Introduction to Databases

Tutorial 2

Dr Paolo Guagliardo

Fall 2020 (week 4)

Problem 1 (mandatory). Consider the following schema:

CUSTOMER(· , · , ·) of arity 3, where the first position is the customer's *ID*, the second is the customer's *Name*, and the third is the *City* where the customer lives.

ACCOUNT (· , · , · , ·) of arity 4, in which the first position is the account's *Number*, the second is its *Branch*, the third is the ID of the customer owning the account, and the fourth is the *Balance* on the account.

Write the following queries in relational calculus:

(1) "ID and name of customers who own an account in a branch in their city."

(2) "ID and name of customers who do **not** own any account."

(3) "ID and name of customers who own an account with a balance which is no less than the balance of any other account."

Note: Write the queries directly in relational calculus, without translating from relational algebra.

Solution.

(1) $\{x_1, x_2 \mid \exists x_3, y_1, y_4\ \textsc{Customer}(x_1, x_2, x_3) \wedge \textsc{Account}(y_1, x_3, x_1, y_4)\}$

(2) $\{x_1, x_2 \mid \exists x_3\ \textsc{Customer}(x_1, x_2, x_3) \wedge \neg\exists y_1, y_2, y_4\ \textsc{Account}(y_1, y_2, x_1, y_4)\}$

(3) $\{x_1, x_2 \mid \exists x_3\ \textsc{Customer}(x_1, x_2, x_3) \wedge \exists y_1, y_2, y_4\ \textsc{Account}(y_1, y_2, x_1, y_4)$
$\wedge\, \forall z_1, z_2, z_3, z_4\ \big(\textsc{Account}(z_1, z_2, z_3, z_4) \rightarrow \neg(y_4 < z_4)\big)\}$

Problem 2 (optional). Given a schema consisting of a binary relation R and a ternary relation S, write a relational calculus query that computes the active domain.

Solution. Recall that the active domain of a database D is the set of all constants occurring in D. Thus, in RA, a query that computes the active domain (of any given database over which it is executed) must take the union of all projections on each single attributes of every relation. Since projecting over an attribute/position in RC means existentially quantifying all others, the query we are looking for is the following:

$$\Big\{x \mid \big(\exists y\ R(x,y)\big) \vee \big(\exists y\ R(y,x)\big) \vee \big(\exists y, z\ S(x,y,z)\big) \vee \big(\exists y, z\ S(y,x,z)\big) \vee \big(\exists y, z\ S(y,z,x)\big)\Big\} \tag{1}$$

Intuitively the above says: "return all values x that appear in the first column of R, or in the second column of R, or in the first column of S, or in the second column of S, or in the third column of S." Observe that we can use the same variables for all disjuncts (but careful: this would not work with conjunction) and write more succinctly:

$$\Big\{x \mid \exists y, z\ \big(R(x,y) \vee R(y,x) \vee S(x,y,z) \vee S(y,x,z) \vee S(y,z,x)\big)\Big\} \tag{2}$$

Problem 3 (mandatory). Consider the schema of Problem 1 and assume that CUSTOMER is over attributes *ID*, *Name*, *City* (in this order) and ACCOUNT is over attributes *Number*, *Branch*, *CustID*, *Balance* (in this order). Express the following relational algebra query in relational calculus:

$$\text{CUSTOMER} \bowtie \big(\pi_{\text{ID,City}}(\text{CUSTOMER}) \cap \rho_{\text{CustID}\to\text{ID},\,\text{Branch}\to\text{City}}\big(\pi_{\text{Branch,CustID}}(\text{ACCOUNT})\big)\big) \tag{3}$$

Note: Use the translation rules from RA to RC we have seen in class.

Solution. We first need to (equivalently) rewrite the given RA expression into one that uses only primitive operations, for which we have translation rules into RC. This would be the standard procedure to follow[1] but, as a matter of fact, by first rewriting $\cap$ in terms of difference, then translating into RC and finally applying some first-order logic equivalences, we can obtain a direct translation rule for $\cap$, which we can then reuse in other translations from RA to RC.

To this end, let us consider the RA expression $e \cap e'$, in which e and e' are arbitrary RA expressions (over the same set of attributes). From the lectures, we know that $e \cap e' \equiv e - (e - e')$. By applying the translation rule for difference (twice), we get $\varphi \wedge \neg(\varphi \wedge \neg\varphi')$, where φ is the translation of e and φ' is the translation of e'. We now make use of some FO equivalences:

$$\begin{aligned}
\varphi \wedge \neg(\varphi \wedge \neg\varphi') &\equiv \varphi \wedge (\neg\varphi \vee \neg\neg\varphi') && [\text{by DeMorgan's law for } \wedge] \\
&\equiv \varphi \wedge (\neg\varphi \vee \varphi') && [\text{by double negation elimination}] \\
&\equiv (\varphi \wedge \neg\varphi) \vee (\varphi \wedge \varphi') && [\text{by distributivity of } \wedge \text{ over } \vee] \\
&\equiv \varphi \wedge \varphi' && [\text{because } \varphi \wedge \neg\varphi \text{ is false}]
\end{aligned}$$

Therefore, we obtain the following rule for translating $e \cap e'$ into RC:

1. Translate the expression e into a formula φ,
2. Translate the expression e' into a formula φ',
3. Translate $e \cap e'$ into $\varphi \wedge \varphi'$.

This rule is the same as the one for Cartesian product, but there is one major difference: for intersection the formulas φ and φ' of which we take the conjunction have the same free variables, whereas in the case of Cartesian product they have no free variables in common.

Let us now go back to the translation of (3). By rewriting the natural join we obtain:

$$\pi_{\text{ID, Name, City}}\Big(\sigma_{\text{ID}=\text{ID}' \,\wedge\, \text{City}=\text{City}'}\big(\underbrace{\rho_{\text{ID}\to\text{ID}',\,\text{City}\to\text{City}'}(\text{CUSTOMER})}_{E_1} \times \underbrace{(\pi_{\text{ID, City}}(\text{CUSTOMER}) \cap E)}_{E_2}\big)\Big)$$

where $E = \rho_{\text{CustID}\to\text{ID},\,\text{Branch}\to\text{City}}\big(\pi_{\text{Branch,CustID}}(\text{ACCOUNT})\big)$.
Then we associate each attribute in the schema with a unique variable name:

$$\eta = \{\, \text{ID} \mapsto x_1, \text{Name} \mapsto x_2, \text{City} \mapsto x_3, \text{Number} \mapsto y_1, \text{Branch} \mapsto y_2, \text{CustID} \mapsto y_3, \text{Balance} \mapsto y_4 \,\}$$

We start the translation from the innermost expressions:

- The translation of CUSTOMER is $\text{CUSTOMER}(x_1, x_2, x_3)$
- The translation of ACCOUNT is $\text{ACCOUNT}(y_1, y_2, y_3, y_4)$

To translate E_1 we first map ID' and City' to new, arbitrarily chosen[2] variables x'_1 and x'_3, respectively, because the environment η does not have bindings for these attributes:

$$\eta := \eta \cup \{\, \text{ID}' \mapsto x'_1, \text{City}' \mapsto x'_3 \,\}$$

[1] You must do this for all other derived operations except for intersection.
[2] As long as they have not been used already.

Then, since there are no occurrences of x'_1 and x'_3 already present in the translation of CUSTOMER,[3] we can simply replace x_1 by x'_1 and x_3 by x'_3:

$$\textsc{Customer}(x'_1, x_2, x'_3) \tag{φ_1}$$

The translation of $\pi_{\text{Branch, CustID}}(\textsc{Account})$ is $\exists y_1, y_4\ \textsc{Account}(y_1, y_2, y_3, y_4)$ and, by renaming y_3 to x_1 and y_2 to x_3, we obtain the translation of E, which is

$$\exists y_1, y_4\ \textsc{Account}(y_1, x_3, x_1, y_4) \tag{φ}$$

The translation of $\pi_{\text{ID,City}}(\textsc{Customer})$ is $\exists x_2\ \textsc{Customer}(x_1, x_2, x_3)$ and, by taking the conjunction with the translation of E (according to the direct rule for $\cap$ we proved earlier), we get the translation of E_2:

$$\big(\exists x_2\ \textsc{Customer}(x_1, x_2, x_3)\big) \wedge \underbrace{\big(\exists y_1, y_4\ \textsc{Account}(y_1, x_3, x_1, y_4)\big)}_{\varphi} \tag{φ_2}$$

To translate the selection, we replace the attributes in the condition by their corresponding variables and we take the conjunction with the translation of $E_1 \times E_2$, which is $\varphi_1 \wedge \varphi_2$, obtaining:

$$\varphi_1 \wedge \varphi_2 \wedge \big(x_1 = x'_1 \wedge x_3 = x'_3\big)$$

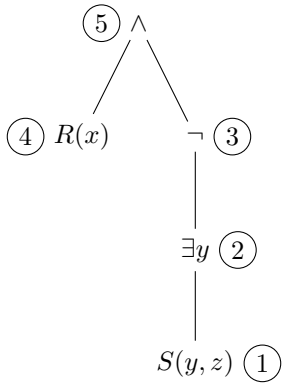
Finally, to translate the outermost projection, we existentially quantify variables x'_1 and x'_3 in the above formula. The final translation (where unnecessary parentheses are removed) is:

$$\exists x'_1, x'_3\ \textsc{Customer}(x'_1, x_2, x'_3) \wedge \exists x_2\ \textsc{Customer}(x_1, x_2, x_3) \\ \wedge \exists y_1, y_4\ \textsc{Account}(y_1, x_3, x_1, y_4) \wedge x_1 = x'_1 \wedge x_3 = x'_3$$

Problem 4 (optional). Given a relation R over attribute A, and a relation S over attributes A, B (in this order), translate the following relation calculus query to relational algebra:

$$\big\{\, x, z \mid R(x) \wedge \neg\exists y\ S(y, z) \,\big\}$$

Solution. The syntax tree of the (body of the) given relational calculus query is:



Under the mapping $\eta = \{x \mapsto A,\ y \mapsto B,\ z \mapsto C\}$, we get the following translations of each subexpression (corresponding to the subtrees above labelled with circled numbers):

[3] If there were such occurrences, say of x'_1, then we would have to replace all of them by a fresh new variable, say w (or whichever other name was not, and will not, be used anywhere else).

$\textcircled{1} \mapsto \rho_{A \to B,\, B \to C}(S)$

$\textcircled{2} \mapsto \pi_C\left(\textcircled{1}\right)$

$\textcircled{3} \mapsto \mathsf{Adom}_C - \textcircled{2}$

$\textcircled{4} \mapsto R$

$\textcircled{5} \mapsto \left(\textcircled{4} \times \mathsf{Adom}_C\right) \cap \left(\textcircled{3} \times \mathsf{Adom}_A\right)$

Putting it all together, we obtain the following:

$$(R \times \mathsf{Adom}_C) \cap \Big(\mathsf{Adom}_A \times \big(\mathsf{Adom}_C - \pi_C\big(\rho_{A \to B, B \to C}(S)\big)\big)\Big)$$