CONSEILS POUR LES REQUÊTES SQL (VERSION 5.0)

Ces conseils ont pour but de vous aider à trouver une bonne requête SQL en fonction de la question posée. Pour une même question, il est possible d'avoir plusieurs requêtes SQL différentes mais qui fournissent le même résultat. Parmi ces requêtes certaines sont plus rapides que d'autres mais cette problématique n'est pas abordée ici.

1) Représenter le MLD (Modèle Logique de Données) permet de visualiser plus rapidement et plus facilement les informations sur les tables. Notamment, on peut repérer les jointures plus aisément : il suffit de suivre les flèches.

2) Dans une sélection, si les informations à récupérer (colonnes présentent immédiatement après la commande *select*) sont réparties sur plusieurs tables utiliser des jointures (en général sur 2 ou 3 tables).

3) Utiliser des alias sur le nom des tables pour alléger les requêtes.

4) Utiliser la syntaxe *join ... on ...* pour réaliser les jointures au lieu de placer les conditions de jointure dans la clause *where*.

5) Si les conditions de la requête (clause *where*) utilisent des informations réparties sur plusieurs tables :
- utiliser des jointures ;
- utiliser les requêtes imbriquées (en général un ou deux niveaux d'imbrication) : en partie gauche d'un opérateur de comparaison (=, >, etc.) ou de l'opérateur *in* (ou *not in*).

6) Si on demande une liste avec un cumul, utiliser la clause *group by* avec une fonction agrégat (*count*, *sum,* etc.). Si il y a une condition sur le regroupement utiliser la clause *having* (il ne peut y avoir de clause *having* sans clause *group by*).

Attention ! : les colonnes qui ne sont pas spécifiées dans la clause *group by* sont inutilisables (une erreur de syntaxe se produira) sauf si elles sont sélectionnées par une fonction agrégat.

7) Si on demande un classement des réponses, utiliser la clause *order by.*

8) Vérifier si il est nécessaire que les lignes retournées soient uniques (pas de doublon). Dans ce cas utiliser la clause *distinct*.

9) Les instructions *update* et *delete* ne peuvent pas utiliser des jointures, il faut utiliser des requêtes imbriquées lorsque les conditions de ces instructions (spécifiées dans la clause *where*) nécessitent l'utilisation de plusieurs tables.

10) Pour une requête du type : « donner l' (les) élément(s) qui a (ont) la plus/moins grande valeur d'un ensemble » :
- créer une vue qui renvoie les valeurs (avec éventuellement d'autres champs) de l'ensemble concerné ;
- écrire la requête qui renvoie l' (les) élément(s) recherché(s) en utilisant la vue créée précédemment (en utilisant *select max/min* sur un champ bien choisi de la vue).

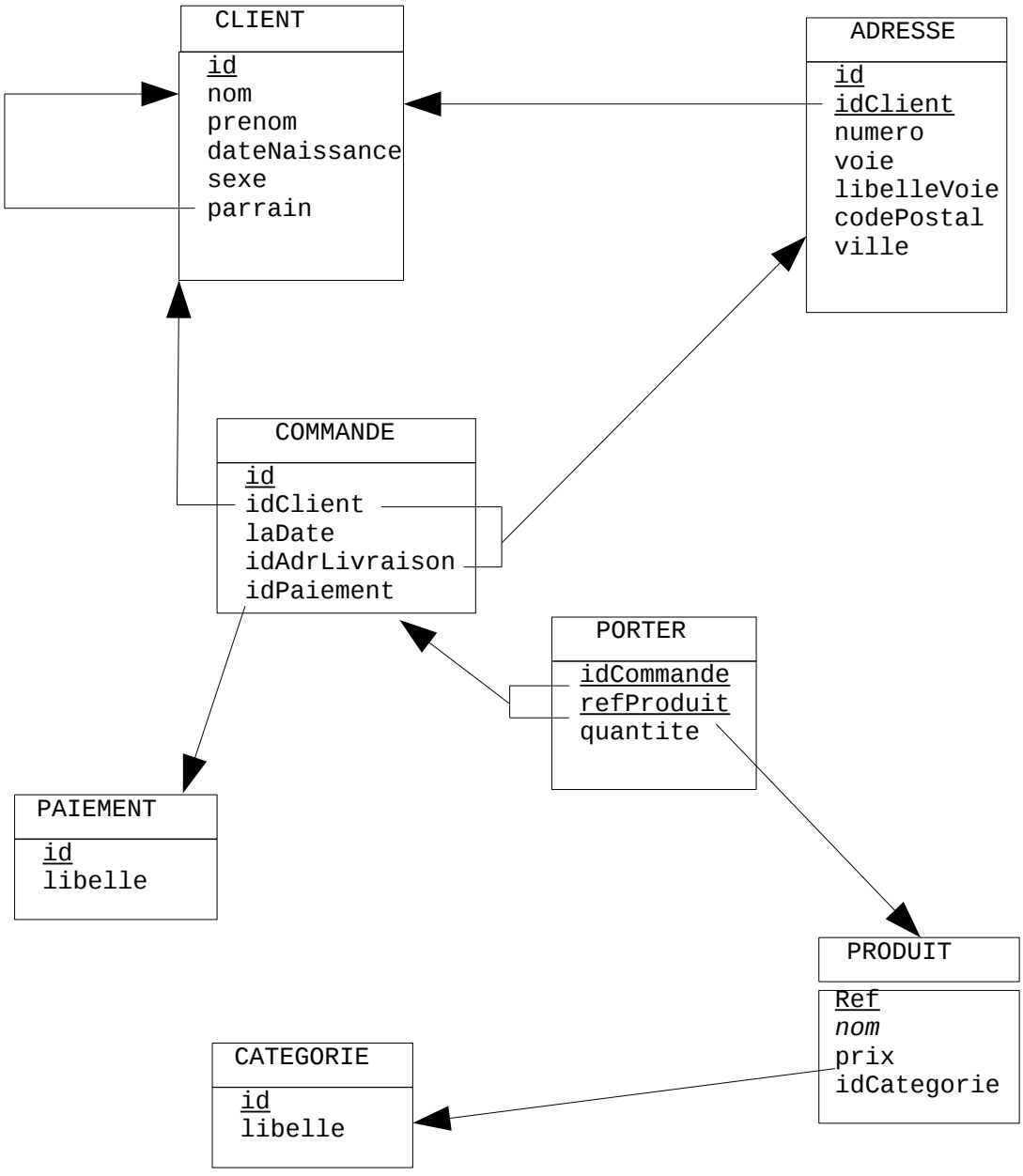
11) Dans le cas d'une requête complexe, il peut être utile de décomposer la requête complexe en utilisant une ou plusieurs vues.

12) Utiliser *grant* et *revoke* pour donner et retirer des permissions sur les objets de la base de données.

13) procédures strockées et déclencheurs

Exemples basés sur le cas « magasin » :

Application du conseil 1 :
Schéma relationnel sous forme graphique (ou modèle logique de données (MLD)) :



Modèle relationnel sous forme textuel :

CLIENT (id, nom, prenom, dateNaissance, sexe, parrain)

ADRESSE (id, idClient, numero, voie, libelleVoie, codePostal, ville)

PAIEMENT (id, libelle)

PORTER (idCommande, refProduit, quantite)

PRODUIT (ref, nom, prix, idCategorie)

CATEGORIE (id, libelle)

Application des conseils 3, 4, 7 et 10 :

```
-- donne les clients qui ont commandé au moins une perceuse
select distinct cl.id, prenom, cl.nom
from client cl
join commande c on (cl.id = c.idClient)
join porter p on (p.idCommande = c.id)
join produit pr on (pr.ref=p.refProduit)
join categorie ca on (ca.id=pr.idCategorie)
where ca.libelle='perceuse'
```

Application du conseil 4 :

```
-- donne les informations sur les produits qui n'ont jamais été commandés
select p.ref, c.libelle, p.prix
from produit p
join categorie c on (p.idCategorie = c.id)
where p.ref not in (select refProduit from porter)
```

Application du conseil 5 :

```
-- donne le nombre de commandes passées par les clients (dont on donnera
l'identifiant, le prénom et le nom)
select cl.id, prenom, nom, count(*) as nb_commandes
from client cl join commande c on (cl.id = c.idClient)
group by cl.id, nom, prenom


-- donne la liste des noms qui sont portés par au moins deux clients
select nom, count(*)
from client
group by nom
having count(*) > 1
```

Application du conseil 6 :

```
-- donne les noms et prénoms des clients classés par ordre alphabétique
– l'odre est d'abord établit sur les noms, en cas de noms identiques le
classement se fait sur les prénoms
select nom, prenom
from client
order by nom, prenom


-- idem mais en utilisant les numéros de colonne au lieu des noms de colonne
select nom, prenom
from client
order by 1, 2


-- classement par orde alphabétique inverse (odre descendant)
select nom, prenom
from client
order by 1 desc, 2 desc
```