



## Chapitre 6

# SQL

## Data Query Language

### *Requêtes avancées*

# Plan

## Partie 1 : Sous-requêtes

- Sous-requête qui retourne un seul élément
- Sous-requête qui retourne un ensemble d'éléments

## Partie 2 : Groupements

- Fonction de groupe
- Sans clause GROUP BY
- GROUP BY sur une colonne
- GROUP BY sur plusieurs colonnes
- HAVING

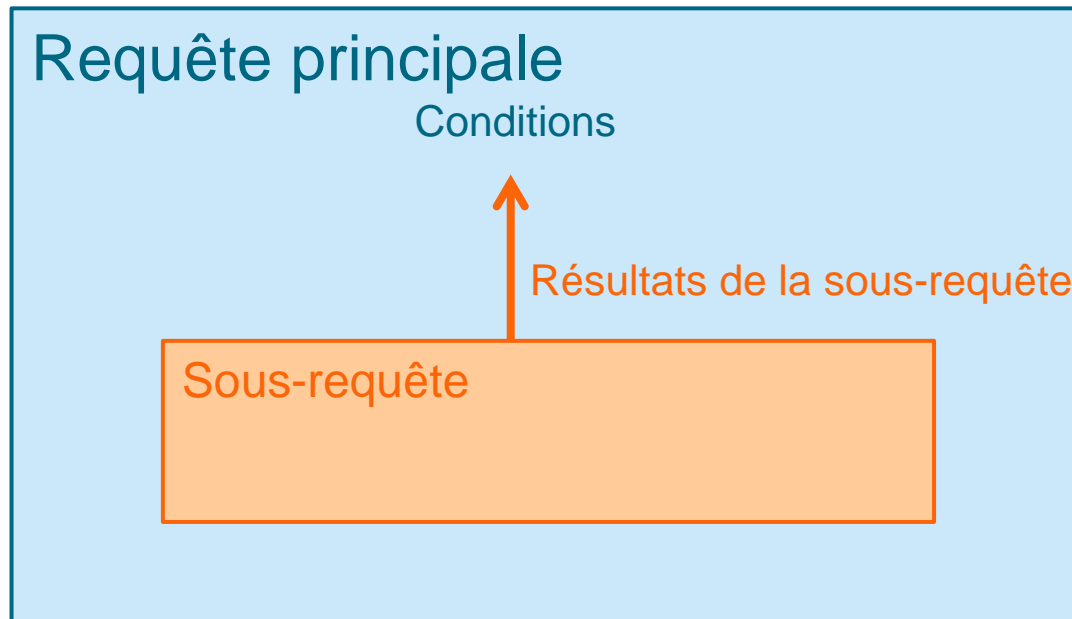
## Partie 3 : Jointures

- Produit cartésien de deux tables
- Jointure de deux tables
- Jointure de plus de deux tables



# Partie 1 : Sous-requêtes

# Sous-requête qui retourne un seul élément



# Sous-requête qui retourne un seul élément

```
SELECT <liste d'expressions>  
FROM   <nom_table1>  
WHERE  <expression1> <opérateur>  
      ( SELECT <expression2>  
        FROM <nom_table2> );
```

*Exemple*

```
select libelle  
from   produit  
where  prix = ← 36 euros  
      (select prix  
       from   produit  
       where  reference = 174);
```

# Sous-requête qui retourne un seul élément

## Opérateurs de comparaison

Opérateur	Signification
=	Égal
>	Plus grand
>=	Plus grand ou égal
<	Plus petit
<=	Plus petit ou égal
<>	Différent

# Sous-requête qui retourne un seul élément

## Exemple

```
select  titre
from    livre
where   nombre_pages > 50 pages
```

```
(select  nombre_pages
from    livre
where   numero = 8563) ;
```

```
and    prix < 37 euros
```

```
(select  prix
from    livre
where   numero = 5569) ;
```

livre

numero

titre

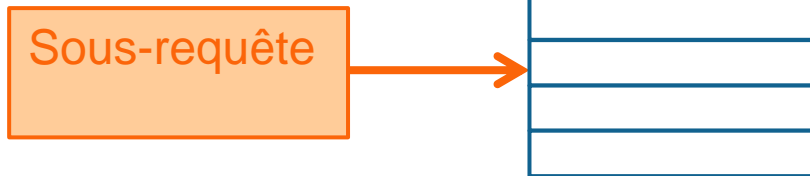
nombre\_pages

date\_parution

prix



# Sous-requête qui retourne un ensemble d'éléments



Attention, si la sous-requête retourne plusieurs éléments, les opérateurs de comparaison sont différents



Opérateur	Signification
IN	Égal à n'importe quel élément de la liste
ANY	Doit être précédé de =, <>, >, <, <=, >=. Doit être vrai pour au moins une valeur de la liste
ALL	Doit être précédé de =, <>, >, <, <=, >=. Doit être vrai pour toutes les valeurs de la liste

# Sous-requête qui retourne un ensemble d'éléments


## Exemple

```
select  titre
from    livre
where   prix > any      ← 90, 60 et 42 euros
      (select prix
        from  livre
        where numero in (1236, 5469, 7898) );
```

- 🔍 Sélectionne les livres qui coûtent plus qu'au moins un livre de la liste  
⇒ qui coûtent **plus de 42 euros** !

# Sous-requête qui retourne un ensemble d'éléments

## Exemple

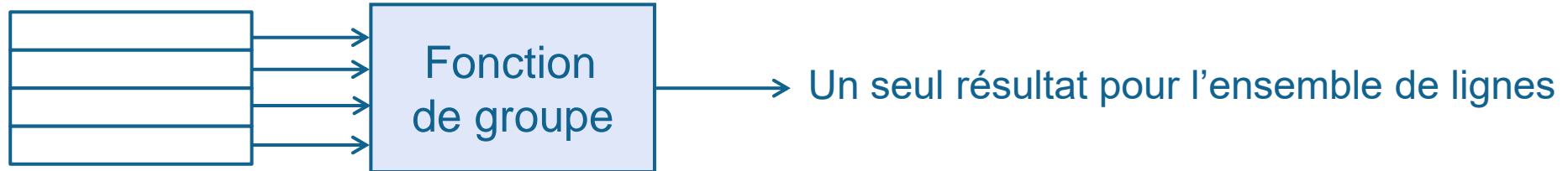
```
select  titre
from    livre
where   prix > all  90, 60 et 42 euros
      (select  prix
       from    livre
       where   numero in (1236, 5469, 7898) );
```

- 👉 Sélectionne les livres qui coûtent plus que tous les livres de la liste  
⇒ qui coûtent **plus de 90 euros** !



# Partie 2 : Groupements

# Fonctions de groupe



## Quelques fonctions de groupe

- **AVG(<nom\_colonne>)** : calcule la moyenne des valeurs de la colonne (*de type numérique*)
- **SUM(<nom\_colonne>)** : calcule la somme des valeurs de la colonne (*de type numérique*)
- **MAX(<nom\_colonne>)** : retourne la plus grande des valeurs de la colonne
- **MIN(<nom\_colonne>)** : retourne la plus petite des valeurs de la colonne
- **COUNT (\*)** : compte le nombre de lignes dans le groupe
- **COUNT(<nom\_colonne>)** : compte le nombre de valeurs non nulles dans la colonne
- **COUNT( DISTINCT <nom\_colonne>)** : compte le nombre de valeurs différentes dans la colonne

# Sans clause GROUP BY

Sans clause GROUP BY, toutes les lignes de la table forment un seul groupe  
⇒ Une seule ligne de résultat

Exemple

livre
<u>numero</u>
titre
nombre_pages
date_parution
prix

```
select count(*),  
       min(numero),  
       max(date_parution),  
       sum(nombre_pages),  
       avg (prix),  
       count(distinct prix)  
from livre ;
```

- ← Nombre de lignes dans la table livre
- ← Le plus petit numéro
- ← La date de parution la plus récente
- ← La somme des pages de tous les livres
- ← Le prix moyen des livres
- ← Le nombre de prix différents

⇒ Une seule ligne à l'affichage comprenant ces 6 valeurs

# GROUP BY sur une colonne

```
SELECT      <nom_colonne> | <fonction_groupe>, ...  
FROM        <nom_table>  
[ WHERE     <condition(s)> ]  
GROUP BY  <nom_colonne>  
[ ORDER BY  <nom_colonne> | <fonction_groupe> [ASC|DESC], ... ] ;
```

GROUP BY permet d'effectuer des regroupements sur base d'une colonne : les lignes partageant une même valeur pour la colonne sont placées dans un même groupe ; des statistiques peuvent ensuite être calculées sur chaque groupe.

*Exemple*

```
select count(*),  
       max(date_naissance),  
from   etudiant  
group by section ;
```

Ce groupement produira une ligne de résultat pour chaque groupe.

*Les étudiants d'une même section sont rassemblés en un même groupe ➡  
⇒ Une ligne de résultat par section*



# GROUP BY sur plusieurs colonnes

```
SELECT      <nom_colonne> | <fonction_groupe>, ...  
FROM        <nom_table>  
[ WHERE     <condition(s)> ]  
GROUP BY  <nom_colonne>, ...  
[ ORDER BY  <nom_colonne> | <fonction_groupe> [ASC|DESC], ... ] ;
```

GROUP BY sur plusieurs colonnes permet de créer des sous-groupes.  
Une ligne de résultat sera produite pour chaque sous-groupe.

Exemple

```
select count(*),  
       max(date_naissance),  
from   etudiant  
group by section, bloc ;
```



Les étudiants d'une même section et d'un même bloc sont rassemblés en un même sous-groupe.  
⇒ Une ligne de résultat par sous-groupe : DA1, DA2, DA3, TI1, TI2, TI3, ...





# GROUP BY sur plusieurs colonnes

```
SELECT      <nom_colonne> | <fonction_groupe>, ...  
FROM        <nom_table>  
[ WHERE     <condition(s)> ]  
GROUP BY    <nom_colonne>, ...  
[ ORDER BY  <nom_colonne> | <fonction_groupe> [ASC|DESC], ... ] ;
```

Attention, les seules colonnes qui peuvent apparaître dans la clause SELECT qui ne sont pas des fonctions de groupe sont les colonnes du GROUP BY



Exemple

```
select section, bloc  
       count(*),  
       max(date_naissance),  
from   etudiant  
group by section, bloc ;
```



Contre\_exemple

```
select matricule, section, bloc  
       count(*),  
       max(date_naissance),  
from   etudiant  
group by section, bloc ;
```



# HAVING

La clause HAVING permet d'ignorer, d'écarter les groupes qui ne satisfont pas certaines conditions ⇒ il faut écrire des conditions portant sur des groupes (*par exemple, en utilisant des fonctions de groupe*)

```
SELECT      <nom_colonne> | <fonction_groupe>, ...  
FROM        <nom_table>  
[ WHERE     <condition(s)> ]  
GROUP BY    <nom_colonne>, ...  
HAVING      <condition(s)_groupe>  
[ ORDER BY  <nom_colonne> | <fonction_groupe> [ASC|DESC], ... ] ;
```

*Exemple*

```
select  section, bloc, count(*), max(date_naissance)  
from    etudiant  
group by section, bloc  
having  count(*) > 20 ;
```



Les sous-groupes (DA1, DA2, DA3, TI1, TI2, TI3, ...) qui ne comptent pas plus de 20 étudiants sont ignorés : aucune statistique ne sera calculée pour ces groupes.



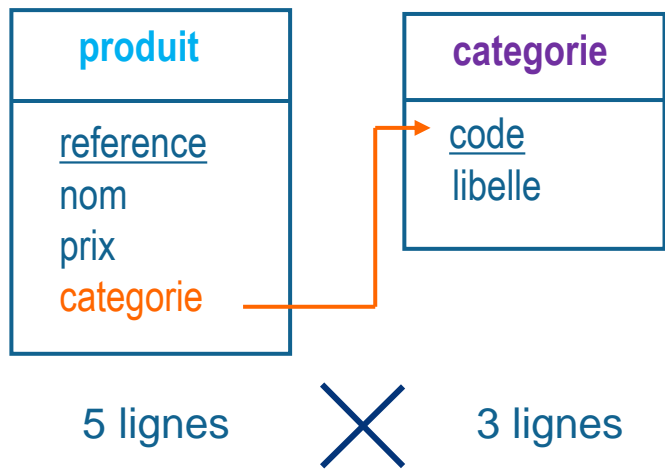
# Partie 3 : Jointures

# Produit cartésien de deux tables

Dans une même requête, il est possible de sélectionner des colonnes provenant de plusieurs tables

Attention, sans condition de jointure  $\Rightarrow$  produit cartésien 

Exemple



$\Rightarrow$  Produit cartésien :  $5 \times 3 = 15$  lignes

```
select produit.nom, produit.categorie,  
       categorie.code, categorie.libelle  
from produit, categorie ;
```

nom	categorie	code	libelle
Poulet	Vi	Fr	Fruit
Carotte	Lg	Fr	Fruit
Orange	Fr	Fr	Fruit
Poireau	Lg	Fr	Fruit
Pomme	Fr	Fr	Fruit
Poulet	Vi	Lg	Légume
Carotte	Lg	Lg	Légume
Orange	Fr	Lg	Légume
Poireau	Lg	Lg	Légume
Pomme	Fr	Lg	Légume
Poulet	Vi	Vi	Viande
Carotte	Lg	Vi	Viande
Orange	Fr	Vi	Viande
Poireau	Lg	Vi	Viande
Pomme	Fr	Vi	Viande

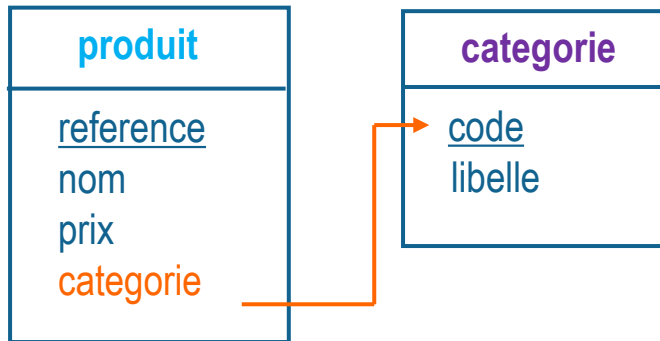
Ben, ça sert pas à grand-chose !



# Jointure de deux tables

Recherche à effectuer

*Sélectionner le nom de chaque produit ainsi que le libellé de sa catégorie*



```
select produit.nom, produit.categorie,  
       categorie.code, categorie.libelle  
from   produit, categorie  
where  produit.categorie = categorie.code ;
```

Condition de jointure

nom	categorie	code	libelle
Poulet	Vi	Fr	Fruit
Carotte	Lg	Fr	Fruit
Orange	Fr	Fr	Fruit
Poireau	Lg	Fr	Fruit
Pomme	Fr	Fr	Fruit
Poulet	Vi	Lg	Légume
Carotte	Lg	Lg	Légume
Orange	Fr	Lg	Légume
Poireau	Lg	Lg	Légume
Pomme	Fr	Lg	Légume
Poulet	Vi	Vi	Viande
Carotte	Lg	Vi	Viande
Orange	Fr	Vi	Viande
Poireau	Lg	Vi	Viande
Pomme	Fr	Vi	Viande

Clé étrangère  
de la table produit

=


Clé primaire  
de la table categorie

# Jointure de deux tables

```
select produit.nom, produit.categorie,  
       categorie.code, categorie.libelle  
from   produit, categorie  
where  produit.categorie = categorie.code ;
```

```
select produit.nom, categorie.libelle  
from   produit, categorie  
where  produit.categorie = categorie.code ;
```

Pas obligatoires dans le select



nom	categorie	code	libelle
Poulet	Vi	Vi	Viande
Carotte	Lg	Lg	Légume
Orange	Fr	Fr	Fruit
Poireau	Lg	Lg	Légume
Pomme	Fr	Fr	Fruit

nom	libelle
Poulet	Viande
Carotte	Légume
Orange	Fruit
Poireau	Légume
Pomme	Fruit

# Jointure de deux tables

Il existe deux syntaxes pour les jointures

Soit

```
SELECT <table1.colonne> | <table2.colonne>, ...  
FROM <table1>, <table2>  
WHERE <table1.colonne1> = <table2.colonne2>  
[AND <condition(s)>];
```

Exemple

```
select produit.nom, categorie.libelle  
from produit, categorie  
where produit.categorie = categorie.code;
```

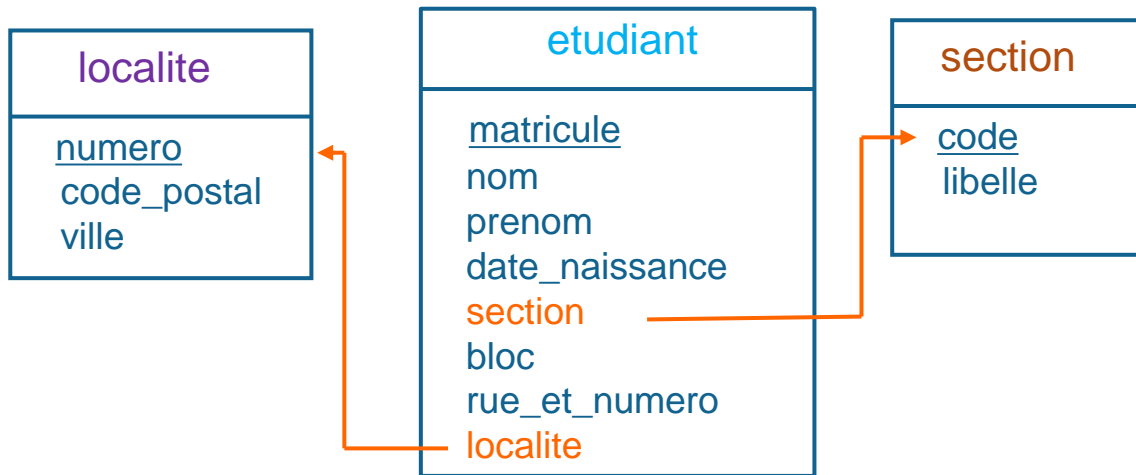
Soit

```
SELECT <table1.colonne> | <table2.colonne>, ...  
FROM <table1> INNER JOIN <table2>  
ON <table1.colonne1> = <table2.colonne2>  
[WHERE <condition(s)>];
```

Exemple

```
select produit.nom, categorie.libelle  
from produit inner join categorie  
on produit.categorie = categorie.code;
```

# Jointure de plus de deux tables



soit

```
select etudiant.prenom, etudiant.nom, etudiant.bloc, section.libelle,
       etudiant.rue_et_numero, localite.ville
from   etudiant, localite, section
where  etudiant.localite = localite.numero and etudiant.section = section.code ;
```

soit

```
select etudiant.prenom, etudiant.nom, etudiant.bloc, section.libelle,
       etudiant.rue_et_numero, localite.ville
from   etudiant inner join localite on etudiant.localite = localite.numero
       inner join section on etudiant.section = section.code ;
```



# Jointure de plus de deux tables

Jointure entre **N** tables



**N-1** conditions de jointures

