Bases de données – Niveau 1
SQL - MySQL – MariaDB - Cours et TP 7
Clé primaire concaténée

MariaDB : https://mariadb.org/

Site officiel MySQL : http://www-fr.mysql.com/

Zone pour les développeurs : http://dev.mysql.com/

Téléchargement : http://dev.mysql.com/downloads/

Documentations MySQL : http://dev.mysql.com/doc/refman/5.0/fr/

http://dev.mysql.com/doc/refman/5.6/en/ http://dev.mysql.com/doc/index.html

Mémo SQL : http://www.volubis.fr/bonus/SQL_memo.htm

Bertrand LIAUDET

SOMMAIRE

Première édition : septembre 2007
Deuxième édition : novembre 2008

SQL : CONSULTATION DE LA BD - SUITE

PRINCIPALES NOTIONS	
Jointure artificielle	**Requête imbriquée**
Opération ensembliste	**ANY, ALL**
EXISTS, NOT EXISTS	**IN, NOT IN**
UNION	**MINUS, INTERSECT**

1. Rappels sur les opérations s'appliquant à plusieurs tables

Classification générale des opérations s'appliquant à plusieurs tables

Il y a trois sortes d'opérations s'appliquant à plusieurs tables :

- les jointures
- les opérations ensemblistes
- les sous interrogations

Classification générale des jointures

Les jointures se divisent en 3 cas :

- Les jointures naturelles
- Les jointures artificielles
- Les jointures externes

Opérations élémentaires constituant une jointure

Une jointure est constituée de :

- Un produit cartésien
- Une restriction de jointure

2. Les jointures artificielles

Définition

Jointure en général

La jointure de deux tables c'est leur **produit cartésien** et la **restriction** consistant à comparer un attribut de la première table à un attribut de la deuxième table.

```
Select *
from table_1, table_2
where table_1.attribut_1  opérateur table_2.attribut_2;
```

Jointure naturelle

Dans une jointure naturelle, la restriction concerne un attribut clé primaire et un attribut clé étrangère y faisant référence.

La comparaison entre les deux attributs est une égalité : opérateur "**=**"

Cette restriction peut être appelée **restriction de jointure naturelle**.

Jointure artificielle

La jointure artificielle est une jointure dont la restriction n'est **PAS une restriction de jointure naturelle** : soit elle met en jeu d'autres opérateurs que celui d'égalité, soit elle met en jeu un couple d'attribut qui n'est pas clé-étrangère et clé-primaire correspondante.

Exemple

Tous les employés ayant le même job que JONES :

```
Select distinct e1.NE, e1.nom, e2.job
from emp e1, emp e2
where e1.job = e2.job          -- restriction de jointure artificielle
and e2.ename = 'JONES'         -- restriction spécifique 1
and e1.ename <> 'JONES'        -- restriction spécifique 2
```

Explications

Chaque employé est croisé avec tous les employés (produit cartésien).

On ne s'intéresse qu'aux employés de e2 qui s'appelle JONES (restriction spécifique 1)

La restriction de jointure artificielle est faite sur job : on ne garde que les employés de e1 qui ont le même job que les JONES.

On supprime JONES de la liste des employés de e1 (pour éviter d'avoir JONES dans la réponse).

Pour finir, on projette les employés de e1.

On met un distinct pour le cas où il y ait plusieurs JONES.

3. Les requêtes imbriquées

Présentation

Généralités

Dans une requête imbriquée, on trouve un ou plusieurs select utilisés dans le select principal (le premier de la requête).

On distingue entre le select imbriquant (ou principal) et le ou les select imbriqués.

On peut avoir autant de niveau d'imbrication qu'on veut.

2 types de select imbriqués

- Les **select imbriqués dans le « from »** : le select imbriqué remplace une table.
- Les **select imbriqués dans le « where »** : le select remplace une valeur ou une liste de valeurs possibles dans une restriction.

2 types de select imbriqués dans le where

Deux types de select imbriqué imbriqué dans le where selon la restriction effectuée :

- Comparaison de chaque tuple du select imbriqué aux tuples du select principal : ALL et ANY.
- Test de la cardinalité vide ou pas de la table résultat du select imbriqué : EXISTS et NOT EXISTS.

Bilan : 3 catégories de requêtes imbriqués

1. Le select imbriqué dans from qui remplace une table.
2. Le select imbriqué dans le where avec comparaison entre le select principal et le select imbriqué : ALL et ANY
3. Les requêtes imbriquées avec test de la cardinalité vide ou pas du select imbriqué : EXISTS et NOT EXISTS

Select imbriqué dans le from

Toute table d'un select peut être remplacée par un autre select.

- ***Exemple : nombre moyen d'employés par département***

```
Select avg(nb)
from (
        select count(*) nb
        from emp
        group by nd ) tnb
;
```

- ***Remarques syntaxiques***
  - Le select imbriqué est mis entre parenthèses.
  - Le select imbriqué est obligatoirement renommé (« tnb » dans notre exemple).
  - Les attributs projetés dans le select imbriqué sont obligatoirement renommés quand il s'agit de fonction de groupe.

Select imbriqué dans le where : ALL et ANY

Présentation

Le select imbrique dans le where avec opérateur ALL ou ANY consiste en une **comparaison de chaque tuple du select principal avec les tuples du select imbriqué.**

Syntaxe

La syntaxe générale est la suivante :

```
Select * from  table
where liste 1 d'attributs opérateur
        (Select liste 2 d'attributs  from ... );
```

- ***L'opérateur***

c'est un opérateur de comparaison booléen classique : =, !=, >, >=, <, <=, suivi d'un opérateur spécial : ANY ou ALL.

- ***Les listes d'attributs***

Elles doivent avoir la même forme dans le where du select principal et dans la projection du select imbriqué : même nombre d'attributs et même type pour les attributs. Habituellement, ce sont les même attributs dans le select principal et dans le select imbriqué.

Opérateur ANY

La comparaison booléenne est vraie si elle est vraie pour au moins un tuple du select imbriqué (le deuxième select).

Autrement dit, « = any » vérifie l'appartenance à la liste des tuples du deuxième select.

- ***Exemple : tous les employés ayant le même job que les employés du département 10***

```
Select NE, nom, job from emp
where job = any  (
        select job from  emp
        where ND = 10
);
```

Opérateur ALL

La comparaison booléenne est vraie si elle est vraie pour tous les tuples du select imbriqué (le deuxième select).

Autrement dit, « != all » vérifie la non-appartenance à la liste des tuples du deuxième select.

Opérateurs IN et not IN

L'opérateur « in » remplace « = any ».

L'opérateur « not in » remplance « != all »

```
Select NE, nom, job from emp
where job in  (
        select job from  emp
```

```
        where ND = 10
    );
```

Cas où on peut se passer de ANY et de ALL

Si le deuxième select fournit un seul tuple, on peut se passer de l'opérateur spécial. On peut par exemple écrire "=" à la place de "= any".

C'est le cas particulièrement quand le select imbriqué renvoie une fonction de groupe ou quand une restriction s'est faite sur une valeur de la clé primaire.

Select imbriqué dans le where : EXISTS et NOT EXISTS

Présentation

Le select imbrique dans le where avec opérateur EXISTS et NOT EXISTS consiste, **pour chaque tuple du select principal, à tester si la cardinalité du select imbriqué est vide ou pas.**

Syntaxe

La syntaxe générale est la suivante :

```
Select * from  table
where opérateur (
        select ...
);
```

L'opérateur peut prendre deux valeurs : **EXISTS** ou **NOT EXISTS**.

En général, le select imbriqué utilise un attribut du select principal qui rend variable son résultat en fonction du tuple en cours du select principal.

Opérateur EXISTS

Les tuples du select principal sont sélectionnés si le select imbriqué produit au moins un tuple.

- ***Exemple : tous les employés travaillant dans un département qui contient au moins un 'ANALYST' (c'est un métier) :***

```
Select NE, nom, ND, job  from emp
where exists (
        select * from emp e1
        where job = 'ANALYST'
        and e1.ND = emp.ND
);
```

- ***Explication***

On regarde s'il existe un ANALYST dont le département est égal à celui du tuple en cours. Si oui, on garde le tuple en cours.

- ***Remarque***

Dans le cas d'un select imbriqué dans le where avec EXISTS, on a forcément une restriction de jointure artificielle dans le select imbriqué, sinon le résultat du select imbriqué serait constant et le test EXISTS donnerait soit toujours vrai, soit toujours faux, donc on obtiendrait finalement soit la conservation de tous les tuples du select imbriquant, soit l'élimination de tous les tuples du select imbriquant.

Opérateur NOT EXISTS

Les tuples du select principal sont sélectionnés si le select imbriqué ne produit aucun tuples.

Equivalence entre IN et EXISTS et entre NOT IN et NOT EXISTS

A la place de l'exemple précédent avec un IN, on pourrait écrire avec un EXISTS :

```
Select NE, nom, ND, job from emp e
where ND in (
        select ND from emp
        where = 'ANALYST'
);
```

4. Transformation des select imbriqués dans le where

Transformation en jointure artificielle

On peut très souvent transformer les select imbriqués en jointure artificielle.

Exemple : tous les employés ayant le même job que les employés du département 10

```
Select NE, nom, job from emp
where job = any  (
        select job from  emp
        where ND = 10
);
```

équivaut à :

```
Select empno, ename, job
from emp e1, emp e2
where e1.job = e2.job
and e2.ND = 10
and e1.ND != 10 ;
```

Du bon usage

On choisira toujours la jointure de préférence à un select imbriqué pour des questions d'optimisation.

Transformation en select imbriqué dans le from

Quand le select imbriqué porte sur un résultat statistique, on ne peut pas la transformer en jointure.

Par contre on peut transformer un select imbriqué dans le where en un select imbriqué dans le from.

Exemple : tous les employés gagnants plus que la moyenne

Version avec select imbriqué dans le where :

```
Select NE, nom, sal
from emp
where sal  > (
        select avg(sal) from  emp
);
```

Version avec select imbriqué dans le from :

```
Select NE, nom, sal, moysal
from emp, (select avg(sal) as moysal from  emp) tmoy
where sal  > moysal;
```

Cas où la transformation est impossible

Quand on a un NOT IN ou un NOT EXISTS, si le select imbriqué renvoie plusieurs tuples, on ne peut pas transformer la requête imbriquée en jointure artificielle.

5. La notion de « Vue »

Présentation

Une commande **select** peut être conservée dans une variable appelée "vue".

Une vue est une table virtuelle :

- elle n'a pas d'existence physique ;
- elle est recalculée à chaque utilisation ;
- elle est équivalente à une requête.

Création d'une vue

La syntaxe de la création d'une vue est la suivante :

```
CREATE VIEW nom_vue AS
select …
```

ou

```
CREATE or REPLACE VIEW nom_vue AS
select …
```

Utilisation d'une vue

Une vue s'utilise comme une table.

Consultation du code d'une vue

```
mysql> SHOW CREATE VIEW nom_vue;
```

Suppression d'une vue

Pour supprimer une vue, on écrira :

```
DROP VIEW nom_vue ;
```

Exemple

Tous les départements contenant au moins un ANALYST (c'est un job) :

```
CREATE or REPLACE VIEW deptAvecAnalyste as
Select distinct nd from emp
where job = 'ANALYST'
order by nd;
```

Les 2 usages des vues

1 : décomposer les requêtes en sous-requêtes

Une vue remplacer n'importe quelle table dans un select.

Les vues permettent de décomposer les requêtes en sous-requêtes en enregistrant les sous-requêtes.

Le select principal joue le rôle de programme principal. Les vues correspondant à des sous-requêtes joue le rôle de fonction (ou de procédure).

- ***Exemple***

Tous les employés travaillant dans un département qui contient au moins un 'ANALYST' (c'est un métier).

On commence par créer la vue correspondant à la requête : « tous les départements contenant au moins un ANALYST (cf. exemple précédent) :

```
CREATE or REPLACE VIEW deptAvecAnalyste as
Select distinct nd from emp
where job = 'ANALYST'
order by nd;
```

Ensuite on traite la requête principale en exploitant la nouvelle vue :

```
Select ne, nom from emp
where nd in (select nd from deptAvecAnalyste);
```

ou :

```
Select distinct e.ne, e.nom from emp e, deptAvecAnalyste d
where e.nd =d.nd;
```

2 : Faciliter et sécuriser l'accès aux données par les utilisateurs

On peut créer des vues et ne donner des droits d'accès que sur ces vues : ça favorise la sécurisation et ça facilite l'accès, les vues étant orientées « usage ».

6. Opérations ensemblistes

Principe général des opérations ensemblistes

Les opérations ensemblistes s'appliquent à deux tables qui ont les mêmes attributs (où plus précisément des attributs de mêmes types).

Les opérations ensemblistes sur ces deux tables consistent soit regrouper tous les tuples (UNION), soit extraire les tuples en commun (INTERSECTION), soit prendre les tuples d'une table qui ne sont pas dans l'autre (DIFFERENCE).

Pour que ces opérations soient significatives, il ne suffit pas que les attributs aient le même type, encore faut-il qu'ils aient la même signification. Dans le cas le plus général, ils seront identiques.

Union

L'union de deux tables ayant des attributs de mêmes types est une table constituée des attributs de la première table et de tous les tuples des deux tables de départ. La syntaxe est la suivante :

```
Select liste_1 d'attributs  from table_1
union
select liste_2 d'attributs  from table 2;
```

- ***Les listes d'attributs***

Elles doivent avoir la même forme dans le where du select principal et dans la projection du select imbriqué : même nombre d'attributs et même type pour les attributs. Habituellement, ce sont les mêmes attributs dans les deux select.

- ***Les tables des select***

Les deux tables de départ (table_1 et table_2) peuvent avoir des attributs différents. Ce sont les tables projetées qui doivent avoir des attributs de mêmes types pour pouvoir être unies.

- ***Suppression des doublons***

Les éventuels doublons sont éliminés automatiquement par l'UNION, comme par toutes les opérations ensemblistes.

- ***Ordre des tuples***

Les tuples apparaissent dans l'ordre du premier select suivi par ceux du deuxième select.

Toutefois, le premier select ne peut pas contenir de ORDER BY

On peut trier le résultat final en mettant un ORDER BY à la fin du deuxième select.

Différence

La différence de deux tables ayant des attributs de mêmes types est une table constituée des attributs de la première table et des tuples de la première table n'appartenant pas à la deuxième :

```
Select liste_1 d'attributs  from table_1
minus
select liste_2 d'attributs  from table 2;
```

Exemple

Tous les numéros des départements vides de la société (c'est-à-dire les numéros des départements dans lesquels aucun employé ne travaille) :

```
Select deptno from dept
minus
select distinct deptno from emp;
```

On soustrait de la table des numéros de tous les départements la table des numéros des départements non vides (ceux dans lesquels travaille au moins un employé).

Intersection

L'intersection de deux tables ayant des attributs de mêmes types est une table constituée des attributs de la première table et des tuples de la première table appartenant aussi à la deuxième :

```
Select liste_1 d'attributs  from table_1
intersect
select liste_2 d'attributs  from table_2;
```

Equivalence entre MINUS, INTERSECT et les requêtes imbriquées

MySQL ne propose que l'opérateur UNION.

Equivalence entre MINUS et NOT IN

tous les numéros des départements vides de la société (c'est-à-dire les numéros des départements dans lesquels aucun employé ne travaille) :

```
Select deptno from dept
minus
select distinct deptno from emp;
```

équivaut à :

```
Select ND from dept
Where ND not in (
        select ND from emp
);
```

Equivalence entre INTERSECT et IN

```
Select listeAtt1 from table1
intersect
Select listeAtt1 from table2
```

équivaut à :

```
Select listeAtt1 from table1
Where listeAtt1  in (
        Select listeAtt1 from table2
);
```

TP N°7 : JOINTURE ARTIFICIELLE, IMBRICATION DE SELECT, OPERATIONS ENSEMBLISTES

Présentation

Bases : « biblio-07 », « employes-07 », « ecoling-07 »

- SELECT : répondre aux questions. Requêtes complexes.

Méthode

Travail à faire

- Dans un fichier texte à votre nom+TP07.txt, écrire les questions et les réponses les unes à la suite des autres.
- Après chaque requête, on met, en commentaire, le nom de l'attribut clé primaire de la table résultat
- Mettez le résultat obtenu dans le fichier (copier-coller).
- Sur papier : faire le graphe de la question.

Méthode de travail

A partir du fichier texte votreNomTP06.txt, faire des copier-coller dans la calculette SQL ou dans utiliser un fichier de test avec un source, et copier coller la bonne requête dans le rapport de TP.

Ordre de projection des attributs

Requête sans fonction de groupe

Attributs de tri, Cle Primaire, Clé Significative, Attributs demandés, Attributs de restriction

Requête avec fonctions de groupe

Attributs de tri, attributs du group by, fonctions de groupe demandées, fonctions de groupe de restriction (du having).

restriction (du having).

Exercice 1 : BD employés et départements

Présentation

On travaillera sur les tables suivantes :

EMPLOYES(<u>NE</u>, NOM, JOB, DATEMB, SAL, COMM, #ND, *NEchef)

- NE numéro de l'employé. **Clé primaire.**
- NOM nom de l'employé.
- JOB intitulé du poste occupé.
- DATEMB date d'embauche.
- SAL salaire de l'employé.
- COMM commission (part de salaire variable).
- #ND n° du département dans lequel travaille l'employé. **Clé étrangère.**
- *NEchef n° d'employé du chef de l'employé. **Clé étrangère réflexive.**

DEPARTEMENTS(<u>ND</u>, NOM, VILLE)

- ND numéro des départements de l'entreprise. **Clé primaire.**
- NOM nom des départements de l'entreprise.
- VILLE nom des villes où sont situés les départements.

Interrogation de la BD

0. Télécharger le script de création de la BD : employes.sql et lancer ce script de création de la BD.
1. Tous les employés travaillant dans un département qui contient au moins un 'ANALYST' (c'est un métier).
   Ecrire 3 versions.
2. Tous les employés ayant le même job que les employés du département 30.
   Ecrire 3 versions.
3. Tous les noms et dates d'embauche des employés embauchés avant BLAKE.
   Ecrire 3 versions.
4. Tous les employés ayant le même chef que ALLEN
   Ecrire 3 versions.
5. Tous les employés n'ayant pas le même chef que ALLEN
   Ecrire 2 versions.
6. Changer le nom de CLARK (le 7782) et passez le à ALLEN.
   Relancer la requête de l'exercice précédent avec jointure artificielle et avec le NOT IN. Que constatez-vous ?
   Repassez le 7782 à CLARK.
7. Tous les employés n'ayant pas de subordonnées.
   Ecrire 3 versions.
8. Tous les départements vides avec leurs noms et villes.
   Ecrire 3 versions.

9. Tous les employés ayant le même job et le même chef que MARTIN.

   Ecrire 3 versions.

10. Tous les employés travaillant à Chicago et ayant le même job qu'ALLEN.

    Ecrire 3 versions.

11. Tous les employés du département RESEARCH embauchés la même année que quelqu'un du département SALES

    Ecrire 3 versions.

12. Tous les employés embauchés avant tous les employés du département 10.

    Ecrire 2 versions.

    Pour vérifier on affichera tous les employés du 10 ordonnés par date d'embauche.

13. Tous les employés qui gagnent plus que la moyenne classés par salaire croissant.

    Ecrire 2 versions.

14. Tous les départements qui contiennent plus d'employés que le nombre moyen d'employés par ville.

    Faire deux versions imbriqués (avec et sans imbrications dans le where)

    Faire plusieurs versions avec des vues.

15. Ecrire un script qui permette de créer l'employé Dupond . Son numéro est le 8000. Il est MANAGER. Sa date d'embauche est le 1 février 2007. Son salaire est 2500. Il n'a pas de commission. Son supérieur hiérarchique est le Président. Il travaille dans le département 30. Turner (7844) et James (7900) passe sous sa responsabilité. Pour ce script, vous n'utiliserez pas de select imbriqués.
16. Dupond et les employés sous sa responsabilité travaille désormais dans le département 40. Ecrivez le script de mise à jour de la base de données. Pour ce script, vous n'utiliserez pas de select imbriqués.
17. Blake (7698) démissionne. Tous les employés sous sa responsabilité passent sous celle de Dupond (8000) et dans son département. Son département est supprimé. Tous les employés de son département passent dans le département de Dupond. Ecrivez le script de mise à jour de la base de données. Dans le script on commencera par lister tous les subordonnés de Blake et tous les employés de son département. Pour ce script, vous n'utiliserez pas de select imbriqués et vous n'utilisez que les numéros de Blake et de Dupond comme valeur « en dur » (7698, 8000).

Exercice 2 : BD biblio

Présentation

On travaillera sur les tables suivantes :

OEUVRES(**NO**, TITRE, AUTEUR)

- **NO** numéro de l'œuvre. **Clé primaire.**
- LIVRE titre de l'oeuvre
- AUTEUR auteur de l'oeuvre

ADHERENTS(**NA**, NOM, VILLE)

- **NA** numéro d'adhérent. **Clé primaire.**
- NOM nom de l'adhérent.
- PRENOM prénom de l'adhérent.
- ADR adresse de l'adhérent.
- TEL téléphone de l'adhérent.

LIVRES (**NL**, EDITEUR, #NO)

- **NL** numéro du livre. **Clé primaire.**
- EDITEUR éditeur du livre
- NO numéro de l'oeuvre. **Clé étrangère.**

EMPRUNTER(#**NL**, **DATEMP**, DATRETMAX, DATRET, #NA)

- **#NL** numéro de livre. **Clé primaire. Clé étrangère.**
- **DATEMP** date d'emprunt du livre. **Clé primaire.**
- DATRETMAX date limite de retour autorisée.
- DATRET date de retour effectif du livre.
- #NA numéro d'adhérent. **Clé étrangère.**

Interrogation de la BD

0. Utiliser le script corrigé de création de la BD : BiblioTP06.txt
1. Combien d'exemplaires du titre : « Narcisse et Goldmund » sont disponibles ?
   On utilisera deux méthodes :

- Méthode 1 qui vérifie si un exemplaire est sorti ou pas.
- Méthode 2 qui soustrait les exemplaires sortis tous les exemplaires.

  Dans la première méthode :

  a) on commencera par afficher tous les exemplaires du livre.

b) puis les exemplaires disponibles

c) Et enfin le nombre d'exemplaires disponibles

2. Pour le titre « Narcisse et Goldmund » afficher le nombre d'exemplaire total, le nombre d'exemplaires disponibles et le nombre d'exemplaires actuellement empruntés, avec le titre.
3. Quelle est la moyenne du nombre de livres empruntés par adhérent.
4. Refaire la question 2 pour tous les titres actuellement sortis au moins une fois.