Database Theory

**Exercise 2: First-order Queries**

24 April 2015
9. Find the actors cast in at least one film by “Smith”.

10. Find the actors cast in every film by “Smith”.

11. Find the actors cast only in films by “Smith”.

12. Find all pairs of actors who act together in at least one film.

13. Find all pairs of actors cast in exactly the same films.

14. Find the directors such that every actor is cast in one of his or her films.
Exercise 2  Let \( R \) be a table (relational instance) with attributes \( A \) and \( B \). Use the construction from the lecture to express the following RA\(_{\text{named}}\) query as a DI\(_{\text{unnamed}}\) query:

\[
q[A, B] := (\pi_A(R) \bowtie \pi_B(R)) - (R \bowtie (\delta_{A,B\rightarrow B,A}(R)))
\]

Given an RA query \( q[a_1, \ldots, a_n] \), we construct a DI query \( \varphi_q[x_{a_1}, \ldots, x_{a_n}] \)

- if \( q = R \) with signature \( R[a_1, \ldots, a_n] \), then \( \varphi_q = R(x_{a_1}, \ldots, x_{a_n}) \)
- if \( n = 1 \) and \( q = \{a_1 \mapsto c\} \), then \( \varphi_q = (x_{a_1} \approx c) \)
- if \( q = \sigma_{a_i = c}(q') \), then \( \varphi_q = \varphi_{q'} \land (x_{a_i} \approx c) \)
- if \( q = \sigma_{a_i = a_j}(q') \), then \( \varphi_q = \varphi_{q'} \land (x_{a_i} \approx x_{a_j}) \)
- if \( q = \delta_{b_1, \ldots, b_n \rightarrow a_1, \ldots, a_n}(q') \), then \( \varphi_q = \exists y_{b_1}, \ldots, y_{b_m} (x_{a_1} \approx y_{b_1}) \land \ldots \land (x_{a_n} \approx y_{b_n}) \land \varphi'_{[y_{a_1}, \ldots, y_{a_n}]} \)
  (Here we assume that the \( a_1, \ldots, a_n \) in \( \delta_{b_1, \ldots, b_n \rightarrow a_1, \ldots, a_n} \) are written in the order of attributes, whereas \( b_1, \ldots, b_n \) might be in another order. \( \varphi'_{[y_{a_1}, \ldots, y_{a_n}]} \) is like \( \varphi_{q'} \) but using variables \( y_{a_i} \).)
- if \( q = \pi_{a_1, \ldots, a_n}(q') \) for a subquery \( q'[b_1, \ldots, b_m] \) with \( \{b_1, \ldots, b_m\} = \{a_1, \ldots, a_n\} \cup \{c_1, \ldots, c_k\} \), then \( \varphi_q = \exists x_{c_1}, \ldots, x_{c_k}. \varphi_{q'} \)
- if \( q = q_1 \bowtie q_2 \) then \( \varphi_q = \varphi_{q_1} \land \varphi_{q_2} \)
- if \( q = q_1 \cup q_2 \) then \( \varphi_q = \varphi_{q_1} \lor \varphi_{q_2} \)
- if \( q = q_1 - q_2 \) then \( \varphi_q = \varphi_{q_1} \land \neg \varphi_{q_2} \)
Excercise 3  It was stated in the lecture that query mappings under named perspective can be translated into query mappings under unnamed perspective. Specify this translation.

Solution.  Let \( M[q] : D_{\text{named}} \to D_{\text{named}} \) be a query mapping, where \( D_{\text{named}} \) denotes the set of all database instances over a named perspective. Similarly, \( D_{\text{unnamed}} \) denotes the set of all database instances over an unnamed perspective. We define the functions

\[
uu : D_{\text{named}} \to D_{\text{unnamed}} \quad \text{and} \quad \unu : D_{\text{unnamed}} \to D_{\text{named}}
\]

For any table \( R[a_1, \ldots, a_n] \) of a named database instance \( I \), \( \unu(R[a_1, \ldots, a_n]) = R \) fixes the order of the columns so that \( a_i \) becomes the \( i \)th column and deletes the attribute names.

For any table \( R \) of an unnamed database instance \( I \), \( \unu(R) = R[a_1, \ldots, a_n] \) creates the attribute names so that the \( i \)th column obtains the name \( a_i \).

Then the required translation is

\[
M[q] : D_{\text{named}} \to D_{\text{named}} \quad \mapsto \quad \nuu \circ M[q] \circ \unu : D_{\text{unnamed}} \to D_{\text{unnamed}}
\]
Excercise 4  Complete the proof that $RA_{\text{named}} \sqsubseteq DI_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are (a) domain independent and (b) equivalent to the input query. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries. Use the mappings from the previous exercise to compare named and unnamed results.
Complete the proof that RA\textsubscript{named} \sqsubseteq DI\textsubscript{unnamed} from the lecture by showing that the results of the transformation are \textbf{domain independent}. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If \( q = R \) with signature \( R[a_1, \ldots, a_n] \), then \( \varphi_q = R(x_{a_1}, \ldots, x_{a_n}) \). Query \( \varphi_q \) is DI, since the values of \( x_{a_i} \) belong to \( \text{adom}(R) \subseteq \text{adom}(I) \).

- If \( q = \{a_1 \mapsto c\} \), then \( \varphi_q = (x_{a_1} \approx c) \). Query \( \varphi_q \) is DI, since \( c \in \text{adom}(q) \).
Complete the proof that $\mathrm{RA}_{\text{named}} \subseteq \mathrm{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are domain independent. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = \sigma_{a_i = a_j}(q')$, then $\varphi_q = \varphi_{q'} \land (x_{a_i} \approx x_{a_j})$.

1. By IH: $\varphi_{q'}$ is DI.

2. Let us assume that $\varphi_{q'} \land (x_{a_i} \approx x_{a_j})$ is not DI.

3. By (2): There is some $D, D' \subseteq \text{dom}$ and a tuple $[c_1, \ldots, c_n]$ such that $a$ is an answer to $\varphi_{q'} \land (x_{a_i} \approx x_{a_j})$ w.r.t. $D$ but not w.r.t. $D'$.

4. By (3): $[c_1, \ldots, c_n]$ is an answer to $\varphi_{q'}$ w.r.t. $D$.

5. By (3): $[c_1, \ldots, c_n]$ is not an answer to $\varphi_{q'}$ w.r.t. $D'$ (note that, $c_i = c_j$).

6. By (4) and (5): Contradiction with (1).
Complete the proof that $\text{RA}_{\text{named}} \sqsubseteq \text{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are domain independent. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = q_1 \cup q_2$, then $\varphi_q = \varphi_{q_1} \lor \varphi_{q_2}$.

  1. By IH: $\varphi_{q_1}$ and $\varphi_{q_2}$ are DI.

  2. Let us assume that $\varphi_q$ is not DI.

  3. By (2): There is some $D, D' \subseteq \text{dom}$ and a tuple $[c_1, \ldots, c_n]$ such that $a$ is an answer to $\varphi_q$ w.r.t. $D$ but not w.r.t. $D'$.

  4. By (3): For some $i \in \{1, 2\}$, $[c_1, \ldots, c_n]$ is an answer $a$ is an answer to $\varphi_{q_i}$ w.r.t. $D$ but not w.r.t. $D'$.

  5. By (4): Contradiction with (1).
Complete the proof that $\text{RA}_{\text{named}} \sqsubseteq \text{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are domain independent. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = \delta_{b_1, \ldots, b_n} \rightarrow a_1, \ldots, a_n q'$, then $\varphi_q = \exists y_{b_1}, \ldots, y_{b_n}. (x_{a_1} \approx y_{b_1}) \land \ldots \land (x_{a_n} \approx y_{b_n}) \land \varphi_{q'}[y_{a_1}, \ldots, y_{a_n}]$.

  1. By IH: $\varphi_{q'}$ is DI.

  2. Let us assume that $\varphi_q$ is not DI.

  3. By (2): There is some $D, D' \subseteq \text{dom}$ and a tuple $[c_1, \ldots, c_n]$ such that $a$ is an answer to $\varphi_q$ w.r.t. $D$ but not w.r.t. $D'$. Then, there is some (exactly one) permutation $[d_1, \ldots, d_n]$ of $[c_1, \ldots, c_n]$ such that:

    (a) By (3): $[d_1, \ldots, d_n]$ is an answer to $\varphi_{q'}$ w.r.t. $D$.

    (b) By (3): $[d_1, \ldots, d_n]$ is not an answer to $\varphi_{q'}$ w.r.t. $D'$.

  4. By (3.a) and (3.b): Contradiction with (1).
Complete the proof that $\text{RA}_{\text{named}} \sqsubseteq \text{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are **domain independent**. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = \pi_{a_1,...,a_n}(q')$, then $\varphi_q = \exists x_{c_1}, \ldots, x_{c_k}. \varphi_{q'}$.

- If $q = q_1 \bowtie q_2$, then $\varphi_q = \varphi_{q_1} \land \varphi_{q_2}$.

- If $q = q_1 - q_2$, then $\varphi_q = \varphi_{q_1} \land \neg \varphi_{q_2}$.
Complete the proof that $\text{RA}_{\text{named}} \subseteq \text{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are equivalent to the input query. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = R$ with signature $R[a_1, \ldots, a_n]$, then $\varphi_q = R(x_{a_1}, \ldots, x_{a_n})$.

- If $q = \{a_1 \mapsto c\}$, then $\varphi_q = (x_{a_1} \approx c)$.

- If $q = \sigma_{a_i = a_j}(q')$, then $\varphi_q = \varphi_{q'} \land (x_{a_i} \approx x_{a_j})$.

- If $q = \delta_{b_1, \ldots, b_n \rightarrow a_1, \ldots, a_n} q'$, then $\varphi_q = \exists y_{b_1}, \ldots, y_{b_n}.(x_{a_1} \approx y_{b_1}) \land \ldots \land (x_{a_n} \approx y_{b_n}) \land \varphi_{q'}[y_{a_1}, \ldots, y_{a_n}]$. 
Complete the proof that $\text{RA}_{\text{named}} \subseteq \text{DI}_{\text{unnamed}}$ from the lecture by showing that the results of the transformation are equivalent to the input query. In each case, show that the claimed property holds true for each case of the recursive construction under the assumption (induction hypothesis) that it has already been established for all subqueries.

- If $q = \pi_{a_1, \ldots, a_n}(q')$, then $\varphi_q = \exists x_{c_1}, \ldots, x_{c_k}. \varphi_{q'}$.

- If $q = q_1 \bowtie q_2$, then $\varphi_q = \varphi_{q_1} \land \varphi_{q_2}$.

- If $q = q_1 \cup q_2$, then $\varphi_q = \varphi_{q_1} \lor \varphi_{q_2}$.

- If $q = q_1 - q_2$, then $\varphi_q = \varphi_{q_1} \land \neg \varphi_{q_2}$. 
Excercise 5  Consider a binary predicate $R$ and the \texttt{AD$_{unnamed}$} query

$$\varphi[x, y] = \neg (R(x, y) \land R(y, x)).$$

Use the construction from the lecture to express it as an \texttt{RA$_{named}$} query.

Consider an AD query $q = \varphi[x_1, \ldots, x_n]$. For every variable $x$, we use a distinct attribute name $a_x$

- if $\varphi = R(t_1, \ldots, t_m)$ with signature $R[a_1, \ldots, a_m]$ with variables $x_1 = t_{v_1}, \ldots, x_n = t_{v_n}$ and constants $c_1 = t_{w_1}, \ldots, c_k = t_{w_k}$, then $E_\varphi = \delta_{a_{v_1}\ldots a_{v_n} \rightarrow a_{x_1}\ldots a_{x_n}}(\sigma_{a_{w_1} = c_1} \ldots \sigma_{a_{w_k} = c_k}(R) \ldots)$

- if $\varphi = (x \approx c)$, then $E_\varphi = \{a_x \mapsto c\}$

- if $\varphi = (x \approx y)$, then $E_\varphi = \sigma_{a_x = a_y}(E_{a_x, \text{adom}} \bowtie E_{a_y, \text{adom}})$

- if $\varphi = \neg \psi$, then $E_\varphi = (E_{a_x, \text{adom}} \bowtie \ldots \bowtie E_{a_y, \text{adom}}) - E_\psi$

- if $\varphi = \varphi_1 \land \varphi_2$, then $E_\varphi = E_{\varphi_1} \bowtie E_{\varphi_2}$

- if $\varphi = \exists y. \psi$ where $\psi$ has free variables $y, x_1, \ldots, x_n$, then $E_\varphi = \pi_{a_{x_1}, \ldots, a_{x_n}} E_\psi$
Exercise 6 Complete the constructions for the proof of $AD \subseteq RA$ given in the lecture.

(a) Define the relational algebra expression $E_{a,\text{adom}}$ such that

$$E_{a,\text{adom}}(I) = \{\{a \mapsto c\} \mid c \in \text{dom}(I, q)\}.$$  

*Hint: Assume that the query and the database schema are known.*

(b) Define the expressions $E_\varphi$ for $\varphi = \varphi_1 \lor \varphi_2$ and $\varphi = \forall y.\psi$ in terms of expressions that have already been defined in the lecture.

(c) Give a direct definition for the expression $E_\varphi$ for $\varphi = \varphi_1 \lor \varphi_2$. 
Excercise 7  Use the function \( rr \) from the lecture to compute the set of range-restricted variables for the following FO queries:

1. \( \exists y_{\text{SID}}, y_{\text{Stop}}, y_{\text{To}}. (\text{Stops}(y_{\text{SID}}, y_{\text{Stop}}, "true") \land \text{Connect}(y_{\text{SID}}, y_{\text{To}}, x_{\text{Line}}))[x_{\text{Line}}] \)

2. \( \neg \text{Lines}(x, "bus")[x] \)

3. \( (\text{Connect}(x_1, "42", "85") \lor \text{Connect}("57", x_2, "85"))[x_1, x_2] \)

4. \( \forall y. p(x, y)[x] \)

5. \( \exists x.(((p(x) \rightarrow q(c)) \rightarrow p(x)) \rightarrow p(x)) \)

Which of these queries is a safe-range query? Which of the queries is domain independent?
1. $\exists y_{\text{SID}}, y_{\text{Stop}}, y_{\text{To}}. (\text{Stops}(y_{\text{SID}}, y_{\text{Stop}}, \text{"true"}) \land \text{Connect}(y_{\text{SID}}, y_{\text{To}}, x_{\text{Line}}))[x_{\text{Line}}]$

$$rr(R(t_1, \ldots, t_n)) = \{x \mid x \text{ a variable among the } t_1, \ldots, t_n\}$$

$$rr(x \approx a) = \{x\}$$

$$rr(x \approx y) = \emptyset$$

$$rr(\varphi_1 \land \varphi_2) = \begin{cases} 
rr(\varphi_1) \cup \{x, y\} & \text{if } \varphi_2 = (x \approx y) \text{ and } \{x, y\} \cap rr(\varphi_1) \neq \emptyset \\
rr(\varphi_1) \cup rr(\varphi_2) & \text{otherwise}
\end{cases}$$

$$rr(\varphi_1 \lor \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)$$

$$rr(\exists y . \psi) = \begin{cases} 
rr(\psi) \setminus \{y\} & \text{if } y \in rr(\psi) \\
\text{throw new NotSafeException()} & \text{if } y \notin rr(\psi)
\end{cases}$$

$$rr(\neg \psi) = \emptyset \quad \text{if } rr(\psi) \text{ is defined (no exception)}$$
2. $\neg$Lines($x$, "bus")[$x$]

$$
rr(R(t_1, \ldots, t_n)) = \{ x \mid x \text{ a variable among the } t_1, \ldots, t_n \}
$$

$$
rr(x \approx a) = \{ x \}
$$

$$
rr(x \approx y) = \emptyset
$$

$$
rr(\varphi_1 \land \varphi_2) = \begin{cases} 
rr(\varphi_1) \cup \{ x, y \} & \text{if } \varphi_2 = (x \approx y) \text{ and } \{ x, y \} \cap rr(\varphi_1) \neq \emptyset \\
rr(\varphi_1) \cup rr(\varphi_2) & \text{otherwise}
\end{cases}
$$

$$
rr(\varphi_1 \lor \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)
$$

$$
rr(\exists y. \psi) = \begin{cases} 
rr(\psi) \setminus \{ y \} & \text{if } y \in rr(\psi) \\
\textbf{throw new} \text{ NotSafeException}() & \text{if } y \notin rr(\psi)
\end{cases}
$$

$$
rr(\neg \psi) = \emptyset \quad \text{if } rr(\psi) \text{ is defined (no exception)}
$$
3. \( (\text{Connect}(x_1, "42", "85") \lor \text{Connect}("57", x_2, "85"))[x_1, x_2] \)

\[
rr(R(t_1, \ldots, t_n)) = \{x \mid x \text{ a variable among the } t_1, \ldots, t_n\}
\]

\[
rr(x \approx a) = \{x\}
\]

\[
rr(x \approx y) = \emptyset
\]

\[
rr(\varphi_1 \land \varphi_2) = \begin{cases} 
rr(\varphi_1) \cup \{x, y\} & \text{if } \varphi_2 = (x \approx y) \text{ and } \{x, y\} \cap rr(\varphi_1) \neq \emptyset \\
rr(\varphi_1) \cup rr(\varphi_2) & \text{otherwise}
\end{cases}
\]

\[
rr(\varphi_1 \lor \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)
\]

\[
rr(\exists y. \psi) = \begin{cases} 
rr(\psi) \setminus \{y\} & \text{if } y \in rr(\psi) \\
\textbf{throw new} \text{ NotSafeException()} & \text{if } y \notin rr(\psi)
\end{cases}
\]

\[
rr(\neg \psi) = \emptyset & \text{if } rr(\psi) \text{ is defined (no exception)}
\]
4. $\forall y.p(x, y)[x]$

$rr(R(t_1, \ldots, t_n)) = \{x \mid x \text{ a variable among the } t_1, \ldots, t_n\}$

$rr(x \approx a) = \{x\}$

$rr(x \approx y) = \emptyset$

$rr(\varphi_1 \land \varphi_2) = \begin{cases} 
rr(\varphi_1) \cup \{x, y\} & \text{if } \varphi_2 = (x \approx y) \text{ and } \{x, y\} \cap rr(\varphi_1) \neq \emptyset \\
rr(\varphi_1) \cup rr(\varphi_2) & \text{otherwise}
\end{cases}$

$rr(\varphi_1 \lor \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)$

$rr(\exists y.\psi) = \begin{cases} 
rr(\psi) \setminus \{y\} & \text{if } y \in rr(\psi) \\
throw \text{ new } \text{NotSafeException}() & \text{if } y \notin rr(\psi)
\end{cases}$

$rr(\neg \psi) = \emptyset$ if $rr(\psi)$ is defined (no exception)
5. \( \exists x.(((p(x) \rightarrow q(c)) \rightarrow p(x)) \rightarrow p(x)) \)

\[
rr(R(t_1, \ldots, t_n)) = \{x \mid x \text{ a variable among the } t_1, \ldots, t_n\}
\]

\[
rr(x \approx a) = \{x\}
\]

\[
rr(x \approx y) = \emptyset
\]

\[
rr(\varphi_1 \land \varphi_2) = \begin{cases} rr(\varphi_1) \cup \{x, y\} & \text{if } \varphi_2 = (x \approx y) \text{ and } \{x, y\} \cap rr(\varphi_1) \neq \emptyset \\ rr(\varphi_1) \cup rr(\varphi_2) & \text{otherwise} \end{cases}
\]

\[
rr(\varphi_1 \lor \varphi_2) = rr(\varphi_1) \cap rr(\varphi_2)
\]

\[
rr(\exists y. \psi) = \begin{cases} rr(\psi) \setminus \{y\} & \text{if } y \in rr(\psi) \\ \text{throw new NotSafeException()} & \text{if } y \notin rr(\psi) \end{cases}
\]

\[
rr(\neg \psi) = \emptyset \quad \text{if } rr(\psi) \text{ is defined (no exception)}
\]