Chapitre 3 . Optimisation de requêtes

11 décembre 2001

1 Optimisation physique

1.1 Liaison et incorporation

Définition 1.1 *Une* liaison *est une relation représentée par un pointeur entre des objets associés. Une* incorporation *se produit lorsqu'il existe un objet composite qui englobe la valeur des objets associés.*

exemple :

```
create type Ville as object(nom VARCHAR(30), pays VARCHAR(15))
/
create type personne as object (nom VARCHAR(30), prenom VARCHAR(30), lieu Ville)
/
```

Incorporation de la ville dans la personne.

```
create type personne as object(nom VARCHAR(30), prenom VARCHAR(30), lieu REF Ville)
/
```

Liaison entre les deux objets.
Les objets incorporés dépendent de l'objet englobant :

- même cycle de vie
- accès à l'objet incorporé se fait par l'intermédiaire de l'objet englobant
- placement physique des objets dans la même page
- pb : partage des objets nécessite une recopie!

1.2 Les index

1.2.1 Index classiques

Objectif : accélérer la sélection ou le tri dans une table (seulement quand on s'intéresse aux valeurs de la clé)

Définition 1.2 *Un index est une sorte de table des matières qui donne pour chaque valeur d'une clé, l'adresse de l'article correspondant à cette clé.*

Un index augmente les performances de recherche d'élément mais diminuent les performances de mise à jour.
Il faut mettre des index sur des éléments qui sont plus lus qu'écrits. Créer la table, insérer les lignes puis mettre un index sinon c'est plus long.
Sous Oracle : On peut mettre des index sur les tables et sur les tables imbriquées.

```
CREATE INDEX <nom> ON <nomtable>(<col1,col2,...>);
```

exemple :

```
CREATE TABLE Lespays (nom VARCHAR(30), capitale Ville, population NUMBER);

CREATE INDEX i_pays ON Lespays(nom,capitale);
```

Cela signifie qu'une table des matières est associée à la table Lespays avec pour index le nom et la capitale. Pour un nom et une capitale donnée on a l'adresse de l'enregistrement du pays correspondant.

1.2.2 Index sur tables imbriquées

exemple :

```
CREATE TYPE employe AS OBJECT (ninsee VARCHAR(13), age NUMBER, nom VARCHAR(30))
/
CREATE TYPE employes AS TABLE OF employe;


CREATE TABLE Lesdepartements
                (numdep VARCHAR(11), budget NUMBER, emps employes)
                NESTED TABLE employes STORE AS tabemp;
```

En fait cela crée deux tables : Lesdepartements et tabemp. La deuxième n'est pas accessible directement à l'utilisateur. Les données de la table imbriquée sont stockées dans la table tabemp qui contient une pseudo-colonne permettant de retrouver pour le département qui lui correspond, cette pseudo-colonne a pour nom NESTED_TABLE_ID.
Ex :

```
SELECT NESTED_TABLE_ID, e.nom FROM THE (SELECT d.emps FROM Lesdepartements d
WHERE d.nom='Informatique') e ;
NESTED_TABLE_ID                      nom
-------------------------------     --------
5E6079A15F                          Bannay
5E6079A15F                          Loiseau

WHERE p.nom='Mathématique') e;
NESTED_TABLE_ID                      nom
-------------------------------     --------
5E6079A15D                          Fermat
```

Si l'on désire créer un index sur une table imbriquée, on peut ajouter la pseudo-colonne NESTED_TABLE_ID dans la clé:

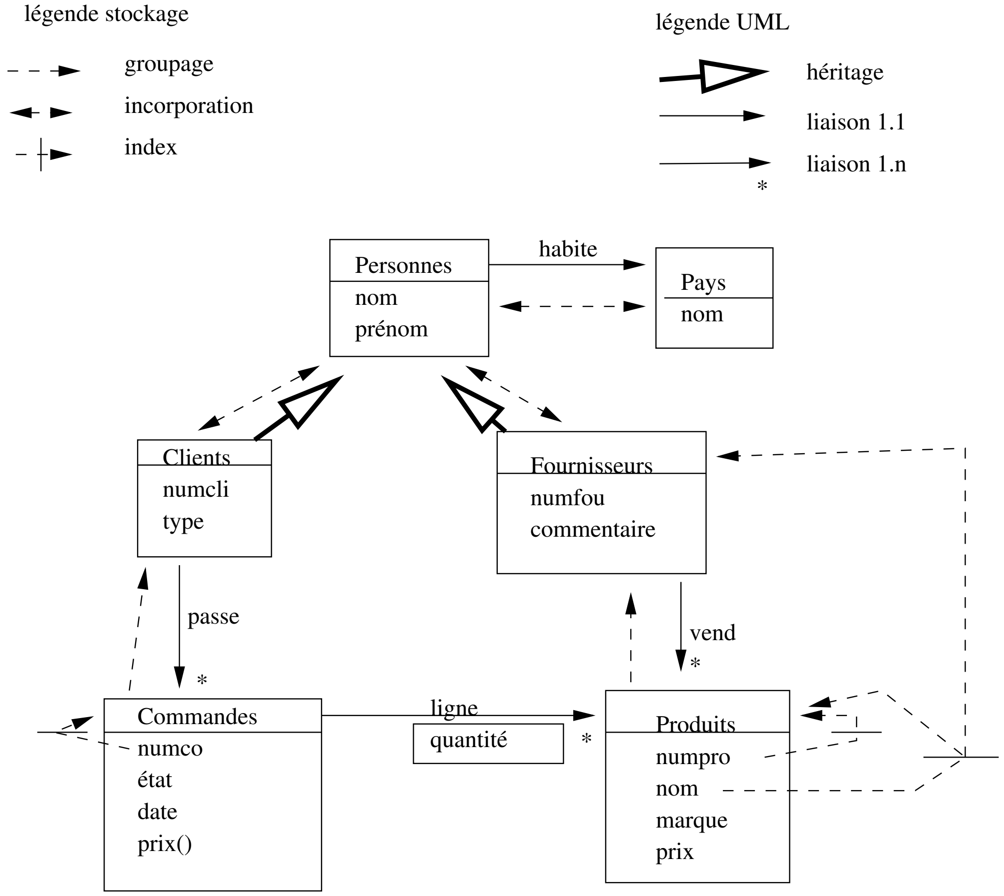
```
CREATE INDEX i_emps ON tabemp(NESTED_TABLE_ID, ninsee);
```

Un index peut également être créé implicitement en ajoutant une contrainte PRIMARY_KEY sur une ou plusieurs colonnes.

1.3 Les clusters

1.3.1 Définition

Un cluster est un stockage groupé des lignes de plusieurs tables ayant une caractéristique commune. Ce qui permet d'accélérer les accès par navigation ou par jointure. On peut stocker ensemble des objets liés par une association.
Schéma de stockage : schéma UML avec symbolisation des groupages, incorporations et index.



1.3.2 Les clusters sous Oracle

Donner un nom pour le cluster qui représente l'association à grouper, préciser le nom des colonnes sur lesquelles se fera le regroupement puis définir les tables en précisant qu'elles appartiennent au cluster. Enfin créer un index pour le cluster.
ex :

```
CREATE CLUSTER paysvilles(codepays CHAR(5));

CREATE TABLE LesPays (numpays CHAR(5), nom VARCHAR(20), pop NUMBER)
CLUSTER paysvilles (numpays);


CREATE TABLE LesVilles
(numville CHAR(5), numpays CHAR(5), nom VARCHAR(20), pop NUMBER, nbhot NUMBER)
CLUSTER paysvilles (numpays);

CREATE INDEX i_pays ON CLUSTER paysvilles; --l'index est obligatoire
```

stockage :	p1	France	57000000	v1	Paris	...
				v2	Toulouse	...
				v3	Angers	...
	p2	Italie	.....			

2 Optimisation logique

2.1 Optimisation syntaxique classique

Transformation des requêtes par le biais de l'algèbre relationnelle
Rappel algèbre relationnelle classique :

- Les opérations ensemblistes ($\cup$, $\cap$, $\setminus$, $\times$).
- Les opérations spécifiques (π, σ, $\bowtie$)
- Les opérations dérivées ($\cap$, complément, jointure externe, division)
- les agrégats et les fonctions de groupe

Propriété 2.1 *Toute question exprimable en logique du premier ordre sans fonction est exprimable en AR*

Les relations unaires (π et σ) réduisent la taille des relations. Les relations binaires ($\bowtie$, $\times$, $\cup$) accroissent la taille des résultats
Optimisation : descendre les relations unaires (règles de réécriture) et utiliser le détachement de sous-questions (INGRES 1976) : transformer une question en deux sous-questions successives.
ex : sous-requêtes en SQL

```
SELECT e.nom FROM employes e, vehicules v
```

```
WHERE e.cdvhc=v.cdvhc
AND v.couleur='BLEUE'

SELECT e.nom FROM employes e
WHERE e.cdvhc IN
        (SELECT v.cdvhc FROM vehicules v
         WHERE v.couleur='BLEUE')
```

2.2 Optimisation objet

2.2.1 Extension de l'algèbre relationnelle

Définition 2.1 (Algèbre relationnelle étendue) *on ajoute des opérateurs pour traduire les concepts objets :*

- *jointures par référence (pour traduire les parcours de chemin)*
- *accès par référence ou par nom*
- *groupage et dégroupage*

groupage d'une collection de tuples sur un attribut Ai : *regroupe les éléments ayant même valeur de A1... Ai-1,Ai+1 ...An*
`Nest(Coll,Ai)` $= \{\langle A1 : e.A1,...,Ai : C,...,An : e.An\rangle \mid \forall r \in C, \exists e \in Coll\ , e.Ai = r\ \}$

dégroupage d'une collection sur attribut Ai
`UnNest(Coll,Ai)` $=\{\langle A1 : e.A1,...,Ai : c,...,An : e.An\rangle \mid e \in Coll, c \in e.Ai\}$

2.2.2 Règles d'optimisation objet

- règles syntaxiques de réécriture comme pour l'algèbre relationnelle
  exemple : $\sigma_{Ai=a,Ak=b}(Nest(X,\{Ai,Aj\}) = \sigma_{Ak=b}(Nest(\sigma_{Ai=a}(X),\{Ai,Aj\})$
- règles sémantiques :
  - les contraintes d'intégrité permettent l'optimisation des requêtes
    ex de CI : `Forall m of type Musée : m.pays= m.adresse.ville.pays`
  - le type abstrait donne des relations d'équivalence entre propriétés
    ex : `Forall f of type formegeométrique : f.deplacer(x,y)= (f.translater_horizontal(x)).translater_vertical(y)`
- règles de plannification : équivalences entre algorithmes de parcours de requêtes
  ex : `parcours(m.adresse.ville.pays.capitale de m in LesMusées)` équivalent à `parcours(m.adresse.ville de m in LesMusées)` $\bowtie$ `parcours(p.capitale de p in Lespays))`