R: relation schema

r(R): relations

F: set of functional dependencies

α ⊆ R and β ⊆ R

attribute: A, B, …

R1, R2 , .., Rn: the decomposition of R

F1, F2, .., FN: all functional dependencies in **F+** that include only attributes of Ri

functional dependency: α → β

t1[α] = t2 [α] ⇒ t1[β ] = t2 [β ]

trival: β ⊆ α

ab->a

a->a

# key

K superkey: K → R

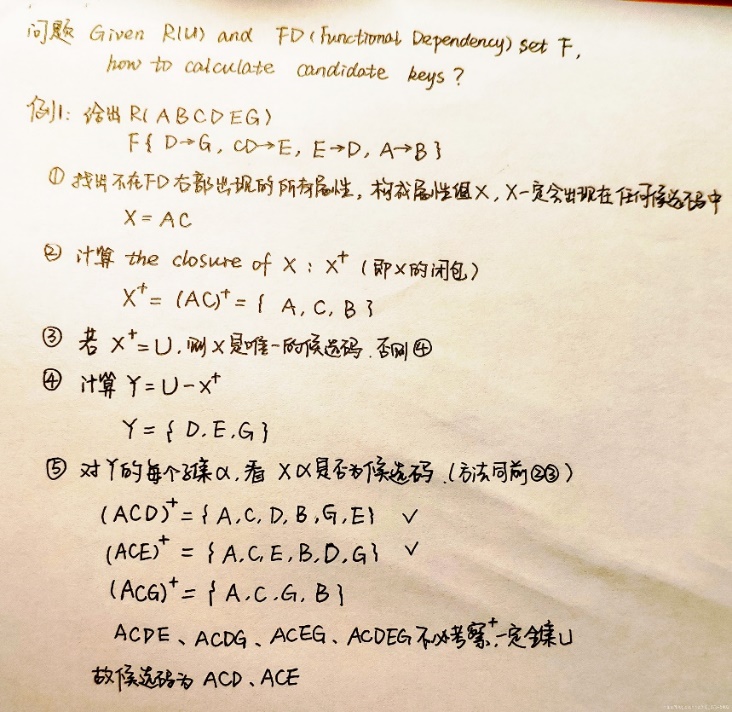
K candidate key: K → R

no α ⊂ K, α → R

candidate: AG

superkey: AG, AGC, AGB, …

superkey: (K)+ ⊇ R



# lossless decompostion

lossless decomposition:

(1) ∏ R1 (r) ∏ R2 (r) = r

(2) **F+**

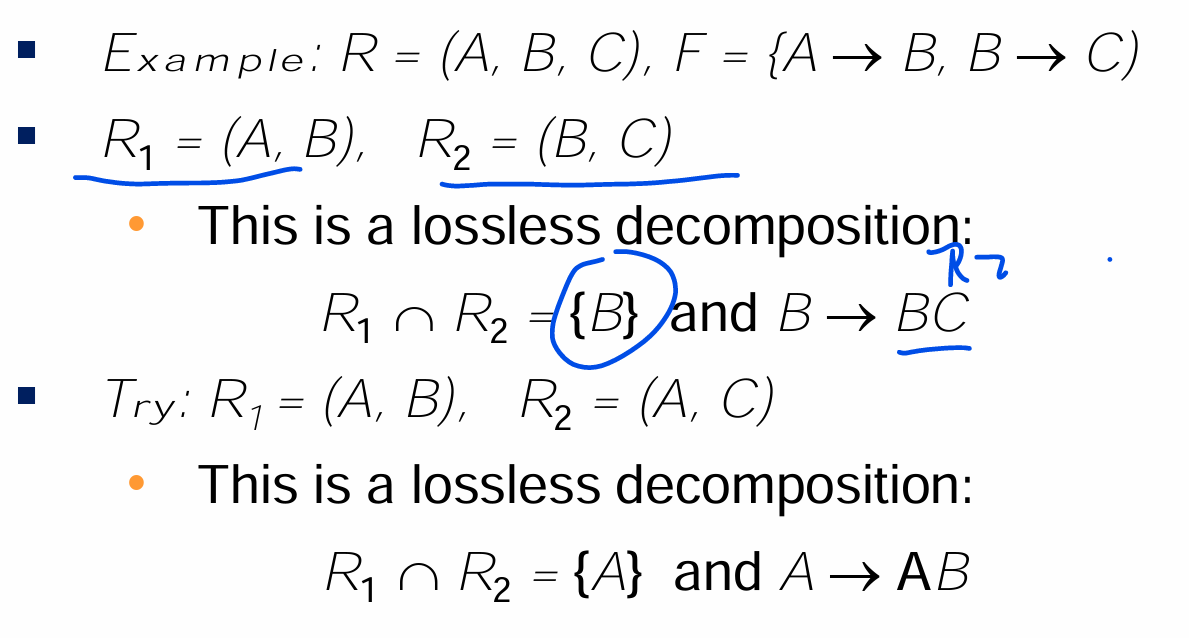
one of:

R1 ∩ R2 R1

R1 ∩ R2 R2

R1∩ R2 = {xx}

xx xx yy (R1 or R2)



# Dependency Preservation

(1) (F1 ∪ F2 ∪ … ∪ Fn )+ = F +

(2)

|  |
| --- |
| for each dependency α → β in **F**:  result = α  for each Ri in the decomposition  t = (result ∩ Ri)+ ∩ Ri  result = result ∪ t  if result ⊇ β: α → β is preserved |

(result ∩ Ri)+ ∩ Ri: 只在Ri里出现的result，的closure，取只在Ri里出现的部分？

找

example1

R (ABCDE)

R1 (A, B, C)

R2 (A, D, E).

F {A → BC, CD → E, B → D, E → A}

1. A → BC

result = A

(1) R1(ABC) 找

t = (A ∩ ABC)+ ∩ ABC = A+ ∩ ABC = ABCDE ∩ ABC = ABC

result = A ∪ ABC = ABC

(2) R2(ADE) 找

t = (ABC ∩ ADE)+ ∩ ADE = A+ ∩ ADE = ABCDE ∩ ADE = ADE

result = ABC ∪ ADE = ABCDE

result = ABCDE contains BC, A → BC preserved

2. CD → E

result = CD

(1) R1(ABC) ) 找

t = (CD ∩ ABC)+ ∩ ABC = C+ ∩ ABC = C ∩ ABC = C

result = CD ∪ C = CD

(2) R2(ADE) ) 找

t = (CD ∩ ADE)+ ∩ ADE = D+ ∩ ADE = D ∩ ADE = D

result = CD ∪ D = CD

result = CD not contain E, CD → E not preserved

…

example2

# Closure

Armstrong’s Axioms:

• Reflexive rule: β ⊆ α => α → β

• Augmentation rule: α → β => γ α → γ β

• Transitivity rule: α → β & β → γ => α → γ

Additional rules:

• α → β & α → γ <=> α → β γ

• α → β & γ β → δ => α γ → δ

---------------------

a->b, b->c: a->c

a->b, a->c: b->c (X)

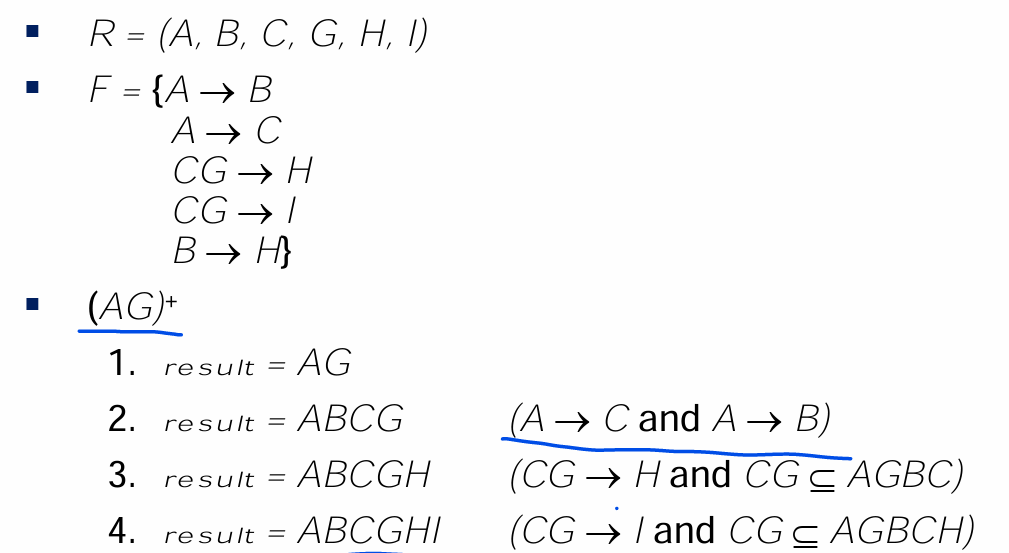
a->b, a->c: a->bc

ab->c: a->c, b->c (X)

----------------------

**Attribute Closure**

α+: closure of α under **F**



use:

Testing for superkey:

α is superkey: α+ ⊇ R

Testing functional dependencies:

α → β: α+ ⊇ β

Computing closure of F:

for each γ ⊆ R:

find γ+

for each S ⊆ γ+:

output γ → S

# Extraneous Attributes

An attribute of a functional dependency in F is extraneous if we can remove it without changing F +

Left (α → β, A ∈ α)

F

(α-{A}) + ⊇ β

|  |
| --- |
| **F**: {…, AX -> UW}  a: {AX}, b: {UW}  X UW ? |

example:

R = (A, B, C)

F = {A → BC, B → C, AB → C}

A is extaneous in AB → C?

B C

Right (α → β, B ∈ β)

F' = (F– {α → β}) ∪ {α →(β– B)}

α + ∋ B

|  |
| --- |
| F: {…, XY -> BW}  a: {XY}, b: {BW}  **F’**: {…, XY -> W}  XY B ? |

example:

R = (A, B, C)

F = {A → BC, B → C}

C is extaneous in A → BC?

A C

# Cononical Cover

Each **left** side of functional dependency in Fc is **unique**.

repeat:

1. replace a->b1 & a->b2, with a->b1 b2

2. A is extraneous in x->y: delete

example:

R = (A, B, C)

F = {A → BC, B → C, A → B, AB → C}

(1) replace A->BC & A->B, with A->BC

F = {A → BC, B → C, AB → C}

(2) A is extaneous in AB → C?

B C

F = {A → BC, B → C, B->C}

replace B → C & B →C, with B → C: F = {A → BC, B → C}

(3) C is extaneous in A → BC?

A C

F = {A → B, B → C}

a->b1 & a->b2 ?

Fc = {A → BC, B → C}

# BCNF

R is BCNF:

for all α →β in **F+**, where α ⊆ R and β ⊆ R,

one of:

α→β is trivial (i.e., β ⊆ α)

α is superkey for R

simplified: α →β in **F**

FD: α → β causes violation of BCNF

decompose R into: • (α U β ) • ( R - ( β - α ) )

**Ri**

**α β, α not superkey(of Ri), and α ∩ β = ∅**

**result := (result - Ri ) ∪ (Ri – {β}) ∪ {α, β}**

example1 Ch7.63

R(ABCDEFGHIJK)

F { A->BCD, HI->J, AEFG->HIK }

1.

(1) R satisifies BCNF? No

candidate key: AEFG

**A->BCD: A not superkey, A ∩ BCD = ∅**

(2) decomposition:

**S1(ABCD)**

**R1(AEFGHIJ),**

**F1{ HI->J, AEFG ->HIK }**

2.

(1) S1 satisifies BCNF?

candidate key: A

R1 satisifies BCNF? No

candidate key: AEFG

**HI->J: HI not superkey, A ∩ BCD = ∅**

(2) decomposition:

S1(ABCD)

**S2(HIJ)**

**R2(AEFGHI),**

**F2{ AEFG ->HIK }**

3.

(1) S1 satisifies BCNF?

S2 satisifies BCNF?

candidate key: HI

R1 satisifies BCNF? Yes

candidate key: AEFG

(ABCD)(HIJ)(AEFGHI)

example2 Tutor9

R(ABCDE)

F { A->C, C->D, B->C, DE->C, CE->A }

1.

(1) R satisifies BCNF?

candidate key: BE

**A->C: A not superkey, A ∩ C = ∅**

(2) decomposition:

**S1(AC)**

**R1(ABDE),**

**F1{ A->D, B->D, BE->A, DE->A }**

|  |  |
| --- | --- |
| A->C  B->C  DE->C | C->D  CE->A |

2.

(1) R1 satisifies BCNF?

candidate key: BE

**A->D: A not superkey, A ∩ D = ∅**

(2) decomposition:

S1(AC)

**S2(AD)**

**R2(ABE),**

**F2{ BE->A }**

|  |  |
| --- | --- |
| A->D  B->D | DE->A |

3.

(1) R2 satisifies BCNF?

candidate key: BE

(AC)(AD)(ABE)

example3 Ass 7.21

R(ABCDE)

F { A->BC, CD->E, B->D, E->A }

1.

(1) R satisifies BCNF?

candidate key: A, E, CD, BC

A->BC: A superkey

CD->E: CD superkey

**B->D: B not superkey, A ∩ BCD = ∅**

(2) decomposition:

**S1(BD)**

**R1(ABCE),**

**F1{ A->BC, BC->E, E->A }**

2.

(1) R1 satisifies BCNF?

candidate key: A, E, BC

A->BC: A superkey

BC->E: BC superkey

E->A: E superkey

(BD)(ABCD)

example4 Ass 7.30

R(ABCDEG)

F { A->BCD, BC->DE, B->D, D->A }

1.

(1) R satisifies BCNF?

candidate key: AG, BG, DG

**A->BCD: A not superkey, A ∩ BCD = ∅**

(2) decomposition:

**S1(ABCD)**

**R1(AEG)**

**F1{A->E }**

(i)

B { A->BCD, BC->DE, B->D, D->A }

|  |  |
| --- | --- |
| A->BCD  A->B  A->CD | BC->DE  B->D |

C { A->CD, AC->DE, D->A }

|  |  |
| --- | --- |
| A->CD  A->C  A->D | AC->DE |

D {A->E}

|  |  |
| --- | --- |
| A->DE  A->D  A->E | D->A |

(ii)

直接看：F左边没有E，F没有G

2.

(1) R1 satisifies BCNF?

candidate key: AG

**A->E: A not superkey, A ∩ E = ∅**

(2) decomposition:

S1(ABCD)

**S2(AE)**

**R2(AG)**

**F2{?}**

3.

(1) R2 satisifies BCNF? Yes

(ABCD)(AE)(AG)

# 3NF

R is 3NF:

for all α →β in **F+**, where α ⊆ R and β ⊆ R,

one of:

α→β is trivial (i.e., β ⊆ α)

α is superkey for R

each attribute A in β – α is contained in a candidate key for R

simplified: α →β in **F**

candidate key: AC

C->B:

(1) C not superkey

(2) α: C

β: B

β – α = {B} – {C} = {B}

B is contained in AC

example1 Ass 7.22

R(ABCDE)

F { A->BC, CD->E, B->D, E->A }

(1) candidate key: A, E, CD, BC

(2) Find Fc:

a->b1 & a->b2 ?

A->BC

B: F’ = { A->C, CD->E, B->D, E->A }

A+ = AC B ∉ A

F = { A->BC, CD->E, B->D, E->A }

C: F’ = { A->B, CD->E, B->D, E->A }

A+ = ABD C ∉ A

F = { A->BC, CD->E, B->D, E->A }

CD->E

C: D+ = D {E} ⊈ D

F = { A->BC, CD->E, B->D, E->A }

D: C+ = C {E} ⊈ C

F = { A->BC, CD->E, B->D, E->A }

Fc = { A->BC, CD->E, B->D, E->A }

(3) decomposition: (ABC)(CDE)(BD)(AE)

(4) exists Ri contains candidate key? Yes

(ABC) yes

(CDE) yes

(BD) no

(AE) yes

(5) Ri contains / subset Rj ? No

example2 Tutor9

R(ABCDEG)

F { A->BCD, BC->DE, B->D, D->A }

(1) candidate key: AG, BG, DG

(2) Find Fc:

a->b1 & a->b2?

A->BCD

B: F’ = { A->CD, BC->DE, B->D, D->A }

A+ = ACD B ∉ ACD

F = { A->BCD, BC->DE, B->D, D->A }

C: F’ = { A->BD, BC->DE, B->D, D->A }

A+ = ABD C ∉ ABD

F = { A->BCD, BC->DE, B->D, D->A }

D: F’ = { A->BC, BC->DE, B->D, D->A }

A+ = ABCDE D ∈ ABCDE

F = { A->BC, BC->DE, B->D, D->A }

a->b1 & a->b2 ?

BC->DE

C: B+ = ABCDE {DE} ⊆ ABCDE

F = { A->BC, B->DE, B->D, D->A }

replace B->DE & B->D with B->DE: F = { A->BC, B->DE, D->A }

B->DE

D: F’ = { A->BC, B->E, D->A }

B+ = BE D ∉ BE

F = { A->BC, B->DE, D->A }

E: F’ = { A->BD, B->D, D->A }

B+ = ABD E ∉ ABD

F = { A->BC, B->DE, D->A }

Fc = { A->BC, B->DE, D->A }

(3) decomposition: (ABC)(BDE)(AD)

(4) exists Ri contains candidate key? No

(ABC) no

(BDE) no

(AD) no

decomposition: (ABC)(BDE)(AD) (AG)

(5) Ri contains / subset Rj ? No

example3 Ch7.70

R (ABCD)

F = {AB->CD, B->C, AC->B}

(1) candidate key: AB, AC

(2) Fc = {AB->D, B->C, AC->B}

(3) decomposition: (ABD)(BC)(ABC)

(4) candidate key be contained? Yes

(ABD) yes

(BC) no

(ABC) yes

(5) Ri contains / subset Rj ? Yes --- (ABC) contains (BC)

decomposition: (ABD) (ABC)