SQL Avancé
Le groupement en SQL

1 Regroupements simples

Le regroupement permet de réaliser des calculs (à l'aide des fonctions ensemblistes) sur des ensembles de tuples qui ont été regroupés dans différents paquets. Ces paquets sont établis en fonction de la valeur du ou des attributs du critère de regroupement. On réalise un regroupement à l'aide du mot clé `GROUP BY` suivi du ou des attributs de regroupement.

Le `GROUP BY` réalise un `DISTINCT` sur les attributs du groupement, puis il applique la fonction qui est dans le `SELECT` à chacun des groupes.

Requête 1 : Pour chaque ville, l'âge moyen des clients

CLIENTS(*codeClient, nomClient, prenomClient, ageClient, villeClient*)

groupe Nîmes	A100	Stiké	Sophie	22	Nimes
	A200	Triser	Jessica	18	Nimes
groupe Béziers	A300	Tare	Guy	28	Béziers
groupe Montpellier	A400	Nemard	Jean	31	Montpellier
	A500	Palleja	Xavier	24	Montpellier
	A600	Ouzy	Jacques	38	Montpellier

Résultat

Nimes	20
Béziers	28
Montpellier	31

```
SELECT villeClient, AVG(ageClient)
FROM Clients
GROUP BY villeClient ;
```

- Si une requête possède dans son `SELECT` un attribut **ET** une fonction ensembliste, elle doit alors posséder obligatoirement un `GROUP BY`.
- Une requête qui possède un `GROUP BY` doit obligatoirement utiliser une fonction (dans le `SELECT` ou bien dans le `ORDER BY`). Sinon le `GROUP BY` est inutile et peut être remplacé par un `DISTINCT`.
- **Tous les attributs qui se trouvent dans le `SELECT` doivent également être dans le `GROUP BY` …**
- … mais il se peut que des attributs qui sont dans le `GROUP BY` ne soient pas dans le `SELECT`.
- Qu'il y ait un `GROUP BY` ou non dans la requête, on rappelle qu'il est interdit d'utiliser des fonctions d'ensemble dans un `WHERE` (par contre il est possible de les utiliser dans une requête imbriquée du `WHERE`).

Requête 2 : Le nombre de clients par ville.

```
SELECT villeClient, COUNT(*)
FROM Clients
GROUP BY villeClient;
```

Requête 3 : L'identifiant des clients qui ont passé une ou plusieurs commandes. Les clients doivent être classés par rapport au nombre de commandes passées.

```
SELECT idClient
FROM Commandes
GROUP BY idClient
ORDER BY COUNT(*);
```

Requête 4 : Pour chacun des clients qui ont passé des commandes, le nom et le prénom du client ainsi que le nombre de commandes.

```
SELECT nomClient, prenomClient, COUNT(*)
FROM Clients cl
     JOIN Commandes co ON cl.idClient = co.idClient
GROUP BY c.idClient, nomClient, prenomClient;
```

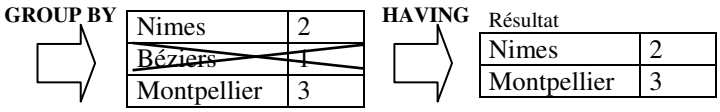
2 La clause HAVING

La clause `HAVING` permet de réaliser un filtre qui applique une restriction sur le résultat trouvé par le `GROUP BY`. Elle peut être utilisée avec toutes les fonctions d'ensembles (`MIN`, `MAX`, `COUNT`, `SUM`, `AVG`). Tout comme le `WHERE` exclut des lignes d'une requête, la clause `HAVING` exclut des groupes.

Requête 5 : Les villes pour lesquelles on a au moins deux clients.

CLIENTS(*codeClient, nomClient, prenomClient, ageClient, villeClient)*

A100	Stiké	Sophie	22	Nimes
A200	Triser	Jessica	18	Nimes
A300	Tare	Guy	28	Béziers
A400	Nemard	Jean	31	Montpellier
A500	Palleja	Xavier	24	Montpellier
A600	Ouzy	Jacques	38	Montpellier




```
SELECT villeClient
FROM Clients
GROUP BY villeClient
HAVING COUNT(*) >= 2 ;
```

- Un `HAVING` doit être suivi d'une fonction ensembliste (`MIN`, `MAX`, `COUNT` ...) sinon il sera interprété comme un `WHERE`.
- … et inversement il est interdit d'utiliser une fonction ensembliste dans un `WHERE`.
- Dans la norme SQL, un `HAVING` sans `GROUP BY` est interprété comme un `WHERE`.
  Toutefois, en pratique, beaucoup de SGBD refusent ce type de requêtes.

Requête 6 : L'identifiant des clients qui ont passé le même nombre de commandes que le client dont l'identifiant est 'C1'.

```
SELECT idClient
FROM Commandes
GROUP BY idClient
HAVING COUNT(*) = (SELECT COUNT(*)
                   FROM Commandes
                   WHERE idClient = 'C1');
```