Introduction to Relational Databases

- Bachelor CS, Lille 1 University
- Lecture 7/12
- Topic: Formal query languages
  - from SQL a step back to
  - Tuple-oriented Relational Calculus - TRC

Rendu du TP4

- Prolongation: deux semaines
  - Date: dimanche 23 octobre pour les groupes ayant TP lundi ou mardi
- Sur Moodle

Relational calculus

One of two major formal languages:

TRC, Tuple Relational Calculus

DRC, Domain Relational Calculus

- Different versions of TRC:
  - Here, similar to Ramakrishnan's textbook

TRC is declarative

Express *what* we want in the result, but not *how* to obtain it.

Quite different from relational algebra

Declarativeness is a typical feature of relational languages. It holds for TRC and SQL!

TRC: definition

- Standard form: { t | p(t) }
  p(t) is a formula, built with atoms
- Definition of atoms:
  - $t \in R$
  - expr comp expr
    - comp is comparison operator: =, <>, >, >=, <, <=
    - expr is an expression using constants and t[A]
    - t[A] is a restriction of a tuple t on its attribute(s) A
- Example: $\{t \mid t \in R\}$

TRC: rules to construct a valid formula

- an atom is a formula
- if p is a formula, $\neg p$ (negation) and (p) are also valid formulas
- if p1 and p2 are valid formulas, then
  $p1 \wedge p2$, $p1 \vee p2$, $p1 \Rightarrow p2$ are valid formulas
  (conjunction $\wedge$, disjunction $\vee$, implication $\Rightarrow$)

- if p is a formula, with variable s, then the following are valid formulas:
  - $\exists\, s \in R\, (p(s))$ existential quantification
  - $\forall\, s \in R\, (p(s))$ universal quantification

RA can be expressed with TRC

It is sufficient to show that the fundamental operators of RA can be expressed in TRC. Let R(A,B,C) be a relation.

Selection, $\sigma_{A=1}$ R:

$$\{ t \mid \exists t \in R\ (t[A]=1) \}$$

Projection, Π_{AC} R:

$$\{ t \mid \exists t1 \in R\ (t[A,C]=t1[A,C]) \}$$

RA can be expressed with TRC

Cartesian Product, R(A,B,C) × S(D,E,F):

$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t[A,B,C]=t1[A,B,C] \wedge$
$t[D,E,F]=t2[D,E,F]) \}$

Join (on common attributes A and B),
R(A,C) * S(B,D):

$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t[A,C] = t1[A,C] \wedge$
$t[B,D] = t2[B,D] \wedge$
$t[A] = t[B]) \}$

RA can be expressed with TRC

Union, $R \cup S$:

$\{ t \mid \exists\, t1 \in R, \exists\, t2 \in S$

$(t = t1 \lor t = t2) \}$

Difference, $R - S$:

$\{ t \mid \exists\, t \in R \; (t \notin S) \}$

Other direction:

TRC can be expressed with RA

However, the proof is ways more complex

Need to exclude unsafe and domain-dependent expressions

Under this hypothesis, TRC and AR have the same expressive power

Equivalence laws for TRC

- De Morgan's Laws
  $p1 \wedge p2 \equiv \neg(\neg p1 \vee \neg p2)$

- Correspondence between quantifiers
  $\forall t \in R\,(p(t)) \equiv \neg \exists t \in R\,(\neg p(t))$

- Implication, defined as:
  $p1 \Rightarrow p2 \equiv \neg p1 \vee p2$

Normal forms

- The above three laws imply that it is possible to write all kinds of expressions without implication:
  - Only one quantifier
  - Only one binary operator
- The most known normal form (similar to SQL) uses the existential quantifier and conjunction

TRC: examples

- Names of students with grade A in mathematics

  - $\{ t \mid \exists t1 \in STUDENT,$
    $\exists t2 \in EXAM,$
    $\exists t3 \in CLASS$
    $(\quad t[name]=t1[name] \wedge$
    $t1[sid]=t2[sid] \wedge$
    $t2[cid]=t3[cid] \wedge$
    $t2[grade]=A \wedge$
    $t3[title]=\text{'mathematics'})$
    $\}$

TRC: examples

Ids of the students who have passed the exam for "mathematics" but not for "databases"

$$
\begin{aligned}
\{\, t \mid\ & \exists\, t1 \in EXAM, \exists\, t2 \in CLASS \\
& (t[sid]=t1[sid] \land \\
& \quad t1[cid]=t2[cid] \land \\
& \quad t2[title]=\text{'mathematics'}) \land \\
& \quad \neg\, (\, \exists\, t3 \in EXAM, \exists\, t4 \in CLASS \\
& \qquad (t[sid]=t3[sid] \land \\
& \qquad\quad t3[cid]=t4[cid] \land \\
& \qquad\quad t4[title]=\text{'databases'}))\ \}
\end{aligned}
$$

Correctness

We avoid to use 'unsafe' formulas:

$\{ t \mid t \notin R \}$ returns an infinite result!

- Only formulas that are independent of the domain are correct
  - the solution does not depend of the domain of the attributes, but solely on the database instances

Exercises

1. Lister les noms des fournisseurs.
2. Donner les noms des articles verts.
3. Les noms d'articles rouges, qui n'existent pas en vert.
4. Les noms des fournisseurs d'articles verts.
5. Les couleurs rares.
6. Les noms des articles non fournissables.
7. Le nom de l'article le plus cher.