CS360

Assignment #4 (for traditional group)

Todd Carter

V01187982

4-10-2025

1. Let *R* = {*ABCDELGHIJK*} and *F* = {*I* → *K, AI* → *BLG, IC* → *ADE, BIG* → *CJ, K* → *HA*}.
   1. **Prove using inference rules that *F |*= *AI* → *H*. [2 points]**

|  |  |  |
| --- | --- | --- |
|  | I -> K | Given |
|  | K -> HA | Given |
|  | AI -> AK | Augmentation 1 |
|  | AI -> K | Decomposition 3 |
|  | AI -> HA | Transitivity 4,2 |
|  | AI -> H | Decomposition 4 |

* 1. **Prove or disprove that *F |≠* *AC* → *K*. [2 points]**

Let *R* = {*ABCDELGHIJK*} and *F* = {*I* → *K, AI* → *BLG, IC* → *ADE, BIG* → *CJ, K* → *HA*}.

A relational table to prove by counterexample:

r =

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | L | G | H | I | J | K |
| a | b1 | c | d1 | e1 | l1 | g1 | h1 | i1 | j1 | k1 |
| a | b2 | c | d2 | e2 | l2 | g2 | h2 | i2 | j2 | k2 |

To disprove AC -> K as a dependency of F, we first show how this relational table is not valid for AC -> K, then we prove that it is valid for each stated dependency:

For AC -> K,

a = a,

c = c, but

k1 ≠ k2,

so this does not hold, since

ac -> k1 < ≠> ac -> k2

For I -> K,

i1 -> k1 does not conflict with i2 -> k2, so this holds.

For AI -> BLG,

ai1 -> b1l1g1 does not conflict with ai2 -> b2l2g2, because ai1 ≠ ai2, so this holds.

For IC -> ADE,

i1c -> ad1e1 does not conflict w/ i2c -> ad2e2 since i1c ≠ i2c and ad1e1 ≠ ad2e2,

so this holds.

For BIG -> CJ,

b1i1g1 -> cj1 does not conflict with b2i2g2 -> cj2 since cj1 ≠ cj2 so this holds.

For K -> HA,

k1 -> h1a does not conflict with k2 -> h2a since h1a ≠ h2a, so this holds.

Since all other dependencies hold, this disproves AC -> K as a dependency of F.

* 1. **Compute *BICF*+. [2 points]**

First iteration:

|  |  |
| --- | --- |
| I -> K: | { BICK } |
| IC -> ADE: | { BICKADE } |
| K -> HA | { BICKADEH } |

Second iteration:

|  |  |
| --- | --- |
| AI -BLG: | { BICKADEHLG } |
| BIG -> CJ: | { BICKADEHLGJ } |

The closure of *BICF*+ is { BICKADEHLG }, which happens to be a superkey.

* 1. **Compute all candidate keys of *R*. [4 points]**

To compute all candidate keys, we start with I, as it is the only attribute that does not appear on the right hand side of any dependency and all attributes appear at least once. If it is a superkey after computing the closure, it will be the only candidate key.

I:

First iteration:

|  |  |
| --- | --- |
| I -> K: | { IK } |
| K -> HA | { IKHA } |

Second iteration:

|  |  |
| --- | --- |
| AI -BLG: | { IKHABLG } |
| BIG -> CJ: | { IKHABLGCJ } |

Third iteration:

|  |  |
| --- | --- |
| IC -> ADE | { IKHABLGCJDE } |

Since the closure of I is { IKHABLGCJDE } which is equivalent to R = { ABCDELGHIJK }, it is a superkey and thus the only candidate key.

1. Let *R* be a relation with scheme over the attributes {A,B,C,D,E,H}, and the set of functional dependencies *F* = {*A* → *B, BD* → *H, E* → *D, C* → *AE*} holds on *R*.
   1. **Prove using inference rules that *F* |= *ADC* → *H*. [2 points]**

|  |  |  |
| --- | --- | --- |
|  | A -> B | Given |
|  | BD -> H | Given |
|  | AD -> BD | Augmentation 1 |
|  | AD -> H | Transitivity 3, 2 |
|  | ADC -> HC | Augmentation 4 |
|  | ADC -> H | Decomposition 5 |

* 1. **Compute *BEF*+. [1 point]**

*BEF*+:

First iteration:

|  |  |
| --- | --- |
| E -> D | { BED } |

Second iteration:

|  |  |
| --- | --- |
| BD -> H | { BEDH } |

The closure of *BEF*+ is { BEDH }.

* 1. **Compute all candidate keys of *R*. [2 points]**

To compute all candidate keys, we start with C, as it is the only attribute that does not appear on the right hand side of any dependency and all attributes appear at least once. If it is a superkey after computing the closure, it will be the only candidate key.

C:

First iteration:

|  |  |
| --- | --- |
| C -> AE: | { CAE } |

Second iteration:

|  |  |
| --- | --- |
| A -> B | { CAEB } |
| E -> D | { CAEBD } |

Third iteration:

|  |  |
| --- | --- |
| BD -> H | { CAEBDH } |

Since the closure of C is { CAEBDH } which is equivalent to R = { ABCDEH }, it is a superkey, and thus the only candidate key.