Institut Supérieur d'Informatique et de Gestion de Kairouan

Bases de Données Relationnelles

Cours 5 : L'algèbre relationnelle

Riadh ZAAFRANI
2ème année Licence Fondamentale : Sciences de l'informatique
Novembre 2009

Langages de manipulation

- Langages formels: base théorique solide
- Langages utilisateurs: version plus ergonomique
- Langages procéduraux: définissent comment deriver le résultat souhaité
- Langages assertionnels ou déclaratifs: définissent le résultat souhaité

LMD classiques

- Langages formels:
  - langages algèbriques: définissent un ensemble d'opérateurs de manipulation
  - langages prédicatifs (calcul): définissent le résultat souhaité en utilisant des expressions de logique
- Langages utilisateurs BDR:
  - inspirés des langage algèbriques: SQL
  - inspirés des langage prédicatifs: QBE, QUEL

L'approche algèbrique

- Une algèbre est un ensemble d'opérateurs de base, formellement définis, qui peuvent être combinés à souhait pour construire des expressions algèbriques
- Une algèbre est dite fermée si le résultat de tout opérateur est du même type que les opérandes (ce qui est indispensable pour construire des expressions)
- Complétude: toute manipulation pouvant être souhaitée par les utilisateurs devrait pouvoir être exprimable par une expression algèbrique

L'algèbre relationnelle

- Opérandes: relations du modèle relationnel (1NF)
- Fermeture: le résultat de toute opération est une nouvelle relation
- Complétude: permet toute opération sauf les fermetures transitives
- Opérations unaires (une seule opérande): sélection (noté σ), projection (π), renommage (a)
- Opérations binaires: produit cartésien (´), jointures (⋈), union (È), intersection (Ç), différence (-), division (/)

Préambule

- Pour chacune de ces 9 opérations, on donne :
  - l'opération
  - la syntaxe (notation)
  - la sémantique (résultat attendu)
  - le schéma
  - d'éventuelles remarques
  - un exemple

Sélection σ

- But: ne retenir que certains tuples dans une relation

Pays

nom	capitale	population	surface
Autriche	Vienne	8	83
UK	Londres	56	244
Suisse	Berne	7	41

On ne veut que les pays dont la valeur de surface est inférieure à 100:

Petit-pays = σ [surface < 100] Pays

Petit-pays

nom	capitale	population	surface
Autriche	Vienne	8	83
UK	Londres	56	244
Suisse	Berne	7	41

Sélection σ

- Opération unaire
- Syntaxe: σ **[p] R**

p: prédicat de sélection (condition de sélection)

< prédicat-élémentaire opérateur-logique prédicat-élémentaire >

opérateur-logique ∈ { et, ou }

prédicat-élémentaire :

[non] attribut opérateur-de-comparaison constante-ou-attribut

attribut est un attribut de la relation R

opérateur-de-comparaison ∈ {=, ?, <, >, ≤, ≥}

Sélection σ

- sémantique : crée une nouvelle relation de population l'ensemble des tuples de R qui satisfont le prédicat p
- schéma (résultat) = schéma (opérande)
- population (résultat) ⊆ population (opérande)
- exemple : **Petit-pays = σ [surface < 100] Pays**

Petit-pays

nom	capitale	population	surface
Autriche	Vienne	8	83
Suisse	Berne	7	41

Projection π

- But: ne retenir que certains attributs dans une relation

Pays

nom	capitale	population	surface
Autriche	Vienne	8	83
UK	Londres	56	244
Suisse	Berne	7	41

On ne veut que les attributs nom et capitale:

Capitales = π [nom, capitale] Pays

Capitales

nom	capitale
Autriche	Vienne
UK	Londres
Suisse	Berne

Projection π

- opération unaire
- syntaxe: p **[attributs] R**
  - attributs: liste l'ensemble d'attributs de R à conserver dans le résultat
- sémantique : crée une nouvelle relation de population l'ensemble des tuples de R réduits aux seuls attributs de la liste spécifiée
- schéma (résultat) ⊆ schéma (opérande)
- nb tuples (résultat) = nb tuples (opérande)
- exemple :

Effet de bord de la projection

- Elimination des doubles
  - une projection qui ne conserve pas la clé de la relation peut générer dans le résultat deux tuples identiques (à partir de deux tuples différents de l'opérande)
  - le résultat ne gardera que des tuples différents (fermeture)

R (B, C, D) p (B, C) R



trois tuples deux tuples

Sélection-projection

- On veut les capitales des petits pays:
  - **Petit-pays =** s **[surface < 100] Pays**
  - **Capitales =** p **[nom, capitale] Petit-pays**

Capitale-petit-pays =

p **[nom, capitale]** s **[surface < 100] Pays**

nom	capitale	population	surface
Irlande	Dublin	3	70
Autriche	Vienne	8	83
UK	Londres	56	244
Suisse	Berne	7	41

(Parties grise et beige à enlever)

Renommage a

- but: résoudre des problèmes de compatibilité entre noms d'attributs de deux relations opérandes d'une opération binaire
- opération unaire
- syntaxe : α [nom_attribut : nouveau_nom] R
- sémantique : les tuples de R avec un nouveau nom de l'attribut
- schéma : schéma (α [n, m] R) le même que schéma (R) avec n renommé en m
- précondition : le nouveau nom n'existe pas déjà dans R
- exemple : R2 = α [B: C]R1

R1

A	B
a	b
y	z
b	b

R2

A	C
a	b
y	z
b	b

Produit cartésien ´

- but: construire toutes les combinaisons de tuples de deux relations (en général, en vue d'une sélection)
- syntaxe : R ´ S
- exemple :

R

A	B
a	b
b	c
c	b

S

C	D	E
c	d	e
b	a	b
a	a	c

R ´ S

A	B	C	D	E
a	b	c	d	e
a	b	b	a	b
a	b	a	a	c
b	c	c	d	e
b	c	b	a	b
b	c	a	a	c
c	b	c	d	e
c	b	b	a	b
c	b	a	a	c

n tuples m tuples n x m tuples

Produit cartésien ´

- opération binaire
- sémantique : chaque tuple de R est combiné avec chaque tuple de S
- schéma : schéma ($R \times S$) = schéma(R) $\cup$ schéma(S)
- précondition: R et S n'ont pas d'attributs de même nom (sinon, renommage des attributs avant de faire le produit)

Jointure naturelle ⋈

- but: créer toutes les combinaisons significatives entre tuples de deux relations
  - significatif = portent la même valeur pour les attributs de même nom
- précondition: les deux relations ont au moins un attribut de même nom
- exemple :

R

A	B
a	b
b	c
c	b

S

B	C	D
b	c	d
a	a	b
d	a	c

R ⋈ S

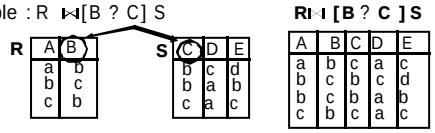
A	B	C	D
a	b	c	d
c	b	a	b

Jointure naturelle ⋈

- opération binaire
- syntaxe : R ⋈ S
- sémantique : combine certains tuples
- schéma : schéma (R ⋈ S) = schéma (R) $\cup$ schéma (S)
  - les attributs de même nom n'apparaissent qu'une seule fois
- la combinaison exige l'égalité des valeurs de tous les attributs de même nom de R et de S
  - si R et S n'ont pas d'attributs de même nom la jointure peut être dynamiquement remplacée par un produit cartesien

Theta-jointure ⋈[p]

- but: créer toutes les combinaisons significatives entre tuples de deux relations
  - ◆ significatif = critère de combinaison explicitement défini en paramètre de l'opération
- précondition: les deux relations n'ont pas d'attribut de même nom
- exemple : R ⋈[B ? C] S

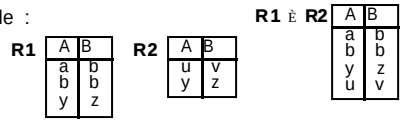


Theta-jointure ⋈[p]

- opération binaire
- syntaxe : R ⋈[p] S
  - ◆ p: prédicat/condition de jointure
  - ◆ < prédicat-élémentaire et/ou prédicat-élémentaire >
- sémantique : combine les tuples qui satisfont le prédicat
- schéma (R ⋈[p] S) = schéma (R) ∪ schéma (S)

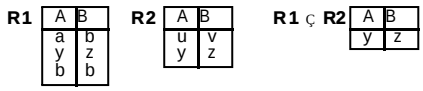
Union È

- opération binaire
- syntaxe : R ∪ S
- sémantique : réunit dans une même relation les tuples de R et ceux de S
- schéma : schéma(R ∪ S) = schéma(R) = schéma(S)
- précondition : schéma(R) = schéma(S)
- exemple :



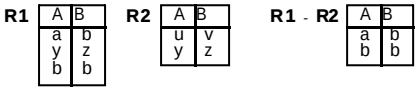
Intersection Ç

- opération binaire
- syntaxe : R ∩ S
- sémantique : sélectionne les tuples qui sont à la fois dans R et S
- schéma : schéma (R ∩ S) = schéma (R) = schéma (S)
- précondition : schéma (R) = schéma (S)
- exemple :



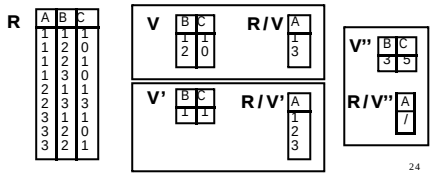
Différence -

- opération binaire
- syntaxe : R – S
- sémantique : sélectionne les tuples de R qui ne sont pas dans S
- schéma : schéma (R – S) = schéma (R) = schéma (S)
- précondition : schéma (R) = schéma (S)
- exemple :



La division /

- but: traiter les requêtes de style «les ... tels que TOUS les ...»
- soient R(A1, ..., An) et V(A1, ..., Am)
  avec n>m et A1, ..., Am des attributs de même nom dans R et V
- R/V = { <am+1, am+2, ..., an> / ∀<a1, a2, ..., am> ∈V,
  ∃<a1, a2, ..., am ,am+1, am+2, ..., an> ∈R}
- exemples :



Exemple

- R

ETUDIANT	COURS	REUSSI
Karim	BD	oui
Karim	Prog	oui
Mohamed	BD	oui
Mohamed	Math	oui
Youssef	Prog	oui
Youssef	BD	non

V

COURS	REUSSI
Prog	oui
BD	oui

R/V

ETUDIANT
Karim

Exemples de requêtes algébriques

- Soient les relations suivantes :

Journal (code-j, titre, prix, type, périodicité)

Dépôt (no-dépôt, nom-dépôt, adresse)

Livraison (no-dépôt, code-j, date-liv, quantité-livrée)

Répondre aux requêtes suivantes :

- Quel sont les prix des journaux ?

  π [prix] Journal

- Donnez tous les renseignements connus sur les journaux hebdomadaires.

  σ [périodicité = "hebdomadaire"] Journal

- Donnez les codes des journaux livrés à Kairouan.

  π [code-j] (σ [adresse = "Kairouan"] Dépôt $\bowtie$ livraison)

Répondre aux requêtes suivantes :

- Donnez les numéros des dépôts qui reçoivent plusieurs journaux (>1).

π [no-dépôt]
(σ [code-j $\neq$ code']
(p **[no-dépôt, code']** a **[code-j, code'] Livraison**) $\bowtie$
π [no-dépôt, code-j] Livraison)

no-dépôt	code-j	date-liv	quantité-livrée
1	Assabah	13/11/2008	1700
1	Letemps	13/11/2008	1400
2	LaPresse	13/11/2008	900
2	LeQuotidien	13/11/2008	1300
3	TunisHebdo	11/11/2008	2000
4	Assarih	13/11/2008	1000

p **[no-dépôt, code']** a
[code-j, code'] Livraison

no-dépôt	Code'
1	Assabah
1	Letemps
2	LaPresse
2	LeQuotidien
3	TunisHebdo
4	Assarih

π [no-dépôt, code-j] Livraison

no-dépôt	code-j
1	Assabah
1	Letemps
2	LaPresse
2	LeQuotidien
3	TunisHebdo
4	Assarih

(p **[no-dépôt, code']** a **[code-j, code'] Livraison**) $\bowtie$ π [no-dépôt, code-j] Livraison)

Les opérations de calcul

- En plus des 9 opérations décrites précédemment, certains auteurs ajoutent des opérations de calcul sur les relations.
- Cet ajout est justifié par le fait que de nombreuses requêtes ont besoin de ce type d'opérations qui d'ailleurs sont implémentées dans tous les langages d'interrogation.
- Ces opérateurs de calcul forment donc une extension aux opérateurs de base et ne peuvent pas être exprimés à l'aide de ceux-ci.

Compte

- **Compte** est une opération courante qui permet de dénombrer les lignes d'une relation qui ont une même valeur d'attributs en commun. La relation résultante ne contient que les attributs de regroupement X_i choisis avec leurs occurrences dans la relation.
- On notera **T = Compte $_{X1,\ldots,Xn}$ (R)** : les $X_1,\ldots,X_n$ étant les attributs de regroupement.
- Si aucun attribut de regroupement n'est précisé, l'opération renvoie alors uniquement le nombre de tuples de la relation.

A Vous :

- **Donner les résultats des opérations Compte** :
- **T1 = Compte $_B$(R)** et T2 = **Compte (R)**

R	A	B	C
	a	n	17
	b	o	14
	c	n	9
	d	n	13
	e	m	20
	f	m	10

Compte $_B$(R)	B	Compte
	n	2
	m	2
	o	1
	p	1

Compte (R)	Compte
	6

Somme

- **Somme** est une opération qui permet de faire la somme cumulée des valeurs d'un attribut Y pour chacune des valeurs différentes des attributs de regroupement $X_1,\ldots,X_n$. Y doit bien sûr être numérique. La relation résultante ne contient que les différentes valeurs des attributs Xi de regroupement choisis ainsi que la somme cumulée des Y correspondants.
- On notera T = **Somme $_{X1,\ldots,Xn}$ (R,Y)**
- Si aucun attribut de regroupement n'est précisé, l'opération renvoie alors la somme de toutes les valeurs de la colonne Y.

A Vous :

- **Donner les résultats des opérations Somme** :
- **T1 = Somme $_B$(R,C)** et T2 = **Somme (R,C)**

R	A	B	C
	a	n	17
	b	o	14
	c	n	9
	d	n	13
	e	m	20
	f	m	10

Somme $_B$(R,C)	B	Somme
	n	26
	m	30
	o	14
	p	13

Somme (R,C)	Somme
	83

Cette opération se généralise facilement à d'autres opérations comme le calcul de la moyenne, du minimum ou du maximum d'une colonne.

Pourquoi une requête est-elle meilleure qu'une autre ?

- Une requête n'est en général pas l'unique solution d'un problème comme le montre la question précédente.
- Dans ce cas, si la relation obtenue est identique avec les deux requêtes, l'efficacité est rarement la même.
- D'où l'utilité d'algorithmes d'optimisation automatique de requêtes que l'on trouve dans les SGBD dignes de ce nom.

Pourquoi une requête est-elle meilleure qu'une autre ?

- **Nous utiliserons comme exemple les trois tables représentant les commandes de produits à des fournisseurs :**
- **produits(<u>numProd</u>, design, prix, poids, couleur)**
- **fournisseurs(<u>numFrs</u>, nom, ville)**
- **commandes(numCde, <u>numFrs</u>, <u>numProd</u>, qté)**

A Vous

- **Déterminer les références, prix et quantités des produits commandés en plus de 10 exemplaires par commande ?**
- Cette requête peut s'écrire de plusieurs manières différentes plus ou moins efficaces.
- **(A):** $\pi_{numProd,prix,qté}(\sigma_{qté>10}(commandes \bowtie produits))$
- **(B):** $\pi_{numProd,prix,qté}(produits \bowtie \sigma_{qté>10}(commandes)))$
- **(C):** $\pi_{numProd,prix}(produits) \bowtie \pi_{numProd,qté}(\sigma_{qté>10}(commandes)))$

Pourquoi une requête est-elle meilleure qu'une autre ?

- Prenons pour hypothèse simplificatrice que chaque colonne nécessite **10** caractères en mémoire.
- La table produits nécessite donc 8 lignes * 5 colonnes * 10 caractères = **400** caractères, tandis que la table commandes nécessite 10 * 4 * 10 = **400** caractères.
- Intéressons nous à la **requête (A).** Elle nécessite une opération de Jointure qui est en général coûteuse. En effet, la table temporaire la plus volumineuse pour effectuer cette requête est engendrée par le produits cartésien des deux tables produits et commandes nécessaire au calcul de la jointure. Cette table temporaire possède un volume de 400 * 400 = **160000** caractères.

Pourquoi une requête est-elle meilleure qu'une autre ?

- Si l'on effectue les projections et restrictions le plutôt possible comme dans la requête (C), on ne conserve alors que le couple (numProd,prix) de la table produits ce qui donne 8 * 2 * 10 = **160** caractères et le couple (numProd,qté) de la table commandes pour les quantités supérieures à 10 ce qui donne 2 * 2 * 10 = **40** caractères. La jointure engendre une table temporaire de 40 * 160 = **6400** caractères, d'où un gain de 75% (facteur 25) en taille mémoire nécessaire (et un gain en vitesse évidemment proportionnel).
- Les tables utilisées dans cet exemple sont très petites et la requête encore très simple, mais on imagine facilement œ que ce genre d'optimisation permet de gagner sur des requêtes à 3 ou 4 jointures sur des tables de milliers d'enregistrements !

Equivalences

- L'algèbre relationnelle permet l'étude des opérateurs entre eux (commutativité, associativité, groupes d'opérateurs minimaux etc ...).
- Cette étude permet de démontrer l'équivalence de certaines expressions et de construire des programmes d'optimisation qui transformeront toute demande en sa forme équivalente la plus efficace.

Equivalences

- $R \cap S = S \cap R$, $R \bowtie S = S \bowtie R$, etc.
- $\sigma[p1](\sigma[p2]R) = \sigma[p2](\sigma[p1]R) = \sigma[p2 \text{ et } p1]R$
- $\sigma[p](\pi[a]R) = \pi[a](\sigma[p]R)$ si $attributs(p) \subseteq a$
- etc.

Conclusion

- D'une manière pratique, les opérations les plus utilisées sont la projection, la restriction et la jointure naturelle.
- L'opération de jointure est en général très coûteuse. Elle est d'ailleurs proportionnelle au nombre de tuples du résultat et peut atteindre m * n tuples avec m et n les nombres de tuples des deux relations jointes.
- Afin d'avoir des requêtes efficaces en temps il est toujours préférable de faire les restrictions et les projections le plutôt possible afin de manipuler des tables les plus réduites possibles.