Requêtes SELECT avec les opérateurs d'ensembles

L'opérateur INTERSECT : cet opérateur permet de ramener l'intersection des données entre deux tables. Les deux commandes SELECT doivent avoir le même nombre d'attributs , et des attributsde même type et dans le même ordre.
Syntaxe:

```
Instruction SELECT1
INTERSECT
Instruction SELECT 2
[ORDER BY].
```

- Le nombre de colonnes renvoyées par SELECT 1 doit être le même que celui renvoyé par SELECT 2
- Le type de données SELECT 1 doit être le même que celui de SELECT 2
- La clause optionnelle ORDER BY doit se faire selon un numéro de colonne et non selon le nom.
- SELECT 1 et SELECT 2 ne peuvent contenir des clauses ORDER BY.

Exemple

```
SELECT NOM, PRENOM FROM ETUDIANTS
INTERSECT
SELECT NOM, PRENOM FROM ENSEIGNANTS
ORDER BY 1
```

Permet de ramener tous les étudiants qui sont en même temps des enseignants.

L'opérateur UNION

Cet opérateur renvoi l'ensemble des lignes des deux tables. Si des lignes sont redondantes elles sont renvoyées une seule fois. Pour renvoyer toutes les lignes, utiliser l'option ALL
Les mêmes contraintes qui s'appliquent pour INTERSECT s'appliquent pour UNION

Syntaxe

```
Instruction SELECT1
UNION [ALL]
Instruction SELECT 2
[ORDER BY].
```

```
SELECT NOM, PRENOM FROM ETUDIANTS
UNION
SELECT NOM, PRENOM FROM ENSEIGNANTS
ORDER BY 1
```

Permet de ramener tous les étudiants et tous les enseignants. Les enseignants qui sont en même temps des étudiants sont ramenés une seule fois.

L'opérateur MINUS

Cet opérateur renvoi l'ensemble des lignes de la première table MOINS les lignes de la deuxième table.

Les mêmes contraintes qui s'appliquent pour INTERSECT s'appliquent pour MINUS

Syntaxe

```
Instruction SELECT1
MINUS
Instruction SELECT 2
[ORDER BY].
```

Les vues

Définition

Une vue c'est une table dont les données ne sont pas physiquement stockées mais se réfèrent à des données stockées dans d'autres tables. C'est une fenêtre sur la base de données permettant à chacun de voir les données comme il le souhaite.
On peut ainsi définir plusieurs vues à partir d'une seule table ou créer une vue à partir de plusieurs tables. Une vue est interprétée dynamiquement à chaque exécution d'une requête qui y fait référence.

Avantages

Les vue permettent de protéger l'accès aux tables en fonction de chacun des utilisateurs. On utilise une vue sur une table et on interdit l'accès aux tables. C'est donc un moyen efficace de protéger les données
Les vues permettent de simplifier la commande SELECT avec les sous requêtes complexes. On peut créer une vue pour chaque sous-requête complexe, ce qui facilite sa compréhension
Il est possible de rassembler dans un seul objet (vue) les données éparpillées

Une vue se comporte dans la plus part des cas comme une table. On peut utiliser une vue comme source d'information dans les commandes SELECT, INSERT, UPDATE ou DELETE. Une vue est créée à l'aide d'une sous-requête.

Syntaxe

```
CREATE [OR REPLACE ][FORCE] VIEW <nom_de_la_vue> AS <sou_requête> [WITH CHECK OPTION]
```

OR REPLACE : commande optionnelle qui indique lors de la création de la vue de modifier la définition de celle-ci ou de la créer si elle n'existe pas
FORCE : permet de créer la vue même si les sources de données n'existent pas.
WITH CHECK OPTION : cette option permet de contrôler l'accès à la vue et par conséquent à la table dont elle est issue.

Contraintes d'utilisation

Vous ne pouvez pas :

- Insérer dans une table à partir d'une vue, si la table contient des champs NOT NULL et qui n'apparaissent dans la vue il y'aura violation de contraintes d'intégrité
- Insérer ou mettre à jour dans la vue si la colonne en question est un résultat calculé.
- Insérer ou mettre à jour (INSERT, UPDATE et DELETE) si la vue contient les clauses GROUP BY ou DISTINCT.
- Utiliser une vue comme source de données pour INSERT, UPDATE et DELETE si elle définie avec :
    - Une jointure
    - Une opération d'ensemble
    - Une clause GROUP BY, CONNECT BY, DISTINCT de l'ordre SELECT
    - Une fonction de groupe (SUM, MAX…)

De manière générale, on peut dire que les commandes SQL INSERT, UPDATE et DELETE ne peuvent s'appliquer qu'a une vue utilisant une table avec restriction et sélection.
Exemple s

```
CREATE VIEW Eleves as
select numad, nom, prenom
from etudiants;

insert into Eleves values (100,'nom1','prn1');

CREATE VIEW ElevesDeMonteral as
select numad, nom, prenom,ville
from etudiant1 where Ville ='MONTREAL'
WITH CHECK OPTION;

insert into eleves values (21,'nom3','prn3','dallas');
```

L’instruction INSERT va renvoyer une erreur à cause du WITH CHECK OPTION

Détruire une VUE :

DROP VIEW nom_de_vue permet de supprimer la vue

RENAME ancien_nom TO nouveau_NOM renomme une vue.