Introduction to Relational Databases

- Bachelor CS, Lille 1 University
- Oct 14th, 2015 (lecture 7/12)
- Topic: Formal query languages
  - from SQL a step back to
  - Tuple-oriented Relational Calculus - TRC

Précisions pour les notes de CC

Formule habituelle de la L3 (voir sur portail)

Note de TP (compte 25% de la note totale):

- 1 pt par séance sur 8 seances et fait d'avoir rendu. Max 8pt.
- Qualité des rendus, sur 2 ou 3 rendus aléatoires par étudiants: max 6 pt.
- Controle de TP (derniere seance): max 6 pt.

Note de TD ("règle du max" avec DS):

- Présence et participation: max 10 pt
- Deux interros surprise à max 5 points

Rendu du TP4

- Prolongation: deux semaines
  - Date: lundi 26 octobre
- Sur Moodle
- Ré-inscription nécéssaire avec clé pour votre groupe
  - **L3S5_15_bdd_Gi**
  - Où i est votre numéro de groupe

Relational calculus

One of two major formal languages:

TRC, Tuple Relational Calculus

DRC, Domain Relational Calculus

- Two versions of TRC:
  - with range-restricted tuples
  - with arbitrary tuples

We will study TRC with arbitrary tuples

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 ∪ S:
$\{ t \mid \exists t1 \in R, \exists t2 \in S$
$(t = t1 \vee t = t2) \}$
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 | ∃ t1 ∈ STUDENT,
  ∃ t2 ∈ EXAM,
  ∃ t3 ∈ CLASS
  (t[name]=t1[name] ∧
  t1[sid]=t2[sid] ∧
  t2[cid]=t3[cid] ∧
  t2[grade]=A ∧
  t3[title]='mathematics')
  }

TRC: examples

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

{ t | ∃ t1 ∈ EXAM, ∃ t2 ∈ CLASS
(t[sid]=t1[sid] ∧
t1[cid]=t2[cid] ∧
t2[title]='mathematics') ∧
¬ (∃ t3 ∈ EXAM, ∃ t4 ∈ CLASS
(t[sid]=t3[sid] ∧
t3[cid]=t4[cid] ∧
t4[title]='databases'))) }

Correctness

We avoid to use 'unsafe' formulas:
{ t | t ∉ 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