# CS360

Assignment 5 – All groups

Todd Carter

V01187982

# Given R = {ABCDELGHIJK} and F = {I → K, AI → BLG, IC → ADE, BIG →CJ, K → HA}. Find a canonical cover for F. Find a 3NF decomposition of R. Is there a BCNF decomposition of R that is both dependency-preserving and also loss-less join. If so, compute such a decomposition.

**Find the canonical cover:**

Apply decomposition:

{I -> K, AI -> B, AI -> L, AI -> G, IC -> A, IC -> D, IC -> E, BIG -> C, BIG -> J, K -> H, K -> A}

Test for redundancy by closure of LHS:

1) I -> K

I+:

{I} – Not redundant

2) AI -> B

AI+:

{AILGKHA} – Not redundant

3) AI -> L

AI+

{AIBGCDEJKHA} – No L, not redundant

4) AI -> G

AI+

{AIBLKHA} – Not redundant

5) IC -> A

IC+

{ICDEKHABLGJ } - Redundant

6) IC -> D

IC+

{ICEKHABLGJ} – No D, not redundant

7) IC -> E

IC+

{ICADKHABLGJ} – No E, not redundant

8) BIG -> C

BIG+

{BIGJKHAL} – Not redundant

9) BIG -> J

BIG+

{BIGCKHAL} – Not redundant

10) K -> H

K+

{KA} – Not redundant

11) K -> A

K+

{KH} – Not redundant

Test for left-redundant attributes:

{I -> K, AI -> B, AI -> L, AI -> G, IC -> D, IC -> E, BIG -> C, BIG -> J, K -> H, K -> A}

1) I -> K

I+:

{I} – Not redundant

2) AI -> B

A+:

{A} – Not redundant

I+:

{IKHABLGCJDE} – Redundant

Replace AI -> B with A -> B

3) AI -> L

A

{A} – Not redundant

I+

{IKHABLGCJDE} – Redundant

Replace AI -> L with A -> L

4) AI -> G

A

{A} – Not redundant

I+

{IKHABLGCJDE} – Redundant

Replace AI -> G with A -> G

5) IC -> D

I+

{IKHABLGCJDE} – Redundant

C+

{C} – Not redundant

Replace IC -> D with C -> D

6) IC -> E

I+

{IKHABLGCJDE} – Redundant

C+

{C} – Not redundant

Replace IC -> E with C -> E

7) BIG -> C

B+

{B} – Not redundant

I+

{IKHABLGCJDE} – Redundant

G+

{G} – Not redundant

Replace BIG -> C with BG -> C

8) BIG -> J

B+

{B} – Not redundant

I+

{IKHABLGCJDE} – Redundant

G+

{G} – Not redundant

Replace BIG -> J with BG -> J

9) K -> H

K+

{KA} – Not redundant

10) K -> A

K+

{KH} – Not redundant

**The canonical cover is:**

{I -> K, A -> B, A -> L, A -> G, C -> D, C -> E, BG -> C, BG-> J, K -> H, K -> A}

**3NF Decomposition of R:**

S =

{ IK, AB, AL, AG, CD, CE, BGC, BGJ, KH, KA }

S =

{ IK, ABLG, CDE, BGCJ, KHA }

Check for superkey:

IK+:

{ IKHABLGCJL } – IK+ is a superkey, so that is valid 3NF.

**BCNF Decomposition of R (using canonical cover):**

R = { ABCDELGHIJK }

1)

{ ABDELGHIJK } – decomp 8. BG -> C

2)

{ ABDELGHIK } – decomp 8. BG -> J

3)

{ ABELGHIK } – decomp 6. C -> D

4)

{ ABLGHIK } – decomp 7. C -> E

5)

{ ABGHIK } – decomp 3. A -> L

6)

{ ABHIK } – decomp 4. A -> G

7)

{ AHIK } – decomp 2. A -> B

8)

{ HIK } – decomp 11. K -> A

9)

{ IK } – decomp 10. K -> H

So BCNF is:

R = {BCG, BCJ, CD, CE, AL, AG, AB, KA, KH, IK } = >

{BC->G, BC->J, C->D, C->E, A->L, A->G, A->B, K->A, K->H, I->K}

Which is a lossless join that preserves dependencies.

# Let R = {ABCDEH} and F = {ABC → DE, AB → D, DE → ABCH, E →C}. Is this scheme in 3NF, BCNF or none? If it isn’t, decompose this scheme into a normal form that is both dependency preserving and loss-less join.

**Test for 3NF and BCNF:**

{ ABCDE, ABD, DEABCH, EC }

It is in 3NF form, but not in BCNF, as AB->D and E->C are not superkeys but they are primary attributes.

**Decompose into lossless join, dependency preserving form:**

Find canonical cover to test for 3NF, BCNF:

Decomposition:

{ ABC -> D,

ABC -> E,

AB -> D,

DE -> A,

DE -> B,

DE -> C,

DE -> H,

E -> C }

Closure of each LHS:

1) ABC -> D

ABC+

{ ABCEDH } – Redundant

2) ABC -> E

ABC+

{ ABCD } – Not redundant

3) AB -> D

AB+

{ AB } – Not redundant

4) DE -> A

DE+

{ DEBCH } – Not redundant

5) DE -> B

DE+

{ DEACH } – Not redundant

6) DE -> C

DE+

{ DEABHC } - Redundant

7) DE -> H

DE+

{ DEABC } – Not redundant

8) E -> C

E+

{ E } – Not redundant

{ABC -> E,

AB -> D,

DE -> A,

DE -> B,

DE -> H,

E -> C }

Test for left-redundant attributes:

1) ABC -> E

A+

{ A } – Not redundant

B+

{ B } – Not redundant

C+

{ C } – Not redundant

2) AB -> D

A+

{ A } – Not redundant

B+

{ B } – Not redundant

3) DE -> A

D+

{ D } – Not redundant

E+

{ EC } – Not redundant

4) DE -> B

D+

{ D } – Not redundant

E+

{ EC } – Not redundant

5) DE -> H

D+

{ D } – Not redundant

E+

{ EC } – Not redundant

6) E -> C

E+

{ E } – Not redundant

The canonical cover is:

{ABC -> E, AB -> D, DE -> A, DE -> B, DE -> H, E -> C}

The lossless join 3NF is:

{ ABCE, ABD, DEABH, EC }

ABCE is a superkey.

**BCNF Decomposition (using canonical cover):**

{ ABCDEH }

AB->D and E->C are not superkeys and are in violation of BCNF.

1)

{ ABCEH } – decomp 2. AB -> D

2)

{ ABCH } – decomp 6. E -> C

The lossless join, dependency preserving BCNF is:

{ ABD, EC, ABCH }

3. Consider the scheme *R* = {*ABCDEGHI*} and the associated functional dependencies *F* = {*A* → *BCDEIGH, BCD* → *AEIGH, BCE* → *ADEIGH, CE* →*H, CD* → *H*}. Find a BCNF decomposition of *R*. Is the decomposition you computed dependency preserving? Why or why not?

**Decompose into BCNF:**

CE -> H and CD -> H are not superkeys and are in violation of BCNF.

{ ABCDEGHI }

1)

{ ABCDEGI } – decomp 4. CE -> H

BCNF is:

{ CEH, ABCDEGI }

This is not dependency preserving, as CD -> H was lost through decomposing CE -> H.