UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

Initiation au langage SQL niveau 2

Morgane De Nicolo, Chloé Cornet, Charles Bénier, Nicolas Pegdwendé Sawadogo, Harry Abogourin

UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

Sommaire

- Introduction
- Les requêtes de mise à jour
- La gestion des privilèges
- Les sous requêtes
- Les jointures en sql
- La notion de TOP
- Pour aller plus loin …
- Bibliographie

Introduction

- le cours de SQL 2 s'inscrit dans la suite du cours SQL 1 mais il traite cette fois-ci non pas les tables, mais les enregistrements

- utilisation d'un langage spécifique à Access
  - Facilement accessible
  - Très intuitif pour débuter
  - Attention : Langage spécifique et fonctionnalités réduites

Les requêtes de mise à jour

But : effectuer des transactions sur les enregistrements de la BDD

INSERER UN ENREGISTREMENT DANS UNE TABLE

Syntaxe SQL :

```
INSERT  INTO nom_table  (nom_attribut)
VALUES (liste_valeurs)
```

Les requêtes de mise à jour

SUPPRIMER UN (DES) ENREGISTREMENT(S) D'UNE TABLE

Cas 1 : Suppression de toutes les lignes

Syntaxe SQL :

```
DELETE FROM nom_table
```

Cas 2 : Suppression conditionnelle

Syntaxe SQL :

```
DELETE FROM nom_table
WHERE [condition]
```

Les requêtes de mise à jour

MODIFIER UN (DES) ENREGISTREMENT(S) D'UNE TABLE

Cas 1 : Mise à jour d'une colonne sans condition

Syntaxe SQL :

```
UPDATE nom_table
SET colonne = valeur
```

Cas 2 : Mise à jour d'une colonne avec filtrage

Syntaxe SQL :

```
UPDATE nom_table
SET colonne = valeur
WHERE [condition]
```

La gestion des privilèges en SQL

But : Contrôler les droits et la création des utilisateurs + manipuler des objets (tables par exemple)

❑ OCTROYER DES DROITS D'ACCES A UNE BASE DE DONNEES

Syntaxe SQL :

GRANT privilèges ON table

TO gratifié

Exemple :

GRANT SELECT ON MAGASIN TO DUPONT

→ *Donne l'autorisation à Dupont de lancer les ordres SQL SELECT sur la table Magasin*

La gestion des privilèges en SQL

❑ REVOQUER (RETIRER) DES DROITS D'ACCES A UNE BASE DE DONNEES

Syntaxe SQL :

REVOKE privilèges ON table

FROM gratifié

Exemple :

REVOKE SELECT ON MAGASIN FROM DUPONT

→ *Supprime l'autorisation à Dupont de lancer les ordres SQL SELECT sur la table Magasin*

Les sous requêtes

But : Permet d'imbriquer une requête dans une autre, pour utiliser son résultat dans la requête supérieure

Syntaxe SQL :

SELECT nom_attribut FROM nom_table WHERE condition avec sous requête entre parenthèses

Exemple Access:

```
SELECT * FROM etudiant WHERE Etudiant.age > (SELECT AVG(Etudiant.age) FROM etudiant)
```

→*Renvoi toutes les informations sur les étudiants dont l'âge est supérieur à la moyenne.*

Remarque:

- Les sous requêtes peuvent être utilisées dans plusieurs clauses (select, having, where …)
- Les sous requêtes peuvent renvoyer une ou plusieurs valeurs

Les jointures en sql

But : Permettent de mettre en relation plusieurs tables pour des interrogations communes

- **La jointure croisée**
  - **Effectue le produit cartésien entre les enregistrements de deux tables**

Syntaxe SQL :

```
SELECT nom_attribut FROM nom_table A CROSS JOIN nom_table B
```

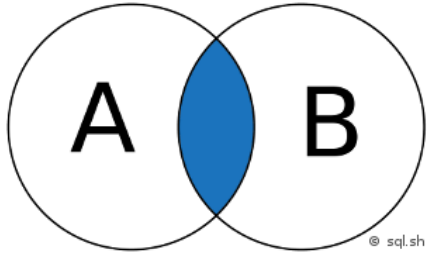
Exemple Access:

```
SELECT nom_attribut FROM nom_table A , nom_table B
```

Remarque:

- Le nombre d'enregistrements peut très vite augmenter !

Les jointures en sql



Les jointures internes

- **Joindre plusieurs tables sur une condition en conservant uniquement les enregistrement correspondant à la condition**

Syntaxe SQL :

SELECT nom_attribut FROM nom_table 1 INNER JOIN nom_table 2 ON condition

Exemple Access:

SELECT * FROM Client INNER JOIN Produit ON Client.Num_p= Produit.Num.p

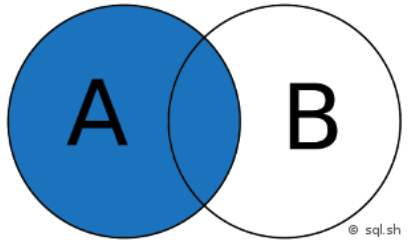
SELECT * FROM commande INNER JOIN (client INNER JOIN produit ON Client.Num_p=Produit.Num.p) ON

Client.Num_co = Commande.Num_co;

Remarque:

- Il est possible d'imbriquer les jointures grâce à des parenthèses.

Les jointures en sql



- **Les jointures externes**
  - **La jointure gauche**
    - **Permet de conserver les résultats d'une requête commune à deux tables mais aussi l'ensemble des enregistrements de la table de gauche**

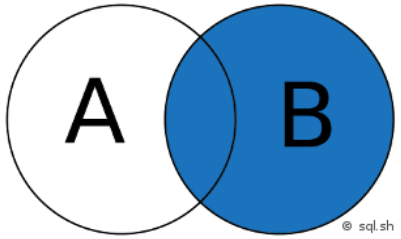
Syntaxe SQL :

```
SELECT nom_attribut FROM nom_table 1 LEFT JOIN nom_table 2 ON condition WHERE … GROUP BY …
```

Exemple Access:

```
SELECT * FROM Client LEFT JOIN  Produit ON Client.Num_p= Produit.Num.p
```

Les jointures en sql



- **Les jointures externes**
  - **La jointure droite**
    - **Permet de conserver les résultats d'une requête commune à deux tables mais aussi l'ensemble des enregistrements de la table de droite**

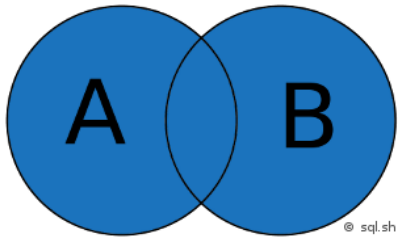
Syntaxe SQL :

```
SELECT nom_attribut FROM nom_table 1 RIGHT JOIN nom_table 2 ON condition WHERE …
GROUP BY …
```

Exemple Access:

```
SELECT * FROM Client RIGHT JOIN  Produit ON Client.Num_p= Produit.Num.p
```

Les jointures en sql



- **Les jointures totales**
  - **Permet de conserver les résultats d'une requête commune à deux tables mais aussi l'ensemble des enregistrements de la table de droite et de gauche**

Syntaxe SQL :

```
SELECT nom_variable FROM nom_table 1 FULL JOIN nom_table 2 ON condition
```

Exemple Access:

```
SELECT * FROM Client RIGHT JOIN  Produit ON Client.Num_p= Produit.Num.p

UNION

SELECT * FROM Client LEFT JOIN  Produit ON Client.Num_p= Produit.Num.p
```

→*Conserve l'ensemble des enregistrements de la table Client et Produit ainsi que les enregistrements présents dans les deux tables*

Remarques:

- Access ne permet pas d'effectuer la clause « full join »
- Il faut donc utiliser une jointure droite et une jointure gauche que nous combinerons par le mot clé « union »

Les jointures en sql

- **La commande union**
  - **Permet de rassembler les éléments issus de plusieurs requêtes**

Syntaxe SQL :

```
SELECT nom_attribut FROM nom_table A

UNION

SELECT nom_attribut FROM nom_table B
```

Remarques:

- Le mot clé « union » ne conserve pas les enregistrements identiques aux deux requêtes, pour cette opération il faut utiliser « union all »
- Les requêtes doivent retourner le même nombre de colonnes, les attributs doivent être de même types et dans le même ordre

La notion TOP

La mot clé « TOP » permet de retourner les n premières lignes d'un résultat sans passer par l'option ORDER BY.

Syntaxe SQL :

```
SELECT TOP 1 nom_attribut

FROM nom_table A

GROUP BY [critères]
```

Remarque:

- Il est fortement conseillé d'utiliser ORDER BY tout de même car sans cette notion, la requête n'est pas déterministe et renverrais un résultat trié sur un critère arbitraire.

Pour aller plus loin …

- Le langage SQL est un outil de requêtage et ne permet pas de faire de la programmation (boucles , fonctions …)
- Le SQL est souvent intégré à des langages de programmation (java, php, pl/sql…) pour permettre des opérations procédurales
- Il est possible d'optimiser les traitements sur une base de données grâce aux procédures stockées et aux déclencheurs

Bibliographie

- http://sqlpro.developpez.com/
- ISO/CEI 9075:2011
- http://sql.sh/