A Safe (but not very flexible) Relational Calculus

Jef Wijsen

Université de Mons (UMONS)

September 29, 2023

Motivation

$$sort(ABUS) = \{Nom, Cru, Annee\}$$
$$sort(CRU) = \{Cru, Millesime, Qualite\}$$

$$\{z \mid \exists x \exists y (ABUS(\text{`An'}, x, y) \wedge CRU(x, y, z))\}$$

Relation name R

Definition

If R is a relation name of arity n, and $x_1, \ldots, x_n$ are distinct variables, then $R(x_1, \ldots, x_n)$ is an SRC formula with $\mathit{free}(R(x_1, \ldots, x_n)) = \{x_1, \ldots, x_n\}$.

Selection $\sigma_{x=\text{'a'}}(E)$ and $\sigma_{x=y}(E)$

Definition

If φ is an SRC formula and $x \in \mathit{free}(\varphi)$, then $\varphi \wedge (x = \text{'a'})$ is an SRC formula with $\mathit{free}(\varphi \wedge (x = \text{'a'})) = \mathit{free}(\varphi)$.

Definition

If φ is an SRC formula and $x, y \in \mathit{free}(\varphi)$, then $\varphi \wedge (x = y)$ is an SRC formula with $\mathit{free}(\varphi \wedge (x = y)) = \mathit{free}(\varphi)$.

Projection $\pi_X(E)$

Definition

If φ is an SRC formula and $x \in \mathit{free}(\varphi)$, then $\exists x(\varphi)$ is an SRC formula with $\mathit{free}(\exists x(\varphi)) = \mathit{free}(\varphi) \setminus \{x\}$.

Join $E_1 \bowtie E_2$

Definition

If φ_1 and φ_2 are SRC formulas, then $\varphi_1 \wedge \varphi_2$ is an SRC formula with $\mathit{free}(\varphi_1 \wedge \varphi_2) = \mathit{free}(\varphi_1) \cup \mathit{free}(\varphi_2)$.

Rename $\rho_{A \mapsto B}(E)$

Definition

Let φ be an SRC formula and $x \in \mathit{free}(\varphi)$.
We can

- rename bound variables; and
- replace all free occurrences of x with a variable that does not occur in φ.

- We cannot replace x with y in $\exists y(R(x, y))$.
- If we rename y as z in $\exists y(R(x, y))$, we obtain $\exists z(R(x, z))$.

Union $E_1 \cup E_2$

Definition

If φ_1 and φ_2 are SRC formulas with $\mathit{free}(\varphi_1) = \mathit{free}(\varphi_2)$, then $\varphi_1 \vee \varphi_2$ is an SRC formula with $\mathit{free}(\varphi_1 \vee \varphi_2) = \mathit{free}(\varphi_1)$.

Difference $E_1 - E_2$

Definition

If φ_1 and φ_2 are SRC formulas with $\mathit{free}(\varphi_1) = \mathit{free}(\varphi_2)$, then $\varphi_1 \land \neg\varphi_2$ is an SRC formula with $\mathit{free}(\varphi_1 \land \neg\varphi_2) = \mathit{free}(\varphi_1)$.

SRC Query

Notation

We write $\varphi(x_1, \ldots, x_n)$ to denote that φ is an SRC formula with $\mathit{free}(\varphi) = \{x_1, \ldots, x_n\}$.

Definition

If $\varphi(x_1, \ldots, x_n)$ is an SRC formula, then $\{\langle x_1, \ldots, x_n\rangle \mid \varphi\}$ is an SRC query. Given database $\mathcal{I}$, the answer to this query is $\{\langle a_1, \ldots, a_n\rangle \mid a_1, \ldots, a_n \text{ constants such that } \mathcal{I} \models \varphi(a_1, \ldots, a_n)\}$.

The symbol $\models$ means "satisfies."

Expressiveness

- SRC was designed so as to capture SPJRUD. Theorem 1 shows that every SPJRUD query has indeed an equivalent SRC query.
- Conversely, SRC is not more expressive than SPJRUD (Theorem 2).

SPJRUD to SRC

- Let $sort(R) = \{A_1, \ldots, A_n\}$, in that order. $\text{A2C}(R) = R(z_{A_1}, \ldots, z_{A_n})$.
- $\text{A2C}(\sigma_{A=\text{`a'}}(E)) = \text{A2C}(E) \wedge (z_A = \text{`a'})$.
- $\text{A2C}(\sigma_{A=B}(E)) = \text{A2C}(E) \wedge (z_A = z_B)$.
- $\text{A2C}(\pi_X(E)) = \exists z_{A_1} \ldots \exists z_{A_n}(\text{A2C}(E))$ where $\{A_1, \ldots, A_n\} = sort(E) \setminus X$.
- $\text{A2C}(E_1 \bowtie E_2) = \text{A2C}(E_1) \wedge \text{A2C}(E_2)$.
- $\text{A2C}(\rho_{A \mapsto B}(E))$ is the SRC formula obtained from $\text{A2C}(E)$ by (1) renaming bound variables z_B in $\text{A2C}(E)$, and then (2) replacing each free occurrence of z_A with z_B.
- $\text{A2C}(E_1 \cup E_2) = \text{A2C}(E_1) \vee \text{A2C}(E_2)$.
- $\text{A2C}(E_1 - E_2) = \text{A2C}(E_1) \wedge \neg\text{A2C}(E_2)$.

Theorem 1 (SPJRUDtoSRC)

Let E be an SPJRUD query with $sort(E) = \{A_1, \ldots, A_n\}$. Let $\varphi = \text{A2C}(E)$. Then, φ is an SRC formula, $free(\varphi) = \{z_{A_1}, \ldots, z_{A_n}\}$, and for every database $\mathcal{I}$, for all constants $a_1, \ldots, a_n$, $\{A_1 : a_1, \ldots, A_n : a_n\} \in [\![E]\!]^{\mathcal{I}}$ if and only if $\mathcal{I} \models \varphi(a_1, \ldots, a_n)$.

Example SPJRUD to SRC

SPJRUD to SRC

Assume $sort(R) = \{A, B\}$ and $sort(S) = \{A, D\}$.

$$\begin{aligned} E &= \pi_A(R \bowtie \rho_{D \mapsto B}(S)) \\ \text{A2C}(E) &= \exists z_B(R(z_A, z_B) \wedge S(z_A, z_B)) \end{aligned}$$

Example SPJRUD to SRC

SPJRUD to SRC

Assume $sort(R) = \{A, B\}$.

$$\begin{aligned} E &= \rho_{A \mapsto B}(\pi_A(R)) \\ \text{A2C}(\pi_A(R)) &= \exists z_B(R(z_A, z_B)) \\ \text{A2C}(E) &= \exists y(R(z_B, y)) \end{aligned}$$

Note:

- First, z_B is renamed as y.
- Then, z_A is replaced with z_B.

SRC to SPJRUD

- Let $sort(R) = \{A_1, A_2, \ldots, A_n\}$, in that order.
  $\text{C2A}(R(x_1, x_2, \ldots, x_n)) = \rho_{A_1 A_2 \ldots A_n \mapsto C_{x_1} C_{x_2} \ldots C_{x_n}}(R)$.
- $\text{C2A}(\varphi \wedge (x = \text{'a'})) = \sigma_{C_x = \text{'a'}}(\text{C2A}(\varphi))$
- $\text{C2A}(\varphi \wedge (x = y)) = \sigma_{C_x = C_y}(\text{C2A}(\varphi))$
- $\text{C2A}(\exists x(\varphi)) = \pi_{\{C_y \mid y \in free(\varphi)\} \setminus \{C_x\}}(\text{C2A}(\varphi))$
- $\text{C2A}(\varphi_1 \wedge \varphi_2) = \text{C2A}(\varphi_1) \bowtie \text{C2A}(\varphi_2)$
- $\text{C2A}(\varphi_1 \vee \varphi_2) = \text{C2A}(\varphi_1) \cup \text{C2A}(\varphi_2)$
- $\text{C2A}(\varphi_1 \wedge \neg\varphi_2) = \text{C2A}(\varphi_1) - \text{C2A}(\varphi_2)$

Theorem 2 (SRCtoSPJRUD)

Let $\varphi(x_1, \ldots, x_n)$ be an SRC formula and $E = \text{C2A}(\varphi)$. Then, E is an SPJRUD query, $sort(E) = \{C_{x_1}, \ldots, C_{x_n}\}$, and for every database $\mathcal{I}$, for all constants $a_1, \ldots, a_n$,
$\mathcal{I} \models \varphi(a_1, \ldots, a_n)$ if and only if $\{C_{x_1} : a_1, \ldots, C_{x_n} : a_n\} \in [\![E]\!]^{\mathcal{I}}$.

Example SRC to SPJRUD

SRC to SPJRUD

Assume $\mathit{sort}(R) = \{A, B\}$ and $\mathit{sort}(S) = \{A, D\}$.

$$\begin{aligned} \varphi &= \exists y (R(x, y) \wedge S(x, y)) \\ \mathrm{C2A}(\varphi) &= \pi_{\{C_x\}}(\rho_{AB \mapsto C_x C_y}(R) \bowtie \rho_{AD \mapsto C_x C_y}(S)) \end{aligned}$$

Discussion

- SRC is a syntactically (highly!) restricted subclass of first-order formulas.
- A first-order formula that is not an SRC formula may be **equivalent** to an SRC formula. For example,

$$R(x, y, z) \wedge \neg(S(x, y) \vee T(y, z)) \tag{1}$$

is equivalent to

$$\begin{aligned} R(x, y, z) \wedge & (\exists z'(R(x, y, z')) \wedge \neg S(x, y)) \\ & \wedge (\exists x'(R(x', y, z)) \wedge \neg T(y, z)) \ . \end{aligned}$$

$$\begin{aligned} R(x, x) &\equiv R(x, y) \wedge x = y \\ R(x, y) \wedge (x \neq y) &\equiv R(x, y) \wedge \neg(R(x, y) \wedge (x = y)) \\ \neg\forall x(R(x, y) \rightarrow T(x, y)) &\equiv \exists x(R(x, y) \wedge \neg T(x, y)) \end{aligned}$$

- $\neg S(x)$ is not equivalent to an SRC formula.
- $S(x) \vee S(\text{'c'})$ is not equivalent to an SRC formula.

Domain Independence

Principle

For database $\mathcal{I}$, we write *adom*($\mathcal{I}$) for the set of constants that occur in $\mathcal{I}$. A first-order formula $\psi(\vec{x})$ is domain independent if for **every** database $\mathcal{I}$, the evaluation of ψ relative to $\mathcal{I}$ does not change if the interpretation domain is changed from *adom*($\mathcal{I}$) to any (possibly infinite) superset of *adom*($\mathcal{I}$).

Theorem 3

Every SRC formula is domain independent.

PROOF. Consequence of the property that every SRC formula has an equivalent SPJRUD expression (Theorem 2). □

It can be shown that every domain independent first-order formula has an equivalent SRC formula.

Safety

Domain independence is a **semantic** notion (it refers to **every** database).

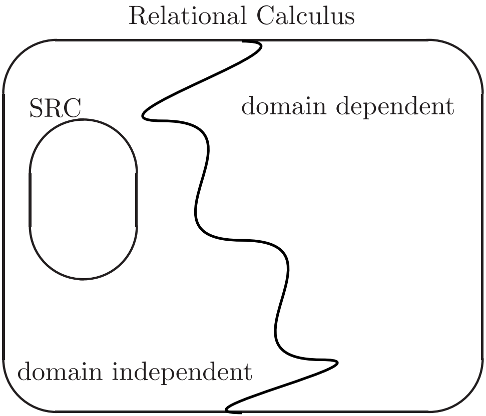
Domain independence is undecidable

There exists no algorithm to decide whether a first-order formula is domain independent.

For this reason, we resort to syntactic restrictions that guarantee domain independence. A relational calculus is called safe if its **syntax** guarantees domain independence.

Overall Picture

We can view a query language as the set of all queries that are syntactically correct.



Historical Note

One of the early results in relational database theory is Codd's completeness theorem [Cod 2](see also [Ull]), which asserts that the relational calculus and relational algebra are equivalent in expressive power, i.e., for any query expressed in the relational calculus there is a semantically equivalent query formulated in the relational algebra, and vice versa (actually, Codd was concerned only with the first part of this theorem).

By this equivalence, the relational algebra can be treated as an "algebraic version" of the predicate calculus.

A fact that seems to have been overlooked by researchers in the relational database theory is that there has been extensive research in mathematical logic concerning the algebraization of the (first-order) predicate calculus. This research led Alfred Tarski to define, about 1952, the notion of a *cylindric algebra*. Cylindric algebras bear the

Source:

Tomasz Imielinski, Witold Lipski Jr.: *The Relational Model of Data and Cylindric Algebras.* **J. Comput. Syst. Sci.** 28(1): 80-102 (1984)

Exercises

- Prove Theorem 1.
- Prove Theorem 2.
- The proposed SRC is expressive (as expressive as SPJRUD) but not very flexible. Think of ways of relaxing the syntactic restrictions of SRC without compromising domain independence. *Hint:* Notice that formula (1) is equivalent to $R(x, y, z) \land \neg S(x, y) \land \neg T(y, z)$.