# La clause GROUP BY

## 1. La clause GROUP BY

La clause BROUP BY est nécessaire dès que l'on utilise des fonctions de calculs statistiques avec des données brutes. Cette clause groupe les lignes sélectionnées en se basant sur la valeur de colonnes spécifiées pour chaque ligne et renvoie une seule ligne par groupe.
On peut la comparer à une opération de découpage de sous ensemble un peut à la manière des "niveaux de rupture" lorsque l'on réalise des états imprimés.

Cherchons à compter le nombre de chambre par étage de notre hôtel :

```
SELECT COUNT(CHB_ID) AS NOMBRE, CHB_ETAGE
FROM   T_CHAMBRE ;
```

La requête, telle que présentée ci dessus n'est pas calculable. En effet comment décider si le comptage doit se faire pour chaque chambre ou pour un groupe de chambre et notamment les groupes formés par chacun des étages ?

Si la requête se faisait pour chaque chambre, le résultat serait :

```
NOMBRE     CHB_ETAGE
---------- ---------
1          RDC
1          RDC
1          RDC
1          RDC
1          1er
1          1er
1          1er
1          1er
1          1er
...
```

En revanche, un regroupement par étage, donne tout son sens à la requête :

```
NOMBRE     CHB_ETAGE
---------- ---------
8          1er
8          2e
4          RDC
```

Pour réaliser un tel regroupement il faut introduire une clause de groupage dans la requête. Pour cela SQL fournit le clause GROUP BY :

```
SELECT COUNT(*) AS NOMBRE, CHB_ETAGE
FROM   T_CHAMBRE
GROUP  BY CHB_ETAGE;
```

**NOTA** :

- La présence de la clause GROUP BY est nécessaire dès que la clause de sélection, ou le filtre WHERE, ou encore les jointures comportent simultanément des calculs d'agrégation et la présence de colonnes de table hors de calculs d'agrégation.
- De plus, outes les colonnes représentées hors des calculs d'agrégation doivent figurer dans la clause GROUP BY.

La plupart du temps, le moteur de requête vous avertira d'un probable incohérence de calcul des agrégats à l'aide d'un message d'erreur (avant exécution de la requête), du genre : "*La colonne ... est incorrecte dans la liste de sélection parce qu'elle n'est pas contenue dans une fonction d'agrégation et qu'il n'y a pas de clause GROUP BY*" (SQL Server).
Cherchons maintenant à compter le couchage de chaque étage :

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE, CHB_ETAGE
FROM   T_CHAMBRE
GROUP  BY CHB_ETAGE;
```

| NOMBRE | CHB_ETAGE |
|---|---|
| 23 | 1er |
| 22 | 2e |
| 9 | RDC |

Un peu plus compliqué, cherchons à savoir quel a été le nombre de nuitées pour chaque chambre au cours de l'année 1999 (une nuitée étant une personne passant une nuit dans une chambre. Si deux personnes occupent la même chambre cela fait deux nuitées) :

```
SELECT SUM(CHB_PLN_CLI_NB_PERS), C.CHB_ID
FROM   T_CHAMBRE C
     JOIN TJ_CHB_PLN_CLI P
          ON C.CHB_ID = P.CHB_ID
WHERE  PLN_JOUR BETWEEN '1999-01-01' AND '1999-12-31'
GROUP  BY C.CHB_ID;
```

| PERSONNE | CHB_ID |
|---|---|
| 558 | 1 |
| 385 | 2 |
| 322 | 3 |
| 367 | 4 |
| 445 | 5 |
| 720 | 6 |
| 390 | 7 |
| 456 | 8 |
| 362 | 9 |
| 364 | 10 |
| 529 | 11 |
| 468 | 12 |
| 347 | 13 |
| 360 | 14 |
| 484 | 15 |
| 720 | 16 |
| 491 | 17 |
| 372 | 18 |
| 504 | 19 |
| 369 | 20 |

Partant de là, nous pouvons rechercher le taux d'occupation de chaque chambre dans cette période.
Le nombre maximal de nuitées étant le nombre de couchage de chaque chambre multiplié par toutes les dates de l'année, ce calcul s'obtient par :

```
SELECT SUM(CHB_COUCHAGE) AS MAX_OCCUPATION, CHB_ID
FROM   T_CHAMBRE C
     CROSS JOIN T_PLANNING P
WHERE  PLN_JOUR BETWEEN '1999-01-01' AND '1999-12-31'
GROUP  BY C.CHB_ID;
```

| MAX_OCCUPATION | CHB_ID |
|---|---|
| 1095 | 1 |
| 730 | 2 |
| 730 | 3 |
| 730 | 4 |
| 1095 | 5 |
| 1825 | 6 |
| 730 | 7 |
| 1095 | 8 |
| 730 | 9 |
| 730 | 10 |
| 1095 | 11 |
| 1095 | 12 |
| 730 | 13 |
| 730 | 14 |
| 1095 | 15 |
| 1825 | 16 |
| 1095 | 17 |
| 730 | 18 |
| 1095 | 19 |
| 730 | 20 |

Il ne suffit plus que de "raccorder" les requêtes des exemples précédents :

```
SELECT (CAST(SUM(CHB_PLN_CLI_NB_PERS) AS FLOAT) /
      CAST(SUM(CHB_COUCHAGE) AS FLOAT)) * 100
            AS TAUX_OCCUPATION_POURCENT,
      C.CHB_ID
FROM   T_CHAMBRE C
      JOIN TJ_CHB_PLN_CLI CPC
          ON C.CHB_ID = CPC.CHB_ID
      CROSS JOIN T_PLANNING P
WHERE  P.PLN_JOUR BETWEEN '1999-01-01' AND '1999-12-31'
GROUP  BY C.CHB_ID;
```

| TAUX_OCCUPATION_POURCENT | CHB_ID |
|---|---|
| 65.293040293040292 | 1 |
| 74.606580829756794 | 2 |
| 74.64488636363636 | 3 |
| 74.660326086956516 | 4 |
| 65.301318267419958 | 5 |
| 60.27972027972028 | 6 |
| 74.759284731774414 | 7 |
| 67.191977077363902 | 8 |
| 75.878378378378372 | 9 |
| 75.681198910081747 | 10 |
| 67.715458276333777 | 11 |
| 65.209471766848822 | 12 |
| 75.920873124147334 | 13 |
| 74.595687331536382 | 14 |
| 67.343256653134858 | 15 |
| 61.162162162162161 | 16 |
| 66.891284815813108 | 17 |
| 77.762982689747005 | 18 |
| 65.900900900900908 | 19 |
| 76.625172890733069 | 20 |

Nous allons voir, maintenant que la clause GROUP BY est souvent utilisée avec la clause HAVING...

# 2. La clause HAVING

La clause HAVING agit comme le filtre WHERE, mais permet de filtrer non plus les données, mais les opérations résultant des regroupements, c'est à dire très généralement toute expression de filtre devant introduire un calcul d'agrégation.

Pour tenter de comprendre l'utilité de la clause having, nous allons procéder par un exemple simple : recherchons un étage de l'hôtel capable de coucher au moins 20 personnes.
Partant de notre exemple 2, nous serions tenter d'écrire :

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE, CHB_ETAGE
FROM   T_CHAMBRE
WHERE  SUM(CHB_COUCHAGE) >= 20
GROUP  BY CHB_ETAGE ;
```

Mais cette requête va imanquablement provoquer une erreur avant exécution du fait de la clause WHERE. En effet, souvenons nous que **le filtre WHERE agit sur les données des tables** et permet de filtrer ligne après ligne. Or le filtrage ne porte plus sur la notion de lignes, mais sur une notion de sous ensemble de la table. En d'autre termes, le filtre, ici, doit porter sur chacun des groupes. C'est pourquoi SQL introduit le filtre **HAVING** qui porte, non pas sur les données, mais sur les calculs résultants des regroupements.

En l'occurence, dans notre exemple, nous devons déporter le filtre WHERE dans la clause HAVING et l'opération deviendra possible :

```
SELECT SUM(CHB_COUCHAGE) AS NOMBRE,
CHB_ETAGE
FROM   T_CHAMBRE
GROUP  BY CHB_ETAGE
HAVING SUM(CHB_COUCHAGE) >= 20 ;
```

```
NOMBRE      CHB_ETAGE
----------- ---------
23          1er
22          2e
```

Autre exemple :
Partant de la requête vue à l'exemple 6, essayons de ne retenir que les chambres occupées à plus de 2/3, soit 66.666666... % ?

Une première tentative consisterait à écrire :

```
SELECT (CAST(SUM(CHB_PLN_CLI_NB_PERS) AS FLOAT) /
       CAST(SUM(CHB_COUCHAGE) AS FLOAT)) * 100 AS
       TAUX_OCCUPATION_POURCENT, C.CHB_ID
FROM   T_CHAMBRE C
       JOIN TJ_CHB_PLN_CLI CPC
           ON C.CHB_ID = CPC.CHB_ID
       CROSS JOIN T_PLANNING P
WHERE  P.PLN_JOUR BETWEEN '1999-01-01' AND '1999-12-31'
GROUP  BY C.CHB_ID
HAVING CAST(SUM(CHB_PLN_CLI_NB_PERS) AS FLOAT) /
      CAST(SUM(CHB_COUCHAGE) AS FLOAT) >= 2.0/3.0;
```

```
TAUX_OCCUPATION_POURCENT   CHB_ID
-------------------------- -----
74.606580829756794         2
74.64488636363636          3
74.660326086956516         4
74.759284731774414         7
67.191977077363902         8
75.878378378378372         9
75.681198910081747         10
67.715458276333777         11
75.920873124147334         13
74.595687331536382         14
67.343256653134858         15
66.891284815813108         17
77.762982689747005         18
76.625172890733069         20
```

Un autre exemple intéressant est de rechercher quelles sont les chambres qui ont été innocupées plus de 10 jours au cours du mois de janvier 2000...

Pour calculer le nombre de jour ou les chambres ont été occupées au cours de janvier 2000, il suffit de faire :

```
SELECT COUNT(C.CHB_ID) AS NOMBRE,
C.CHB_ID
FROM   T_CHAMBRE C
      JOIN TJ_CHB_PLN_CLI CPC
            ON C.CHB_ID = CPC.CHB_ID
WHERE  EXTRACT(MONTH FROM P.PLN_JOUR)
= 1
  AND  EXTRACT(YEAR FROM P.PLN_JOUR) =
2000
GROUP  BY C.CHB_ID;
```

| NOMBRE | CHB_ID |
|---|---|
| 19 | 1 |
| 20 | 2 |
| 18 | 3 |
| 23 | 4 |
| 21 | 5 |
| 19 | 6 |
| 18 | 7 |
| 20 | 8 |
| 21 | 9 |
| 24 | 10 |
| 21 | 11 |
| 22 | 12 |
| 20 | 13 |
| 24 | 14 |
| 23 | 15 |
| 20 | 16 |
| 19 | 17 |
| 24 | 18 |
| 19 | 19 |
| 23 | 20 |

Mais nous voulons le complément de cette colonne NOMBRE avec le nombre de jours du mois de janvier 2000, ce qui représente 31 jours. La requête devient donc :

```
SELECT 31 - COUNT(C.CHB_ID) AS NOMBRE,
C.CHB_ID
FROM   T_CHAMBRE C
      JOIN TJ_CHB_PLN_CLI CPC
            ON C.CHB_ID = CPC.CHB_ID
WHERE  EXTRACT(MONTH FROM P.PLN_JOUR)
= 1
  AND  EXTRACT(YEAR FROM P.PLN_JOUR) =
2000
GROUP  BY C.CHB_ID;
```

| NOMBRE | CHB_ID |
|---|---|
| 12 | 1 |
| 11 | 2 |
| 13 | 3 |
| 8 | 4 |
| 10 | 5 |
| 12 | 6 |
| 13 | 7 |
| 11 | 8 |
| 10 | 9 |
| 7 | 10 |
| 10 | 11 |
| 9 | 12 |
| 11 | 13 |
| 7 | 14 |
| 8 | 15 |
| 11 | 16 |
| 12 | 17 |
| 7 | 18 |
| 12 | 19 |
| 8 | 20 |

Dès lors il ne suffit plus que de filtrer le calcul :
31 - COUNT(C.CHB_ID)
pour ne retenir que les valeurs supérieures à 10. Comme cette expression contient une fonction d'agrégation, ce qui suppose un groupage, il est nécessaire d'utiliser le filtre WHERE :

```
SELECT 31 - COUNT(C.CHB_ID) AS NOMBRE, C.CHB_ID
FROM   T_CHAMBRE C
     JOIN TJ_CHB_PLN_CLI CPC
          ON C.CHB_ID = CPC.CHB_ID
WHERE  EXTRACT(MONTH FROM CPC.PLN_JOUR) = 1
  AND  EXTRACT(YEAR FROM CPC.PLN_JOUR) = 2000
GROUP  BY C.CHB_ID
HAVING 31 - COUNT(C.CHB_ID) > 10;
```

```
NOMBRE      CHB_ID
----------- -----------
12          1
11          2
13          3
12          6
13          7
11          8
11          13
11          16
12          17
12          19
```

Pour dernier exemple, nous allons nous placer dans un contexte plus proche de la réalité. Voici notre directeur qui souhaite inciter les clients avec lesquels il a peu travaillé à venir plus souvent. L'idée lui prend d'offrir un bon de réduction de 15% sur l'ensemble des prestations hôtelières pour tous les clients ayant eut un chiffre d'affaire inférieur à 15 000 F HT au cours de l'année 2000.
Une première approche consisterais par exemple a rechercher toutes les chambres libres entre les deux dates :

```
SELECT F.CLI_ID,
-- le calcul est le suivant :
-- (quantité * montant * remise en %) - remise en francs
-- les colonnes "remises" pouvant avoir des valeurs NULL
-- il faut les remplacer par un zéro à l'aide de la fonction COALESCE
     SUM(LIF_QTE * LIF_MONTANT * ((1 - COALESCE(LIF_REMISE_POURCENT/100, 0)))
                    - COALESCE(LIF_REMISE_MONTANT, 0)) AS CA
FROM   T_FACTURE F
     INNER JOIN T_LIGNE_FACTURE L
          ON F.FAC_ID = L.FAC_ID
WHERE  EXTRACT(YEAR FROM F.FAC_DATE) = 2000
GROUP  BY C.CLI_ID
HAVING SUM(LIF_QTE * LIF_MONTANT * ((1 -
COALESCE(LIF_REMISE_POURCENT/100, 0)))
                    - COALESCE(LIF_REMISE_MONTANT, 0)) < 15000 ;
```

```
CLI_ID  CA
------- ------------------
1       14510.6
37
12488.799999999999
39      13327.4
40
14785.200000000001
52
13890.799999999999
61
14168.799999999999
87      14992.4
```

Les clients 1, 37, 39, 40, 52, 61, 87 étant bien les personnes visées par la promotion.