LIF4 - Initiation aux Bases de données : Calcul relationnel

E.Coquery

emmanuel.coquery@liris.cnrs.fr

http://liris.cnrs.fr/~ecoquery

Notes

- Semaine du 05 octobre 2009, petit contrôle en TD : 5%
- Semaine du 26 octobre 2009, TP de réseaux avec compte-rendu : 5%
- Semaine du 02 novembre 2009, TP de bases de données noté en séance : 5%
- Mardi 03 novembre 2009, contrôle de réseaux : 15%
- Semaine du 23 novembre 2009, contrôle en TD : 5%
- Semaine du 07 décembre 2009, contrôle en TD : 5%
- Semaine du 11 janvier 2010, soutenances de projet.
  Le projet compte en tout pour 20%
- Semaine du 18 janvier 2010, contrôle terminal : 40%

Langages prédicatifs

- Principe de l'algèbre relationnelle : combiner des relations à travers des opérateurs pour formuler une requête.
- Langages prédicatifs :
  - exprimer une requête par définition du résultat ;
  - en faisant abstraction des mécanismes utilisés par le SGBD.

  Langages déclaratifs inspirés de la logique du premier ordre.

Logique du premier ordre : termes

- Les termes sont composés de :
  - Variables $(x, y, z, \ldots)$
  - Constantes $(a, b, c, \ldots, 1, 2, \ldots, \texttt{'toto'}, \texttt{'titi'}, \ldots)$
  - Symboles de fonctions, avec leur arité $(f/2, g/4, +/2, \ldots)$

  Exemple : $f(g(a, x, a, z), y)$
- Évaluation des termes dans un domaine :
  - domaine : ensemble de valeurs possibles
  - les variables prennent leur valeur dans le domaine
  - les constantes prennent leur valeur dans le domaine
  - les fonctions renvoie un résultat dans le domaine dépendant de leurs arguments (dont la valeur est aussi dans le domaine)

  Exemple : $2 + x \rightarrow 6$ lorsque x vaut 4.

Prédicats

- Fonctions dont le résultat est vrai ou faux.
- Les arguments sont des termes.
- Évaluation :
  - on évalue les arguments $\rightarrow$ valeurs dans le domaine ;
  - le résultat est calculé en fonction de ces valeurs.

Dans le cours, on considère deux sortes de prédicats :
- Des prédicats prédéfinis sur les domaines que l'on manipule :
  - $</2$, $=/2$, $geq/2$, ...
- Des prédicats qui correspondent aux relations définies dans le schéma de la base.
  - L'évaluation du prédicat : dépend de l'instance de la relation
    - vrai si les arguments correspondent à un n-uplet de l'instance
    - faux sinon

Formules

Les prédicats peuvent être combinés via des connecteurs logiques :
- $\wedge$, $\vee$, $\Rightarrow$, $\neg$
- $A \wedge B$ s'évalue à vrai si A s'évalue à *vrai et* B s'évalue à *vrai*
- $A \vee B$ s'évalue à vrai si A s'évalue à *vrai ou* si B s'évalue à *vrai*
- $A \Rightarrow B$ s'évalue à vrai si A s'évalue à *faux ou* si B s'évalue à *vrai*
- $\neg A$ s'évalue à vrai si A s'évalue à faux

Les variables peuvent être introduites par des quantificateurs :
- $\forall x$: "Pour tout x"
  $\forall x A$ s'évalue à *vrai* si pour toutes les valeurs possibles pour x, A s'évalue à *vrai*
- $\exists x$: "Il existe x tel que"
  $\exists x A$ s'évalue à *vrai* si on peut trouver une valeur pour x telle que A s'évalue à *vrai*

Exemples de formule, priorités

- $P(x, y) \wedge Q(x, a)$
- $\neg(P(x, y) \wedge Q(x, a))$
- $\forall x\ \forall y\ (\neg(P(x, y) \wedge Q(x, a)))$
- $P(z, f(a)) \Rightarrow \forall x\ \exists y\ (\neg(P(x, y) \wedge Q(x, a)))$

Priorité : (+ prioritaire) $\{\neg\}, \{\wedge, \vee\}, \{\Rightarrow\}, \{\forall, \exists\}$ (- prioritaire)

Exemple : du français à la formule -1-

Prédicats : $Employe/1$, $NeLe/2$

"Il existe un employé né le 9 janvier 1960"

$\exists x\ Employe(x) \wedge NeLe(x, \texttt{'09/01/1690'})$

Exemple : du français à la formule -2-

Prédicats : $Employe/1$, $NeLe/2$

Si tous les employés sont nés en janvier 1960 alors il y a un employé né le 9 janvier 1960

$$(\forall x \forall y\ (Employe(x) \wedge NeLe(x, y) \Rightarrow y \geq \texttt{'01/01/1960'} \wedge y \leq \texttt{'31/01/1960'})) \Rightarrow \exists z\ Employe(z) \wedge NeLe(z, \texttt{'09/01/1960'})$$

Exemple : du français à la formule -3-

Prédicats : $Employe/1$, $NeLe/2$, $Salaire/2$

Si tous les employés nés après 1960 ont un salaire supérieur à 40000 euros, alors il existe au moins un employé né après 1965 ayant un salaire supérieur à 40 000 euros.

$$\begin{array}{l}(\forall x \forall y \forall z\ (Employe(x) \wedge NeLe(x,y) \wedge \\ \quad Salaire(x,z) \wedge y \geq \text{'01/01/1961'} \quad \Rightarrow \quad z \geq 40000)) \\ \Rightarrow \exists x' \exists y' \exists z'\ Employe(x') \wedge NeLe(x',y') \wedge Salaire(x',z') \wedge \\ \qquad y' \geq \text{'01/01/1966'} \wedge\ z' \geq 40000\end{array}$$

Variables libres et liées

Convention pour ce cours : si une variable est introduite par un quantificateur, elle ne peut apparaître que dans la formule sous ce quantificateur :

- $A \wedge (\forall x\ B) \wedge C$
  x peut apparaître dans B mais pas dans A ni C

Variable libre :

- variable *non* introduite par un quantificateur

Variable liée :

- variable introduite par un quantificateur

Ensembles de valeurs décrits par des formules

- À toute formule, on peut faire correspondre les valeurs pour les variables libres qui rendent cette formule vraie.
- Une formule peut être utilisée pour décrire les valeurs recherchées.
  ⇒ Il est possible d'utiliser une formule pour spécifier une requête dans une base de données.

Le calcul relationnel de domaine s'appuie directement sur cette constatation.

Exemple

Prédicats : $Employe/1$, $NeLe/2$, $Salaire/2$

Trouver l'ensemble des employés ayant un salaire inférieur à 20000 euros et nés avant le 1er janvier 1970 :

$Employe(x) \wedge$
$\exists y NeLe(x,y) \wedge y < \text{'01/01/1970'} \wedge$
$\exists z Salaire(x,z) \wedge z < 20000$

Ici, la variable x est libre

La formule est vraie lorsque la valeur de x est un employé vérifiant les condition du texte ci-dessus.

Attributs, Relations et Prédicats

Jusqu'ici pas d'attributs dans les formules.

Pour les prédicats liés aux relations de la base :

- Relation $R(A_1, \ldots, A_n)$ représentée par le prédicat R/n
- Nouvelle syntaxe : lorsque l'on utilise le prédicat R/n, on précise les arguments : $R(A_1 : e_1, \ldots, A_n : e_n)$
- Comme on précise les attributs, l'ordre n'est plus important.

Requêtes en calcul de domaine

Syntaxe :

$$\{x_1, \ldots, x_n \mid F\}$$

où F est une formule dont les variables libres sont $x_1, \ldots, x_n$ et composée à partir :

- De prédicats des relations de la base appliqués à des variables : $R(A_1 : y_1, \ldots, A_k : y_k)$
- De comparaisons entre variables et variables ou variables et constantes : $x < y$, $x = 12$, ...
- De connecteurs logiques et de quantificateurs : $\wedge, \vee, \Rightarrow, \neg, \forall, \exists$
- Il existe des restrictions sur les formules utilisables
  - Imposent que tout valeur d'une variable soit issue d'une table.

Exemple

On considère le schéma :
$Employe(Nom)$ $\quad NeLe(Nom, DateN)$ $\quad Salaire(Nom, Revenu)$

Trouver les employés nés après 1965 et donner leur salaire

$$\begin{array}{ll}\{x, y \mid & Employe(Nom : x) \wedge \\ & Salaire(Nom : x, Revenu : y) \wedge \\ & \exists z\ NeLe(Nom : x, DateN : z) \wedge z > \text{'31/12/1965'}\}\end{array}$$

En algèbre relationnelle :

$$\pi_{Nom,Salaire}(\sigma_{DateN > \text{'31/12/1965'}}(Employe \bowtie Salaire \bowtie NeLe))$$

Exemple, suite

Employe

Nom
Mohammed
Marc
Lucie

NeLe

Nom	DateN
Lucie	'25/10/1962'
Marc	'18/07/1970'
Mohammed	'15/04/1975'

Salaire

Nom	Revenu
Lucie	30000
Mohammed	20000
Marc	15000

$$\begin{array}{ll}\{x, y \mid & Employe(Nom : x) \wedge \\ & Salaire(Nom : x, Revenu : y) \wedge \\ & \exists z\ NeLe(Nom : x, DateN : z) \wedge z > \text{'31/12/1965'}\}\end{array}$$

Résultat : { (Mohammed, 20000) , (Marc, 15000) }

Calcul relationnel de n-uplet ("tuple")

Autre version du calcul relationnel :

- notion de n-uplet avec attributs
- une variable ↔ un n-uplet
- les relations de la base sont traduites par des prédicats unaires

N-uplets avec attributs

Extension de la notion de n-uplet en ajoutant des nom d'attributs :

- À chaque composant du n-uplet, on associe un attribut différent.
- Notations :
  - $x.A$ est le composant associé à l'attribut A dans le n-uplet x.
  - $x^{(A_1,\dots,A_n)}$ indique que x est un n-uplet dans lequel le premier composant est associé à l'attribut $A_1, \dots$ et le n^{ieme} composant est associé à l'attribut A_n.
  - on s'autorise à ne pas indiquer les attributs d'un n-uplet si on peut les déduire

Exemple : soit $p^{(Nom,Prenom,Age)} = (Lechat, Sylvestre, 8)$.

- $p.Nom = Lechat$
- $p.Prenom = Sylvestre$
- $p.Age = 8$

Relations et prédicats

En calcul relationnel "tuple", pour chaque relation $R(A_1, \dots, A_n)$ de la base :

- La relation R est traduit par un prédicat $R/1$.
- Les arguments de $R/1$ sont des n-uplets de la forme $x^{(A_1,\dots,A_n)}$.
- Étant donnée une instance de R :
  - l'instance est un ensemble de n-uplets $\{t_1, \dots, t_k\}$, *i.e.* chaque t_i est de la forme $(v_1^i, \dots, v_n^i)$;
  - $R(x)$ s'évalue à vrai si la valeur de x est un des t_i.

Requêtes en calcul relationnel "tuple"

Syntaxe :

$$\{x_1.A_1, \dots, x_n.A_n \mid F\}$$

où F est une formule :

- dont les variables libres sont $x_1, \dots, x_n$, avec la possibilité d'avoir plusieurs fois la même variable dans parmi les x_i ;
- composée de la manière suivante :
  - $R(x)$ ou x est une variable représentant un n-uplet ;
  - $x.A \square x'.A'$ ou $x.A \square c$ avec $\square \in \{<, \leq, >, \geq, =\}$ et où c est une constante ;
  - $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \Rightarrow F_2$, $\neg F_1$, $\exists x^{(A_1,\dots,A_n)} F_1$, $\forall x^{(A_1,\dots,A_n)} F_1$ où F_1 et F_2 sont des formules.

Exemple

On considère le schéma :
$Employe(Nom)$ $NeLe(Nom, DateN)$ $Salaire(Nom, Revenu)$

Trouver les employés nés après 1965 et donner leur salaire

$$\{x.Nom, y.Revenu \mid Employe(x) \wedge Salaire(y) \wedge \exists z^{(Nom,DateN)}\ NeLe(z) \wedge x.Nom = y.Nom \wedge x.Nom = z.Nom \wedge z.DateN > \text{'31/12/1965'}\}$$

Exemple, suite

Employe

Nom
Mohammed
Marc
Lucie

NeLe

Nom	DateN
Lucie	'25/10/1962'
Marc	'18/07/1970'
Mohammed	'15/04/1975'

Salaire

Nom	Revenu
Lucie	30000
Mohammed	20000
Marc	15000

$$\{x.Nom, y.Revenu \mid Employe(x) \wedge Salaire(y) \wedge \exists z^{(Nom,DateN)}\ NeLe(z) \wedge x.Nom = y.Nom \wedge x.Nom = z.Nom \wedge z.DateN > \text{'31/12/1965'}\}$$

Résultat : $\{\ (Mohammed, 20000)\ ,\ (Marc, 15000)\ \}$

Du calcul de "tuple" au calcul de domaine

Les deux formes de calcul relationnel sont équivalentes en termes d'expressivité.
Passage du calcul de "tuple" au calcul de domaine :

- Remplacer $\exists x^{(A_1,\dots,A_n)}$ par $\exists x_{A_1} \dots \exists x_{A_n}$ où les x_{A_i} sont des variables du calcul de domaine ;
- Procéder similairement pour les $\forall$;
- Remplacer $R(x^{(A_1,\dots,A_n)})$ par $R(A_1 : x_{A_1}, \dots, A_n : x_{A_n})$;
- Remplacer $x.A$ par x_A ;
- Ajouter $\exists x_A$ au début de la formule pour tous les attributs A des variables x tel que x apparaît dans la partie résultat mais pas $x.A$.

Exemple

Calcul de "tuple" :

$$\{x.Nom, y.Revenu \mid Employe(x) \wedge Salaire(y) \wedge \exists z^{(Nom,DateN)}\ NeLe(z) \wedge x.Nom = y.Nom \wedge x.Nom = z.Nom \wedge z.DateN > \text{'31/12/1965'}\}$$

Traduction en calcul de domaine :

$$\{x_{Nom}, y_{Revenu} \mid \exists y_{Nom}\ Employe(Nom : x_{Nom}) \wedge Salaire(Nom : y_{Nom}, Revenu : y_{Revenu}) \wedge \exists z_{Nom} \exists z_{DateN}\ NeLe(Nom : z_{Nom}, DateN : z_{DateN}) \wedge x_{Nom} = y_{Nom} \wedge x_{Nom} = z_{Nom} \wedge z_{DateN} > \text{'31/12/1965'}\}$$

Du calcul de domaine au calcul de "tuple"

- Dans le résultat de la requête et dans les comparaisons, remplacer x par $x^{(A)}.A$, où A est le premier attribut correspondant à x dans les prédicats.
- Remplacer *existsx* par $\exists x^{(A)}$, où A est le premier attribut correspondant à x dans les prédicats.
- Procéder similairement pour $\forall$.
- Transformer $R(A_1 : x_1, \dots, A_N : x_n)$ en $(\exists x^{(A_1,\dots,A_n)} R(x) \wedge \bigwedge_{i=1}^{n} x.A_i = x_i^{(A_i)}.A_i)$

Exemple

$$\{x, y \mid Employe(Nom : x) \wedge Salaire(Nom : x, Revenu : y) \wedge \exists z\ NeLe(Nom : x, DateN : z) \wedge z > \text{'31/12/1965'}\}$$

$$\{\ x^{(Nom)}.Nom, y^{(Revenu)}.Revenu \mid (\exists u_1^{(Nom)}\ Employe(u_1) \wedge u_1.Nom = x.Nom) \wedge (\exists u_2^{(Nom,Revenu)}\ Salaire(u_2) \wedge u_2.Nom = x.Nom \wedge u_2.Revenu = y.Revenu) \wedge \exists z^{(DateN)} (\exists u_3^{(Nom,DateN)}\ NeLe(u_3) \wedge u_3.Nom = x.Nom \wedge u_3.DateN = z.DateN) \wedge z.DateN > \text{'31/12/1965'}\}$$

Calcul relationnel et algèbre relationnelle

Le calcul relationnel a même puissance d'expression que l'algèbre relationnelle.
Indications pour passer de l'algèbre relationnelle au calcul relationnel "tuple" :

- Une relation se traduit par le prédicat correspondant.
- On peut insérer les conditions des sélections directement dans les formules (en les combinant en général avec $\wedge$).
- La projection correspond à ajouter des $\exists$ devant les variables n-uplets dont aucun attribut n'est projeté.
- L'union se traduit par un $\vee$
- La différence $A - B$ se traduit par $F_A \wedge \neg F_B$ où F_A et F_B ont les mêmes variables libres.

Algèbre relationnelle vers calcul "tuple", suite

- Le produit $A \times B$ se traduit par un $F_A \wedge F_B$ avec les variables libres de F_A et F_B disjointes.
- Le renommage peut, lorsque cela est nécessaire, se traduire via l'introduction d'une nouvelle variable n-uplet combinée avec des égalités.
- La jointure naturelle se traduit par l'ajout d'égalités entre les attributs communs au relations constituant la jointure.

Exemple

$$\pi_{Nom,Salaire}(\sigma_{DateN > \text{'31/12/1965'}}(Employe \bowtie Salaire \bowtie NeLe))$$

$$\{y.Nom, y.Revenu \mid \exists x \exists z\ Employe(x) \wedge Salaire(y) \wedge NeLe(z) \wedge x.Nom = y.Nom \wedge x.Nom = z.Nom \wedge z.DateN > \text{'31/12/1965'}\}$$

Du calcul de domaine vers l'algèbre relationnelle

Guide intuitif pour la traduction :

- Un prédicat se traduit par la relation correspondante.
- Un $\exists$ se traduit en général par une projection.
- Un $\wedge$ se traduit par un produit cartésien ou une intersection
- Un $\vee$ se traduit par une union
- Une $\neg$ se traduit par une différence, mais la traduction n'est général pas directe.
- On peut utiliser le renommage en cas de conflit sur les attributs.
- On peut utiliser une jointure naturelle lorsque les attributs commun à plusieurs relations sont liés par des variables.
- Une comparaison se traduit par une sélection.

Exemple

$$\{x, y \mid Employe(Nom : x) \wedge Salaire(Nom : x, Revenu : y) \wedge \exists z \exists u\ NeLe(Nom : u, DateN : z) \wedge u = x \wedge z > \text{'31/12/1965'}\}$$

$$\pi_{Nom,Revenu}(\sigma_{Nom=Nom'}((Employe \bowtie Salaire) \times \rho_{Nom/Nom'}(\sigma_{DateN > \text{'31/12/1965'}}(NeLe))))$$

Autres notations

En calcul relationnel "tuple", on utilise parfois la position dans les n-uplets au lieu des attributs :

- $t^{(n)}$ au lieu de $t^{(A_1,\ldots,A_n)}$
- $t.n$ au lieu de $t.A_n$

On utilise parfois des [ ] au lieu du .

- $t[A_i]$ ou $t[i]$ au lieu de $t.A_i$