Michael Placzek

2/28/2022

CSC 355

Assignment 5

**Problems:**

1. *F* = { a, c → b ; b → d, e ; c → a ; e → b }.
   1. There are ten subsets of the set of attributes {a, b, c, d, e} that contain exactly two attributes (i.e., {a, b}, {a, c}, {a, d}, {a, e}, {b, c}, … and so on). For each of these ten subsets, find its closure using the set of functional dependencies *F*.

{a, b}+ = {ABDE}

{a ,c}+ = {ACBDE}

{a, d}+ = {AD}

{a, e}+ = {AEBD}

{b, c}+ = {BCADE}

{b, d}+ = {BDE}

{b, e}+ = {BED}

{c, d}+ = {CDABE}

{c, e}+ = {CEBDA}

{d, e}+ = {DEB}

* 1. Which of the ten subsets you considered in part a. are superkeys of *R*?

{a, c}, {b, c}, {c, d}, {c, e}

* 1. Which of the ten subsets you considered in part a. are candidate keys of *R*? If your answer is “none of them”, give some other subset of *R* that is a candidate key of *R*.

None of them, because the only attribute not determined by another attribute in *F* is C. And all the Super Keys can be “made smaller” into just {c}.

The only Candidate Key for *R* would be just {c}.

* 1. Is *R* in BCNF? Explain why or why not.

No, becauseif we test the closure of b → d, e

We can determine that {b}+ = {b, d, e}.

{b, d, e} are not all the values of *R*.

Meaning {b} is not a Super Key and *R* is not in BCNF.

* 1. Is *R* in 3NF? Explain why or why not.

|  |  |  |
| --- | --- | --- |
|  | **Left Super Key?** | **Right Prime?** |
| **A, C 🡪 B** | Yes | No |
| **B 🡪 D, E** | No | No |
| **C 🡪 A** | Yes | No |
| **E 🡪 B** | No | No |

Since C is the only prime attribute, and none of the functional dependencies point to C, therefore the Right Prime is always No.

*R* is not in 3NF, because every row does not contain a Yes value for Left Super Key or Right Prime.

1. Consider the following relation with six attributes:

INSURANCE(CustomerID, PlanID, Type, Payment, StartDate, EndDate)

The Functional Dependencies in INSURANCE are:

CustomerID → Type

CustomerID, PlanID → StartDate, EndDate

Type → Payment

The only Candidate Key of INSURANCE is {CustomerID, PlanID}.

* D = {INSURANCE}
* Remove from INSURANCE (Q); CustomerID (X) → Type (Y)

X+ = {CustomerID, Type, Payment}

X+ - X = {Type, Payment}

* Q – (X+ - X) = R1(CustomerID, PlanID, StartDate, EndDate)
* CustomerID, PlanID → StartDate, EndDate

Determinant is a Super Key, so in BCNF

* X+ = R2(CustomerID, Type, Payment)

CustomerID → Type

Type → Payment

Determinant is a Super Key, so in BCNF

* D = {R1, R2} … and we’re done.

1. Suppose that the relation R(A, B, C, D) has the set of functional dependencies

*F* = { D → C ; A → B, C ; B → C ; C → A }.

Consider the following decomposition of R:

{ R1(C, D) ; R2(A, B) ; R3(B, C) }

* 1. Give the projections of *F* on R1, R2, and R3.

|  |  |  |
| --- | --- | --- |
| R1(C, D) | R2(A, B) | R3(B, C) |
| D→C | A→B | B→C |
|  | B→A | C→B |

* 1. Does the decomposition have the dependency preservation property? Give a detailed explanation why or why not. (Don’t just state the definition of the dependency preservation property, but show why the decomposition either has or does not have this property by showing whether or not each functional dependency in *F* can be derived from the union of the projections.)

|  |  |
| --- | --- |
| Derive D→C | Yes, D→C in R1 |
| Derive A→B,C | No |
| Derive B→C | Yes, B→C in R3 |
| Derive C→A | No |

The decomposition does not have the Dependency Preservation Property because not every functional dependency can be derived from the projections stated above.

* 1. Does the decomposition have the lossless join property? Give a detailed explanation why or why not. (Use the matrix test, and show each step of your work. That is, each time you change the matrix, state which functional dependency you are applying, and show the change to the state of the matrix. Answers that show only the final matrix without showing the steps taken to obtain it will receive little or no credit.

Initial Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | (B11) | (B12) | C | D |
| **R2** | A | B | (B23) | (B24) |
| **R3** | (B31) | B | C | (B34) |

D→C

(can’t apply)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | (B11) | (B12) | C | D |
| **R2** | A | B | (B23) | (B24) |
| **R3** | (B31) | B | C | (B34) |

A→B,C

(can’t apply)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | (B11) | (B12) | C | D |
| **R2** | A | B | (B23) | (B24) |
| **R3** | (B31) | B | C | (B34) |

B→C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | (B11) | (B12) | C | D |
| **R2** | A | B | C | (B24) |
| **R3** | (B31) | B | C | (B34) |

C→A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | A | (B12) | C | D |
| **R2** | A | B | C | (B24) |
| **R3** | A | B | C | (B34) |

A→B,C

(final matrix)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** |
| **R1** | A | B | C | D |
| **R2** | A | B | C | (B24) |
| **R3** | A | B | C | (B34) |

No need to update the matrix with D🡪C again because all aj’s of R1 of the matrix already filled.

The decomposition has the Lossless Join Property because all aj’s of R1 of the matrix are filled.