4. Exercise

Timo Bergerbusch 344408
Thomas Näveke 311392
Shu Xing 381176

29.11.2017

Exercise 4.1

1.

$\underline{T_1 \text{ and } T_2}$

$T_1 \subseteq T_2$**?**

$$\text{Find mapping } h : T_2 \to T_1, x \rightarrowtail \begin{cases} b_3 & x = b_1 \\ 5 & x = b_2 \\ b_4 & x = b_3 \\ b_3 & x = b_4 \\ a_1 & x = a_1 \\ a_2 & x = a_2 \end{cases} \Rightarrow T_1 \subseteq T_2$$

$T_2 \subseteq T_1$**?**

Impossible since we would have to map the constant 5 to an other constant and T_2 does not contain a constant. So $T_2 \nsubseteq T_1$

Overall

$\Rightarrow T_1 \subset T_2$, but $T_2 \nsubseteq T_1$ and therefore $T_1 \not\equiv T_2$

T_2 and T_3

$T_2 \subseteq T_3$?

$T_3 \subseteq T_3$?

Overall

T_1 and T_3

$T_1 \subseteq T_3$?

$T_3 \subseteq T_3$?

Analogously to $T_2 \subseteq T_1$ we would have to map the constant 5 to an other constant and T_3 does not contain a constant. So $T_3 \not\subseteq T_1$

Overall

$\Rightarrow T_1 \not\subseteq T_3, T_3 \not\subseteq T_1$ and therefore especially $T_1 \not\equiv T_3$

2.

T_1:

T_1 is minimal, because if we delete any row and so create T' we can not have a function $h : T \to T'$ such that $T' \subseteq T$

T_2:

1. delete row (b_4, a_2), since we have a function h_1, with $h_1(x) = \begin{cases} b_1 & \text{if } x = b_4 \\ x & \text{if } x \neq b_4 \end{cases}$

New Tableau T_1':

a_1	a_2	
b_1	a_2	(R)
a_1	b_3	(R)
b_2	b_4	(R)
b_2	b_1	(R)
b_3	b_2	(R)

2. delete row (a_1, b_3), since we have a function h_2, with $h_2(x) = \begin{cases} b_1 & \text{if } x = a_1 \\ a_2 & \text{if } x = b_3 \\ x & else \end{cases}$

New Tableau $T_1^{''}$:

a_1	a_2	
b_1	a_2	(R)
b_2	b_4	(R)
b_2	b_1	(R)
b_3	b_2	(R)

3. delete row (b_2, b_4), since we have a function h_2, with $h_3(x) = \begin{cases} b_1 & \text{if } x = b_4 \\ x & else \end{cases}$

New Tableau $T_1^{'''}$:

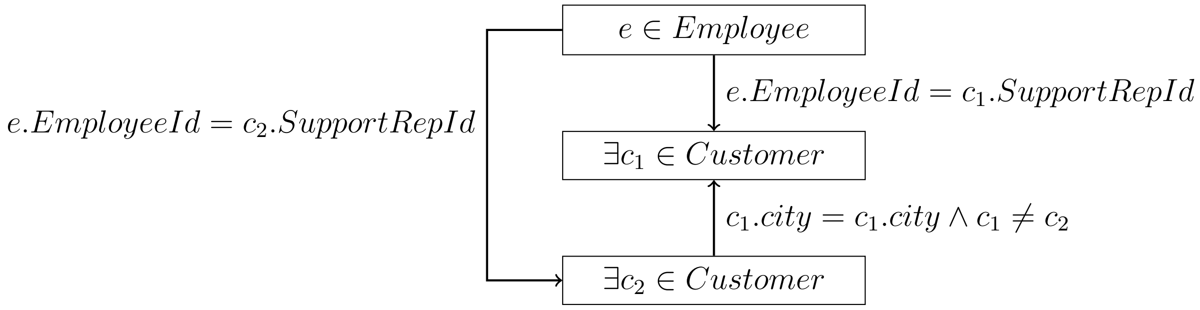
a_1	a_2	
b_1	a_2	(R)
b_2	b_1	(R)
b_3	b_2	(R)

4. This table $T_1^{'''}$, which es derived by $h_3(h_2(h_1(T)))$, is now minimal and equivalent

Exercise 4.2

1.a)

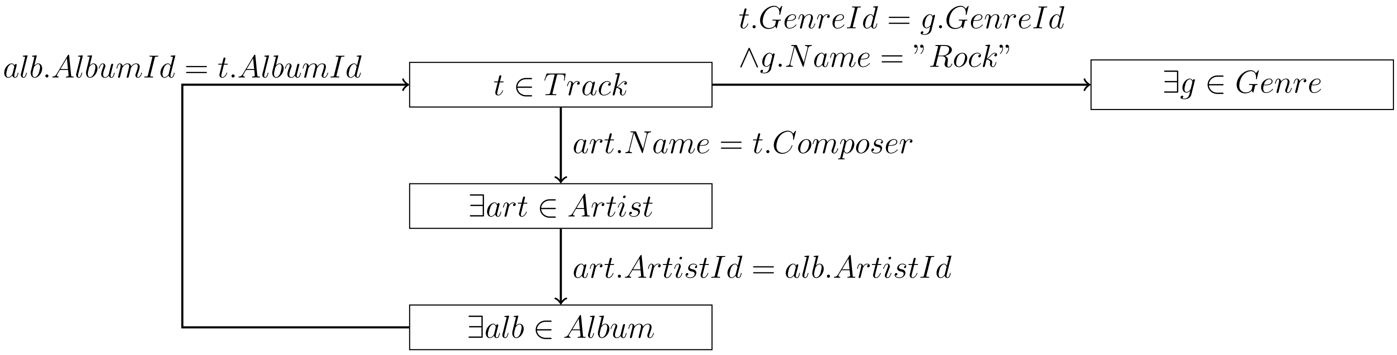
$\{< e.EmployeeId, e.LastName >| \ e \in Employee \wedge \exists c_1 \in Customer \wedge c_1.SupportedRepId = e.EmployeeId \wedge \exists c_2 \in Customer \wedge c_2.SupportRepId = e.EmployeeId \wedge c_1.City = c_2.City \wedge c_1 \neq c_2\}$


$e \in Employee$
$e.EmployeeId = c_1.SupportRepId$
$e.EmployeeId = c_2.SupportRepId$
$\exists c_1 \in Customer$
$c_1.city = c_1.city \wedge c_1 \neq c_2$
$\exists c_2 \in Customer$

This graph is cycle free, which means the query is optimizable using semi-joins.

1.b)

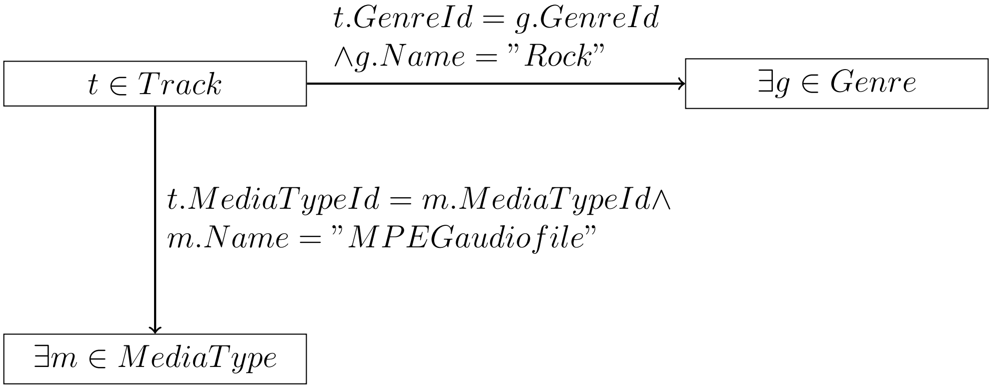
$\{< t.name, t.composer >| \ t \in Track \wedge \exists g \in Genre \wedge g.GenreId = t.GenreId \wedge g.Name = "Rock" \wedge \exists art \in Artist \wedge \exists alb \in Album \wedge alb.ArtistId = art.ArtistId \wedge t.AlbumId = alb.AlbumId \wedge t.Composer = art.Name\}$



This Graph has a cycle. This means the query is not optimizable using semi-joins.

2.

$\{< t.Name >|\ t \in Track \land t.Miliseconds \leq 90000 \land \exists g \in Genre \land t.GenreId = t.GenreId \land g.Name = "Rock" \land \exists m \in MediaType \land t.MediaTypeId = m.MediaTypeid \land m.Name = "MPEGaudiofile"\}$


$t.GenreId = g.GenreId$
$\land g.Name = "Rock"$
$t \in Track$
$\exists g \in Genre$
$t.MediaTypeId = m.MediaTypeId \land$
$m.Name = "MPEGaudiofile"$
$\exists m \in MediaType$

TODO: es fehlt das mit den $\leq$90000ms

The graph is cycle free, which means it can be optimized using semi-joins.