Calcul Relationnel

Réda DEHAK reda@lrde.epita.fr

2012-2013

http://www.lrde.epita.fr/~reda/bdd

Plan

- Introduction
- Calcul relationnel de tuples
- Quel
- Calcul relationnel de domaines
- Query By Example (QBE)
- Conclusion

Introduction

- Deux grandes classes :
  1. Langage algébrique :
     - Algèbre relationnelle.
     - Opérateurs relationnels.
     - Définit les opérations : utile pour la définition des plans d'exécution.
  2. Langage prédicatif :
     - Logique des prédicats.
     - Langage déclaratif : on définit les données recherchées et non la manière de les obtenir.

Langages algébriques

- L’algèbre relationnelle permet de spécifier la suite des opérations à exécuter pour calculer le résultat de la requête :

$$\pi\langle nom \rangle(\ \sigma\langle\ age > 20\ \rangle(CLIENT)\ )$$

SQL est un mélange entre les langages algébriques et les langages prédicatifs.

```
SELECT Client.nom
FROM Client
WHERE Client.age > 20
```

Calcul Relationnel

Un **langage non procédural** qui sert à décrire formellement l'information que l'on veut extraire **sans spécifier une séquence particulière de recherche** :

- Spécification des prédicats qui doivent être vérifiés pas les données pour former le résultat

Basé sur la logique des prédicats du 1er ordre :

- **Alphabet de symboles**
  - Symboles logiques (ex : ¬, ⇒, ∧, ∨, …)
  - Un ensemble de variables
  - Un ensemble de prédicats n-aires
  - Un ensemble de fonctions n-aires
  - Parenthèses
- Expressions (appelées **formules bien formées**) construites à partir de cet alphabet

Logique du premier ordre

La logique du premier ordre (ou calcul des prédicats) permet de manipuler des formules dont la vérité dépend de variables, c-a-d de prédicats :

- Généralise le calcul propositionnel
- Grande puissance d'expression
- Très utilisée en pratique :
  - Systèmes experts
  - Langage PROLOG
  - BD relationnelles, BD déductives

Syntaxe

L’alphabet des symboles suivants :

- Des variables : x, y, z, …
- Des constantes : a, b, c, …
- Des prédicats : P, Q, R, …
- Des connecteurs logiques : $\neg$, $\Rightarrow$, $\vee$, $\wedge$
- Des fonctions n-aires : f, g, h
- Les quantificateurs : $\forall$ et $\exists$

Un terme est une variable, une constante ou le résultat de l’application d’une fonction à un terme

- Exemple : x, a, f(x) et f(f(x)) sont des termes
  - x une personne; f : père de; a : Réda

Syntaxe

Soit T l'ensemble des termes.

Les formules F_{PR} : le plus petit ensemble de formules tel que :

- $\forall t_1, t_2, \ldots, t_n \in T : P(t_1, t_2, \ldots, t_n) \in F_{PR}$
- $\forall A \in F_{PR}, \forall B \in F_{PR}$:

  $\neg A \in F_{PR}, (A \Rightarrow B) \in F_{PR}$,

  $(A \wedge B) \in F_{PR}$, $(A \vee B) \in F_{PR}$
- Si x est une variable et $A \in F_{PR}$ alors :

  $\forall x (A) \in F_{PR}$

  $\exists x (A) \in F_{PR}$

Tables de vérité des connecteurs

La signification des connecteurs logiques est définie par les applications de {V, F} dans {V, F} suivante :

A	B	¬ A	A⇒B	A∨B	A∧B	A⇔B
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	V	V	F	F
F	F	V	V	F	F	V

Equivalence de formules

Pour simplifier et transformer des formules :

- $(A \Rightarrow B) \equiv (\neg A \vee B)$
- $(A \wedge B) \equiv \neg(\neg A \vee \neg B)$
- $(A \Leftrightarrow B) \equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$
- $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$
- $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$
- $(A \vee (B \wedge C)) \equiv (A \vee B) \wedge (A \vee C)$
- $(A \wedge (B \vee C)) \equiv (A \wedge B) \vee (A \wedge C)$
- $\neg(\neg A) \equiv A$
- $\neg(\exists x\ A) \equiv \forall x\ (\neg A)$
- $\neg(\forall x\ A) \equiv \exists x\ (\neg A)$

Interprétation des formules

Table de vérité : ensemble exhaustive de valeurs de l'interprétation d'une formule dans {V, F}

Tautologie (formule valide)

– Vraie pour toute interprétation

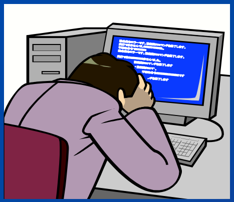
– « A est une tautologie » est notée $\models A$

Contradiction (formule inconsistante)

– Fausse pour toute interprétation

Formule consistante(satisfiable)

– Vraie pour au moins une interprétation



Sémantique

Une formule est interprétée dans {V, F} sur un ensemble d'objets appelé domaine de discours :

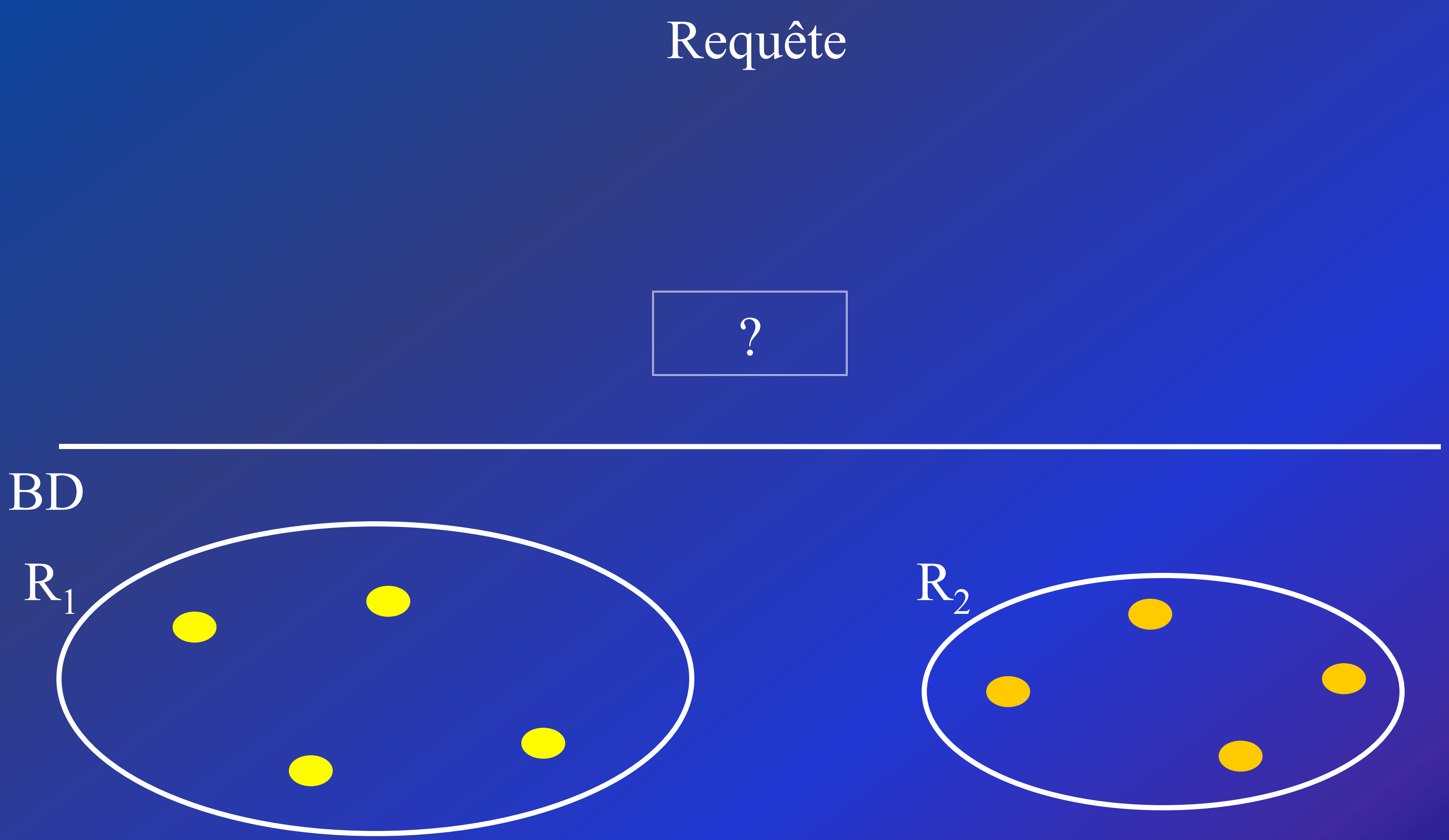
- Constantes : objets particuliers
- Variables : objets quelconques
- Prédicats : relations entre objets
- Fonctions : fonctions particulières entre objets

Exemple :

$\forall x \exists y \ (x * y == y)$

$\forall y \ (x * y == y)$

Modèle général



Types de calcul relationnel

Selon les variables de base utilisées pour spécifier les requêtes

Deux classes :

- **Calcul relationnel de tuples (QUEL).**
- **Calcul relationnel de domaines (QBE).**

Calcul relationnel de tuples

- Variables:
  - x, y, z parcourent les relations
    - les relations sont définies par « ∈ »
    - l'ordre de parcours n'est pas défini
  - les valeurs sont les **tuples** correspondants

s ∈ S

S = (s#, nom, age, ville)

s = ('123', dupont, 30, Paris)

Exemples

PROD(<u>nprod</u>, design, couleur, volume)

Les produits de couleur rouge

$$\{ t \mid t \in Prod \wedge t[couleur] = \text{‘}rouge\text{’} \}$$

Le volume des produits de couleur rouge

$$\{ t[volume] \mid t \in Prod \wedge t[couleur] = \text{‘}rouge\text{’} \}$$

Spécification du format du résultat

Déclaration de variable

Prédicat à satisfaire par le résultat

Calcul relationnel de tuples

- Forme générale :

  $$\{\ t \mid P(t)\ \}$$

  Où :

  t est une variable tuple

  P est un prédicat (formule) construit à partir d'atomes et d'opérateurs.

  ***t* représente la seule variable libre de *P*.**
- *t* peut être qualifiée par des attributs : *t[A]*
- La réponse est constituée de tous les tuples *t* pour lesquels *P* est vrai.

Atomes

Les atomes sont :

1. Variables tuples
   – La variable est qualifiée par le nom de la relation, noté : $t \in R$
2. Conditions
   – $s[A]\ \theta\ t[B]$, où s *et* t sont des variables tuples et A *et* B sont des attributs de s *et* t, respectivement :

   $$\theta = \{ <, >, =, \neq, \leq, \geq \}$$

   Par ex. x[nom] = y[nom]

   – $s[A]\ \theta\ c$

   par ex. *s[prenom] = 'jerome'*

Formules

Une formule est composée de :

- Atomes
- Opérateur booléens $\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$
- Quantificateur existentiels $\exists$
- Quantificateur universels $\forall$

Formation des formules

- Tout atome est une formule
- Si F et G sont des formules, alors :
  $F \vee G$, $F \wedge G$, $\neg F$, $F \Rightarrow G$, $F \Leftrightarrow G$ sont des formules
- Si F est une formule et t une variable libre dans F, alors $\exists t(F)$ et $\forall t(F)$ sont des formules (notées aussi $\exists t\ F$ et $\forall t\ F$ resp.)

Exemples

PROD(<u>nprod</u>, design, couleur, volume)

Les produits de couleur rouge

$$\{ t \mid t \in Prod \wedge t[couleur] = \text{'rouge'} \}$$

Le volume des produits de couleur rouge

$$\{ t[volume] \mid t \in Prod \wedge t[couleur] = \text{'rouge'} \}$$

Exemples

PROD(nprod, design, couleur, volume)

CMD(nclt, nprod, qte, date)

Liste des nprod, couleur commandés par le client n° 13 :

$$\{\ t[nprod], t[couleur]\ |\ t \in Prod \wedge \exists c\ (\ c \in Cmd \wedge t[nprod] = c[nprod] \wedge c[nclt]=13\ )\}$$

Liste des nprod, couleur et qte commandés par le client n° 13 :

$$\{\ t[nprod], t[couleur], c[qte]\ |\ t \in Prod \wedge c \in Cmd \wedge t[nprod] = c[nprod] \wedge c[nclt]=13\}$$

Exemples

Etudiants(id, nom, prenom, année)

- $\exists x\ (x \in Etudiant \wedge x[nom] = "Dehak")$
  Est vrai s'il y a un tuple étudiant avec le nom "Dehak".
  x est une variable liée

- $\forall x (x \notin Etudiant \vee x.année > 1980)$
  $\forall x (x \in Etudiant \Rightarrow x.année > 1980)$
  est vrai si tous les tuples étudiants ont une année > 1980
  x est une variable liée

Sémantique d'une requête

$\{x[a], y[b] \mid x \in R \wedge y \in S \wedge P_z(x, y)\}$

Pour tout tuple x de R faire

  Pour tout tuple y de S faire

    Si $P_z(x, y)$ est vrai alors

      ajouter le tuple <x[a], y[b]> au résultat

    Fsi

  Fin pour

Fin pour

Le test $P_z(x, y)$ implique pour chaque variable liée (ici z) le parcours de la relation correspondante.

Formule saine

- Quantificateurs universel, existentiel où négation peuvent rendre des expressions sans sens (impossible à évaluer)
  - $\{ s \mid \neg (s \in Prod) \}$

    le résultat comprend tous les tuples qui ne sont pas dans Prod

Formule saine

$\{ s \mid \neg (s \in Prod) \}$???

- Soit P une formule formée à partir des relations $R_1, R_2, \ldots, R_n$ et des constantes $a_1, a_2, \ldots a_k$. Le domaine de P, noté $Dom(P)$, est un ensemble fini défini par :

$$Dom(P) = \{a_1, a_2, \ldots, a_k\} \cup \text{ensemble des composants des tuples de } R_1, R_2, \ldots, R_n$$

- Une formule saine vérifie les 3 conditions suivantes :
  - Si s est un n-uplet tel que $P(s)$ est vrai, alors chaque composant de s est un élément de $Dom(P)$.
  - Pour chaque sous-formule P' de P de la forme $\exists u\, P'(u)$, si u est un n-uplet tel que $P'(u)$ est vrai. Alors chaque composant de u est un élément de $Dom(P')$.
  - Pour chaque sous-formule P' de P de la forme $\forall u\, P'(u)$, si un quelconque composant de u n'est pas un élément de $Dom(P')$, alors $P'(u)$ est vrai.

Théorème d'équivalence

Toute proposition formulable en algèbre relationnelle est formulable en calcul de tuple et vice versa.

– Codd, 1978

Même puissance d'expression

Utilisations dans les SGBDs

- ALPHA (Codd, 1978)
  - Jamais implémenté dans un SGBD commercial
- QUEL (Stonebraker, Wong, Rowe, 1979)
  - Le langage initial de INGRES
  - Plus puissant que SQL
- SQL (Salinger & al)
  - System R
  - Éléments de syntaxe algébrique

QUEL (INGRES puis POSTGRES)

$\{ t[nprod], t[couleur], c[qte] \mid t \in Prod \wedge c \in Cmd \wedge t[nprod] = c[nprod] \wedge c[nclt]=13\}$

RANGE OF t IS Prod

RANGE OF c IS Cmd

RETRIEVE [INTO résultat] (t.nprod, t.couleur, c.qte)

WHERE t.nprod = c.nprod AND c.nclt = 13

QUEL (INGRES puis POSTGRES)

- RANGE : définition des variables
- RETRIEVE : définition du résultat
- WHERE : formule bien formée (Prédicat)

Pas de quatificateur

QUEL (INGRES puis POSTGRES)

La formule en QUEL ne contient pas de quantificateurs

⇒ ∃ implicite pour toutes les variables non citées dans la liste après le RETRIEVE

$$\{\ t[nprod], t[couleur] \mid t \in Prod \wedge \exists c\ (\ c \in Cmd \wedge t[nprod] = c[nprod] \wedge c[nclt]=13\ )\}$$

RANGE OF t IS Prod

RANGE OF c IS Cmd

RETRIEVE [INTO résultat] (t.nprod, t.couleur)

WHERE t.nprod = c.nprod AND c.nclt = 13

QUEL (INGRES puis POSTGRES)

$$\{\ t[nprod], t[couleur] \mid t \in Prod \wedge \forall c\ (\ c \in Cmd \Rightarrow t[nprod] \neq c[nprod])\}$$

RANGE OF t IS Prod

RANGE OF c IS Cmd

RETRIEVE [INTO résultat] (t.nprod, t.couleur)

WHERE COUNT(t.nprod WHERE t.nprod= c.nprod) = 0

Exemples

Soit les trois tables suivantes :

CLIENT(nclt, nom, age, adresse)
PROD(nprod, design, couleur, volume)
CMD(nclt, nprod, qte, date)

1. La liste des noms de clients qui ont un age > 20.
2. La liste des noms de clients ayant commandés le produit numéro 13.
3. La liste des noms de clients ayant commandés un produit de couleur rouge.
4. La couleur des produits commandés par monsieur Dupont.
5. La liste des noms de clients ayant commandés au moins un produit.
6. La liste des noms de clients ayant commandés un produit vert ou rouge.
7. La liste des noms de clients ayant commandés un produit vert ou bien rouge.
8. La liste des noms de clients ayant commandés au moins deux produits.
9. La liste des clients qui ont un age > 50 et qui n'ont pas commandé un produit vert.
10. La liste des noms de clients qui ont commandé tous nos produits.
11. La liste des noms de clients qui ont commandé tous nos types de pince.
12. Le Nombre de clients habitant à paris.
13. Le nclt et le nom du ou des clients les plus jeunes de la table client.

Résumé

- Le calcul relationnel sur tuple est un langage non procédural : Exprimer les propriétés des données recherchées et non pas la manière de les obtenir.
- L'Algèbre relationnelle et le calcul relationnel sur tuple (formule saine) offrent les mêmes possibilités.