

Chapitre 5

LANGAGES DE MANIPULATION RELATIONNELS

1. Pourquoi des langages de manipulation de données relationnelles ?

La structure des relations étant semblable à celle des tableaux en mémoire centrale, on peut se demander pourquoi avoir inventé des langages spéciaux d'interrogation et de mise à jour pour les relations. Ne pouvait-on pas employer n'importe quel langage de programmation? Cela aurait l'avantage de ne pas obliger les utilisateurs à apprendre un autre langage spécifique des bases de données. L'inconvénient d'une telle solution est que les programmes des utilisateurs doivent connaître l'organisation des tuples dans les relations pour pouvoir accéder plus rapidement aux informations dont ils ont besoin. Par exemple, si la relation Etudiant est triée selon les noms des étudiants, un programme efficace cherchant des informations sur l'étudiante Zoé, partira de la fin de la relation. L'indépendance - recherchée dans les SGBD - des programmes et des données n'est plus réalisée. Les utilisateurs doivent connaître l'organisation des données (existence de tri, d'index,...), et il est impossible de changer cette organisation (à des fins d'optimisation par exemple) sans affecter les programmes des utilisateurs.

Les langages de manipulation de données (LMD) doivent donc être indépendants de l'organisation interne des relations, c'est-à-dire ne porter que sur les concepts du modèle relationnel (relations, attributs, domaines). Les LMD permettent ainsi aux utilisateurs de définir ce qu'ils veulent chercher dans la base de données, mais pas comment y arriver. C'est le module d'optimisation des requêtes du SGBD qui va étudier quelles sont les différentes possibilités pour faire cette recherche, et qui va choisir la plus rapide.

Par ailleurs, les LMD doivent être efficaces, c'est-à-dire avoir des temps de réponse courts même si la base de données est très grande. C'est pourquoi, les LMD offrent un éventail de fonctions limité à celles qu'on sait optimiser de façon efficace, mais assez vaste pour permettre d'exprimer la plupart des requêtes.

2. LMD algébriques et LMD prédicatifs

Le modèle relationnel a été à l'origine proposé avec deux LMD de base, l'algèbre relationnelle et le calcul des tuples, équivalents en puissance et qui ont fixé l'ensemble des fonctions que tout LMD relationnel doit offrir. A partir de ces deux langages, d'autres LMD ont pu être définis, qui sont plus conviviaux pour les utilisateurs, et qui ont au moins la même puissance d'expression que l'algèbre ou le calcul.

En plus des fonctions de l'algèbre ou du calcul, ces LMD offrent généralement des possibilités de mise à jour de la base de données, et d'emploi, dans les requêtes, d'expressions arithmétiques et de fonctions d'agrégation telles que cardinalité, somme, minimum, maximum et moyenne.

On dit d'un LMD qu'il est complet s'il offre au moins les mêmes fonctions que l'algèbre ou le calcul relationnels.

L'intérêt de l'algèbre relationnelle est multiple:

- l'algèbre a identifié les opérateurs fondamentaux d'utilisation d'une base de données relationnelle;
- ces opérateurs ont défini les principales fonctions à optimiser dans les SGBD relationnels;
- l'algèbre a donné naissance à des LMD pour les utilisateurs. C'est le cas de ISBL, qui a été développé par IBM en habillant l'algèbre d'une syntaxe plus agréable.

Un autre type de LMD est constitué des langages issus du calcul des prédicats de la logique du premier ordre. Deux adaptations de ce calcul au modèle relationnel ont été proposées, qui toutes deux ont conduit à des langages utilisateurs:

- le calcul des tuples, qui a donné naissance au LMD QUEL du SGBD relationnel Ingres,
- le calcul des domaines qui a donné naissance à des LMD de type graphique dont QBE, proposé par IBM.

Les langages de type prédictif sont actuellement très prisés par les utilisateurs. Une requête exprimée dans un tel langage, spécifie par des prédicats uniquement les caractéristiques du résultat qu'elle veut obtenir (c'est-à-dire la définition du résultat désiré). Tandis que la même requête exprimée dans un langage algébrique, spécifie un enchaînement d'opérations conduisant au résultat désiré (c'est-à-dire comment construire le résultat à partir des relations de la base de données).

SQL qui est le LMD relationnel le plus répandu du fait que c'est la seule norme existante pour les LMD relationnels, comporte des caractéristiques de type algébrique et d'autres de type prédictif.

Dans ce cours, nous étudierons successivement ces différents types de langages.

L'ALGÈBRE RELATIONNELLE

1 Introduction

L'algèbre relationnelle est un ensemble d'opérateurs qui, à partir d'une ou deux relations existantes, créent en résultat une nouvelle relation temporaire (c'est-à-dire qui a une durée de vie limitée, généralement la relation est détruite à la fin du programme utilisateur ou de la transaction qui l'a créée). La relation résultat a exactement les mêmes caractéristiques qu'une relation de la base de données et peut donc être manipulée de nouveau par les opérateurs de l'algèbre.

Formellement l'algèbre comprend:

- cinq opérateurs de base: sélection, projection, union, différence et produit,
- un opérateur syntaxique, renommer, qui ne fait que modifier le schéma et pas les tuples.

A partir de ces opérateurs, d'autres opérateurs ont été proposés qui sont équivalents à la composition de plusieurs opérateurs de base. Ces nouveaux opérateurs, appelés opérateurs déduits, sont des raccourcis d'écriture, qui n'apportent aucune fonctionnalité nouvelle, mais qui sont pratiques pour l'utilisateur lors de l'écriture des requêtes. Nous présentons dans ce chapitre, outre les opérateurs de base et renommer, les opérateurs déduits les plus fréquents: intersection, jointure naturelle, thêta jointure et division.

Les opérateurs de l'algèbre peuvent être regroupés en deux classes:

- les opérateurs provenant de la théorie mathématique sur les ensembles (applicables car chaque relation est définie comme un ensemble de tuples): union, intersection, différence, produit;
- les opérateurs définis spécialement pour les bases de données relationnelles: sélection, projection, jointure, division et renommage.

2 Les opérateurs

2.1. Projection

Cet opérateur construit une relation résultat où n'apparaissent que certains attributs de la relation opérande (en termes de tableau, cela revient à extraire certaines colonnes).

Définition : soit $R (A_1, A_2, \dots, A_n)$ une relation, et soit $A_{i1}, A_{i2}, \dots, A_{ij}$ un sous-ensemble de ses attributs, la projection de R sur $A_{i1}, A_{i2}, \dots, A_{ij}$, notée :

$$\pi [A_{i1}, A_{i2}, \dots, A_{ij}] R$$

créé une nouvelle relation, temporaire, de schéma $(A_{i1}, A_{i2}, \dots, A_{ij})$ et de population égale à l'ensemble des tuples de R "tronqués à $A_{i1}, A_{i2}, \dots, A_{ij}$ ", i.e. :

$$\{t / \exists r (r \in R \wedge t.A_{i1} = r.A_{i1} \wedge t.A_{i2} = r.A_{i2} \wedge \dots \wedge t.A_{ij} = r.A_{ij})\}$$

Remarque: le résultat est un ensemble de tuples sans double, c'est-à-dire que si la projection crée des tuples en double - cas d'une projection éliminant tous les identifiants de R - ces doubles sont supprimés automatiquement.

Exemple : soit la relation

| Personne | nom | prénom | jour-nais | mois-nais | an-nais | sexe |
|----------|--------|---------|-----------|-----------|---------|------|
| | Dupont | Jean | 30 | 07 | 72 | M |
| | Talon | Achille | 20 | 11 | 75 | M |
| | Rochat | Marie | 13 | 05 | 72 | F |

| | | | | | |
|--------|-------|----|----|----|---|
| Martin | Régis | 27 | 03 | 74 | M |
| Picard | Anne | 10 | 10 | 76 | F |
| Martin | Jules | 05 | 03 | 74 | M |

On construit l'ensemble des noms et prénoms des personnes avec l'opération:

NP := p [nom, prénom] Personne

On obtient : NP

| nom | prénom |
|--------|---------|
| Dupont | Jean |
| Talon | Achille |
| Rochat | Marie |
| Martin | Régis |
| Picard | Anne |
| Martin | Jules |

L'opération : NA := p [nom, an-nais] Personne

donnera en résultat : NA

| nom | an-nais |
|--------|---------|
| Dupont | 72 |
| Talon | 75 |
| Rochat | 72 |
| Martin | 74 |
| Picard | 76 |

soit une population de 5 tuples au lieu des 6 tuples de la relation Personne.

2.2. Sélection

Cet opérateur construit une relation résultat où n'apparaissent que certains tuples de la relation opérande (en termes de tableau, cela revient à extraire certaines lignes). Les tuples retenus sont ceux satisfaisant une condition explicite, appelée prédicat de sélection.

Définition: soit R (A₁, A₂, ..., A_n) une relation, la sélection de R selon un prédicat p, notée:

s [p] R

crée une nouvelle relation, temporaire, de schéma identique à celui de R, et de population l'ensemble des tuples de R pour lesquels le prédicat p est vrai.

Exemple: pour créer une relation Femmes contenant l'ensemble des personnes de sexe féminin, on écrira:

Femmes := s [sexe = 'F'] Personne

ce qui donne en résultat :

| Femmes | nom | prénom | jour-nais | mois-nais | an-nais | sexe |
|--------|--------|--------|-----------|-----------|---------|------|
| | Rochat | Marie | 13 | 05 | 72 | F |
| | Picard | Anne | 10 | 10 | 76 | F |

Le prédicat de sélection permet de comparer la valeur d'attributs de R à celle d'autres attributs de R ou à des constantes. Sa forme est la suivante ¹:

<p> ::= <condition> | <p> <opérateur logique> <p> | ¬ <p> | "(" <p> ")"

<opérateur logique> ::= ∧ | ∨

<condition> ::= nom-attribut <opérateur comparaison> valeur |
nom-attribut <opérateur comparaison> nom-attribut

<opérateur comparaison> ::= = | ? | = | < | > | =

Remarquons que les opérateurs de comparaison = , < , > et = ne peuvent être appliqués qu'aux attributs dont les domaines contiennent des valeurs ordonnées (valeurs numériques, dates, chaînes de caractères). Les domaines de chaînes de caractères alphabétiques sont triés alphabétiquement,

¹ ∧ représente le connecteur logique "et"

∨ représente le connecteur logique "ou"

¬ représente l'opérateur logique de négation "non"

tandis que les domaines de chaînes de caractères alphanumériques sont triés selon les codes numériques des caractères. Si le domaine d'un attribut est un ensemble de valeurs non triées, les seuls opérateurs de comparaison utilisables sont = et ? .

2.3 Expressions d'algèbre

Les opérateurs de l'algèbre peuvent être combinés dans des expressions pour exprimer des requêtes non élémentaires.

Exemple : on obtient la liste des noms et prénoms des hommes nés avant 1975 par l'expression :

$H := p [\text{nom}, \text{prénom}] \text{ } S [\text{sexe} = 'M' \wedge \text{an-nais} < 75] \text{ Personne}$

| H | nom | prénom |
|---|--------|--------|
| | Dupont | Jean |
| | Martin | Régis |
| | Martin | Jules |

Le même résultat pourrait être obtenu en écrivant deux opérations l'une après l'autre en créant une relation intermédiaire (dans ce cas il faut nommer les relations intermédiaires):

$H1 := S [\text{sexe} = "M" \wedge \text{an-nais} < 75] \text{ Personne}$

$H := p [\text{nom}, \text{prénom}] H1$

2.4 Jointure (naturelle) de deux relations ayant au moins un attribut commun.

Définition : étant donné deux relations $R(X, Y)$ et $S(Y, Z)$, où X, Y, Z symbolisent soit un attribut, soit un ensemble d'attributs, et où Y n'est pas vide, la jointure (naturelle) de R et S , notée :

$R * S$

crée une nouvelle relation temporaire, de schéma (X, Y, Z) . La population de $R * S$ est l'ensemble des tuples $\langle x, y, z \rangle$ créés par composition d'un tuple $\langle x, y \rangle$ de R et d'un tuple $\langle y, z \rangle$ de S , tels que les deux tuples ont la même valeur pour Y .

On remarque que la population de $R * S$ comporte n tuples, $n \in [0 : \text{card}(R) \times \text{card}(S)]$, les valeurs extrêmes étant obtenues dans les cas suivants:

- 0 : il n'existe pas de tuple de R et S qui ont même valeur pour Y ,
- $\text{card}(R) \times \text{card}(S)$: les tuples de R et de S ont tous la même valeur, y_0 , pour Y .

Pour les exemples de requêtes d'algèbre, nous utiliserons par la suite la base de données relationnelle FormaPerm (du chapitre 4).

Exemples: on désire tous les renseignements sur les étudiants (nom, adresse, date de naissance, numéros d'étudiant et de personne)

$\text{Etudiant} * \text{Personne}$

Noms des étudiants ayant réussi le cours d'algorithmique:

$p [\text{nom}] (\text{Personne} * \text{Etudiant} * S [\text{nomC} = "algo"] \text{ Obtenu})$

On obtient en résultat , d'après la base de données fournie en exemple: Walter, Bernard.

2.5 Renommer un ou des attributs d'une relation

L'opérateur renommer, noté α , permet de changer le nom d'un (ou plusieurs) attribut d'une relation R :

$\alpha [\text{nom_attr1: nouveau_nom_pour_attr1}, \dots] R$

Cet opérateur est utile avant les jointures s'il y a un problème d'homonymie ou de synonymie, ou avant les opérations ensemblistes (union, différence, intersection) qui requièrent que les attributs correspondants aient le même nom.

2.6 La thêta-jointure de deux relations n'ayant aucun attribut commun

Définition : soient deux relations $R (A_1, A_2, \dots, A_n)$ et $T (B_1, B_2, \dots, B_p)$ n'ayant pas d'attribut de même nom, la thêta jointure de R et T selon le prédicat p , notée :

$R * [p] T$

crée une nouvelle relation temporaire de schéma $(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_p)$, et de population égale à l'ensemble des tuples de R et de T concaténés qui satisfont le prédicat.

Le prédicat est de la même forme que le prédicat d'une sélection, sauf pour les conditions élémentaires qui comparent un attribut de R à un attribut de T :

$\langle \text{condition} \rangle ::= \text{nom-attribut-de-R} \langle \text{opérateur de comparaison} \rangle \text{nom-attribut-de-T}$

Exemple: liste des couples de numéros d'étudiants, tels que ces deux étudiants soient nés le même jour

/ on crée d'abord une autre relation Etudiant avec des attributs renommés/

$\text{Etudiant2} := a [n^{\circ}E : n^{\circ}E2, \text{dateN} : \text{dateN2}] p [n^{\circ}E, \text{dateN}] \text{Etudiant}$

$p [n^{\circ}E, n^{\circ}E2] (\text{Etudiant} * [n^{\circ}E < n^{\circ}E2 \wedge \text{dateN} = \text{dateN2}] \text{Etudiant2})$

On obtient en résultat, d'après la base de données fournie en exemple, une relation vide.

2.7 Union, Différence, Intersection de deux relations de même schéma

Définition: soient R et S deux relations de même schéma : $R (A_1, A_2, \dots, A_n)$, $S (A_1, A_2, \dots, A_n)$.

Union : $R \dot{\cup} S$ crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples de R et de ceux de S (avec élimination des doubles éventuellement créés).

Différence : $R - S$ crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples de R moins ceux de S , c'est à dire : les tuples qui se trouvent dans R mais pas dans S .

Intersection : $R \dot{\cap} S$ crée une relation temporaire de même schéma et de population égale à l'ensemble des tuples de R qui ont un tuple de même valeur dans S .

Exemples:

liste des numéros des personnes qui sont soit enseignant de BD soit étudiant en BD:

/* on crée deux relations temporaires: */

$\text{EnsBD} := a [n^{\circ}\text{Ens} : n^{\circ}P] p [n^{\circ}\text{Ens}] s [\text{nomC} = \text{"BD"}] \text{Cours}$

On obtient en résultat d'après la base de données fournie en exemple: 2222.

$\text{EtudBD} := p [n^{\circ}P] \text{Etudiant} * s [\text{nomC} = \text{"BD"}] \text{Inscrit}$

On obtient en résultat d'après la base de données fournie en exemple: 5555.

/* Le résultat recherché est: */

$\text{EnsBD} \dot{\cup} \text{EtudBD}$

On obtient en résultat: 2222, 5555.

liste des numéros des personnes qui n'ont rien à voir avec le cours de BD:

$p [n^{\circ}P] \text{Personne} - (\text{EnsBD} \dot{\cup} \text{EtudBD})$

On obtient en résultat: 1111, 6666, 3333.

liste des numéros des personnes qui sont enseignants et étudiants simultanément (assistants-doctorants,...) :

$(p [n^{\circ}P] \text{Enseignant}) \dot{\cap} (p [n^{\circ}P] \text{Etudiant})$

On obtient un résultat vide.

2.8 Produit cartésien de deux relations n'ayant aucun attribut commun

Définition : Soient deux relations, $R (A_1, A_2, \dots, A_n)$ et $T (B_1, B_2, \dots, B_p)$, n'ayant pas d'attribut de même nom, alors le produit de R par T , noté $R \times T$, crée une relation temporaire de schéma $(A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_p)$ et de population toutes les concaténations possibles de tuples de R et de T .

Remarque: La différence entre les opérateurs de thêta-jointure et de produit cartésien consiste dans le fait que dans la thêta-jointure, seules les combinaisons des tuples qui satisfont le prédicat de jointure apparaissent dans le résultat, tandis que dans le produit toutes les combinaisons de tuples sont présentes.

Exemple: existe-t-il des personnes dont le nom est le même que celui d'un cours? Donner leurs noms.

$p[nom] \text{ } s[nom=nomC] (\text{ Personne } \times \text{ Cours })$ avec un produit et une sélection

ou

$p[nom] (\text{ Personne } * [nom=nomC] \text{ Cours })$ avec une thêta-jointure

Le résultat est une relation vide.

2.9 Division

Définition : Soient deux relations $R (A_1, \dots, A_n)$ et $V (A_1, \dots, A_f)$ ($f < n$) telles que tous les attributs de V sont aussi attributs de R , alors la division de R par V , notée :

R / V

crée une nouvelle relation temporaire de schéma $(A_{f+1}, A_{f+2}, \dots, A_n)$, et de population égale aux tuples de R , tronqués à $[A_{f+1}, A_{f+2}, \dots, A_n]$, et qui existent dans R concaténés à tous les tuples de V , c'est-à-dire:

$\{ \langle a_{f+1}, a_{f+2}, \dots, a_n \rangle / \forall \langle a_1, a_2, \dots, a_f \rangle \in V, \exists \langle a_1, a_2, \dots, a_f, a_{f+1}, a_{f+2}, \dots, a_n \rangle \in R \}$

| | | | | | | | | | |
|------------|---|-----|----|-----|-----|-----|---------|--------|-----|
| Exemples : | R | (A, | B, | C) | V | (B, | C) | R / V | (A) |
| | | 1 | 1 | 1 | | 1 | 1 | | 1 |
| | | 1 | 2 | 0 | | 2 | 0 | | 3 |
| | | 1 | 2 | 1 | | | | | |
| | | 1 | 3 | 0 | V' | (B, | C) | R / V' | (A) |
| | | 2 | 1 | 1 | | 1 | 1 | | 1 |
| | | 2 | 3 | 3 | | | | | 2 |
| | | 3 | 1 | 1 | | | | | 3 |
| | | 3 | 2 | 0 | | | | | |
| | 3 | 2 | 1 | V'' | (B, | C) | R / V'' | (A) | |
| | | | | | 3 | 5 | | / | |

Exemple : liste des étudiants qui peuvent s'inscrire au cours de système (c'est-à-dire qui ont réussi tous les prérequis de ce cours)

- cours prérequis pour système :

$\text{ReqSyst} := p [nomCprérequis] \text{ } s [nomC = \text{"système"}] \text{ } Prérequis$

Cette relation ReqSyst contient alors deux tuples: algo, C.

- numéros des étudiants qui peuvent s'inscrire au cours de système :

$(p [nomC, n^oE] \text{ } Obtenu) / (a[nomCprérequis : nomC] \text{ } ReqSyst)$

On obtient en résultat un tuple: 22.

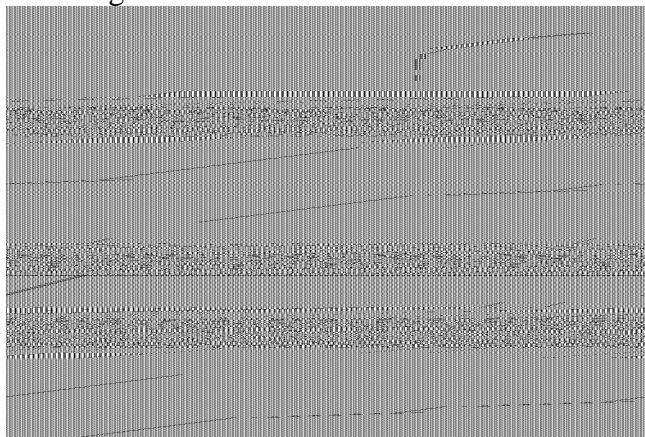
3 Comment écrire une requête compliquée ?

Exemple : noms des étudiants qui suivent un des cours de l'enseignant numéro 3333.

La méthode consiste à représenter visuellement la requête sur le schéma des relations. Elle comprend les étapes suivantes :

1. identifier les relations utiles pour exprimer la requête. Dans l'exemple, ce sont :
Personne pour le nom de l'étudiant,
Cours pour les cours de l'enseignant numéro 3333,
Inscrit et Etudiant pour faire le lien entre ces deux premières relations.
2. recopier le schéma de ces relations, et indiquer sur ces schémas :
 - les attributs qui font partie du résultat de la requête. Dans l'exemple: nom de Etudiant,
 - les conditions portant sur les attributs. Dans l'exemple: dans Cours, n°Ens=3333,
 - les liens entre les relations. Dans l'exemple: n°P de Personne = n°P de Etudiant,
n°E dans Etudiant = n°E dans Inscrit, nomC dans Inscrit = nomC dans Cours.

On obtient donc la figure :



3. traduire cette figure en expression d'algèbre :
 - faire les sélections selon les conditions portant sur les attributs,
 - faire les jointures (naturelle ou thêta) selon les liens entre relations (une jointure par lien),
 - projeter sur les attributs qui font partie du résultat.

On obtient ainsi l'expression :

$p[nom] (Personne * Etudiant * Inscrit * (\sigma [n^{\circ}Ens = 3333] Cours))$

Cette méthode est valable pour la plupart des requêtes. Cependant, certains types de requêtes nécessitent de compliquer la méthode. C'est le cas des requêtes où la même relation est utilisée plusieurs fois avec des ensembles de tuples différents. Par exemple : liste des noms des étudiants qui habitent dans la même ville que l'étudiant Jean Dupont. Ici la relation Etud doit être représentée par son schéma deux fois, une fois pour Jean Dupont, une fois pour les étudiants recherchés.

Enfin, les requêtes qui comportent l'équivalent sémantique d'un "pour tout" ou d'un "aucun" se représentent difficilement visuellement.

4 Les opérateurs déduits

Les opérateurs déduits présentés dans ce cours sont équivalents aux expressions suivantes qui ne comportent que les opérateurs de base et renommer :

Intersection :

$$R \cap S = R - (R - S) = S - (S - R) \quad \text{ou} \\ R \cap S = (R \setminus S) \cup ((R - S) \cap (S - R))$$

Jointure naturelle :

Soient $R(X,Y)$ et $S(Y,Z)$

$$R * S = \{ [X,Y,Z] \mid \exists Y' (Y = Y' \wedge \exists a [Y : Y'] S) \}$$

Thêta jointure :

Soient $R(X,Y)$ et $S(U,V)$

$$R *_{\theta} S = \{ [p] \mid [p] \in (R \times S) \}$$

Division :

Soient $R(X,Y)$ et $S(Y)$

$$R/S = \{ [X] \mid R - \{ [X] \mid ((\{ [X] \} \times S) - R) \}$$

5. Complexité des opérateurs

Définition : la complexité d'un algorithme c'est l'ordre de grandeur du nombre d'instructions à exécuter. Pratiquement on compte les itérations.

5.1 Sélection : $\sigma [\text{condition}] R$

Dans le cas le pire (pas de chemin d'accès selon la condition), cela implique de balayer la relation et tester la condition sur chaque tuple.

Complexité = $\text{card}(R)$.

Taille du résultat $\in [0 : \text{card}(R)]$.

La relation résultat est en général plus petite que la relation initiale.

5.2 Projection : $\pi [A_i, A_k \dots] R$

Cela implique de balayer la relation et ne conserver que les valeurs des attributs A_i, \dots, A_k si c'est une projection toute seule. En général, dans une expression, la projection se fait en même temps que l'opérateur qui la précède.

Exemple : $\pi [\text{nom}] \sigma [\text{adr} = \text{"Lausanne"}] \text{Personne}$.

Les deux opérations, sélection et projection, sont exécutées en même temps par le SGBD.

Mais la projection peut aussi nécessiter de supprimer les valeurs doubles créées, par exemple par un tri.

Complexité : 0 (ou celle d'un tri).

Taille du résultat $\in [1 : \text{card}(R)]$.

Si la liste de projection inclut un identifiant de la relation, alors le résultat aura le même nombre de tuples que la relation initiale.

5.3 Jointure (naturelle ou thêta)

Pour calculer $R * S$, on prend un tuple de R et on cherche le ou les tuples de S qui correspondent à la jointure. Ce qui nécessite deux itérations emboîtées dans le cas le pire (pas de lien de jointure préexistant, pas d'index ...)

L'algorithme est donc :

- balayer R et pour chaque tuple de R faire :
- balayer S et comparer chaque tuple de S avec celui de R .

Complexité = $\text{card}(R) \times \text{card}(S)$.

Taille du résultat $\in [0 : \text{card}(R) \times \text{card}(S)]$.

On remarque que dans le cas du produit cartésien (qui peut être vu comme une théta-jointure sans condition) la taille du résultat est exactement égale à $\text{card}(R) \times \text{card}(S)$.

On a intérêt à faire "petites" jointures (i.e., qui portent sur des relations avec peu de tuples); sinon on peut obtenir en résultat une relation énorme! Dans les expressions optimisées d'algèbre, les sélections seront faites le plus tôt possible. Elles porteront autant que possible sur les relations de la base de données (et pas sur les jointures).

Exemple : cours assurés par Muller :

expression non optimisée: $s[\text{nom} = \text{"Muller"}] (\text{Personne} * [n^{\circ}P = n^{\circ}Ens] \text{Cours})$

expression optimisée: $(s[\text{nom} = \text{"Muller"}] \text{Personne}) * [n^{\circ}P = n^{\circ}Ens] \text{Cours}$

L'optimisation des requêtes sera traitée plus en détail dans un chapitre ultérieur.

6 Propriétés des opérateurs

Dans ce paragraphe, nous donnons les propriétés principales des opérateurs. Ces propriétés sont utiles pour l'optimisation des requêtes. Elles définissent pour chaque expression d'algèbre les transformations qu'on peut lui appliquer pour obtenir d'autres expressions équivalentes (qui donnent le même résultat). Parmi ces autres expressions équivalentes, certaines sont plus rapides à exécuter, comme par exemple celles qui font le plus tôt possible les sélections qui réduisent le nombre de tuples du résultat.

6.1 Cascades de projections

Soient deux ensembles d'attributs tels que: $\{A_{i1}, A_{i2}, \dots, A_{ij}\} \subseteq \{A_{k1}, A_{k2}, \dots, A_{kl}\}$, alors:

$$p[A_{i1}, A_{i2}, \dots, A_{ij}] (p[A_{k1}, A_{k2}, \dots, A_{kl}] R) = p[A_{i1}, A_{i2}, \dots, A_{ij}] R$$

6.2 Cascades de sélections

$$s[p_1] (s[p_2] R) = s[p_2] (s[p_1] R) = s[p_1 \wedge p_2] R$$

6.3 Propriétés des jointures et du produit

Commutativité:

Jointure naturelle:

$$R * S = S * R$$

Thêta-jointure:

$$R *[p] S = S *[p] R$$

Produit cartésien:

$$R \text{ ' } S = S \text{ ' } R$$

Associativité:

Jointure naturelle: si les relations R et S ont au moins un attribut commun, et si les relations R et T ont au moins un attribut commun, alors:

$$(R * S) * T = S * (R * T)$$

Thêta-jointure: si les relations R, S et T n'ont aucun attribut commun, alors:

$$(R *[p_1] S) *[p_2] T = S *[p_1] (R *[p_2] T)$$

Produit cartésien: si les relations R, S et T n'ont aucun attribut commun, alors:

$$(R \cup S) \cap T = S \cap (R \cup T)$$

6.4 Propriétés des opérateurs ensemblistes

Les opérateurs d'union et d'intersection sont commutatifs et associatifs. Ces propriétés sont directement dérivées de la théorie des ensembles. Par contre, l'opérateur de différence n'est ni associatif ni commutatif.