텍스트이(가) 표시된 사진

자동 생성된 설명

A decomposition of R into R 1 and R 2 is lossless join if at least one of the following dependencies is in F+

One of them is *R*1∩*R*2→*R*1.

In R1(A, B, C) and R2(A, D, E), *R*1∩*R*2 is a. If A->BC, then A->ABC with augmentation.

So it satisfies *R*1∩*R*2→*R*1.

텍스트이(가) 표시된 사진

자동 생성된 설명

7.4

Union rule : If α -> β holds and α -> γ holds, then α -> βγ holds

α -> β

αα -> αβ with augmentation rule

α -> αβ union of identical sets

α -> γ

αβ -> γβ with augmentation rule

α -> γβ with transitivity (if α -> αβ and αβ -> γβ, then α -> γβ)

7.5

Pseudo-transitivity rule : If α -> β holds and βγ -> δ holds, then αγ -> δ holds

αγ -> βγ (augmentation with γ)

βγ -> δ is given

So we can see αγ -> δ (with transitivity)

텍스트이(가) 표시된 사진

자동 생성된 설명

(A)+

1. result = A

2. result = ABC (A->BC)

3. result = ABCD (B->D)

4. result = ABCDE (CD->E)

So A can be a candidate key

(B)+

1. result = B

2. result = BCD (B->D)

“B” can’t be a candidate key since it doesn’t contain everything in schema R.

So we find something with B. For a, it already can be a candidate key. Now we find out a candidate key containing B.

(BC)+

1. Result = BC

2. result = BCD (B->D)

3. result = BCDE (CD->E)

4. result = ABCDE (E->A)

(BD)+

1. Result = BD

(BE)+

1. result = BE

2. result = BDE (B->D)

3. result = ABDE (E->A)

4. result = ABCDE (A->BC)

We can see BC and BE can be a candidate key.

C+

1. Result = C

(CD)+

1. result : CD

2. result : CDE (CD->E)

3. result : ACDE (E->A)

4. result : ABCDE (A->BC)

(CE)+

1. Result : CE

2. result : ACE (E->A)

3. result : ABCE (A->BC)

4. result : ABCDE (B->D)

Now we know CD and CE can be a candidate key.

E+

1. E

2. AE (E->A)

3. ABCE (A->BC)

4. ABCDE (B->D)

A, BC, BE, CD, CE, E are the candidate keys. But BE and CE can be denoted by E. So, A, BC, CD, E are the candidate keys.

텍스트이(가) 표시된 사진

자동 생성된 설명

First, to test if attribute Y is extraneous on the first functional dependence, We remove Y from the first one.

FD` = {X->Z, Y->XZ, Z->XY}

X+->X,Y,Z

To test if attribute Z is extraneous on the first functional dependence, We remove Z from the first one.

FD` = {X->Z, Y->X, Z->XY}

Y+ = {X, Y,Z}

Now we remove X from the first one to see if it’s extraneous.

FD` = {X->Z, Y->X, Z->Y}

Z+ = {X,Y,Z}

{X->Z, Y->X, Z->Y} is one of the canonical covers.

Now we find another canonical cover.

First, to test if attribute Z is extraneous on the first functional dependence, We remove Y from the first one.

FD` = {X->Y, Y->XZ, Z->XY}

X+->X,Y,Z

To test if attribute X is extraneous on the first functional dependence, We remove X from the first one.

FD` = {X->Y, Y->Z, Z->XY}

Y+ = {X, Y,Z}

Now we remove Y from the first one to see if it’s extraneous.

FD` = {X->Y, Y->Z, Z->X}

Z+ = {X,Y,Z}

{X->Y, Y->Z, Z->X} can be a canonical cover, too.

This shows there can be more than one canonical cover for this given set of functional dependencies.

텍스트이(가) 표시된 사진

자동 생성된 설명

a)

B+

1. Result : BD (B->D)]

2. result : ABD (D->A)

3. result : ABCD (A->BCD)

4. result : ABCDE (BC->DE)

b)

A->BCD

A->ABCD (with augmentation with A)

ABCD -> ABCD (with augmentation with BCD)

ABCD -> ABCDE (with BC->DE)

A->ABCD, ABCD -> ABCDE then A->ABCDE (transitivity)

AG -> ABCDEG (augmentation with G)

c)

if we assume D is extraneous. Then,

A->BC

BC->E

B->D

D->A

D+= ABCDE

Now, we assume C is extraneous.

A->BC

B->E

B->D

D->A

Then,

A->BC

B->DE

D->A

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

d) D is extraneous from

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

So,

(A, B, C), (B, D, E), (A, D), (A, G)

e)

(A, B, C, D), (A, G), (A, E)

텍스트이(가) 표시된 사진

자동 생성된 설명

a)

A->BC

AD -> BCD (with augmentation with D)

BD->E (given)

BCD -> CE (with augmentation with C)

if AD->BCD and BCD -> CE, then AD->CE.

It can be divided into AD->C and AD->E.

So, AD->E

b)

A+ = {A, B, C}

So R(A, B, C, D, E, G) can be decomposed into R1(A, B, C) and R2(A, D, E, G)

And R2 is decomposed into R3(A, D, E) and R4(A, D, G) since AD->E

So, BCNF decomposition of R is (A,B,C), (A,D,E), (A,D,G).

c) In R1(A, B, C) and R2(A, D, E), R1∩R2 = (A)

(A) -> R1 satisfies. So it’s lossless-join.

d) we had BD->E and CD->AB in original relation R but we can’t get those two. So it doesn’t preserve it and this means it is not dependency preserving.

텍스트이(가) 표시된 사진

자동 생성된 설명

A)

AB+ -> ABCDEG

A+, B+, C+, D+, E+, G+ and ADE+ don’t have exact “ABCDEG”.

AB is the only candidate key for this.

b)

AB->C is redundant since B->C is reachable (B->GC)

AB->D is redundant since B->G and G->D.

ADE->D and ADE->E are required in canonical cover.

So,

ADE->G

B->CG

G->DE

c)

G->E and G->D (with 4th functional denpendency)

G+ = (G, E, D)

So the schema R will be decomposed into (A, D, E, G), (B, C, G), (D, E, G)

d.

none of them contain candidate key (A, B)

So the final decomposition is (A, B), (B, C, G), (A, D, E, G)

텍스트이(가) 표시된 사진

자동 생성된 설명

Isbn -> (title, publisher, author)

|  |
| --- |
| book(isbn, title, author, publisher) book\_accnno(accessionno, isbn) |

Isbn -> author

|  |
| --- |
| book(isbn, title, publisher)  book\_author(isbn, author) |

Deptid -> dept\_name

|  |
| --- |
| user(userid, name, deptid)  dept(deptid, deptname) |

book(isbn, title, publisher)

user(userid, name, deptid)

dept(deptid, deptname)

book\_author(isbn, author)

book\_accnno(accessionno, isbn)

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |
| --- |
| A->B  A->C |

A->AC (with augmentation with C with A->C)

AC -> BC (with augmentation with A->B)

Now we can see A->BC