LIF4 - Initiation aux Bases de données : Optimisation à base de règles

E.Coquery

emmanuel.coquery@liris.cnrs.fr

http ://www710.univ-lyon1.fr/~ecoquery/enseignement/lif4

Motivation

Opérations coûteuses dans les requêtes relationnelles :

- Jointure ($\bowtie$)
- Produit cartésien ($\times$)

Évaluation du coût d'un produit cartésien $R \times S$:

- Lire $R \Rightarrow k_R$ n-uplets.
- Lire S (k_S n-uplets) k_R fois $\Rightarrow k_R \times k_S$ n-uplets.

Temps d'accès et débit :

- Mémoire : accès $\sim$ 10 ns, débit $\sim$ 2 Go/s
- Disque dur : accès $\sim$ 5 ms, débit $\sim$ 50 Mo/s

$\Rightarrow$ important d'évaluer le coût des lectures disque.

Coût du produit cartésien

Hypothèses :

- R et S contiennent n_R et n_S n-uplets.
- Un bloc disque peut contenir b_R n-uplets de R ou b_S n-uplets de S.
- Nombre de blocs pour R et S : $k_R = \frac{n_R}{b_R}$ et $k_R = \frac{n_R}{b_R}$
- m blocs peuvent tenir en mémoire centrale.

- En gardant 1 bloc pour la lecture de S, on peut garder $m-1$ blocs de R en mémoire.
- Il faut faire $\frac{k_R}{m-1}$ fois lire $m-1$ blocs de R et k_S blocs de S.

$\Rightarrow$ coût de $\frac{k_R}{m-1} \times (m - 1 + k_S)$ lectures de bloc.

Exemple

- Nombre de blocs lus par secondes : $n_{bps} = 20$.
- R : 10000 nuplets, 10 nuplets/bloc $\Rightarrow k_R = 1000$.
- S : 10000 nuplets, 10 nuplets/bloc $\Rightarrow k_S = 1000$.
- Nombre de blocs en mémoire : 100

$\Rightarrow \frac{1000}{99} \times (99 + 1000) \approx 11100$ accès (9,25 min).

Si, à cause d'une sélection, on utilise 10% de R, on peut réduire à 1110+1000 accès soit 1,76 min.

Exemple SQL et Algèbre relationnelle

```
SELECT R.A
FROM R,S
WHERE R.B=S.C
AND S.D = 99
```

Évaluation naïve :

- $\pi_A(\sigma_{B=C \wedge D=99}(R \times S))$

Autre possibilités :

- $\pi_A(\sigma_{B=C}(R \times \sigma_{D=99}(S)))$
- $\pi_A(R \bowtie_{B=C} \sigma_{D=99}(S))$

Optimisation

Objectif : trouver une stratégie d'évaluation permettant d'accélérer l'évaluation.

- Par manipulations algébriques, indépendante de la manière dont sont stockée les informations.
- En utilisant une stratégie dépendante du stockage (clés).

Dans ce cours : manipulations algébriques.

Manipulations algébriques

Idées :

- Effectuer les sélections aussitôt que possible.
- Combiner les sélections et les produits cartésiens pour faire des jointures.
- Combiner les séquences d'opérations unaires, comme les sélections et les projections.
- Chercher les sous-expressions communes dans une expression.

Pour cela, on exploite des identités remarquables de l'algèbre relationnelle.

Lois sur les jointures et les produits

Commutativité :

$$E_1 \bowtie_C E_2 \equiv E_2 \bowtie_C E_1 \quad (1)$$
$$E_1 \bowtie E_2 \equiv E2 \bowtie E_1 \quad (2)$$
$$E_1 \times E_2 \equiv E_2 \times E_1 \quad (3)$$

Associativité :

$$E_1 \bowtie_C (E_2 \bowtie_D E_3) \equiv (E_1 \bowtie_C E_2) \bowtie_D E_3 \quad (4)$$
$$E_1 \bowtie (E_2 \bowtie E_3) \equiv (E_1 \bowtie E_2) \bowtie E_3 \quad (5)$$
$$E_1 \times (E_2 \times E_3) \equiv (E_1 \times E_2) \times E_3 \quad (6)$$

Lois sur les projections et les sélections

Cascade de projections :

$$\pi_{A_1,\ldots,A_n}(\pi_{B_1,\ldots,B_k}(E)) \equiv \pi_{A_1,\ldots,A_n}(E) \quad (7)$$

Cascade de sélections :

$$\sigma_C(\sigma_D(E)) \equiv \sigma_{C \wedge D}(E) \quad (8)$$
$$\sigma_C(\sigma_D(E)) \equiv \sigma_D(\sigma_C(E)) \quad (9)$$

Permutations de projections et sélections :

- Si C ne porte que sur des attributs parmi $A_1, \ldots, A_n$:

$$\pi_{A_1,\ldots,A_n}(\sigma_C(E)) \equiv \sigma_C(\pi_{A_1,\ldots,A_n}(E)) \quad (10)$$

Lois sur les sélections et les autres opérations

Sélections et produits cartésiens :

- Si C ne porte que sur les attributs de E_1 :

$$\sigma_C(E_1 \times E_2) \equiv \sigma_C(E_1) \times E_2 \quad (11)$$
$$\sigma_{C \wedge D}(E_1 \times E_2) \equiv \sigma_D(\sigma_C(E_1) \times E_2) \quad (12)$$

- Si C ne porte que sur les attributs de E_1 et D sur les attributs de E_2 :

$$\sigma_{C \wedge D}(E_1 \times E_2) \equiv \sigma_C(E_1) \times \sigma_D(E_2) \quad (13)$$

Sélections et union / différence :

$$\sigma_C(E_1 \cup E_2) \equiv \sigma_C(E_1) \cup \sigma_C(E_2) \quad (14)$$
$$\sigma_C(E_1 - E_2) \equiv \sigma_C(E_1) - \sigma_C(E_2) \quad (15)$$

Projection et autres opérations

Projection et

- produit cartésien :

$$\pi_{A_1,\ldots,A_n}(E_1 \times E_2) \equiv \pi_{B_1,\ldots,B_k}(E_1) \times \pi_{C_1,\ldots,C_l}(E_2) \quad (16)$$

- union :

$$\pi_{A_1,\ldots,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,\ldots,A_n}(E_1) \cup \pi_{A_1,\ldots,A_n}(E_2) \quad (17)$$

Plan d'exécution

Un plan d'exécution est un programme qui vise à évaluer une requête en algèbre relationnelle et qui consiste en une suite d'étapes parmi les suivantes :

- Application d'un opérateur unaire (sélection ou projection)
- Application d'une sélection puis d'une projection
- Application d'un opérateur binaire ($\times, \cup, \cap, \setminus, \bowtie$)
  - Eventuellement précédé et/ou suivi d'opérateurs unaires.

Un algorithme d'optimisation a pour but de trouver un bon (le meilleur si possible) plan d'exécution.

Optimisation a base de règles

Consiste à appliquer des règles pour transformer une expression de l'algèbre relationnelle en plan d'exécution optimisé :

- Entrée : arbre représentant l'expression à évaluer
- Sortie : plan d'exécution pour la requête.

L'arbre passé en entrée est construit comme suit :

- Les noeuds internes de l'arbre sont les opérateurs de l'expression.
- Chaque noeud a pour fils le ou les sous-arbres construits à partir de son ou ses arguments.
- Les feuilles sont des relations de la bases de données.

Règles - 1

1. Utiliser (8) pour transformer les sélections $\sigma_{C_1 \wedge \cdots \wedge C_n}(E)$ en cascades $\sigma_{C_1}(\ldots \sigma_{C_n}(E))$
2. Pour chaque sélection, utiliser (9) (10) (11) (12) (13) (14) et (15) pour pousser les sélections en bas de l'arbre.
3. Pour chaque projection, utiliser les règles (7) (16), (17) et (10) pour pousser la projection au plus bas dans l'arbre. Eliminer une projection qui conserve tous les attributs.
4. Utiliser les règles (7) (8) et (10) pour pour combiner les cascades de sélections et projections en une sélection unique, une projection unique, ou une sélection suivie d'une projection.

Règles - 2

1. Partionner l'arbre résultant en groupes comme suit :
   - Créer un groupe par noeud binaire ($\times, \cup, \cap, \setminus, \bowtie$).
   - Le groupe inclus tous les ancêtres unaires intermédiaires
   - Le groupe inclus toute branche étiquetée par des opérateur unaires et se terminant à une feuille (table)
2. Chaque groupe correspond à une étape. Les étapes sont à évaluer dans n'importe quel ordre tant qu'un groupe est évalué après ses descendants.