

UFR de Mathématiques et Informatique

Licence professionnelle



"Les métiers de l'Internet"

Réf. Regles_MCD_MPD.doc Module BD1
(Partiel et examen)

Date dernière version : Avril 2002 Diffusion
: apprenants

Auteur : Stéphane LAMBERT

Disponible sur :

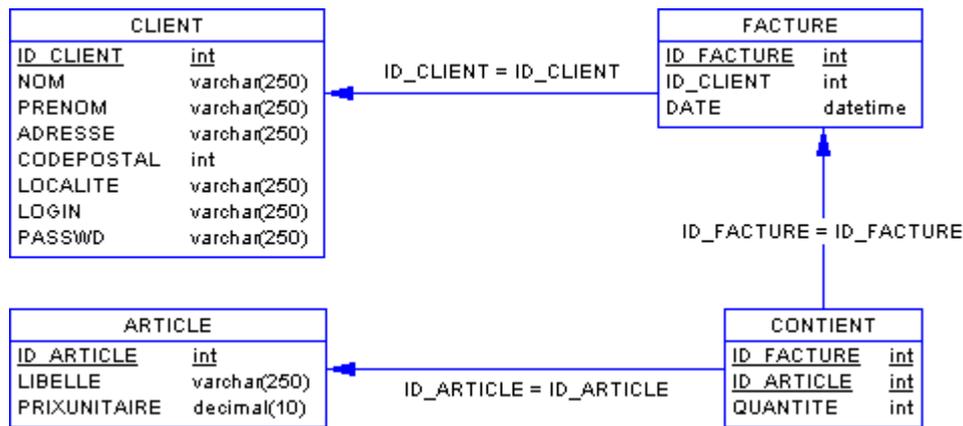
http://www.vediovis.net/licence_pro/

SQL
APPROCHE PRATIQUE

LANGAGE SQL : SYNTAXE D'EXTRACTION DES DONNEES

Le langage SQL (Structured Query Language, ou Langage d'Interrogation Structuré) date de la fin des années 70. Nous allons déjà commencer par voir comment lire les données, et ce que cela signifie réellement.

Appuyons-nous sur ce schéma de base :



Syntaxe de la commande SELECT :

```
SELECT [ALL | DISTINCT] liste de colonnes
FROM Table_1, table_2, ..., Table_n
WHERE .....
GROUP BY .....
HAVING .....
ORDER BY .....
COMPUTE .....
```

Exemples simples :

SELECT NOM, PRENOM FROM CLIENT

renvoie le contenu des colonnes NOM et PRENOM de la table CLIENT.

SELECT NOM as lastname, PRENOM as firstname FROM CLIENT

renverra le même contenu, mais en renommant le nom des colonnes de sorties.

SELECT LIBELLE, PRIXUNITAIRE * 0.85 as prixpromo FROM ARTICLE

affichera le nom des articles et le prix diminué de 15%. Il est donc possible d'effectuer des calculs directement au niveau de la base de donnée, ce qui évitera de faire appel à une partie de script spécialement développée pour l'occasion.

Détail des clauses optionnelles :

La clause **WHERE** exprime les conditions de recherches.

La clause **GROUP BY** groupe les résultats par critères.

La clause **HAVING** permet de restreindre les résultats en imposant une propriété.

La clause **ORDER BY** trie les résultats suivant un critère de qualification.

La clause **COMPUTE** génère des statistiques.

LANGAGE SQL : LA CLAUSE WHERE

Les conditions de recherches peuvent faire référence à des colonnes qui ne sont pas incluse dans la liste de sélection. Elle est très utilisée pour définir ce que l'on appelle des jointures :

```
SELECT CLIENT.NOM, CLIENT.PRENOM, FACTURE.DATE
```

```
FROM CLIENT, FACTURE
```

```
WHERE CLIENT.ID_CLIENT=FACTURE.ID_CLIENT
```

renverra le nom et le prénom des clients ayant des factures, ainsi que la/les date des factures correspondantes. Il est important de faire une jointure sur ID_CLIENT dans la clause Where, pour indiquer au moteur SQL que la liaison entre les deux tables s'effectue sur cette colonne 'commune'. (Nous reviendrons dans un prochain article sur le fonctionnement d'un moteur SQL).

Nous avons plusieurs outils pour les types de condition :

Opérateur de comparaison : =, >, <, >=, <=, <>

```
SELECT LIBELLE, PRIXUNITAIRE
```

```
FROM ARTICLE
```

WHERE PRIXUNITAIRE <>'100'

renverra les articles dont le prix est différent de 100. Il est important de mettre les côtes '' dans la clause Where, car sinon le moteur SQL interprétera 100 comme étant une colonne.

Intervalles : BETWEEN, NOT BETWEEN

```
SELECT LIBELLE, PRIXUNITAIRE
```

```
FROM ARTICLE
```

```
WHERE PRIXUNITAIRE BETWEEN '100' AND '200'
```

renverra les articles dont le prix est compris entre 100 et 200.

Listes : IN, NOT IN

```
SELECT LIBELLE, PRIXUNITAIRE
```

```
FROM ARTICLE
```

```
WHERE PRIXUNITAIRE IN ('100','200','300')
```

renverra les articles dont le prix est soit de 100, soit de 200, soit de 300.

Correspondances de chaînes : LIKE , NOT LIKE

Quelques jokers :

- % joker : toute chaîne de caractère
- - joker simple: un caractère, n'importe lequel
- [] un caractère de l'intervalle ou de l'ensemble spécifié
- [^] un caractère en dehors de l'intervalle ou de l'ensemble spécifié

```
SELECT LIBELLE, PRIXUNITAIRE
```

```
FROM ARTICLE
```

```
WHERE LIBELLE LIKE '%teu%'
```

renverra les articles dont le nom contient 'teu' (pelleteuse, par exemple....)

Valeurs inconnues : IS NULL, IS NOT NULL

```
SELECT LIBELLE, PRIXUNITAIRE
```

```
FROM ARTICLE
```

```
WHERE PRIXUNITAIRE IS NOT NULL
```

renverra les articles dont le prix existera, c'est à dire aura une valeur correspondante dans la table.

ATTENTION !!! En sql, NULL ne signifie pas 0 ,il représente une valeur indéterminée, il signifie 'rien', comme l'ensemble vide en Mathématique.

Combinaisons : AND , OR

```
SELECT LIBELLE, QUANTITE, DATE
```

```
FROM ARTICLE , CONTIENT, FACTURE
```

```
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
```

```
AND CONTIENT.ID_FACTURE=FACTURE.ID_FACTURE
```

```
AND PRIXUNITAIRE NOT BETWEEN '100' AND '200'
```

```
OR PRIXUNITAIRE='150'
```

renverra les articles, leur quantité par facture, et la date, et ceci pour les articles dont le prix n'est pas compris entre '100' et '200' ou est égal à '150'.

LANGAGE SQL : LES AGREGATS

Les fonctions d'agrégats calculent des valeurs à partir des données d'une colonne particulière. Les opérateurs d'agrégat sont : sum (somme), avg (moyenne), max (le plus grand), min (le plus petit), et count (le nombre d'enregistrements retournés).

```
SELECT sum(PRIXUNITAIRE)
```

```
FROM ARTICLE, CONTIENT
```

```
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
```

```
AND CONTIENT.ID_FACTURE='123456'
```

renverra le montant total de la facture '123456'

```
SELECT avg(PRIXUNITAIRE)
```

```
FROM ARTICLE, CONTIENT
```

```
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
```

```
AND CONTIENT.ID_FACTURE='123456'
```

renverra le prix moyens par article de la facture '123456'

```
SELECT max(PRIXUNITAIRE)
FROM ARTICLE, CONTIENT
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
AND CONTIENT.ID_FACTURE='123456'
```

renverra le plus grand prix de la facture '123456'

```
SELECT min(PRIXUNITAIRE)
FROM ARTICLE, CONTIENT
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
AND CONTIENT.ID_FACTURE='123456'
```

renverra le plus petit prix de la facture '123456'

```
SELECT count(PRIXUNITAIRE)
FROM ARTICLE, CONTIENT
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
AND CONTIENT.ID_FACTURE='123456'
```

renverra le nombres d'articles figurants sur la facture '123456'

LANGAGE SQL : LA CLAUSE GROUP BY

La clause **GROUP BY**, utilisée dans une instruction **SELECT**, divise le résultat d'une table en groupe. Il y a actuellement 16 niveaux de sous-groupes possibles.

La clause **GROUP BY** est présente dans les requêtes qui comportent également des fonctions d'agrégats, ces dernières produisant alors une valeur pour chaque groupe, également appelé agrégat vectoriel.

```
SELECT NOM, avg (PRIXUNITAIRE), sum (PRIXUNITAIRE), ID_FACTURE
FROM ARTICLE, CONTIENT, FACTURE, CLIENT
WHERE ARTICLE.ID_ARTICLE=CONTIENT.ID_ARTICLE
AND CONTIENT.ID_FACTURE=FACTURE.ID_FACTURE
AND FACTURE.ID_CLIENT=CLIENT.ID_CLIENT
GROUP BY ID_FACTURE
```

renverra pour chaque facture, par nom de client, le prix moyens ainsi que la quantité des produits présents sur chaque facture.

LANGAGE SQL : LA CLAUSE HAVING

La clause **HAVING** définit les critères de la clause **GROUP BY** de la même façon que la clause **WHERE** dans une instruction **SELECT**. L'avantage de la clause **HAVING** est surtout qu'elle peut comporter des agrégats, ce que ne permet pas la clause **WHERE**.

```
SELECT NOMJOUEUR
FROM JOUEUR, BUT
WHERE JOUEUR.IDJOUEUR=BUT.IDJOUEUR
HAVING count (BUT.IDJOUEUR) > '2'
```

retournera la liste des joueurs ayant marqué plus de deux buts.

Il est possible de combiner les critères avec les opérateurs **AND** et **OR**.

LANGAGE SQL : LA CLAUSE ORDER BY

La clause **ORDER BY** permet de trier les résultats d'une requête par colonnes. Il est possible de demander un tri croissant (**asc**), ou décroissant (**desc**). Par défaut, le tri est croissant (**asc**). Plusieurs colonnes peuvent être indiquées : dans ce cas, le tri sera effectué selon la première colonne indiquée, puis la deuxième, etc.

```
SELECT TAILLE, POIDS, NOMJOUEUR
FROM JOUEUR
ORDER BY TAILLE desc, POIDS desc
```

retournera la taille, le poids et le nom des joueurs participants au tournoi, mais du plus grand au plus petit : en cas de taille équivalente, les plus lourds seront retournés en premier.

Cette clause est très importante : en effet, elle permet à elle seule d'effectuer un classement des données, et donc d'éviter de programmer un traitement supplémentaire généralement très lourd en ressource système et en mémoire.

Il est possible d'utiliser le numéro des colonnes plutôt que leur nom.

```
SELECT POSITION, avg (TAILLE)
FROM JOUEUR
GROUP BY POSITION
ORDER BY 2
```

retournera la taille moyenne des joueurs positions par positions, de la taille moyenne la plus petite à la plus grande. Cela peut se révéler très utile, notamment pour des tableaux de statistiques. Et surtout, les données arrivent déjà triées.

LANGAGE SQL : UPDATE

L'instruction **UPDATE** sert à mettre à jour les données de la table spécifiée. Elle accepte la condition **WHERE** au même titre que l'instruction **SELECT**.

```
UPDATE AGENCE
SET NomAgence='MonAgence'
WHERE idAgence='3'
```

attribuera « MonAgence » comme valeurs de la colonne NomAgence pour les enregistrements de la table AGENCE dont la valeur de idAgence est égale à 3.

LANGAGE SQL : DELETE

L'instruction **DELETE** sert à supprimer des enregistrements de la table spécifiée. Elle accepte la conditions **WHERE** au même titre que l'instruction **SELECT**.

```
DELETE FROM AGENCE
WHERE idAgence='5'
```

Supprimera les enregistrements de la table AGENCE dont la valeur de idAgence est égale à 5.

LANGAGE SQL : INSERT

L'instruction **INSERT** sert à insérer des enregistrements de la table spécifiée.

```
INSERT INTO AGENCE SET
NomAgence='MonAgence', AdresseAgence='Batiment1',
Adresse2Agence='Rue de l'immobilier',
CodePostalAgence='$CodePostalAgence',
```

Créera l'enregistrements correspondant dans la table AGENCE