

02 – Les fonctions et fonctions de groupe

SI3 – Exploitation des données

Introduction

- Le but des fonctions :faire des calculs ou manipuler des données dans une requête
- 2 types de fonctions:
 - Fonctions ou opérations sur des valeurs déjà présentes dans la table
 - Fonctions sur des valeurs issues d'un premier calcul sur la table

Les opérateurs simples

- On peut utiliser les opérateurs standard dans une requête :
 - + - / *
- Les opérations et fonctions simples peuvent se faire sur une ou plusieurs colonnes de tables.
- S'utilisent dans le SELECT ou dans le WHERE ou dans le ORDER BY

Exemple opérations

- Liste des prix TTC en stock pour chaque produit:

```
SELECT (pxUnit*1.2)*unitEnStock  
FROM PRODUIT;
```

- Liste des prix TTC en stock pour chaque produit dont ce prix est supérieur à 20000€.

```
SELECT (pxUnit*1.2)*unitEnStock  
FROM PRODUIT  
WHERE (pxUnit*1.2)*unitEnStock > 20000 ;
```

Exemple de fonctions simples

- MySQL contient déjà une liste de fonctions.
- Différentes d'un SGBD à l'autre
- Consulter la documentation MySQL pour connaître l'utilisation de ces fonctions:
<http://dev.mysql.com/doc/refman/5.0/fr/functions.html>
- S'utilisent dans le SELECT ou dans le WHERE ou dans ORDER BY

Catégories de fonctions simples

- Les fonctions de contrôle
- Fonctions de chaînes de caractères
- Fonctions numériques
- Fonctions de dates et d'heures
- Recherche en texte intégral (Full-text)
- Fonctions de transtypage
- Plus d'autres fonctions

Exemple d'utilisation de fonctions simples

- Donner l'âge actuel des employés:

```
SELECT year(curdate()) - year(ddn)  
FROM Personnel ;
```

- Les fonctions prennent 0 ou plusieurs paramètres.
- Pour appeler une fonction à 0 paramètre, on utilise quand même les parenthèses :
curdate()

Les fonctions de groupes

- Précédemment, le calcul s'effectuait sur une seule ligne de la ou des tables.
- But des fonctions de groupes -> pouvoir effectuer des calculs à la verticale sur des colonnes.
- Ne peuvent pas s'utiliser dans un WHERE.

Les fonctions de groupes


- AVG: effectue la moyenne des valeurs
- SUM: effectue la somme des valeurs
- MIN: retourne la valeur minimale
- MAX: retourne la valeur maximale
- COUNT:
 - COUNT(unChamp) : retourne le nombre de valeurs NON nulle
 - COUNT(*) : retourne le nombre de valeur même nulles
 - COUNT(DISTINCT....): retourne le nombre de valeur distinct non nulles

Les fonctions de groupes (suite)

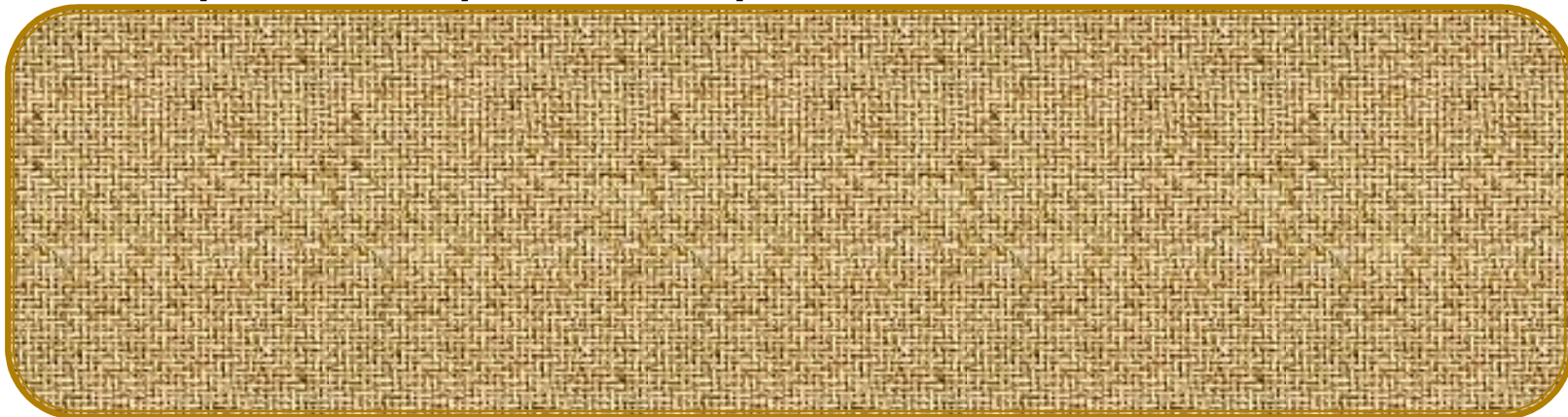
- STDDEV: retourne la déviation standard (racine carrée de la variance)
- VARIANCE: retourne la variance
- GROUP_CONCAT: retourne la concaténation des valeurs
- BIT_AND: retourne la combinaison AND de toutes les valeurs
- BIT_OR: retourne la combinaison OR de toutes les valeurs
- BIT_XOR: retourne la combinaison XOR de toutes les valeurs

Exemple de fonctions de groupe

- Donner le prix moyen des prod du rayon 2:



- Nom des produits dont le prix est supérieur au prix moyen des produits:



Pause Exercices

- Nom du produit le moins cher du rayon Boucherie
- Montant total du passage en caisse 327 (Attention à la qté)
- Nombre de produits différents dans le passage en caisse 32.
- Nombre de produits dans le même rayon que le rayon du Lapin.

Le groupement

- La clause GROUP BY (après le WHERE) permet de regrouper des lignes sur des valeurs identiques afin d'effectuer des calculs.
- Démonstration : Nombre de produits par rayon:

```
SELECT idRayon , COUNT(*) as nb  
FROM Produit  
GROUP BY idRayon ;
```

Explications GROUP BY -> COUNT

```
SELECT idRayon , COUNT(*) as nb  
FROM Produit  
GROUP BY idRayon
```



idProduit	idRayon	Nom	pxUnit	UnitEnS
P1	R1	Lapin	7	28
P2	R1	Poulet	4	33
P3	R2	Coca Cola	2	578
P4	R2	Barbecue	12	345
P5	R2	Olivier	8	23
P6	R2	Chaise	13	456
P7	R2	Frites	5	1234
P8	R3	Salade	1	45

idRayon	nb
R1	2
R2	5
R3	1

Exemple GROUP BY -> SUM

- Nombre d'unités en stock par rayon:

```
SELECT idRayon , SUM(unitEnStock) as nbStock  
FROM Produit  
GROUP BY idRayon ;
```

Explications GROUP BY -> SUM

```
SELECT idRayon , SUM (unitEnStock) as nbStock  
FROM Produit  
GROUP BY idRayon
```



idProduit	idRayon	Nom	pxUnit	UnitEnS
P1	R1	Lapin	7	28
P2	R1	Poulet	4	33
P3	R2	Coca Cola	2	578
P4	R2	Barbecue	12	345
P5	R2	Olivier	8	23
P6	R2	Chaise	13	456
P7	R2	Frites	5	1234
P8	R3	Salade	1	45

idRayon	nbStock
R1	61
R2	2636
R3	45

Le SELECT avec un GROUP BY

- Obligation de retrouver les champs du SELECT dans le Group By (logique!!!):
 - Donner le nom des rayons ainsi que le nombre de produits par rayon:

```
SELECT Rayon.nom , COUNT(*) as nbStock  
FROM Produit , Rayon  
WHERE Produit.idRayon=Rayon.idRayon  
GROUP BY idRayon , Rayon.nom;
```

- Possibilité d'avoir plusieurs fonctions dans le SELECT

Sélection de groupement (HAVING)

- La sélection de lignes se fait habituellement dans le WHERE
- HAVING permet de sélectionner des lignes qui ont déjà été agrégées.
- Exemple : Rayons qui comportent plus de 15 produits différents:

```
SELECT idRayon  
FROM Produit  
GROUP BY idRayon  
HAVING COUNT(*) > 15 ;
```

Fonctionnement HAVING



```
SELECT idRayon  
FROM Produit  
GROUP BY idRayon  
HAVING COUNT(*) > 3
```

idProduit	idRayon	Nom	pxUnit	UnitEnS
P1	R1	Lapin	7	28
P2	R1	Poulet	4	33
P3	R2	Coca Cola	2	578
P4	R2	Barbecue	12	345
P5	R2	Olivier	8	23
P6	R2	Chaise	13	456
P7	R2	Frites	5	1234
P8	R3	Salade	1	45

idRayon	nb
R1	2
R2	5
R3	1

Rappels HAVING

- HAVING fonctionne forcément avec un GROUP BY.
- La clause WHERE peut filtrer les lignes avant.
- La clause HAVING filtre après le traitement du GROUP BY.

HAVING + Ss-Requête

- Donner les rayons qui contiennent plus de produits que le rayon 2.

```
SELECT idRayon  
FROM Produit  
GROUP BY idRayon  
HAVING COUNT(*) >
```

Renvoie le nombre de produits dans le rayon 2



25

Sélectionne les rayons en contenant plus que 25

Exercices

- Liste des passages en caisse qui comportent plus de 15 produits du rayon Liquide.

```
Select idPassageCaisse  
From Rayon as r, Produit as p , Conteneur as c  
Where r.nom = "Liquide"  
And r.idRayon = p.idRayon  
And c.idProduit = p.idProduit  
Group by idPassageCaisse  
Having count(*) > 15 ;
```

Exercices

- Donner le prix total de chaque passage en caisse durant l'année 2005 trié par ordre de prix croissant

```
SELECT C.idPassageCaisse , sum( pxUnit * qte )  
FROM PassageCaisse as PC , Produit as P , Contenir as C  
WHERE year(datePassage) = 2005  
AND PC.idPassageCaisse = C.idPassageCaisse  
AND P.idProduit = C.idProduit  
GROUP BY C.idPassageCaisse  
ORDER BY 2 ;
```

Exercices

- Nombre de passage en caisse par caisse durant le mois de juillet 2005. On veut également la date de production de la caisse.

```
select PC.idCaisse , count(*) , dateProd  
from PassageCaisse as PC , Caisse as C  
where PC.idCaisse = C.idCaisse  
and
```

```
    (year(datePassage),month(datePassage))=(2005,7)  
group by PC.idCaisse ;
```


Exercices

- Donner le idPassageCaisse du passage en caisse le + onéreux de l'année 2005

```
SELECT C.idPassageCaisse , sum( pxUnit * qte ) as PxTotal
FROM PassageCaisse as PC , Produit as P , Contenir as C
WHERE year(datePassage) = 2005
AND PC.idPassageCaisse = C.idPassageCaisse
AND P.idProduit = C.idProduit
GROUP BY C.idPassageCaisse
HAVING pxTotal >= ALL( SELECT sum( pxUnit * qte )
                        FROM PassageCaisse as PC , Produit as P , Contenir as C
                        WHERE year(datePassage) = 2005
                        AND PC.idPassageCaisse = C.idPassageCaisse
                        AND P.idProduit = C.idProduit
                        GROUP BY C.idPassageCaisse );
```

FIN

- Questions

