

1 Aufgabe 1 - Funktionale Abhängigkeiten

1.1 Teilaufgabe a)

Gilt $F \Rightarrow f$?

	Funktionale Abhängigkeit f		Begründung
1	$DI \rightarrow BCDEFI$	✗	kein F
2	$BDG \rightarrow ABCDEFHH$	✓	
3	$DHI \rightarrow ACDGI$	✗	kein A
4	$BCDEGI \rightarrow ABCDEFGHI$	✓	
5	$AF \rightarrow ABCEGH$	✓	
6	$ABCDE \rightarrow ABCEFHI$	✗	kein I
7	$DI \rightarrow CFGI$	✗	kein C
8	$ADFI \rightarrow BCDEFGHI$	✓	
9	$GI \rightarrow BDEFG$	✗	kein B

1.2 Teilaufgabe b)

Schlüssel in R:

- $\{I, A\}$
- $\{I, B\}$
- $\{I, C\}$

Die Relation befindet sich nur in 1NF, da $I \rightarrow H$ eine partielle Abhängigkeit darstellt. Daher kann die Relation nicht in 2NF sein.

1.3 Teilaufgabe c)

$$F^{(2)} = \{A \rightarrow B, \\ AI \rightarrow \delta, \\ B \rightarrow C, B \rightarrow D, \\ C \rightarrow A, C \rightarrow D, C \rightarrow F, \\ D \rightarrow E, D \rightarrow F, D \rightarrow G, D \rightarrow H, D \rightarrow H, \\ F \rightarrow G, \\ I \rightarrow H\}$$

$$F^{(3)} = \{A \rightarrow B, \\ AI \rightarrow \delta, \\ B \rightarrow C, B \rightarrow D, \\ C \rightarrow A, \\ D \rightarrow E, D \rightarrow F, D \rightarrow H, \\ F \rightarrow G, \\ I \rightarrow H\}$$

aufgelöst wurden wie folgt:

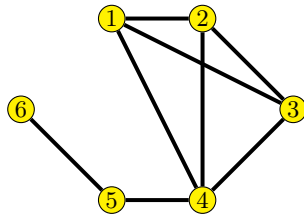
	redundant durch
$C \rightarrow D$	$C \rightarrow A \wedge A \rightarrow B \wedge B \rightarrow D$
$C \rightarrow F$	$C \rightarrow A \wedge A \rightarrow B \wedge B \rightarrow D \wedge D \rightarrow F$
$D \rightarrow G$	$D \rightarrow F \wedge F \rightarrow G$

ergibt die Zerlegung

$$R = \{ \begin{aligned} &(\{ A, B, C, D \}, \{ \{ A \}, \{ B \}, \{ C \} \}), \\ &(\{ A, I \}, \{ \{ A, I \} \}), \\ &(\{ D, E, F, H \}, \{ \{ D \} \}), \\ &(\{ F, G \}, \{ \{ F \} \}), \\ &(\{ I, H \}, \{ \{ I \} \}) \end{aligned} }$$

2 Aufgabe 2 - SQL

2.1 Teilaufgabe a)



2.2 Teilaufgabe b)

```
1 CREATE VIEW FriendshipSymmetric AS (  
2     (  
3         SELECT person1, person2  
4         FROM Friendship  
5     )  
6     UNION  
7     (  
8         SELECT person2 AS person1, person1 AS person2  
9         FROM Friendship  
10    )  
11 )
```

2.3 Teilaufgabe c)

2.3.1 Version A

```
1 SELECT f1.person2, f2.person2  
2 FROM FriendshipSymmetric f1  
3 JOIN FriendshipSymmetric f2  
4     ON f1.person1 = f2.person1  
5 LEFT JOIN FriendshipSymmetric f3  
6     ON f1.person2 = f3.person1 AND f2.person2 = f3.person2  
7 WHERE f1.person2 != f2.person2  
8     AND f1.person1 = <id>  
9     AND p3.person1 IS NULL  
10    AND p3.person2 IS NULL
```

Weitere Erklärungen: Ansatz:

1. Suche alle Personenpaare, die beide <id> als Freund haben (wobei nur ungleiche paare gesucht sind, da man nicht mit sich selbst befreundet sein kann)
2. Prüfe über die Menge dieser Paare, welche noch nicht befreundet sind

Dazu:

Ein LEFT JOIN ergänzen, um zu ermitteln, welche Paare nicht tatsächlich in FriendshipSymmetric stehen (diese werden NULL joinen). Daher nach NULL selektieren

Beispielhaftes Ergebnis für gegebene Situation und id=4:

"1", "5"
"2", "5"
"3", "5"

```
"5","1"  
"5","2"  
"5","3"
```

2.3.2 Version B

```
1 SELECT f1.person2, f2.person2  
2 FROM (  
3     SELECT * FROM FriendshipSymmetric WHERE person1 = <id>  
4 ) f1  
5 JOIN ON  
6 (  
7     SELECT * FROM FriendshipSymmetric WHERE person1 = <id>  
8 ) f2  
9 EXCEPT  
10 (  
11     SELECT * FROM FriendshipSymmetric  
12 )  
13 WHERE f1.person2 != f2.person2
```

Ohne EXCEPT (da ich mir nicht sicher bin, ob es nun SQL-Standard ist oder nicht, z.B. SQLite kenn kein EXCEPT, auf einer Übersicht stand es aber bei SQL89 angehakt dabei). Hinweis: NOT EXISTS ist True, gdw die Unterabfrage genau 0 Zeilen enthält.

```
1 SELECT f1.person2, f2.person2  
2 FROM (  
3     SELECT * FROM FriendshipSymmetric WHERE person1 = 4  
4 ) f1  
5 JOIN  
6 (  
7     SELECT * FROM FriendshipSymmetric WHERE person1 = 4  
8 ) f2 ON f1.person1 = f2.person1  
9 WHERE f1.person2 != f2.person2  
10 AND NOT EXISTS  
11 (  
12     SELECT * FROM FriendshipSymmetric f WHERE f.person1 = f1.person2 AND f.person2 = f2.person2  
13 )
```

3 Aufgabe 3 - Histories

3.0.3 Teilaufgabe a)

H1 Es gibt folgende Kanten: (12, xyz), (13, xy), (23, y), (32, y).
Somit ist ein Zykel zwischen 2 und 3 \Rightarrow nicht serialisierbar

H2 (21, xyz), (23, y), (31, xy).
Somit keine Zykel und serialisierbar

3.0.4 Teilaufgabe b) und c)

(Y = erfüllt, N = nicht erfüllt)

- H1
 - T3 reads y from T2 NN
 - T2 reads x from T3 YN
 - T2 reads z from T1 YN

- T3 reads x from T1 YN
- H2
 - T1 reads y from T3 YN
 - H1 ist nicht rücksetzbar (also weder in RC, ACA oder ST)
 - H2 ist in RC (also nicht in ACA oder ST)