Tuple Relational Calculus

Jef Wijsen

Université de Mons (UMONS)

October 22, 2024

La base de données

S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

P

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12	London
P2	Bolt	Green	17	Paris
P3	Screw	Blue	17	Rome
P4	Screw	Red	14	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

Motivation

Get all pairs of city names such that a supplier located in the first city supplies a part stored in the second city.

```
SELECT  s.CITY, p.CITY
FROM    S AS s, SP AS r, P AS p
WHERE   s.S# = r.S#
AND     r.P# = p.P# ;
```

$$\{s.4, p.5 \mid S(s) \wedge P(p) \wedge \exists r\,(SP(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\}$$

$$\{s.4, p.5 \mid \exists r\,(S(s) \wedge P(p) \wedge SP(r) \wedge s.1 = r.1 \wedge r.2 = p.1)\}$$

TRC versus DRC (RC = Relational Calculus)

S	1 S#	2 SNAME	3 STATUS	4 CITY
s	$s.1$	$s.2$	$s.3$	$s.4$

P	1 P#	2 PNAME	3 COLOR	4 WEIGHT	5 CITY
p	$p.1$	$p.2$	$p.3$	$p.4$	$p.5$

$s.1 = r.1 \wedge r.2 = p.1$

SP	1 S#	2 P#	3 QTY
r	$r.1$	$r.2$	$r.3$

s, r et p sont des variables qui sont interprétées comme des tuples dans la base de données.
En revanche, dans le SRC (Safe Relational Calculus), les variables (e.g., $x_1, y_1, z_1, s_1, p_1, r_1, x_2, y_2, z_2, s_2, p_2, r_2, \ldots$) sont interpétées comme des valeurs dans le domaine actif:

$$\left\{ s_4, p_5 \mid \exists s_1 \exists s_2 \exists s_3 \exists p_1 \exists p_2 \exists p_3 \exists p_4 \exists r_3 \left(\begin{array}{l} S(s_1, s_2, s_3, s_4) \wedge \\ P(p_1, p_2, p_3, p_4, p_5) \wedge \\ SP(s_1, p_1, r_3) \end{array} \right) \right\}.$$

On parle aussi de TRC (Tuple RC) et DRC (Domain RC).

Motivation

Get supplier names for suppliers who supply all red parts.

```
SELECT   s.SNAME
FROM     S AS s
WHERE    NOT EXISTS (  SELECT   *
                       FROM     P AS p
                       WHERE    p.COLOR = 'Red'
                       AND      NOT EXISTS (  SELECT   *
                                              FROM     SP AS r
                                              WHERE    r.S# = s.S#
                                              AND      r.P# = p.P# ) ) ;
```

$$\{s.2 \mid S(s) \wedge \forall p \in P\big(p.3 = \text{`red'} \rightarrow \exists r \in SP\,(r.1 = s.1 \wedge r.2 = p.1)\big)\}$$

$$\{s.2 \mid S(s) \wedge \neg\exists p \in P\big(p.3 = \text{`red'} \wedge \neg\exists r \in SP\,(r.1 = s.1 \wedge r.2 = p.1)\big)\}$$

Formulas

Alphabet

- Relation names $R, S, T, \ldots$, each of fixed arity in $\{1, 2, \ldots\}$.
- Tuple variables $r, s, t, \ldots$, each of fixed arity.

Terms

- Every constant is a term.
- If r is a tuple variable of arity n and $i \in \{1, 2, \ldots, n\}$, then $r.i$ is a term.

Atomic formulas

- If R is a relation name and r a tuple variable, both of the same arity, then $R(r)$ is an atomic formula.
- If u_1 and u_2 are terms, then $u_1 = u_2$ is an atomic formula.

Formulas

- Every atomic formula is a formula.
- If φ_1 and φ_2 are formulas, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ are formulas.
- If φ is a formula with free tuple variable r, then $\exists r \varphi$ and $\forall r \varphi$ are formulas.

Formulas

Alphabet

- Relation names $R, S, T, \ldots$, each of fixed arity in $\{1, 2, \ldots\}$.
- Tuple variables $r, s, t, \ldots$, each of fixed arity.

Terms

- Every constant is a term.
- If r is a tuple variable of arity n and $i \in \{1, 2, \ldots, n\}$, then $r.i$ is a term.

Atomic formulas

- If R is a relation name and r a tuple variable, both of the same arity, then $R(r)$ is an atomic formula.
- If u_1 and u_2 are terms, then $u_1 = u_2$ is an atomic formula.

Formulas

- Every atomic formula is a formula.
- If φ_1 and φ_2 are formulas, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ are formulas.
- If φ is a formula with free tuple variable r, then $\exists r\varphi$ and $\forall r\varphi$ are formulas.

Formulas

Alphabet

- Relation names $R, S, T, \ldots$, each of fixed arity in $\{1, 2, \ldots\}$.
- Tuple variables $r, s, t, \ldots$, each of fixed arity.

Terms

- Every constant is a term.
- If r is a tuple variable of arity n and $i \in \{1, 2, \ldots, n\}$, then $r.i$ is a term.

Atomic formulas

- If R is a relation name and r a tuple variable, both of the same arity, then $R(r)$ is an atomic formula.
- If u_1 and u_2 are terms, then $u_1 = u_2$ is an atomic formula.

Formulas

- Every atomic formula is a formula.
- If φ_1 and φ_2 are formulas, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ are formulas.
- If φ is a formula with free tuple variable r, then $\exists r \varphi$ and $\forall r \varphi$ are formulas.

Formulas

Alphabet

- Relation names $R, S, T, \dots$, each of fixed arity in $\{1, 2, \dots\}$.
- Tuple variables $r, s, t, \dots$, each of fixed arity.

Terms

- Every constant is a term.
- If r is a tuple variable of arity n and $i \in \{1, 2, \dots, n\}$, then $r.i$ is a term.

Atomic formulas

- If R is a relation name and r a tuple variable, both of the same arity, then $R(r)$ is an atomic formula.
- If u_1 and u_2 are terms, then $u_1 = u_2$ is an atomic formula.

Formulas

- Every atomic formula is a formula.
- If φ_1 and φ_2 are formulas, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$ are formulas.
- If φ is a formula with free tuple variable r, then $\exists r\varphi$ and $\forall r\varphi$ are formulas.

Queries

A query is an expression of the form

$$\{L \mid \varphi\}$$

where

- L is a list of terms;
- φ is a formula;
- whenever $r.i$ is a term in L, then r is a free tuple variable of φ.

Abbreviations

- $\varphi_1 \rightarrow \varphi_2$ is an abbreviation for $\neg\varphi_1 \vee \varphi_2$
- $\exists r \in R\,(\varphi)$ is an abbreviation for $\exists r\,(R(r) \wedge \varphi)$
- $\forall r \in R\,(\varphi)$ is an abbreviation for $\forall r\,(R(r) \rightarrow \varphi)$
- $u_1 \neq u_2$ is an abbreviation for $\neg\,(u_1 = u_2)$

Notice that these abbreviations make sense:

$$
\begin{aligned}
\forall r \in R\,(\varphi) &\equiv \neg\neg\forall r \in R\,(\varphi) \\
&\equiv \neg\neg\forall r\,(R(r) \rightarrow \varphi) \\
&\equiv \neg\exists r\neg\,(\neg R(r) \vee \varphi) \\
&\equiv \neg\exists r\,(R(r) \wedge \neg\varphi) \\
&\equiv \neg\exists r \in R\,(\neg\varphi)
\end{aligned}
$$

Semantics

- A tuple variable of arity n ranges over $\mathbf{dom}^n$.
- $R(r)$ is true if tuple r belongs to relation R.
- $\exists r \varphi$ is true if there exists $r \in \mathbf{dom}^n$ that makes φ true (where n is the arity of r).
- ...
- In tuple relational calculus, we also have the problem of domain dependence.

$$\{r.1 \mid R(r) \vee \exists s(S(s))\}$$

$$\{r.1 \mid \neg R(r)\}$$

- How to express $\{r.1 \mid R(r) \vee S(r)\}$ in SQL?
  SQL is a mix of tuple relational calculus and relational algebra.

Semantics

- A tuple variable of arity n ranges over $\mathbf{dom}^n$.
- $R(r)$ is true if tuple r belongs to relation R.
- $\exists r \varphi$ is true if there exists $r \in \mathbf{dom}^n$ that makes φ true (where n is the arity of r).
- ...
- In tuple relational calculus, we also have the problem of domain dependence.

$$\{r.1 \mid R(r) \vee \exists s \big(S(s)\big)\}$$

$$\{r.1 \mid \neg R(r)\}$$

- How to express $\{r.1 \mid R(r) \vee S(r)\}$ in SQL?
  SQL is a mix of tuple relational calculus and relational algebra.

Exercise

Get pairs (n_1, n_2) of supplier names such that the parts supplied by n_1 is a subset of the parts supplied by n_2.

$$\{s_1.2, s_2.2 \mid S(s_1) \land S(s_2) \land \forall r_1 \in SP \\ \big(r_1.1 = s_1.1 \rightarrow \exists r_2 \in SP\,(r_2.1 = s_2.1 \land r_2.2 = r_1.2)\big)\}$$

$$\{s_1.2, s_2.2 \mid S(s_1) \land S(s_2) \land \neg\exists r_1 \in SP \\ \big(r_1.1 = s_1.1 \land \neg\exists r_2 \in SP\,(r_2.1 = s_2.1 \land r_2.2 = r_1.2)\big)\}$$

```
SELECT  s1.SNAME, s2.SNAME
FROM    S AS s1, S AS s2
WHERE   NOT EXISTS (  SELECT  *
                      FROM    SP AS r1
                      WHERE   r1.S# = s1.S#
                      AND     NOT EXISTS (  SELECT  *
                                            FROM    SP AS r2
                                            WHERE   r2.S# = s2.S#
                                            AND     r2.P# = r1.P# ) ) ;
```