Langage SQL (2)

Sébastien Limet Denys Duchier

IUT Orléans

4 septembre 2007

Le langage SQL : requêtes avancées

- opérations ensemblistes
- requêtes imbriquées
- fonctions d'agrégat
- regroupements
- vues

Opérations ensemblistes

⟨requête⟩ ::= ⟨requête⟩ ⟨opens⟩ ⟨requête⟩

⟨opens⟩ ::= **UNION** | **INTERSECT** | **MINUS**

Exemple : les prénoms portés à la fois par un intervenant et un adhérent :

```
select prenom from adherent
intersect
select prenom from intervenant;
```

Requêtes imbriquées

- facilité dans la construction des requêtes
- puissance d'expression accrue
- difficultés d'implantation dans le SGBD (pb de lenteur)

Bien que la norme SQL prévoit ce genre de requêtes, certains SGBD relationnels n'acceptent pas les requêtes imbriquées (MySQL avant version 5).

Exemple

Liste des adhérents ayant participé à au moins une activité :

```
select nom
  from adherent, participe
 where adherent.numad=participe.numad;
```

Autre solution

```
select nom
  from adherent
 where adherent.numad in (
   select participe.numad
     from participe);
```

on recherche les adhérents dont le numéro fait partie des numéros d'adhérents ayant participé à une activité.

Prédicat `IN`

⟨bool⟩ ::= ⟨colonne⟩ ⟨pred⟩ **(**⟨requête⟩**)**

⟨pred⟩ ::= **`IN`** | **`NOT IN`**

Attention : la sous-requête doit avoir une seule colonne comme résultat

Exemple

Liste des intervenants qui n'animent aucune activité en 1999 :

```
select nom
  from intervenant
 where numinter not in (
    select anime.numinter
      from anime
     where anneeanim=1999);
```

Le prédicat `EXISTS`

⟨bool⟩ ::= **`EXISTS`** **`(`**⟨requête⟩**`)`**
| **`NOT EXISTS`** **`(`**⟨requête⟩**`)`**

Retourne vrai si le résultat de la sous-requête contient au moins une ligne et faux sinon.

Exemple

Liste des intervenants qui n'animent aucune activité en 1999 :

```
select nom
  from intervenant I
 where not exists (
    select A.numinter
      from anime A
     where I.numinter=A.numinter
       and anneeanim=1999);
```

Le prédicat `ALL`

⟨bool⟩ ::= ⟨expr⟩ ⟨op⟩ **ALL** (⟨requête⟩)

⟨op⟩ est un opérateur de comparaison.

Attention : la sous-requête ne doit avoir qu'une seule colonne en résultat.

Vrai seulement quand ⟨expr⟩ satisfait la comparaison ⟨op⟩ avec tous les résultats de la sous-requête.

Exemple ALL

Liste des adhérents qui ont l'âge requis pour toutes les activités :

```
select nom
  from adherent
 where datanais <= ALL (
    select sysdate-(agerequis*365)
      from activite);
```

Fonctions d'agrégat et regroupements

Fonctions d'agrégat : ensemble de fonctions qui permettent d'effectuer des statistiques sur le résultat d'une requête (minimum, maximum, somme, etc. . .)

Regroupements : possibilité de regrouper plusieurs lignes d'une même requêtes (souvent associés aux fonctions d'agrégat)

Fonctions d'agrégat : exemples

Afficher la date de naissance du plus agé des adhérents :

```
select min(datanais) from adherent;
```

Afficher le nombre d'adhérents inscrits à une activité en 2000 :

```
select count(DISTINCT A.numadh)
  from adherent A, partitipe P
 where A.numadh=P.numadh
   and anneeparticipe=2000;
```

Fonctions d'agrégats

$$
\begin{array}{rcl}
\langle \text{agreg} \rangle & ::= & \texttt{count}(\langle \text{exp} \rangle) \\
& | & \texttt{avg}(\langle \text{exp} \rangle) \\
& | & \texttt{sum}(\langle \text{exp} \rangle) \\
& | & \texttt{min}(\langle \text{exp} \rangle) \\
& | & \texttt{max}(\langle \text{exp} \rangle) \\
& | & \ldots
\end{array}
$$

Regroupements : exemple

Client	Nom	Prenom	Age	Ville
	Dupont	Jean	52	Tours
	Duval	Paul	25	Tours
	Martin	Martine	39	Blois
	Bon	Jean	41	Blois

Regroupements : exemple

Client	Nom	Prenom	Age	Ville
	Dupont	Jean	52	Tours
	Duval	Paul	25	Tours
	Martin	Martine	39	Blois
	Bon	Jean	41	Blois

```
select ville
  from client
 group by ville;
```

Groupe 1 et Groupe 2. Résultat :

	Ville
	Tours
	Blois

Regroupements

C'est la clause **group by** :

⟨group by⟩ ::= **group by** ⟨bycols⟩
| **group by** ⟨bycols⟩ **having** ⟨cond⟩
⟨bycols⟩ ::= ⟨colonne⟩
| ⟨colonne⟩, ⟨bycols⟩

Regroupements

- les colonnes du **`select`** doivent toutes apparaître dans le **`group by`** (sauf les opérations)
- les noms de colonnes dans le **`having`** doivent aussi être dans le **`group by`** (ou être fonction d'agrégat)
- la clause **`group by`** va regrouper les lignes qui sont identiques sur les colonnes mentionnées dans le **`group by`**

Exemple

```
select ville, count(*) nbcli
  from client
 group by ville;
```

Client	Nom	Prenom	Age	Ville
	Dupont	Jean	52	Tours
	Duval	Paul	25	Tours
	Martin	Martine	39	Blois
	Bon	Jean	41	Blois

Résultat :

	Ville	Nbcli
	Tours	2
	Blois	2

Exemple

```
select ville, sum(age) total
  from client
 group by ville;
```

Client	Nom	Prenom	Age	Ville
	Dupont	Jean	52	Tours
	Duval	Paul	25	Tours
	Martin	Martine	39	Blois
	Bon	Jean	41	Blois

Résultat :

	Ville	Total
	Tours	77
	Blois	80

Exemple

```
select ville, avg(age) moyenne
  from client
 group by ville;
```

Client	Nom	Prenom	Age	Ville
	Dupont	Jean	52	Tours
	Duval	Paul	25	Tours
	Martin	Martine	39	Blois
	Bon	Jean	41	Blois

Résultat :

	Ville	Moyenne
	Tours	38.5
	Blois	40

Test sur les groupes

Permet de sélectionner des groupes de la requête de regroupement.

Exemple :

```
select ville, avg(age) moyenne
  from client
 group by ville
having avg(age) < 40;
```

Résultat :

Ville	Moyenne
Tours	38.5

Notion de vue

⟨defvue⟩ ::= **create view** ⟨vue⟩ **as** ⟨requête⟩
| **create view** ⟨vue⟩**(**⟨cols⟩**)** **as** ⟨requête⟩

- permet de donner un nom à une requête qui sera utilisée ensuite comme une table (pour la consultation)
- à chaque appel de la vue la requête est réexécutée
- attention : ce n'est pas une vraie table !
- la variante ⟨vue⟩**(**⟨cols⟩**)** permet de renommer les colonnes

Différence entre `where` et `having`

Trouver pour chaque ville la moyenne d'âge des personnes de moins de 40 ans :

```
select ville, avg(age) agm
  from client
 where age < 40
 group by ville;
```

Trouver les villes dont l'âge moyen des habitants est inférieur à 40 ans :

```
select ville, avg(age) agm
  from client
 group by ville
having avg(age) < 40;
```

Différence entre `where` et `having`

- **`where`** sélectionne les lignes de la requête avant de faire les groupes. Il agit donc avant les regroupements
- **`having`** sélectionne les groupes une fois qu'ils sont constitués. Il agit donc après les regroupements

Exemples

Une vue s'appellant `LesAdresses` et ayant une colonne de nom `adresse` :

```
create view LesAdresses as
   select adresse from adherent;
```

Une vue s'appellant `LesVilles` et ayant une colonne de nom `ville` :

```
create view LesVilles(ville) as
   select adresse from adherent;
```

Suppression d'une vue

Une vue (comme une table) persiste même après une déconnexion d'Oracle. Pour supprimer une vue :

```
drop view <VIEW>;
```

Exemple :

```
drop view LesVilles;
```