BASE DE DONNEES ET SYSTEMES DE GESTION DE BASE DE DONNEES (SGBD)

CHAPITRE VI : TRIS ET REGROUPEMENTS

Sommaire

CLAUSE ORDER BY

L'ordre dans lequel le résultat d'une opération de sélection est présenté correspond à un parcours séquentiel de la table lors de l'exécution. L'utilisateur peut cependant souhaiter voir ce résultat dans un ordre particulier, qu'il peut choisir par la clause ORDER BY.

Les arguments en sont :

```
… ORDER BY < nom_col  | entier > [ASC  |  DESC]

[ , < nom_col  | entier > [ASC  | DESC], …]
```

Ainsi dans la table PASSAGERS, on peut relever par ordre alphabétique les noms et adresses des passagers dont le numéro de téléphone commence par 041/ :

```
SELECT nom, prenom, rue, boite, localite, tel
        FROM passagers
                WHERE tel LIKE '041/%'
                        ORDER BY nom ;
```

Les options ASC et DESC permettent d'ordonner les valeurs respectivement par ordre croissant et décroissant. L'option ASC est prise par défaut.

Un classement peut être réalisé sur plusieurs colonnes, auquel cas le tri principal se fait sur la première colonne spécifiée, les classements secondaires se faisant sur les colonnes suivantes.

Ainsi, dans la requête ci-dessus, il peut arriver que plusieurs passagers aient le même nom de famille, donc il est nécessaire de trier en se basant sur le prénom également. On modifie, la requête qui devient :

```
SELECT nom, prenom, rue, boite, localite, tel
        FROM passagers
                WHERE tel LIKE '041/%'
                        ORDER BY nom, prenom ;
```

Autre exemple :

```
SELECT nom, prenom, rue, boite, localite, tel
    FROM passagers
        ORDER BY nom, prenom
```

Note

Les colonnes indiquées dans la clause ORDER BY doivent se retrouver dans le SELECT (N'est pas vérifié pour MySQL). Si l'on souhaite faire un tri en fonction d'une expression et non d'une colonne, on peut indiquer la position de l'expression dans la clause SELECT, plutôt que son nom. Ce numéro peut bien entendu se rapporter à une simple colonne plutôt qu'à une expression :

```
SELECT nom, prenom, rue, boite
    FROM passagers
        WHERE tel LIKE '041/%'
                ORDER BY 1, 2 ;
```

CLAUSE GROUP BY

Syntaxe

```
… GROUP BY < nom_col 1 |  entier >
        [ , < nom_col 2  | entier > , …]
```

Cette clause effectue un groupement des lignes de la table en fonction d'une ou de plusieurs colonnes spécifiées dans le GROUP BY. Pour chaque valeur de ces colonnes, un groupe distinct sera créé. Une valeur NULL engendre également un groupe particulier.

Le SELECT peut prendre deux formes :

- soit il contient exclusivement des noms de colonnes, dont certaines se retrouvent dans le GROUP BY ;
- soit il contient des noms de colonnes et des fonctions statistiques, tous ces noms de colonnes devant se retrouver dans le GROUP BY.

Ainsi, la requête suivante compte le nombre de réservations effectuées pour chaque vol :

```
SELECT no_vol, COUNT(*) AS nbEnreg
    FROM reservations
        GROUP BY no_vol ;
```

CLAUSE HAVING

Elle a la forme syntaxique :

… HAVING condition …

Cette clause impose une condition aux groupes formés par la clause GROUP BY. De même que la clause WHERE conduit à la suppression des lignes, HAVING permet d'éliminer des groupes.

Notons que la condition fait intervenir une expression qui possède une valeur unique pour chaque groupe ; il faut donc obligatoirement qu'une fonction statistique apparaisse dans la clause HAVING.

Exemple :

Supposons que l'on souhaite connaître les vols pour lesquels cinquante places au plus sont réservées. On écrira :

```
SELECT no_vol, COUNT(*)
    FROM reservations
        GROUP BY no_vol
            HAVING COUNT(*) <= 50 ;
```

COMMANDE UNION

Elle s'intercale entre des instructions SELECT et permet ainsi de combiner les ensembles de résultats issus de chacune de ces instructions ; de plus, elle élimine les tuples qui se présentent plusieurs fois.

La syntaxe en est :

```
    SELECT … FROM … WHERE …
    UNION
    SELECT … FROM … WHERE …
    UNION
…
    [ ORDER BY entier [ASC   DESC]   [ , entier [ASC  DESC], …] ] ;
```

Les tables qui résultent de divers SELECT doivent comporter chacune le même nombre de colonnes, les colonnes correspondantes étant de même type. Les colonnes de type LONG VARCHAR ne sont pas autorisées.

La clause optionnelle ORDER BY peut servir à trier les composantes du résultat final. L'entier associé à cette clause sera le numéro de la colonne en fonction de laquelle s'effectue le tri.

Exemple :

Créons une table nommée USER :

```
CREATE
        TABLE utilisateur
                (id INT, name VARCHAR(20));
```

Insérons quelques enregistrements :

```
        INSERT INTO utilisateur (id, name) VALUES (1, 'KOSSI');
        INSERT INTO utilisateur (id, name) VALUES (2, 'ALI');
        INSERT INTO utilisateur (id, name) VALUES (319, 'Ancion');
```

Si alors l'administrateur souhaite faire la liste de toutes les personnes connues de la base de données, en vue, par exemple, de vérifier qu'aucun numéro d'identification ne se retrouve deux fois, il entrera la commande :

```
        SELECT   no_pass, nom
                FROM   passagers
        UNION
        SELECT   id, name
                FROM   utilisateur
        ORDER   BY  1;
```

Ainsi, les doublets de tuples identiques seront éliminés, de tels doublets signifiant simplement qu'un utilisateur de la base de données est également passager. L'ensemble des tuples, après fusion, sera redonné en fonction du numéro

d'identification par la clause ORDER BY 1, permettant ainsi de repérer facilement les paires de passagers à numéro identique.

Remarque.

Dans certaines versions de SQL (comme par exemple SQL*PLUS) existent deux autres opérateurs de liaison : INTERSECT et MINUS.

- INTERSECT combine les requêtes et fournit en retour uniquement les lignes communes aux deux tables qui le constituent ;
- MINUS combine les requêtes et retourne les lignes issues de la première requête, si elles ne se retrouvent pas dans la deuxième.

En bref, trois clauses peuvent être annexées à une instruction SELECT : ORDER BY, GROUP BY et HAVING. Plusieurs SELECT seront eux-mêmes réunis par l'opérateur UNION ou, dans certaines versions de SQL, par INTERSECT ou MINUS.