Bases de données relationnelles

Calculs relationnels

LMD relationnels algèbriques (rappel)

- L'algèbre relationnelle permet de spécifier quelles sont les opérations à exécuter pour calculer le résultat de la requête
  - L'implantation des opérateurs algébriques définit un moteur algèbrique pour l'exécution des requêtes.
  - Ce composant "moteur algèbrique" constitue le noyau des SGBD relationnels
  - SQL est une version orientée utilisateur de l'algèbre relationnelle

LMD prédicatifs

- Un langage prédicatif permet de ne spécifier que le résultat cherché (pas comment le calculer)
  - spécification des prédicats qui doivent être vérifiés par les données pour former le résultat
- Ces langages sont dits prédicatifs car il sont basés sur le calcul de prédicats (logique 1er ordre)
- Un langage prédicatif est donc plus simple qu’une algèbre

LMD relationnels prédicatifs

- Il existe deux types de langages prédicatifs relationnels
- Calcul de tuples : les variables dans les expressions logiques portent sur les tuples des relations (QUEL)

$$x \in \text{Etudiant}$$

- Calcul de domaines : les variables dans les expressions logiques portent sur les valeurs des attributs des tuples (QBE)

$$x \in \text{Etudiant.nom}$$

Calcul de tuples

Exemple de requête

- Etudiant (n°, nom, prénom, année)
- Requête : nom et prénom des étudiants nés après 1980

  π [nom, prénom] σ [année > 1980] Etudiant

déclaration de variable

e $\in$ Etudiant
{ e.nom, e.prénom | e.année > 1980 }

spécification du format du résultat

prédicat à satisfaire par les tuples désignés par e

Requête multi-relation

Etudiant (n°, nom, prénom, année)

Inscription (noét, nomC, note1, note2)

- Requête : no, prénom, notes des étudiants de nom "Rochat" inscrits au cours "BD"

e Î **Etudiant , i** Î **Inscription**

{ e.no, e.prénom, i.note1, i.note2 |

e.nom = "Rochat" Ù **i.nomC = "BD"** Ù **e.no = i.noét }**

Ù : ET

Ú : OU

Ø : NON

Algèbre / calcul

π [nº, prénom, note1, note2] ((σ [nom = "Rochat"] Etudiant)
*[nº = nºét] (σ [nomC ="BD"] Inscription))
ou

π [nº, prénom, note1, note2]
σ [nom = "Rochat" Ù nomC = "BD"]
(Etudiant *[nº = nºét] Inscription)

e Î Etudiant , i Î Inscription
{ e.nº, e.prénom, i.note1, i.note2 |
e.nom = "Rochat" Ù i.nomC = "BD" Ù e.nº = i.nºét }

Même puissance d'expression

Format d'une requête

x_1 Î R_1 , x_2 Î R_2 , ... x_i Î R_i È R_j , .. x_n Î R_n

déclaration des variables (tuples)

sur des relations ou des unions de relations compatibles

{ x_1.A, x_1.B, ... x_i.D / $f_{x_1,x_2,\ldots x_n}$ }

spécification du format du résultat /

spécification du prédicat

$f_{x_1,x_2,\ldots x_n}$ est une formule logique valide ayant pour variables libres exactement x_1 … x_i

(Les autres variables x_{i+1} … x_n doivent être liées)

x_i.D représente la valeur de l'attribut D dans le tuple x_i

Formule valide

- Formule élémentaire :
  - x.A oper-comparaison constante
  - x.A oper-comparaison y.B

  oper-comparaison : =, ? , <, >, =, =
  - x et y sont des variables libres
- Exemples
  - x.nom="Rochat"
  - x.nom = y.nom
- Une formule avec variables libres ne peut pas être évaluée

Formule valide (suite)

- Formule valide :
  - formule élémentaire
  - formule $\wedge$ formule
  - formule $\vee$ formule
  - $\neg$ formule
  - (formule)
  - Si fx est une formule valide où x est une variable libre, alors :

    $\exists$x fx   est une formule valide où x est liée
    "il existe au moins un tuple x tel que fx soit vrai"

    $\forall$x fx   est une formule valide où x est liée
    "pour tous les tuples x, fx est vrai"

Formules valides - Exemples

Etudiant (<u>n°</u>, nom, prénom, année)

x $\in$ Etudiant

- x.nom = "Rochat"   ???
  - x est une variable libre
- $\exists$x (x.nom="Rochat")
  est vrai s'il y a un Rochat dans Etudiant
  - x est une variable liée
- $\forall$x (x.année>1980)
  est vrai si tous les étudiants sont nés après 1980
  - x est une variable liée

Quantificateur "il existe"

- Etudiant (n°, nom, prénom, année)
- Inscription (n°ét, nomC, note1, note2)
- Requête : nom, prénom des étudiants
  ayant réussi brillamment **un** cours

e ∈ Etudiant, i ∈ Inscription

{ e.nom, e.prénom |

$ **i** (e.nº = i.nºét Ù i.note1 = 6 Ù i.note2 = 6) }

Quantificateur "pour tous"

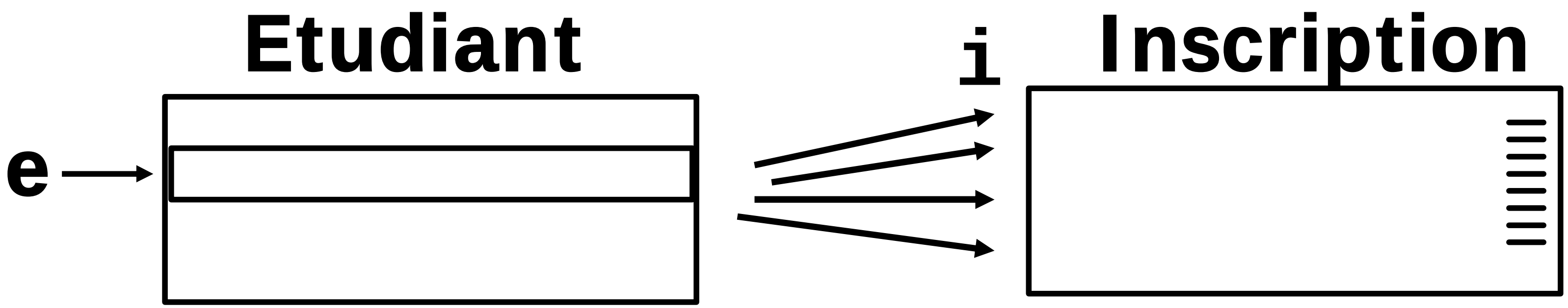
- Requête : nom, prénom des étudiants
  ayant réussi brillamment **tous les** cours

e ∈ Etudiant , i ∈ Inscription

{ e.nom, e.prénom |

" **i** (e.n^{o} **?** i.n^{o}ét Ú (i.note1 = 6 Ù i.note2 = 6))

Ù $ i (e.n^{o} = i.n^{o}ét) }



" x (Ø A Ú B) Û " x (A Þ B)

Logique du 1er ordre - Rappels

- L'opérateur logique => (implique) ne fait pas partie des opérateurs du calcul
  f1 => f2 est équivalent à : Ø f1 Ú f2
- ∀x fx est équivalent à : Ø ∃x (Øfx)
- Quantificateurs sur un ensemble vide
  Soit x∈R
  si R est vide , alors
  - ∀x fx = Vrai
  - ∃x fx = Faux

Rappels (suite)

- Pour mieux comprendre
  - La structure du prédicat (variables libres et liées, occurrences de relations)
  - La sémantique du prédicat

on peut écrire les formules sous forme prenexe (tous les quantificateurs en début de la formule)

- **Formule sous forme prenexe** :

  $Q_1 x_1, Q_2 x_2, \ldots Q_n x_n \ (f\, x_{1, \ldots,} x_n)$

  pas de quantificateur dans $f\, x_{1, \ldots,} x_n$

  et Q : $\forall$ ou $\exists$

- Toute formule peut se mettre sous forme prenexe

Sémantique d'une requête (1)

- $x \in R$, $y \in S$, $z \in T$
  $\{ x.A, y.B \mid f_{x,y,z} \}$
- Faire le produit des relations à variables libres
  R x S
- Sélectionner les tuples du produit qui satisfont la formule $f_{x,y,z}$

Sémantique d'une requête (2)

- $x \in R$, $y \in S$, $z \in T$
  { x.A, y.B | $f_{x,y,z}$ }

- Pour tout tuple, x, de R faire :
    Pour tout tuple, y, de S faire :
      Si $f_{x,y,z}$ est vrai alors ajouter <x.A,y.B> au résultat
    fin pour tout y de S
  fin pour tout x de R

NB Le test de $f_{x,y,z}$ implique pour chaque variable liée (ici z) le parcours de la relation correspondante (ici T)

Equivalence algèbre / calcul

- Toute expression d'algèbre peut s'écrire en calcul
  - chaque opérateur peut se traduire
  - démonstration par récurrence
- Toute requête de calcul peut s'écrire en algèbre

  R (A, B, C)

  S (D, E, F)

  - $x \in R$, $y \in S$
  - { x.A, x.B } p[A, B]R
  - { x.A, y.E } p[A, E] (R**x**S)
  - { x.A | $\exists$y (y.E=x.B)} } p[A] (R*****[B=E]S)
  - . . .

Exemples - Bars à bières

- Bière (bière, degré, couleur, pays, goût)
- Bar (bar, quartier)
- Personne (nom, quartier, age, sexe)
- Sert (bar, bière)
  tel bar sert telle bière
- Abu (buveur, bière, bar, jour, mois, année, qté)
  tel jour, tel buveur a bu telle bière dans tel bar en telle quantité (qté)

Exemples (1)

- Noms des bars fréquentés par Philippe ce mois ci

u ∈ Abu
{ u.bar / u.buveur="Philippe" Ù u.mois=12 Ù
u.année=2003 }

- Noms des bars fréquentés par Philippe ce mois ci avec les bières bues et leur pays

u ∈ Abu ,   i ∈ Bière
{ u.bar, u.bière, i.pays / u.buveur="Philippe" Ù
u.mois=12 Ù u.année=2003 Ù u.bière=i.bière }

Exemples (2)

- Nom, âge et quartier des personnes qui ont fréquenté au moins un bar

p ∈ Personne , u ∈ Abu

{ p.nom, p.age, p.quartier / ∃u (p.nom=u.buveur)}

- Nom, âge et quartier des personnes qui ont fréquenté tous les bars

p ∈ Personne , a ∈ Bar , u ∈ Abu

{ p.nom, p.age, p.quartier / ∀a ∃u (p.nom=u.buveur Ù u.bar=a.bar) }

Exemples (3)

- Nom des personnes qui n'ont jamais fréquenté un bar

p ∈ Personne , u ∈ Abu
{ p.nom / Ø ∃u (p.nom=u.buveur) }

- Nom, âge et quartier des personnes qui ont fréquenté au moins un bar du quartier de la gare

p ∈ Personne , u ∈ Abu , a ∈ Bar
{ p.nom, p.age, p.quartier / ∃a ∃u (a.quartier="gare" Ù u.bar=a.bar Ù p.nom=u.buveur) }

Exemples (4)

- Nom, âge et quartier des personnes qui ont fréquenté tous les bars du quartier de la gare

p ∈ Personne , u ∈ Abu , a ∈ Bar
{ p.nom, p.age, p.quartier / ∀a (a.quartier? "gare" Ú ∃u (u.bar=a.bar Ù p.nom=u.buveur)) }

OU

p ∈ Personne , u ∈ Abu , a ∈ Bar
{ p.nom, p.age, p.quartier / Ø ∃a (a.quartier="gare" Ù Ø ∃u (u.bar=a.bar Ù p.nom=u.buveur)) }

Exemples (5)

- Nom des personnes qui ont bu une brune et une blonde le même jour dans le même bar

u1 ∈ Abu , u2 ∈ Abu , b1 ∈ Bière, b2 ∈ Bière

{ u1.buveur / ∃u2 (u1.bar=u2.bar Ù u1.jour=u2.jour Ù u1.mois=u2.mois Ù u1.année=u2.année Ù u1.buveur=u2.buveur Ù
 ∃b1 (u1.bière=b1.bière Ù b1.couleur="blonde") Ù
 ∃b2 (u2.bière=b2.bière Ù b2.couleur="brune")
) }