Database Systems - Homework 3 - Functional Dependencies & Normal Forms

Yosef Goren & Yonatan Kahalani

June 22, 2023

Contents

1

1.1

X is not in $BCNF$.
Proof.
From transitivity and reflexivity, $CG \to H$.
Note CG is not a superkey in R_3 since D is not implied by any FD since it only appears as the left operand.
Thus $CG \to D$ is not satisfied and $R_3 \notin BCNF \Rightarrow X \notin BCNF$.

1.2

No.
Proof.

- e only appears as the right operand.
- e is not prime.
- Any properkey does not contain e.
- c is not a superkey in R_2.
- $c \to e$ violates the properties of $3NF$.

1.3

By running the algorithm seen in class we get to:

A	B	C	D	E	H	G
	b	c	d			
		c		e	h	
		c	d		h	g

$\Longrightarrow_{C \to E}$

A	B	C	D	E	H	G
	b	c	d	**e**		
		c		e	h	
		c	d	**e**	h	g

And thus no more steps can be deduces and via the correctness of the algorithm - the decomposition does not preserve information.

1.4

Let

$$R := \{(a_1, b, c, d, e, g_1, a_1), (a_2, b, c, d, e, g_2, a_2)\}$$

Note that $R \models F$, additionally $(a_2, b, c, d, e, g_1, a_2) \in R_1 \bowtie R_2 \bowtie R_3$, thus the decomposition does not preserve information.

1.5

First we apply the confusing algorithm seen in the tutorial:

- $Z_F := \{A, H\}$
- On inspecting R_1:
  $\{A, H\} \cup (\{A, H\} \cap (\{A, H\} \cap \{B, C, D\})^+ \cap \{B, C, D\}) = \{A, H\}$
- On inspecting R_2:
  $B \notin R_2 \Rightarrow B \notin Z_F$
- On inspecting R_3:
  $B \notin R_3 \Rightarrow B \notin Z_F$

Thus $B \notin Z_F \Rightarrow$ Dependencies are not preserved.

2 Scheme Analysis

2.1

The keys:

- $K_1 := EG$.
  By applying the algorithm we see:
  Note that A can be derrived from $BCDEG$ by the FD $G \to ABCD$ and the Decomposition property.
  By applying the algorithm for finding keys 3 more times, we get rid of BCD. And we are left with EG.
  Note that $E \not\to G$ and $G \not\to E$ - hence K_1 is a proper-key.
- $K_2 := ABCD$.
  By the rule $BD \to G$ we get rid of G, and by the rule $AC \to D$ we get rid of D.
  Thus we are left with K_2 while not more attributes can be removed - so by the correctness of the algorithm - K_2 is a proper-key.

2.2

2.3

The claim is correct.
Proof.
Let $K := A_{i_1}A_{i_2}...A_{i_k}$ s.t. K is the left operand in f,
and so $\forall j \in \{1...k\} : i_j \in \{1...n\}$.

Now we show that K is a super-key:
Let $j \in \{1...n\} : A_j \notin K$.

A_j must be the right operand of f, Thus by Decomposition be can derive that $K \to A_j$. Thus $K^+ = R$.
Thus F satisfies the BCNF property and from a theorem from the lectures, this sufficient to show that $(R, F) \in BCNF$.

2.4

The claim is incorrect.
Example.
Let $F := A_1 \to A_2 A_3$.
Each A_i appears exactly once. But by the definition of a minimal cover - on the right side of each operand in the cover - there must be exactly one attribute.

3 Scheme Analysis

3.1

The key is $K := ADEG$.
Now we show for each attribute in R that it can be derrived from K.

- H:
  - $G \to H$ (from F)
  - $K \to KH$ (Augmentation)
  - $KH \to H$ (Reflexivity)
  - $K \to H$ (Transitivity)
- B:
  - $G \to B$ (from F)
  - $K \to KB$ (Augmentation)
  - $KB \to B$ (Reflexivity)
  - $K \to B$ (Transitivity)
- C:
  - $E \to CH$ (from F)
  - $CH \to C$ (Reflexivity)
  - $E \to C$ (from F)
  - $K \to KC$ (Augmentation)
  - $KC \to C$ (Reflexivity)
  - $K \to C$ (Transitivity)

3.2

1. The claim is correct.
   Proof.

   (a) $B \to G$ (from F)
   (b) $BG \to G$ (Augmentation)
   (c) $BG \to E$ (from F)
   (d) $G \to H$ (from F)
   (e) $BG \to BH$ (Augmentation)
   (f) $BH \to H$ (Reflexivity)
   (g) $BG \to H$ (e, f, Transitivity)
   (h) $BH \to EGH$ (b,c,g, Union)

2. The claim is incorrect.
   Example.
   Let $R := \{(a_1, b_1, c_1, d_1, e_1, g_1, h_1), (a_2, b_2, c_1, d_1, e_2, g_2, h_1)\}$.
   Note that $R \models F$, while $R \not\models (CH \to AE)$ thus, from the soundness theorem - we can deduce there is no proof.

3.3

A	B	C	D	E	G	H
a	b				g	
	b	c		e		
			d	e		h

$\Longrightarrow_{B\to G}$

A	B	C	D	E	G	H
a	b				g	
	b	c		e	g	
			d	e		h

$\Longrightarrow_{BG\to E}$

A	B	C	D	E	G	H
a	b			e	g	
	b	c		e	g	
			d	e		h

$\Longrightarrow_{E\to CH}$

A	B	C	D	E	G	H
a	b	c		e	g	h
	b	c		e	g	h
		c	d	e		h

$\Longrightarrow_{C\to D}$

A	B	C	D	E	G	H
a	b	c	d	e	g	h
	b	c	d	e	g	h
		c	d	e		h

3.4

	X_2	X_3
BCNF	✓	✗
3NF	✓	✗

Both ✗'s can be demostrated by $C \to D$.

4 Vatiations on Armstrong's Axioms

4.1

Yes. Since this system has less of freedom than the armstrong system, any proof generated by the reduced system is also a proof in the armstrong system.
In particular - our system is satisfied by the correctness of the armstrong system.

4.2

No.
For example, let $F := \{A \to B, A \to C\}$.

Lemma 1 *In our reduced axiom system - regarding F, for any rule $X \to Y$: The number of attributes in the right operand is never larger than the number of attributes in the left operand.*
We denote this as $|X| \geq |Y|$.

Lemma Proof.
By induction on the number of lines in a formal proof the rule.
Base: By reflexivity or FD from F.
Step: Reflexivity or FD like in the Base case. Transitivity:
We have already shown $W \to Y$, and that $Z \to W$. We now show $Z \to Y$.
From the induction hypothesis - $|W| \geq |Y| \wedge |Z| \geq |W|$ Thus $|Z| \geq |Y|$.

Due to the lemma, in the reduced system, $A \to BC$ cannot be implied by F. While union is provable in the armstrong system (which means $A \to BC$ can be implied by F in the armstring system).

5 MongoDB

5.1

```
db.s.distinct("university", {seniority_year: 2023})
```

5.2

```
db.s.aggregate([
  {
    $match: {
      seniority_year: { $gt: 2013 }
    }
  },
  {
    $group: {
    _id: "$university",
    staff_members: {
      $push: {
```

```
        staff_member_name: "$staff_member_name",
        staff_member_id: "$staff_member_id"
      }
    }
    }
  }
])
```

5.3

```
db.s.aggregate([
  {
    $group: {
    _id: {
      university: "$university",
      estimated_retirement_year: "$estimated_retirement_year"
    },
    total_current_salary: { $sum: "$salary" },
    total_retirement_salary: { $sum: "$estimated_retirement_salary" }
    }
  },
  {
    $group: {
    _id: "$_id.university",
    retirement_years: {
      $push: {
      estimated_retirement_year: "$_id.estimated_retirement_year",
      salary_delta: { $subtract: ["$total_current_salary", "$total_retirement_salary"] }
      }
    }
    }
  }
  ])
```