COMPSCI 2DB3 Assignment 7

Prakhar Saxena

MacID: saxenp4

Student number: 400451379

30 March 2024

Contents

Part 1

Question 1

Functional Dependency	**Reasoning**
id → date	Each event (identified via *id*) has a unique time and date (stored in *date*).
id → movie_id	Each event (identified via *id*) is related to exactly one movie (identified via *movie_id*).
movie_id → genre	Each movie (identified via *movie_id*) has exactly one main genre (identified via *genre*).
date → weekend	Each date (identified via *date*) has a Boolean flag (stored in *weekend*) indicating whether the event is on a weekend.
id → visitor	Each event (identified via *id*) has an expected number of visitors (stored in *visitors*).

Table 1: Functional Dependencies and their Reasoning

Question 2

The snacks ordered at the event are independent of the actors coming to the event. We can express this independence via either the multivalued dependency "id → snack", the multivalued dependency "id → actor", or the join dependency "$\bowtie \{X, Y\}$" with $X = \{id, date, actor, movie\_id, genre\}$ and $Y = \{id, date, snack, movie\_id, genre\}$.

Part 2

Question 3

The relational schema is not in 3NF. For example, the functional dependency:

$$\text{"}employer\_id \rightarrow employer\_rank\text{"}$$

is a 3NF violation as this functional dependency is not trivial, the attribute "employer_id" is not a (super)key of the relational schema, and the attribute "emplyer_rank" is not part of any key.
Next, we use the algorithm Decompose-3NF to put the relational schema into 3NF. To do so, we first determine a minimal cover of all the provided functional dependencies. We obtain:

$\mathfrak{S}'$ ={product_id → description
product_id, option → option_price,
id → user_id, id → date, id → employer_id,
id, product_id → amount, id, product_id → price,
emplyer_id → employer_rank,
category_id → category, }

The for-loop of Decompose-3NF will construct a relational schema for each $A \rightarrow X$ in the minimal cover with attributes $A \cup B$, $B = \{Y | A \rightarrow Y \in \mathfrak{S}'\}$. This will result in the following relational schemas:

- We get r_1(product_id, description) due to "product_id → description".
- We get r_2(product_id, option, option_price) due to "product_id, option → option_price".
- We get r_3(id, user_id, date, employer_id) due to "id → user_id", "id → employer_id", and "id → date".
- We get r_4(id, product_id, amount, price) due to "id, product_id → amount" and "id, product_id → price".
- We get r_5(employer_id, employer_rank) due to "employer_id → employer_rank".
- We get r_6(category_id, category) due to "category_id → category".

The if condition of Decompose-3NF will construct a relational schema r_7(id, product_id, category_id, option) for the key ”id, product_id, category_id, option”.
Finally, the while-loop will remove redundant relational schemas. However, in our solution we do not have any redundant schemas so we dont worry about it.
The following functional dependencies hold in each relational schema of the resulting decomposition:

Relation Scheme	**Functional Dependencies**
r_1(product_id, description)	“product_id → description”
r_2(product_id, option, option_price)	“product_id, option → option_price”
r_3(id, user_id, date, employer_id)	“id → user_id”, “id → employer_id”, and “id → date”
r_4(id, product_id, amount, price)	“id, product_id → amount” and “id, product_id → price”
r_5(employer_id, employer_rank)	“employer_id → employer_rank”
r_6(category_id, category)	“category_id → category”
r_7(id, product_id, category_id, option)	

This decomposition is lossless-join and dependency-preserving (with respect to the original func- tional dependencies), as we have used the Decompose-3NF decomposition algorithm (which guar- antees a lossless-join and dependency-preserving decomposition). Finally, we decompose the example dataset according to the relational schema obtained from the decomposition algorithm:

P	**Ds**
4	Ink
7	Scanner
12	Laptop

Table 1: Relational schema for r_1

P	**O**	**Op**
4	Warranty	$12.00
7	Installation	$76.00
7	Warranty	$14.00
12	Instalation	$15.00
12	Warranty	$140.00

Table 2: Relational schema for r_2

Id	**U**	**D**	**E**
1	21	18 April	3
2	17	19 April	8

Table 3: Relational schema for r_3

Id	**P**	**A**	**Pr**
1	4	2	$5.00
1	7	1	$55.00
2	12	1	$1200.00

Table 4: Relational schema for r_4

E	**Er**
3	Junior
8	Senior

Table 5: Relational schema for r_5

Ci	**C**
43	Printer
12	Office
23	PCs

Table 6: Relational schema for r_6

Id	**P**	**Ci**	**O**
1	4	43	Warranty
1	4	12	Warranty
1	7	43	Installation
1	7	12	Installation
1	7	43	Warranty
1	7	12	Warranty
2	12	23	Mouse
2	12	12	Mouse
2	12	23	Warranty
2	12	12	Warranty

Table 7: Relational schema for r_7

Question 4

The relational schema is not in BCNF, as it is not in 3NF (alternatively, one can show that a BCNF violation exists, for example, the functional dependency "*employer_id* → *employer_rank*" is a BCNF violation as this functional dependency is not trivial and the attribute *employer_id*" is not a (super)key of the relational schema.
Next, we use the algorithm Decompose-BCNF to put the relational schema into BCNF. In the below, we use $\rho(....)$ with $\rho(....)$ a relational schema to indicate that $\rho(....)$ is in BCNF as there are no functional dependencies in $\rho(....)$ that are BCNF violations and we use $\rho(....)$ with $\rho(....)$ a relational schema to indicate that $\rho(....)$ is in BCNF as it is a binary relationship. We perform the following steps:

- $R(Id, E, Er, U, D, P, A, Pr, Ds, C, Ci, O, Op)$
  - (a) Id− >E,Er,U,D
    $R_1(Id, E, Er, U, D)$
    - (b) E− >Er
      $R_{1,1}(E, Er)$
    - $R_{1,2}(Id, E, U, D)$
  - $R_2(Id, P, A, Pr, Ds, C, Ci, O, Op)$
    - $R_{2,1}(Id, P, A, Pr, Ds)$
      - (c) Id,P− >A,Pr
        $R_{2,1,1}(Id, P, A, Pr)$
      - (d) P− >D
        $R_{2,1,2}(P, Ds)$
    - $R_{2,2}(Id, P, C, Ci, O, Op)$
      - $R_{2,2,1}(Id, P, Ci, O, Op)$
        - (f) P,O − > Op
          $R_{2,2,1,1}(P, O, Op)$
        - $R_{2,2,1,2}(Id, P, Ci, O)$
      - (e) Ci− >Ci
        $R_{2,2,2}(C, Ci)$

(a) We begin with R(Id, E, Er, U, D, P, A, Pr, Ds, C, Ci, O, Op). The attribute "id" is not a key of R. Hence, all functional dependencies of the form id $->$ X are BCNF violations. We split using:

$$id-> employer\_id, emplyer\_rank, user\_id, date$$

and obtain R_1(with all attributes in id^+) and R_2 with "id" and all remaining attributes of R:

$$R_1(Id, E, Er, U, D); \quad R_2(L, P, R_p).$$

(b) The relational schema R_1 is not in BCNF. E.g., the functional dependency

$$employer\_id-> emplyer\_rank$$

is a BNCF violation. We split this violation and obtain $R_{1,1}$ and $R_{1,2}$

$$R_{1,1}(E, Er); \quad R_{1,2}(Id, E, U, D).$$

$R_{1,1}$ is binary and is in the BCNF. The relational schema $R_{1,2}$ is in the BCNF.

(c) The relational schema R_2 is not in BCNF. E.g., the functional dependency

$$id, product\_id-> product, price$$

is a BNCF violation. We split this violation and obtain $R_{2,1}$ and $R_{2,2}$

$$R_{2,1}(Id, P, A, Pr, Ds); \quad R_{2,2}(Id, P, C, Ci, O, Op).$$

(d) The relational schema $R_{2,1}$ is not in BCNF. E.g., the functional dependency

$$product\_id-> description$$

is a BNCF violation. We split this violation and obtain $R_{2,1,1}$ and $R_{2,1,2}$

$$R_{2,1,1}(Id, P, A, Pr); \quad R_{2,1,2}(P, Ds).$$

The relational schema $R_{2,1,1}$ is in the BCNF as "id, product_id $->$ amount, price is the only non trivial dependancy that holds in $R_{2,1,1}$.
$R_{2,1,2}$ is binary and is in the BCNF.

(e) The relational schema $R_{2,2}$ is not in BCNF. E.g., the functional dependency

$$category\_id- > category$$

is a BNCF violation. We split this violation and obtain $R_{2,2,1}$ and $R_{2,2,2}$

$$R_{2,2,1}(Id, P, Ci, O, Op); \quad R_{2,2,2}(C, Ci).$$

$R_{2,2,2}$ is binary and is in the BCNF.

(f) The relational schema $R_{2,2,1}$ is not in BCNF. E.g., the functional dependency

$$product\_id, option- > option\_price$$

is a BNCF violation. We split this violation and obtain $R_{2,2,1,1}$ and $R_{2,2,1,2}$

$$R_{2,2,1,1}(P, O, Op); \quad R_{2,2,1,2}(Id, P, Ci, O).$$

The relational schema $R_{2,2,1,1}$ is in the BCNF as "product_id,option $- >$ option_price", price is the only non trivial dependancy that holds in $R_{2,2,1,1}$. Finally, $R_{2,2,1,2}$ is in the BCNF.

The following functional dependencies hold in each relational schema of the resulting decomposition:

Relation Scheme	**Functional Dependencies**
$r_{2,1,2}$(product_id, description)	"product_id → description"
$r_{2,2,1,1}$(product_id, option, option_price)	"product_id, option → option_price"
$r_{1,2}$(id, user_id, date, employer_id)	"id → user_id", "id → employer_id", and "id → date"
$r_{2,1,1}$(id, product_id, amount, price)	"id, product_id → amount" and "id, product_id → price"
$r_{1,1}$(employer_id, employer_rank)	"employer_id → employer_rank"
$r_{2,2,2}$(category_id, category)	"category_id → category"
$r_{2,2,1,2}$(id, product_id, category_id, option)	

This decomposition is lossless-join, as we have used the Decompose-BCNF decomposition algorithm (which guarantees a lossless-join decomposition).
Remark. The Decompose-BCNF decomposition algorithm cannot guarantee that the decomposition is dependency preserving, as there are cases in

which the only lossless-join BCNF decompositions are not dependency preserving.
Finally, we decompose the example dataset according to the relational schema obtained from the decomposition algorithm:

P	**Ds**
4	Ink
7	Scanner
12	Laptop

Table 1: Relational schema for $r_{2,1,2}$

P	**O**	**Op**
4	Warranty	$12.00
7	Installation	$76.00
7	Warranty	$14.00
12	Instalation	$15.00
12	Warranty	$140.00

Table 2: Relational schema for $r_{2,2,1,1}$

Id	**U**	**D**	**E**
1	21	18 April	3
2	17	19 April	8

Table 3: Relational schema for $r_{1,2}$

Id	**P**	**A**	**Pr**
1	4	2	$5.00
1	7	1	$55.00
2	12	1	$1200.00

Table 4: Relational schema for $r_{2,1,1}$

E	**Er**
3	Junior
8	Senior

Table 5: Relational schema for $r_{1,1}$

Ci	**C**
43	Printer
12	Office
23	PCs

Table 6: Relational schema for $r_{2,2,2}$

Id	**P**	**Ci**	**O**
1	4	43	Warranty
1	4	12	Warranty
1	7	43	Installation
1	7	12	Installation
1	7	43	Warranty
1	7	12	Warranty
2	12	23	Mouse
2	12	12	Mouse
2	12	23	Warranty
2	12	12	Warranty

Table 7: Relational schema for $r_{2,2,1,2}$

Question 5

The relational schema is not in 4NF, as it is not in BCNF (alternatively, one can show that a 4NF violation exists, e.g., due to Replication, the functional dependency "$employer\_id \rightarrow employer\_rank$" implies the multivalued dependency "$employer\_id \twoheadrightarrow employer\_rank$" and this multivalued dependency is a 4NF violation as this multivalued dependency is not trivial and the attribute "employer id" is not a (super)key of the relational schema.
Next, we use the algorithm Decompose-4NF to put the relational schema into 4NF. We will use the BCNF decomposition as a starting point: we can do so, as every step with violation $\alpha \rightarrow \beta$ in the algorithm Decompose-BCNF can

be translated to a valid step in the algorithm Decompose- 4NF. Namely, in the algorithm Decompose-4NF, we use the violation $\alpha \twoheadrightarrow \alpha^+$ in which α^+ is the attribute closure of α with respect to only the functional dependencies. Note that $\alpha \rightarrow \alpha^+$ implies $\alpha \twoheadrightarrow \alpha^+$ due to replication.
In the below, we use $\rho(....)$ with $\rho(....)$ a relational schema to indicate that $\rho(....)$ is in the BCNF decomposition but is not in 4NF, we use $\rho(....)$ with $\rho(....)$ a relational schema to indicate that $\rho(....)$ is in 4NF as there are no multivalued dependencies in $\rho(....)$ that are 4NF violations and we use $\rho(....)$ with $\rho(....)$ a relational schema to indicate that $\rho(....)$ is in 4NF as it is a binary relationship.

Hence, we start by applying all splitting steps we can re-use from the BCNF decomposition:

(a) The multivalued dependency

$$id \twoheadrightarrow employer\_id, emplyer\_rank, user\_id, date$$

which we obtain by applying Replication on id $\rightarrow id^+$, is a 4NF violation. We split using this violation and obtain

$$R_1(Id, E, Er, U, D); \quad R_2(Id, P, A, Pr, Ds, C, Ci, O, Op).$$

(b) The multivalued dependency

$$employer\_id \twoheadrightarrow emplyer\_rank$$

which we obtain by applying Replication on employer_id $\rightarrow employer\_id^+$, is a 4NF violation. We split using this violation and obtain

$$R_{1,1}(E, Er); \quad R_{1,2}(Id, E, U, D).$$

$R_{1,1}$ is binary and, hence, is in 4NF.

(c) The multivalued dependency

$$id, product\_id \twoheadrightarrow amount, price$$

which we obtain by applying Replication on id,product_id $\rightarrow (id, product\_id)^+$, is a 4NF violation. We split using this violation and obtain

$$R_{2,1}(Id, P, A, Pr, Ds); \quad R_{2,2}(Id, P, C, Ci, O, Op).$$

(d) The multivalued dependency

$$product\_id \twoheadrightarrow description$$

which we obtain by applying Replication on product_id $\rightarrow$ $product\_id^+$, is a 4NF violation. We split using this violation and obtain

$$R_{2,1,1}(Id, P, A, Pr); \quad R_{2,1,2}(P, Ds).$$

$R_{2,1,1}$ is binary and is in 4NF.

(e) The multivalued dependency

$$category\_id \twoheadrightarrow category$$

which we obtain by applying Replication on category_id $\rightarrow$ $category\_id^+$, is a 4NF violation. We split using this violation and obtain

$$R_{2,2,1}(Id, P, Ci, O, Op); \quad R_{2,2,2}(C, Ci).$$

$R_{2,2,1}$ is binary and is in 4NF.

(f) The multivalued dependency

$$product\_id, option \twoheadrightarrow option\_price$$

which we obtain by applying Replication on product_id,option $\rightarrow$ $(product\_id, option)^+$, is a 4NF violation. We split using this violation and obtain

$$R_{2,2,1,1}(P, O, Op); \quad R_{2,2,1,2}(Id, P, Ci, O).$$

Next, we evaluate whether each resulting non-binary relation scheme is in 4NF. Hence, we have the following additional steps:

(g) $R_{2,2,1,2}(Id, P, Ci, O)$ We have "id, product" $\rightarrow$ $\mathcal{J}$. We apply Replication on "id, product"$\mathcal{J}$. Next, we apply Transitivity on "id, product"$\mathcal{J}$ and $\mathcal{J} \twoheadrightarrow$ "option" to obtain "id, product $\rightarrow$ option". As $id, product$ is not a key of R_2, the multivalued dependency "id, product $\twoheadrightarrow$ option" is a 4NF violation. We split using this violation and obtain:

$$R_{2,1,2,1}(Id, P, O); \quad R_{2,1,2,2}(Id, P, Ci).$$

$R_{2,1,2,1}$ and $R_{2,1,2,2}$ are in 4NF.

(i) The introduced multivalued dependencies do not introduce new dependencies for the relational schemas $R_{2,1,1}(Id, P, A, Pr)$, $R_{2,2,1,1}(P, O, Op)$; Hence, these schemas are in 4NF.

The following functional dependencies hold in each relational schema of the resulting decomposition:

Relation Scheme	**Functional Dependencies**
$r_{2,1,2}$(product_id, description)	"product_id $\rightarrow$ description"
$r_{2,2,1,1}$(product_id, option, option_price)	"product_id, option $\rightarrow$ option_price"
$r_{1,2}$(id, user_id, date, employer_id)	"id $\rightarrow$ user_id", "id $\rightarrow$ employer_id", and "id $\rightarrow$ date"
$r_{2,1,1}$(id, product_id, amount, price)	"id, product_id $\rightarrow$ amount" and "id, product_id $\rightarrow$ price"
$r_{1,1}$(employer_id, employer_rank)	"employer_id $\rightarrow$ employer_rank"
$r_{2,2,2}$(category_id, category)	"category_id $\rightarrow$ category"
$r_{2,2,1,2}$(id, product_id, category_id, option)	
$r_{2,2,1,2,1}$(id, product_id, option)	
$r_{2,2,1,2,2}$(id, product_id, category_id)	

We note that non-trivial multivalued dependencies hold in this decomposition (that cannot be derived from the above functional dependencies).
This decomposition is lossless-join, as we have used the Decompose-4NF decomposition algo- rithm (which guarantees a lossless-join decomposition).
Remark. The Decompose-4NF decomposition algorithm cannot guarantee that the decom- position is dependency preserving, as there are cases in which the only lossless-join 4NF decompositions are not dependency preserving.

Finally, we decompose the example dataset according to the relational schema obtained from the decomposition algorithm:

E	**Er**
3	Junior
8	Senior

Table 1: Relational schema for $r_{1,1}$

Id	**U**	**D**	**E**
1	21	18 April	3
2	17	19 April	8

Table 2: Relational schema for $r_{1,2}$

Id	**P**	**A**	**Pr**
1	4	2	$5.00
1	7	1	$55.00
2	12	1	$1200.00

Table 3: Relational schema for $r_{2,1,1}$

P	**Ds**
4	Ink
7	Scanner
12	Laptop

Table 4: Relational schema for $r_{2,1,2}$

Ci	**C**
43	Printer
12	Office
23	PCs

Table 5: Relational schema for $r_{2,2,2}$

P	**O**	**Op**
4	Warranty	$12.00
7	Installation	$76.00
7	Warranty	$14.00
12	Instalation	$15.00
12	Warranty	$140.00

Table 6: Relational schema for $r_{2,2,1,1}$

Id	**P**	**O**
1	4	Warranty
1	7	Installation
1	7	Warranty
2	12	Mouse
2	12	Mouse
2	12	Warranty

Table 7: Relational schema for $r_{2,2,1,2,1}$

Id	**P**	**Ci**
1	4	43
1	4	12
1	7	12
1	7	43
2	12	12
2	12	23

Table 8: Relational schema for $r_{2,2,1,2,2}$

Question 6

No, not all the issues are solved for example, category_id is linked with both product_id and Id even though there is a clear functional dependancy of category_id on product. This introduces redundancy.
Also, the price of a product is independent of order_id and should only depend on product_id, which has also not been resolved in our decompositions.