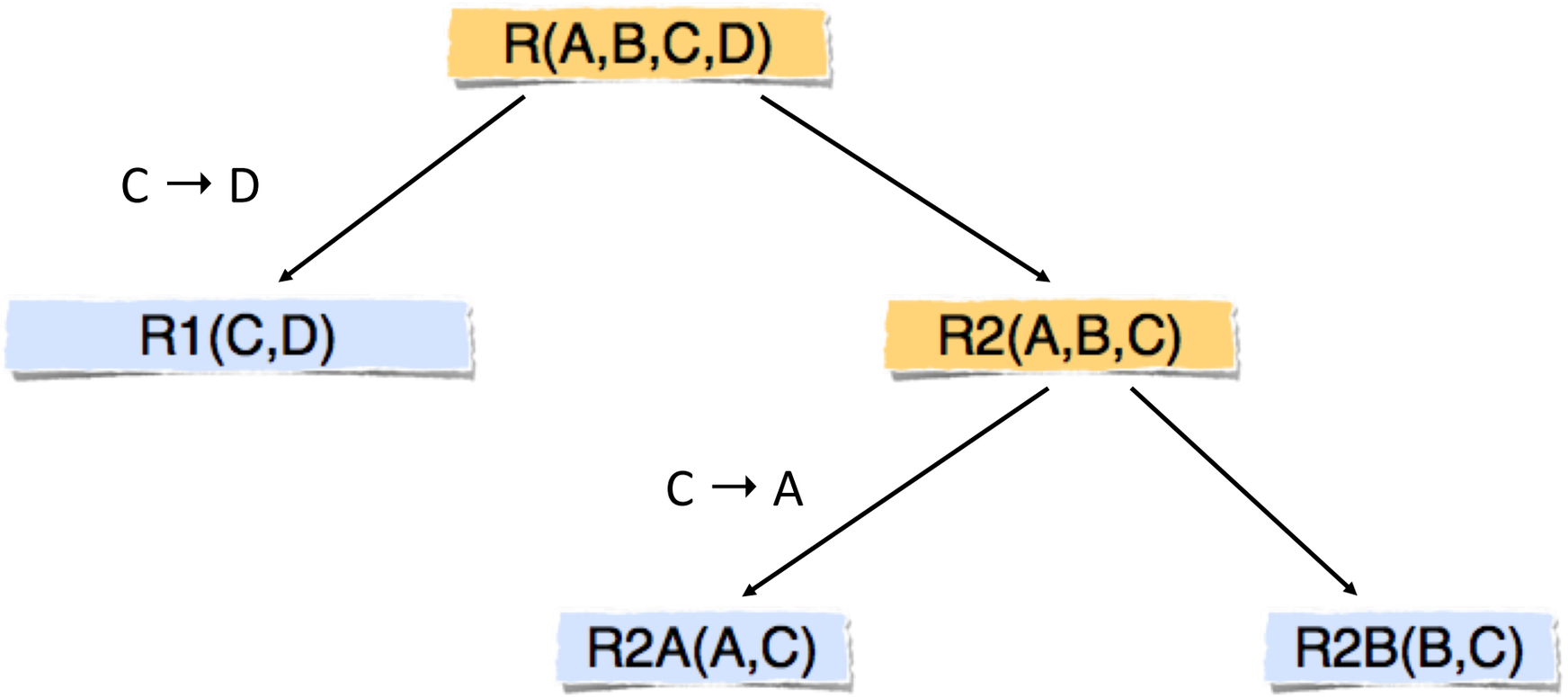
Solution 2

Given R(A, B, C, D)

FDs D → A
C → D
AB → C

Derived: C → A, AB → D, AC → D, BC → A,
BC → D, BD → A, BD → C, CD → A
ABC → D, ABD → C, BCD → A

Keys{AB}, {BC}, {BD}



Solution 2

Given R(A, B, C, D)

FDs D → A, C → D, AB → C

Derived: C → A, AB → D, AC → D, BC → A, BC → D, BD → A, BD → C, CD → A, ABC → D, ABD → C, BCD → A

Keys{AB}, {BC}, {BD}

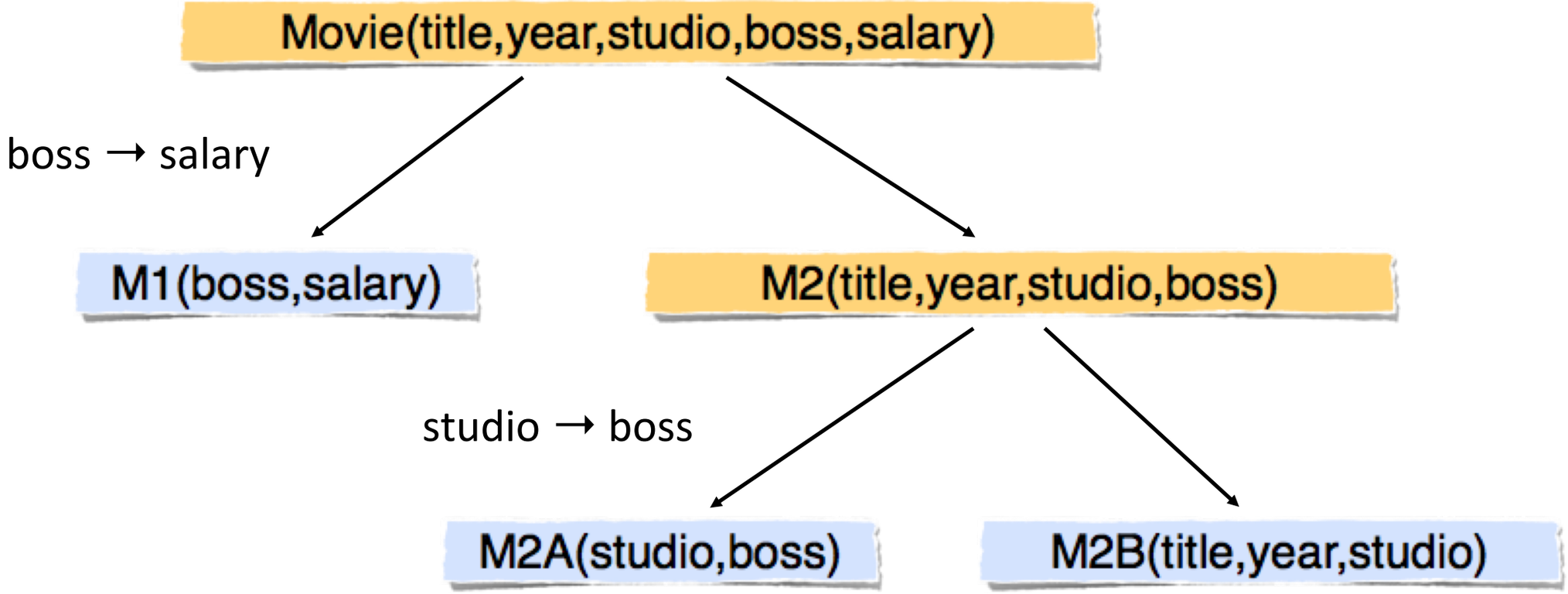
FD	Action	Relations
		R(A,B,C,D)
C → D	C is not a superkey for R with R1 and R2.	R1(C,D) i.e. LHS ∪ RHS R2(A,B,C) i.e. R-RHS
	Note: R1 is in BCNF, R2 isn't	
C → A	C is not a superkey for R2. Replace with R2A, R2B	R1(C,D) R2A(A,C) i.e. LHS ∪ RHS R2B(B,C) i.e. R-RHS
	R2A and R2B are in BCNF	Note: D → A and AB → C require joins

Exercise

Given	Movie(title, year, studio, boss, salary)
FDs	boss → salary
	studio → boss
	title year → studio
Key	{title, year}

Solution

Given	Movie(title, year, studio, boss, salary)
FDs	boss → salary
	studio → boss
	title year → studio
Key	{title, year}



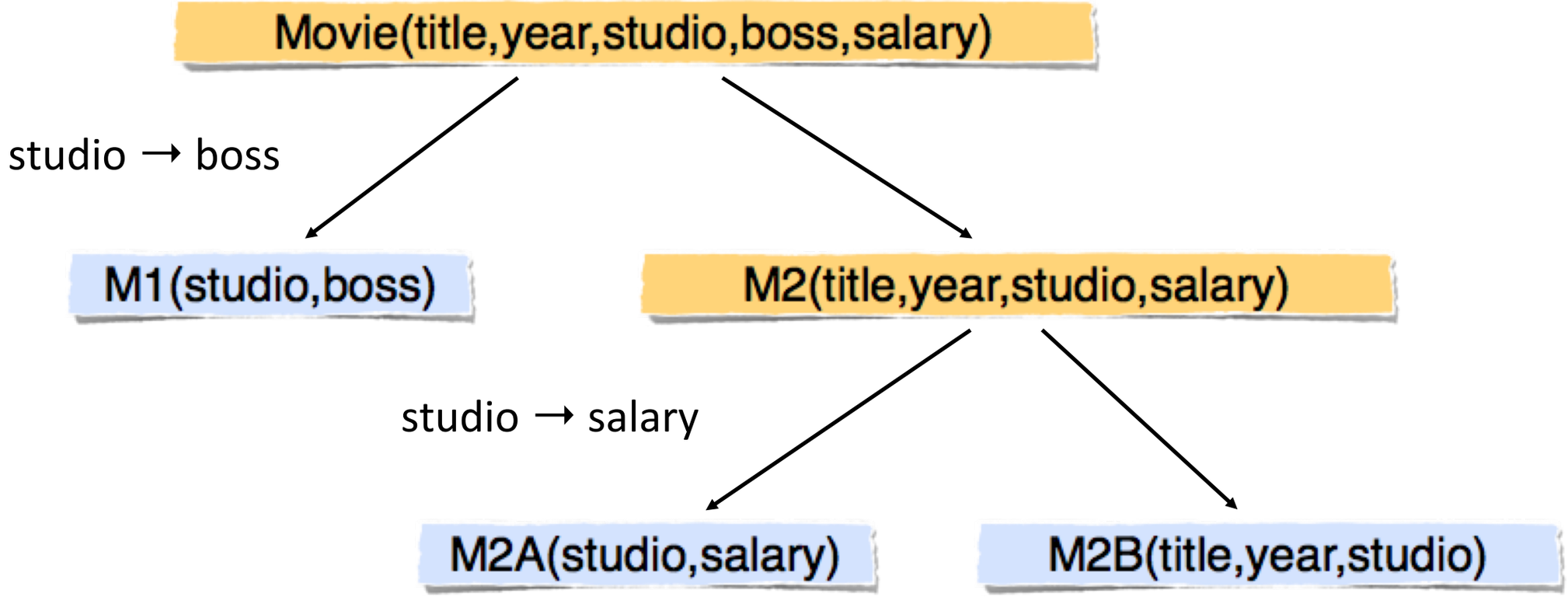
Solution

Given Movie(title, year, studio, boss, salary)
FDs boss → salary
studio → boss
title year → studio
Key {title, year} - can work out key by closing sets of 5 attributes

FD	Action	Relations
		Movie(title, year, studio, boss, salary)
boss → salary	boss is not a superkey, decompose Movie	M1(boss, salary) i.e. LHS ∪ RHS M2(title, year, studio, boss) i.e. R-RHS
	Note: M1 is in BCNF, M2 isn't	
studio → boss	studio is not a superkey for M2, so decompose M2.	M1(boss, salary) M2A(studio, boss) i.e. LHS ∪ RHS M2B(title, year, studio) i.e. R-RHS
	M2A and M2B are in BCNF	

Solution 2

Given	Movie(title, year, studio, boss, salary)
FDs	boss → salary
	studio → boss
	title year → studio
Key	{title, year}

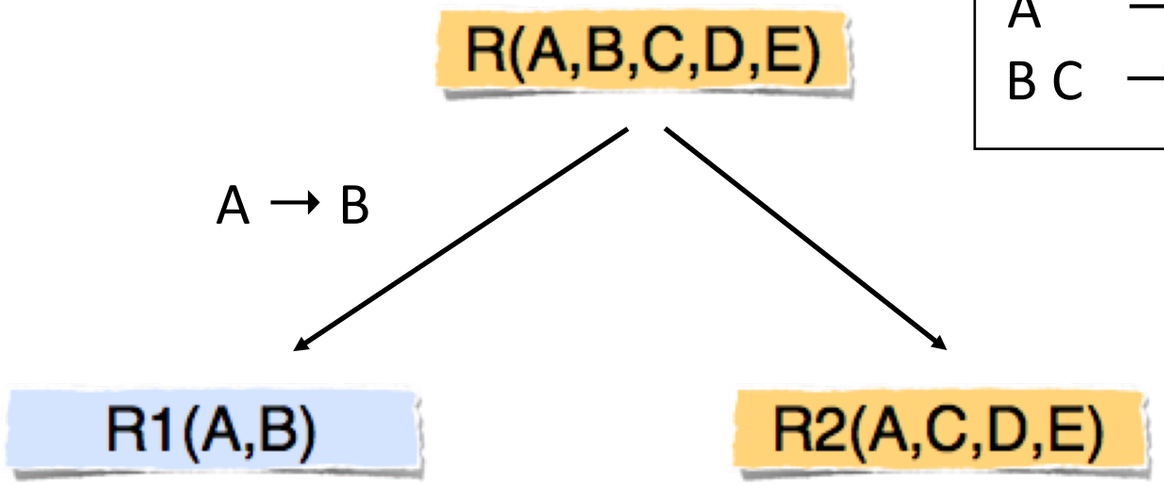


Solution 2

Given Movie(title, year, studio, boss, salary)
FDs boss → salary
studio → boss
title year → studio
Key {title, year}

FD	Action	Relations
		Movie(title, year, studio, boss, salary)
studio → boss	studio is not a superkey, decompose Movie	M1(studio, boss) i.e. LHS U RHS M2(title, year, studio, salary) i.e. R-RHS
	Note: M1 is in BCNF, M2 isn't	
studio → salary (inferred)	studio is not a superkey for M2, so decompose M2.	M1(studio, boss) M2A(studio, salary) i.e. LHS U RHS M2B(title, year, studio) i.e. R-RHS
	M2A and M2B are in BCNF	Note: boss → salary is not preserved requires a join

Another Example

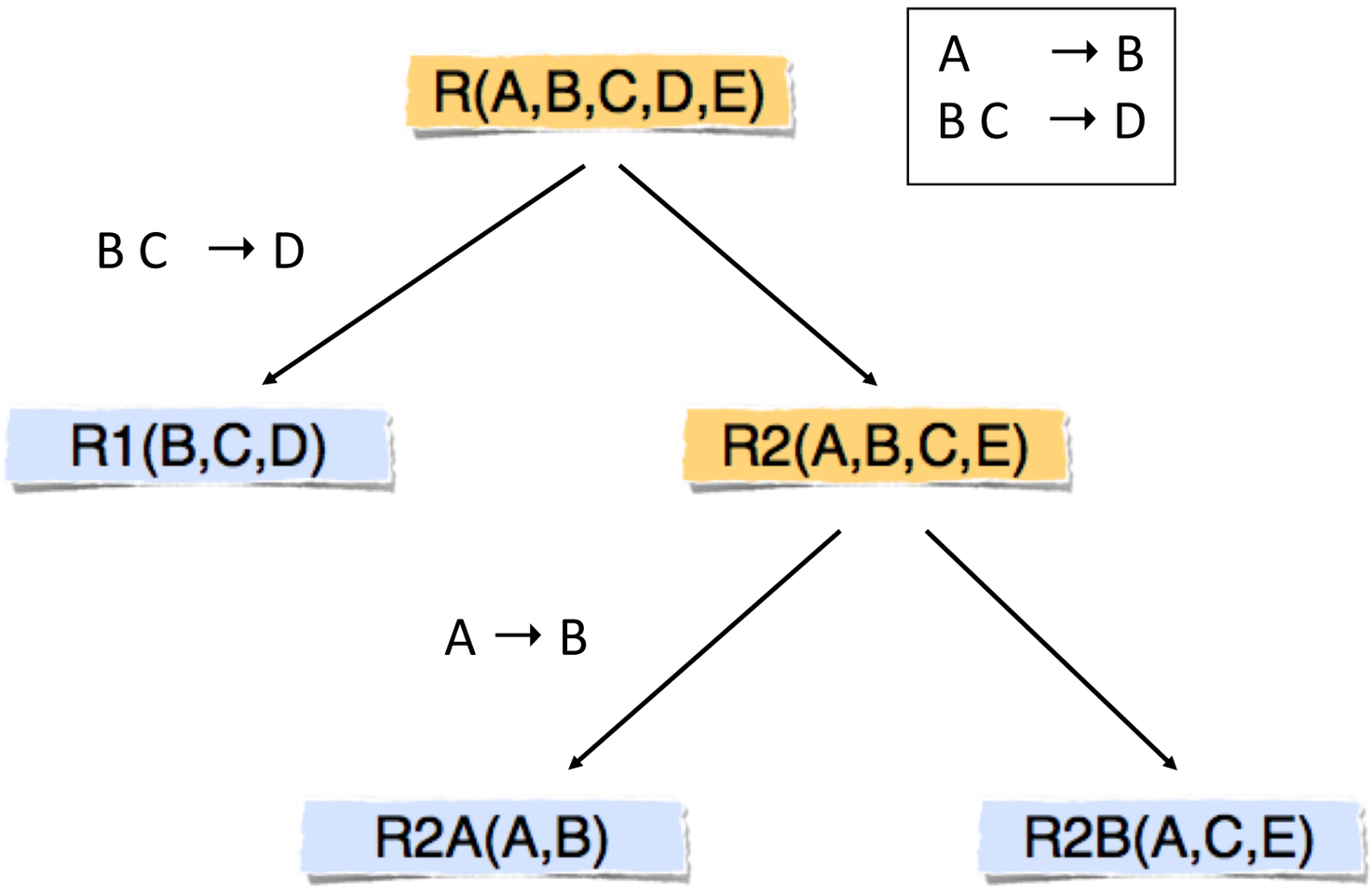


A → B
B C → D

But we can infer
A C → D (*pseudotransitivity*)
i.e. AC^+=ABCD doesn't include E
AC is not a key

Closures: A^+=AB, B^+=B, C^+=C, D^+=D
AB^+=AB, $\underline{AC}^+$=ABCD, AD^+=ABD, BC^+=BCD, BD^+=BD, CD^+=CD

Alternative Solution

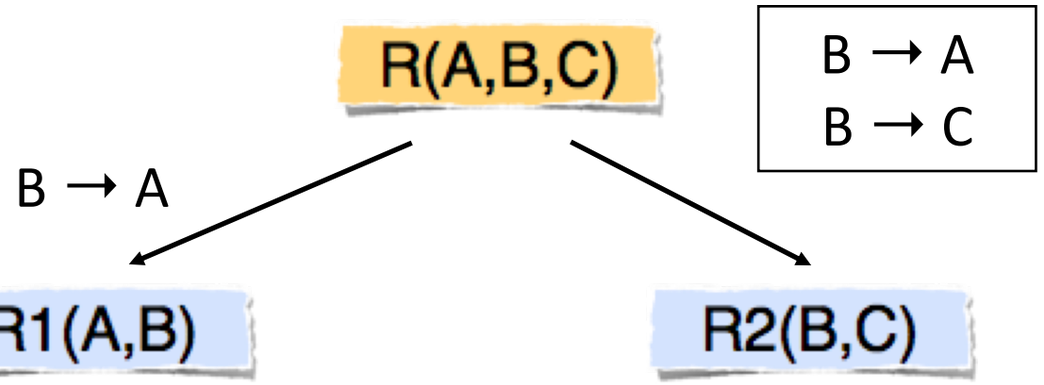


Closures: $A^+=AB$, $B^+=B$, $C^+=C$, $D^+=D$

$AB^+=AB$, $\underline{AC}^+=ABCD$, $AD^+=ABD$, $BC^+=BCD$, $BD^+=BD$, $CD^+=CD$

Recovering the Original Data

When decomposing into BCNF it is important that we can correctly recover the original relation by *joining* the decomposed relations. This is always possible provided the FDs hold for the original relation.



Provided the FDs hold for R then the decomposition R1, R2 is fine. For example:

R

A	B	C
1	2	3
4	5	6

$R1=\prod_{A,B}(R)$

A	B
1	2
4	5

⋈

$R2=\prod_{B,C}(R)$

B	C
2	3
5	6

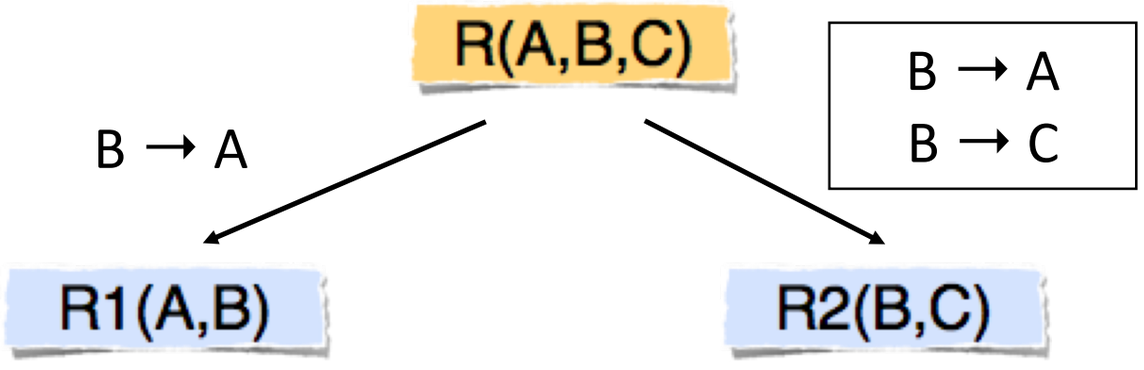
=

Join=R

A	B	C
1	2	3
4	5	6

Recovering the Original Data

However if the FDs don't hold we'll get spurious data. For example:



Key: A

The Join returns two spurious tuples (1,2,6) and (4,2,3)

R

A	B	C
1	2	3
4	2	6

$R1=\prod_{A,B}(R)$

A	B
1	2
4	2

$R2=\prod_{B,C}(R)$

B	C
2	3
2	6

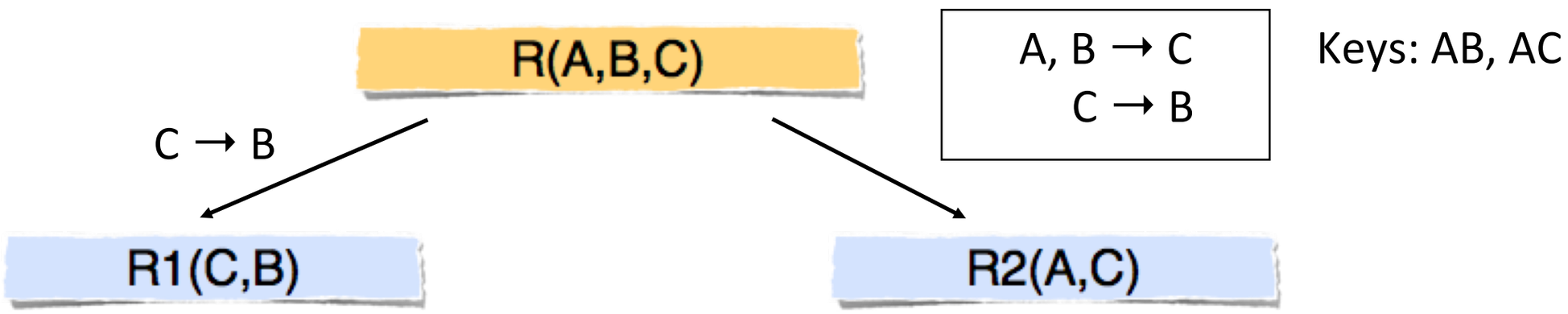
Join≠R

$\bowtie$ =

A	B	C
1	2	3
1	2	6
4	2	3
4	2	6

What about Dependency Preservation?

We have already seen that dependency preservation is not always possible with BCNF decomposition. The following small example illustrates this:



Although no FDs are violated in the decomposed relations, the FD A B → C is violated by the database as a whole, and the RDBMS would need to join the relations in order to check the FD.

Third Normal Form (3NF)

If a BCNF decomposition doesn't preserve the original FDs then we have two choices:

(1) "Live with" the violating dependencies.

(2) Use a weaker normal form that is lossless and preserves FDs but allows some redundancies (that we're happy to "live with").

Third Normal Form (3NF)

With 3NF, there is always a lossless, dependency preserving decomposition that we can do. Essentially it weakens (BCNF) so that we do not need to decompose in problematic situations.

Third Normal Form (3NF)

A relation R is in 3NF, if and only if, the LHS of every nontrivial FD is a superkey (the BCNF test) or if every attribute on the RHS of a FD is prime.

An attribute is **prime** if it is a member of **any key** of the relation. Each attribute on the RHS could be a member of a different key.

Consider the previous example:

R(A,B,C)

A, B → C
C → B

Keys: {A,B} {A,C}

Although A → C violates BCNF it does not violate 3NF because C is prime (C is a member of the key {A, C}

Decomposition into 3NF

Given a relation R and a set of FDs F for R, we can decompose R into a set of decomposed relations D (each of which is in 3NF) as follows:

Let C **be a canonical cover for** F **(i.e. a minimal FD set for R)**

Initialise the set of decomposed relations D **to {}**

Foreach FD: LHS → RHS in C
    add a new relation(LHS ∪ RHS) to the set of decomposed relations D

Foreach relation R in D that is a subset of another relation in D
    remove R from D

If none of the relations in D includes a key for R
    add a new relation(key) to D

Example

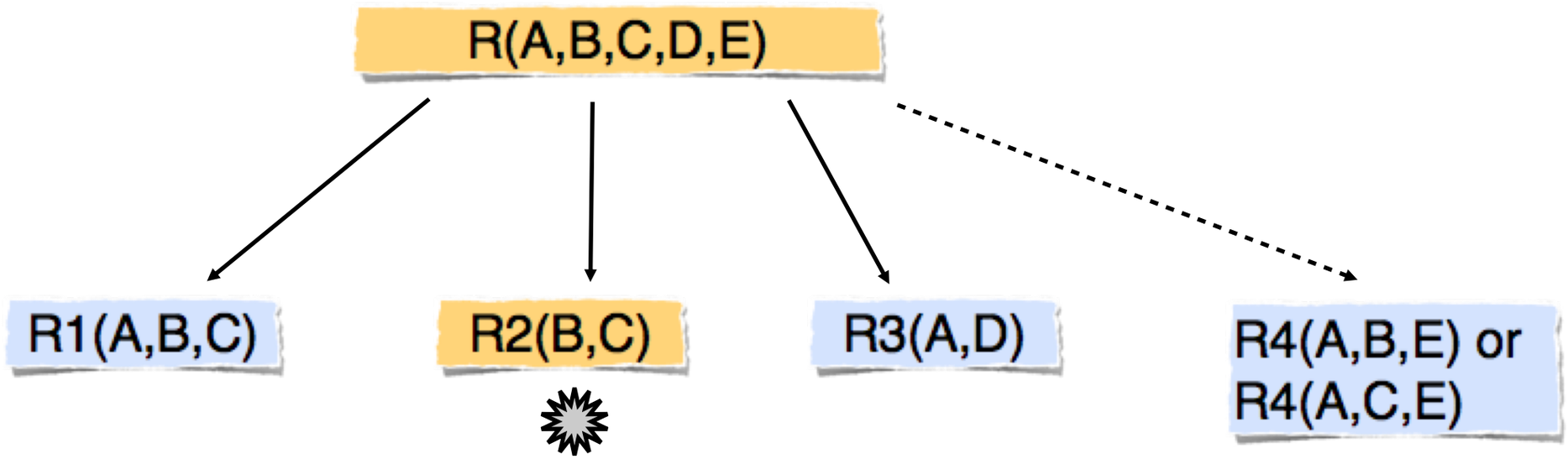
Decompose R(A,B,C,D,E)
into 3NF

FD Set

$A\ B \rightarrow C$
$C \rightarrow B$
$A \rightarrow D$

Keys

{A, B, E}
{A, C, E}



Exercise

Decompose R(A,B,C,D)
into 3NF

FD Set

A B	→	C D
B	→	C
A C	→	B

Keys

{A, B}
{A, C}

Solution

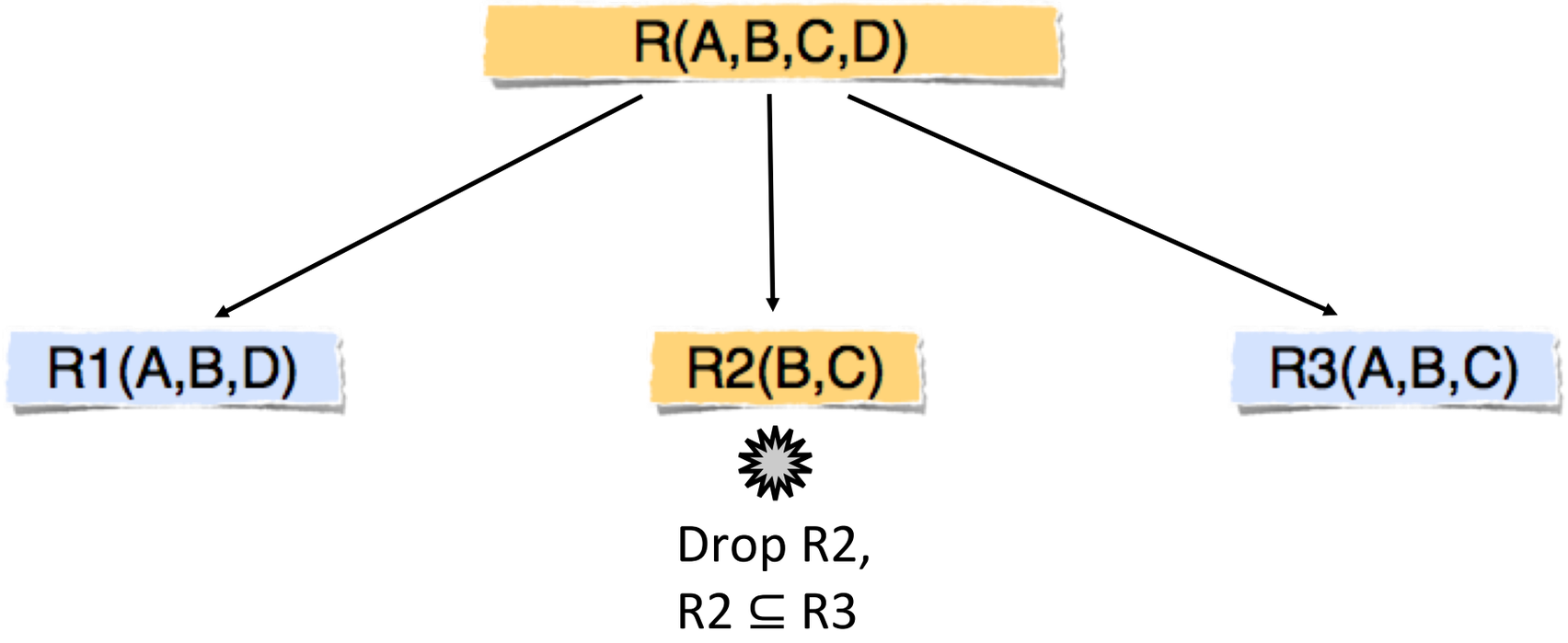
Decompose R(A,B,C,D) into 3NF

Canonical cover

A B	→	⊄ D
B	→	C
A C	→	B

Keys

{A, B}
{A, C}



Exercise

Decompose R(A,B,C,D,E,F,G,H,I)
into 3NF

FD Set

A B → D E
A → I
B → C F G H
G → H
A D → B C E F G H
B D → A I E

Keys

{A, B}
{A, D}
{B, D}

Solution

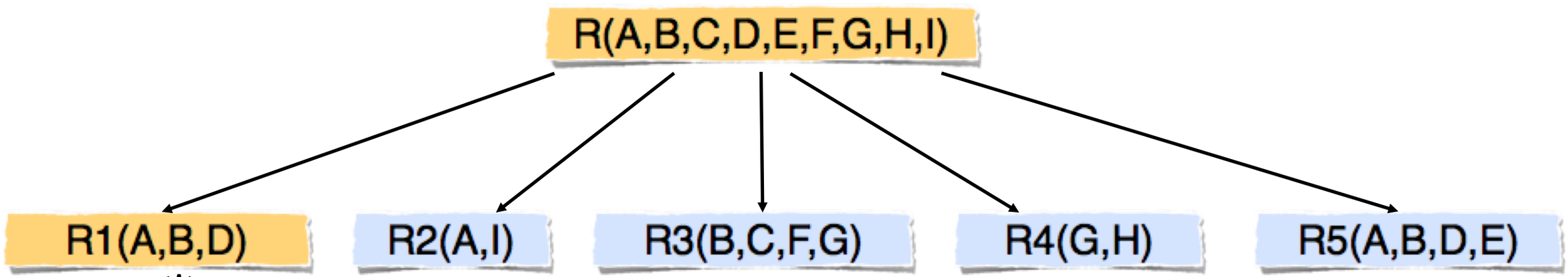
Decompose R(A,B,C,D,E,F,G,H,I)
into 3NF

Canonical Cover

$A\,B \rightarrow D$ ~~E~~
$A \rightarrow I$
$B \rightarrow C\,F\,G$ ~~H~~
$G \rightarrow H$
$A\,D \rightarrow B$ ~~C~~ ~~E~~ ~~F~~ ~~G~~ ~~H~~
$B\,D \rightarrow A$ ~~I~~ E

Keys

{A, B}
{A, D}
{B, D}



Drop R1,
R1 ⊆ R5

Exercise

Decompose
R(A,B,C,D,E,F,G)
into 3NF

FD Set

A B	→ C D E F G
E	→ F
B E	→ G
B C G	→ A D E F
B C E	→ A D

Keys

{A, B}
{B, C, G}
{B, C, E}

Solution

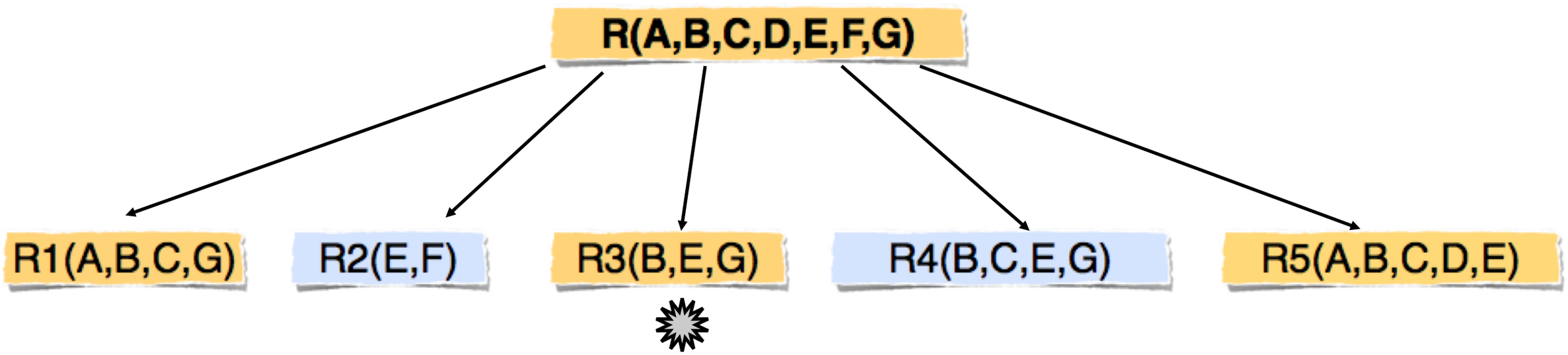
Decompose
R(A,B,C,D,E,F,G)
into 3NF

Canonical cover

A B	→	C ~~D~~ ~~E~~ ~~F~~ G
E	→	F
B E	→	G
B C G	→	~~A~~ ~~D~~ E ~~F~~
B C E	→	A D

Keys

{A, B}
{B, C, G}
{B, C, E}



Solution 2

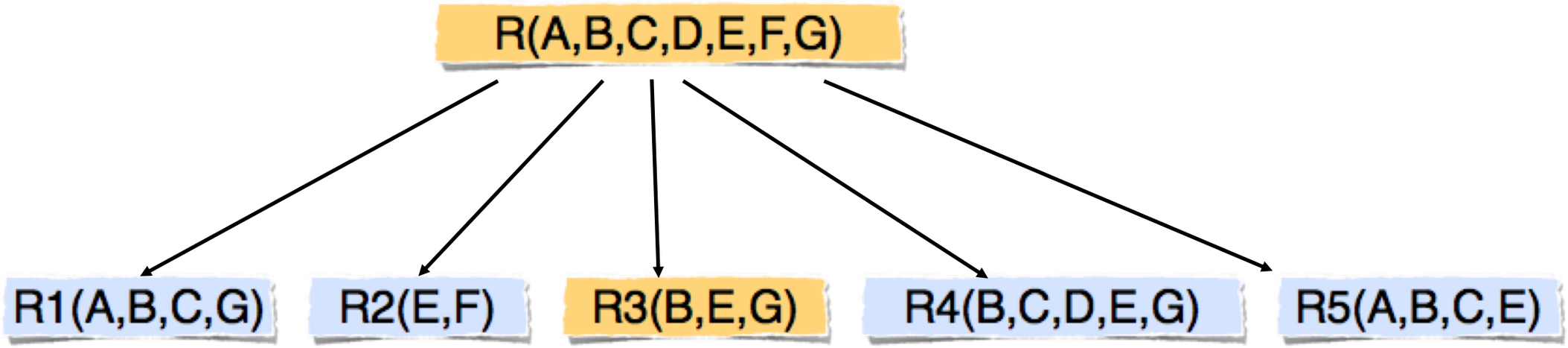
Decompose
R(A,B,C,D,E,F,G)
into 3NF

Alternative Canonical cover

$$
\begin{aligned}
A\,B &\rightarrow C \not\emptyset \not\in \not\in G \\
E &\rightarrow F \\
B\,E &\rightarrow G \\
B\,C\,G &\rightarrow \not A\,D\,E \not F \\
B\,C\,E &\rightarrow A \not\emptyset
\end{aligned}
$$

Keys

{A,B}
{B,C,G}
{B,C,E}



Drop R3,
R3 ⊆ R4

Other Normal Forms

First Normal Form (1NF)	Every value is indivisible (atomic), that is we do not operate on parts of the value. No lists, arrays, nested relations etc. Strings, Dates, Times are okay provided they are considered as a whole - for example a name “John Smith” is okay provided we are not interested in Firstname or Lastname.
Second Normal Form (2NF)	A weaker form of 3NF
Fourth Normal Form (4NF)	A stronger form of BCNF that handles so-called Multi-valued Dependencies (MVDs), constraints that two or more attributes are independent of each other.
Fifth Normal Form (5NF)	Also known as Project-Join Normal Form (PJFN) that handles so-called join dependencies. Nearly all 4NF relations are also in 5NF.
Sixth Normal Form (6NF)	Also known as Domain-Key Normal Form(DKNF). Primarily used for temporal or interval data.

In general, each “higher” normal form satisfies the constraints for a lower one, i.e. 4NF satisfies BCNF which satisfies 3NF and 2NF and 1NF. Higher forms are harder to reason about however.

2NF vs BCNF

Properties	3NF	BCNF	4NF
Eliminates redundancy due to FDs	✗	✓	✓
Guarantees to preserve FDs	✓	✗	✗
Eliminates redundancy due to Multi-valued Dependencies (MVD)	✗	✗	✓
Guarantees to preserve MVDs	✗	✗	✗

When designing a relational database, go for BCNF to eliminate redundancies due to FDs, and 5NF to eliminate redundancies due to MVDs (and join dependencies).

Exercise

For the relation R(S,T,U,V,W,X) and set of functional dependencies F:

$S \rightarrow T\ U\ V$
$T \rightarrow V$
$V \rightarrow S$
$T\ U \rightarrow V\ W$

1. Give a BCNF decomposition of R.
2. Give a canonical cover for F.
3. Give a 3NF decomposition of R.

Solution 1. Decomposition into BCNF (1)

Relation R(S,T,U,V,W,X)

FD Set

$S \rightarrow T\ U\ V$
$T \rightarrow V$
$V \rightarrow S$
$T\ U \rightarrow V\ W$

Keys Let's work out the keys first. The closures of the single attributes are

$\{S\}^+ = \{S,T,U,V,W\}$
$\{T\}^+ = \{S,T,U,V,W\}$
$\{U\}^+ = \{U\}$
$\{V\}^+ = \{S,T,U,V,W\}$
$\{W\}^+ = \{W\}$
$\{X\}^+ = \{X\}$

So we need to augment S, T and V with X to get two attribute keys
{S,X},{T,X},{V,X}

Solution 1. Decomposition into BCNF (2)

Relation R(S,T,U,V,W,X)

FD Set S → T U V, T → V, V → S, T U → V W

Keys {S,X},{T,X},{V,X}

Split R We could try to work out the derived FDs but they are very many and they're not needed here since all the given FDs are violating.
So let's pick one, e.g. S → T U V.

This will split **R** into **R1(S,T,U,V)** and **R2(S,W,X)**

We now need to apply the BCNF decomposition to both **R1** and **R2** in turn.

Solution 1. Decomposition into BCNF (3)

Relation R1(S,T,U,V)

FD Set

S → T U V
T → V
V → S
T U → V W

We can't use this since W is not in R1

Keys Let's work out the keys again. The closures of the single attributes are:

$\{S\}^+ = \{S,T,U,V\}$
$\{T\}^+ = \{S,T,U,V\}$
$\{U\}^+ = \{U\}$
$\{V\}^+ = \{S,T,U,V\}$

So the keys are {S}, {T} and {V}

Solution 1. Decomposition into BCNF (4)

Relation R1(S,T,U,V)

FD Set
S → T U V
T → V
V → S

Keys {S}, {T} and {V}

FDs None of these FDs are violating. What about the derivable FDs? The issue with derivable FDs is, that they can be very many of them, so we'd like to avoid enumerating them. If we look at the given FDs any derivable FD with one of the keys on the LHS will not be violating since its LHS will be a superkey. The only possibility for violation is to have an FD that includes U on the LHS. However the remaining attributes are keys so any derivable FD with U on the LHS will be a superkey therefore will not be a violating. **So R1 is in BCNF**. What about R2?

Solution 1. Decomposition into BCNF (5)

Relation R2(S,W,X)

FD Set

S → T U V
T → V
V → S
T U → V W

All of these FDs have attributes that are not in R2. But what about the derivable FDs? Are there any derivable FDs with only S, W and X?

Yes, S → W can be derived from the FD set.

Let's work out the keys again. The closures of the single attributes are:

$\{S\}^+ = \{S,W\}$

$\{W\}^+ = \{W\}$

$\{X\}^+ = \{X\}$

From this we can see that the only key is {S, X} making S → W a violating FD. So let's split R2(S,W,X) into R2A(S,W) and R2B(S,X)

Final Decomposition **R1(S,T,U,V)**, **R2A(S,W)**, **R2B(S,X)**

Solution 2. 3NF

Relation R(S,T,U,V,W,X)

Cover **S → T U, V → S, T → V W**

Canonical cover from 2d.

3NF Decompose into relations based on the FDs in the canonical cover:

R1(S,T,U), R2(S,V), R3(T,V,W)

Since none of these relations includes a key, we need to add one, e.g. {S,X} to give our 3NF decomposition

R1(S,T,U), R2(S,V), R3(T,V,W), R4(S,X)