Chapter 8

The Logic of Query Languages

8.1. Using the \texttt{part\_cost} relation of Example 8.14, write safe Datalog rules to find those suppliers who supply all basic parts.

\textbf{Answer:}

\begin{verbatim}
supply\_all(Supplier) ← part\_cost(_,Supplier,_,_),
part\_cost(Part,_,_),
¬not\_supply(Supplier,Part).
not\_supply(Sup,Part) ← part\_cost(Part,_,_),
part\_cost(_,Sup,_,_),
¬part\_cost(Part,Sup,_,_).
\end{verbatim}

8.2. Write a Datalog query to find students who have taken at least two classes, and got the highest grade (possibly with others) in every class they took.

\textbf{Answer:}

\begin{verbatim}
elite\_students(X) ← took(X,Z1,_,),took(X,Z2,_,),Z1 ≠ Z2,
¬outperformed(X).
outperformed(X) ← took(X,C,GX),took(Y,C,GY),GY > GX.
\end{verbatim}

Here, we have used only one negation. In fact we have transformed the counterexample query: “there is no class in which the student did not get the highest grade” into “there is no class in which some other student got a higher grade.” Reformulating the query as the negation of a counterexample remains the first step for replacing universal quantifiers with negated existential ones.

8.3. Express the previous query in SQL using the \textbf{EXISTS} construct.

\textbf{Answer:}

\begin{verbatim}
SELECT t0.name
FROM took t0, took t1
WHERE t0.name=t1.name AND
      t0.course <>t1.course AND
      NOT EXISTS (  

does not compile
does not compile

\end{verbatim}
8. Universal quantified queries can also be expressed in SQL using the set aggregate \texttt{COUNT}. Reformulate the last query and that of Example 8.13 in SQL, using the \texttt{COUNT} aggregate.

\textbf{Answer:}

\begin{verbatim}
SELECT t1.name, 
FROM took t1, took t2 
WHERE t2.name = t1.name AND 
t1.course <> t2.course AND 
COUNT(SELECT t3.* 
    FROM took t3 
    WHERE t3.course=t1.course AND 
t3.grade > t1.grade) = 0
\end{verbatim}

8.5. A relationally complete relational algebra (RA) contains set union, set difference, set intersection, Cartesian product, selection, and projection.

\begin{itemize}
    \item a. Show that the expressive power of RA does not change if we drop set intersection.
    \item b. List the monotonic operators of RA.
    \item c. Show that there is a loss in expressive power if we drop set difference from the RA described above.
\end{itemize}

\textbf{Answer:}

\begin{itemize}
    \item a. Intersection can be expressed in terms of other operators: \( A \cap B = A - (A - B) = B - (B - A) \).
    \item b. Monotonic operators (from the five basic RA operators): Union, Cartesian product, Selection, Projection.
    \item c. As illustrated by the reformulation of set intersection using set difference, relational algebra expressions containing set difference can be monotonic. However, expressions containing only monotonic operators are monotonic in their operand(s). Now, if we assume that set difference can be represented as an expression of monotonic operators, then it would follow that set difference is monotonic—a contradiction. Thus relational algebra with only the monotonic operators is not relationally complete.
\end{itemize}
8.6. Define generalized projection using the other RA operators.

**Answer:**
The two new features provided by generalized projections are (i) repeated columns, and (ii) constant columns. The following two examples illustrate how to emulate them using the basic RA. Let $A$ be an $n$-ary relation, then:

Adding a copy of the $j^{th}$ column ($1 \leq j \leq n$) as the $(n+1)^{th}$ a column of $A$.

$$\pi_{1, \ldots, n, j} A = \sigma_{\delta_j = \delta_{(n+1)}} A \times \pi_{\delta_j} A$$

Adding a column containing the constant $c$ at the end of $A$:

$$\pi_{1, \ldots, n, c} A = A \times \{c\}$$

We can then move these added columns to any position we desire, by the standard projection.

8.7. Which of the following rules and predicates are safe if $b_1$ and $b_2$ are database predicates?

- $r_1 : p(X, X) \leftarrow b_2(Y, Y, a), b_1(X), X > Y$.
- $r_2 : q(X, Y) \leftarrow p(X, Z), p(Z, Y)$.
- $r_3 : s(X) \leftarrow b_2(Y, Y, a), X > Y, \neg b_1(X)$.

Translate the safe Datalog rules and predicates in this program into equivalent relational algebra expressions.

**Answer:**

- $r_1$ is safe. Its RA translation is: $\pi_{X,Y} (\sigma_{X,Y=a} (b_2 \times b_1))$
- $r_2$ is safe. Its RA translation is: $\pi_{X,Y} (\sigma_{X,Y=a} (P \times P))$
- $r_3$ is not safe.

8.8. Improve the translation proposed for negated rule goals into RA by minimizing the number of columns in the relations involved in the set difference. Translate the following rule:

$$r_4 : n(X, X) \leftarrow b_1(X, Y, \_), \neg b_2(X, a, \_).$$

**Answer:**
The main improvement consists in projecting out columns so the set difference is performed on smaller relations. The translation of Chapter 8, which models the body of the rules through a positive rule $rp$ and a negative rule $rn$, yields the following rules for the example at hand:

- $r_{4p} : ph(X, Y) \leftarrow b_1(X, Y, \_)$.
- $r_{4n} : nh(X, Y) \leftarrow b_1(X, Y, \_), b_2(X, a, \_)$.

Here the head of the rules contains all the variable in the body (since the rule is safe these must be the same in both rules). Now the original rule $r_4$ can be rewritten as follows:

- $r'_4 : n(X, X) \leftarrow \text{diff}(X, Y)$.
- $r''_4 : \text{diff}(X, Y) \leftarrow ph(X, Y), \neg nf(X, Y)$. 
Now, the rules $r_4p, r_4n$, and $r'_4$ are translated into RA according to the standard procedure for positive rules given in Chapter 8, while $r''_4$ is translated into: $DIFF = PH − NH$.

We can now answer the original question, and minimize the number of columns, by observing that the variables actually needed in $ph$ and $nh$ are: (i) those in the head of the original rule, and (ii) those in the negated goal. In the example at hand, for instance, only $X$ is needed; $Y$ is not needed. Therefore, our rules can be simplified as follows:

\[
\begin{align*}
  r_4p & : \ ph(X) \leftarrow b1(X, Y, \_). \\
  r_4n & : \ nh(X) \leftarrow b1(X, Y, \_), b2(X, a, \_). \\
  r'_4 & : n(X, X) \leftarrow \text{diff}(X). \\
  r''_4 & : \text{diff}(X) \leftarrow \text{ph}(X), \neg\text{nf}(X).
\end{align*}
\]

8.9. Prove the converse of Theorem 8.3: that is, show that for every RA expression, there exists an equivalent Datalog program, which is safe and nonrecursive.

**Answer:**

This is simple. Look at Section 8.3 where the definition of relational algebra operators are given using TRC notation and just write equivalent rules for each one. For instance for Cartesian product $R \times S$ you only need to write:

\[
\begin{align*}
  r_{cp}s(X_1, \ldots, X_n, Y_1, \ldots, Y_m) & \leftarrow r(X_1, \ldots, X_n), s(X_1, \ldots, X_n, Y_1, \ldots, Y_m).
\end{align*}
\]

8.10. Express in Datalog the division operation $R(A, B) \div S(B)$. By translating the rules so obtained into RA, express relation division in terms of the other RA operators.

**Answer:**

\[
Q(A) = R(A, B) \div S(B)
\]

We can define division via a quotient rule and a remainder rule. Thus $\text{rem}(A)$ contains each value $a$ for which $\exists b \in S$ s.t. $(a, b) \notin R$. The quotient is then the others as in the first column of $R$.

\[
\begin{align*}
  r_1 : q(A) & \leftarrow r(A, \_), \neg\text{rem}(A). \\
  r_2 : \text{rem}(A) & \leftarrow r(A, \_), s(B), \neg r(A, B).
\end{align*}
\]

Observe that these rules are safe, and thus can be translated into an equivalent relational algebra expression.

To obtain a simpler relational expression we can start by capturing the positive body and negative body through the following two rules:

\[
\begin{align*}
  r_{2p}(A, B) & \leftarrow r(A, \_), s(B). \\
  r_{2n}(A, B) & \leftarrow r(A, \_), s(B), r(A, B).
\end{align*}
\]

Now these two rules can be expressed by the following relational expressions: $R2p = \pi_{S1}R \times S$ and $R2n = R2p \cap R$. Their difference can be simplified as follows: $R2p − R2n = R2p − (R2p \cap R) = R2p − R$. Thus, the expression equivalent to rule $r_2$ is: