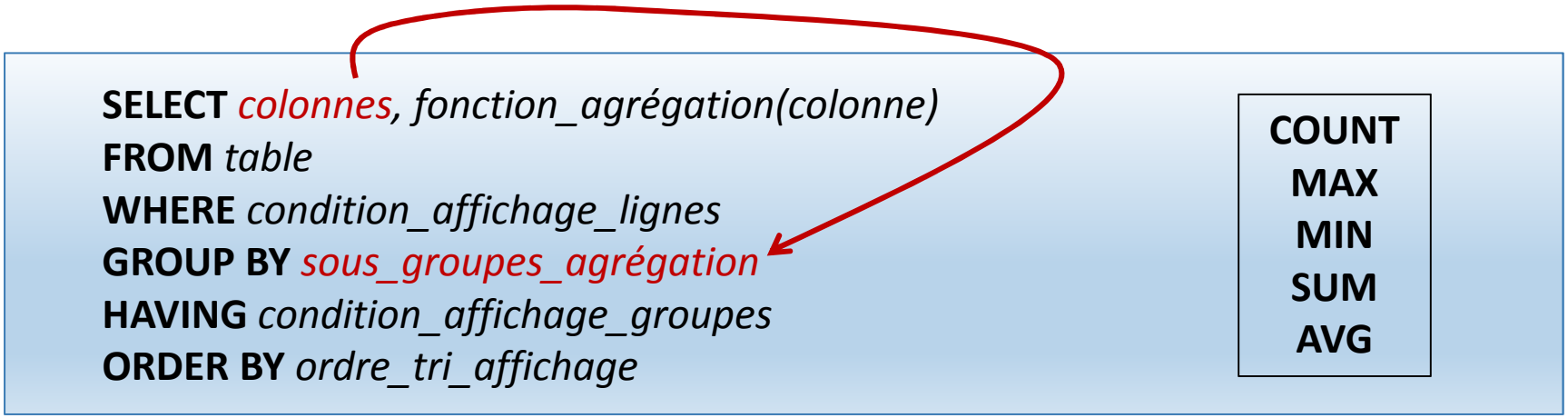
GROUP BY



- La clause ***« GROUP BY »*** permet de créer des sous-regroupements de lignes au niveau de la table, afin de leur appliquer une même fonction d'agrégation
- La clause ***« HAVING »*** ne peut être présente que si la clause ***« GROUP BY »*** est présente également. Le ***« HAVING »*** pose une condition d'affichage sur les groupes créés par la clause ***« GROUP BY »***. Cette condition doit porter sur une fonction d'agrégation également
- **Première règle d'or**
  *Dès que la clause **« SELECT »** combine l'affichage d'une ou plusieurs fonctions d'agrégation **ET** des colonnes non-agrégées, la clause **« GROUP BY »** est obligatoire*
- **Seconde règle d'or**
  *Toutes les colonnes non-agrégées présentes dans la clause **« SELECT »** doivent impérativement se retrouver dans la clause **« GROUP BY »***

GROUP BY

```
SELECT section_id, AVG(year_result)
  FROM student
GROUP BY section_id
```

section_id	Moyenne par section
1010	4
1020	7
1110	8
1120	17
1310	11
1320	10

Sans le ***« GROUP BY »***, le système produit l'erreur suivante :

```
Column 'student.section_id' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.
```

GROUP BY : + **WHERE**

```
    SELECT section_id, AVG(year_result)
      FROM student
     WHERE LEFT(last_name,1) IN ('B', 'C', 'D')
  GROUP BY section_id
```

section_id	last_name	year_result
1020	Connery	12
1110	De Niro	3
1120	Bacon	16
1310	Basinger	19
1110	Depp	11
1020	Clooney	4
1020	Cruise	4
1010	Bullock	2
1320	Doherty	2
1320	Berry	18

Moyenne = 6,67

section_id	Moyenne par section
1010	2
1020	6
1110	7
1120	16
1310	19
1320	10

Solution de la requête

*Ensemble de lignes triées grâce à la clause **« WHERE »***
*et sur lequel la clause **« GROUP BY »** sera appliquée*

GROUP BY : + **WHERE + HAVING**

```
    SELECT section_id, AVG(year_result)
      FROM student
     WHERE LEFT(last_name,1) IN ('B', 'C', 'D')
  GROUP BY section_id
    HAVING AVG(year_result) >= 10
```

section_id	last_name	year_result
1020	Connery	12
1110	De Niro	3
1120	Bacon	16
1310	Basinger	19
1110	Depp	11
1020	Clooney	4
1020	Cruise	4
1010	Bullock	2
1320	Doherty	2
1320	Berry	18

« WHERE » uniquement

section_id	Moyenne par section
1010	2
1020	6
1110	7
1120	16
1310	19
1320	10

WHERE + GROUP BY

section_id	Moyennes > 10
1120	16
1310	19
1320	10

WHERE + GROUP BY + HAVING

GROUP BY : **colonnes multiples**

```
    SELECT section_id, course_id, AVG(year_result)
      FROM student
     WHERE section_id IN (1010, 1020)
  GROUP BY section_id, course_id
    HAVING SUM(year_result) >= 2
  ORDER BY section_id
```

- *Toutes les colonnes non-agrégées présentes dans la clause **« SELECT »** doivent impérativement se retrouver dans la clause **« GROUP BY »***
- La condition du ***« HAVING »***, portant sur l'affichage des groupes créés par la clause ***« GROUP BY »***, peut utiliser d'autres fonctions d'agrégation et d'autres colonnes que celles utilisées dans la clause ***« SELECT »***

section_id	course_id	Moyenne
1010	EG1020	2
1010	EG2210	4
1020	EG1020	7
1020	EG2110	7
1020	EG2210	10

GROUP BY : **ROLLUP et CUBE**

SELECT *colonnes, fonction_agrégation(colonne)*
FROM *table*
GROUP BY ROLLUP (*sous_groupes_agrégation***)**

SELECT *colonnes, fonction_agrégation(colonne)*
FROM *table*
GROUP BY CUBE (*sous_groupes_agrégation***)**

- Les mots-clés ***« ROLLUP »*** ou ***« CUBE »*** peuvent être rajoutés à la clause ***« GROUP BY »*** de façon à afficher des sous-totaux
- ***« ROLLUP »*** applique un sous-total en remontant dans les colonnes indiquées, présentant un sous-total à partir de la colonne la plus détaillée, en remontant vers la colonne présentant des résultats groupés de façon plus vaste (sous-total par section et global)
- ***« CUBE »*** permet d'appliquer la fonction d'agrégation sur tout regroupement possible au niveau des données agrégées (sous-total par section, global et par cours, sans tenir compte des sections). Le ***« CUBE »*** englobe le ***« ROLLUP »***

GROUP BY : ROLLUP

```
SELECT section_id, course_id, AVG(year_result)
  FROM student
 WHERE section_id IN (1010, 1020)
GROUP BY ROLLUP (section_id, course_id)
```

section_id	course_id	Moyenne
1010	EG1020	2
1010	EG2210	4
1020	EG1020	7
1020	EG2110	7
1020	EG2210	10

Sans « ROLLUP »

section_id	course_id	Moyenne	
1010	EG1020	2	
1010	EG2210	4	
1010	NULL	4	*Total section 1010*
1020	EG1020	7	
1020	EG2110	7	
1020	EG2210	10	
1020	NULL	7	*Total section 1020*
NULL	NULL	6	*Total général*

Avec « ROLLUP »

GROUP BY : CUBE

```
    SELECT section_id, course_id, SUM(year_result)
      FROM student
     WHERE section_id IN (1010, 1020)
  GROUP BY CUBE (section_id, course_id)
```

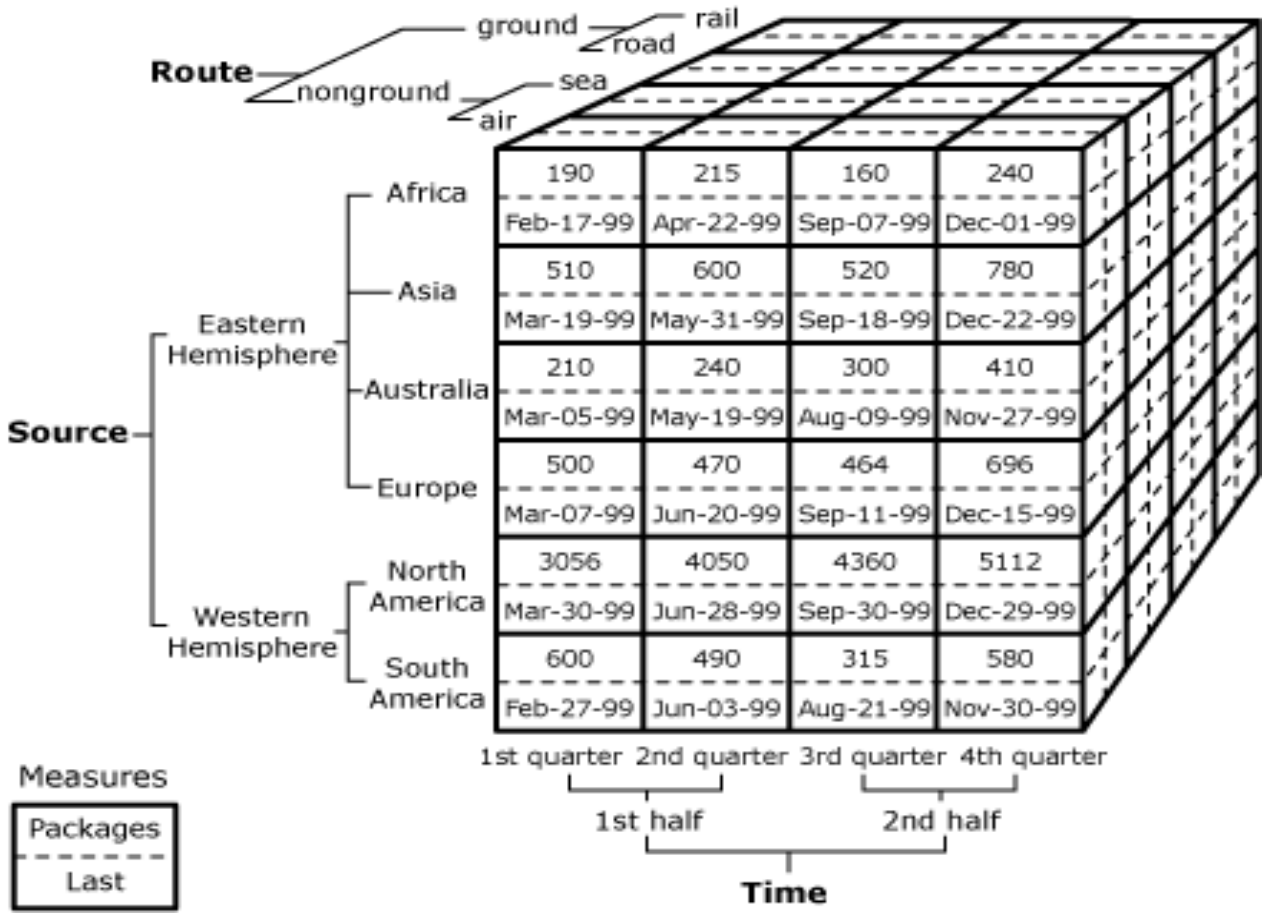
section_id	course_id	Somme
1010	EG1020	2
1020	EG1020	7
1020	EG2110	35
1010	EG2210	14
1020	EG2210	10

Sans « CUBE »

section_id	course_id	Somme	
1020	EG2210	10	
1020	EG2110	35	
1020	EG1020	7	
1020	NULL	52	*Total section 1020*
1010	EG2210	14	
1010	EG1020	2	
1010	NULL	16	*Total section 1010*
NULL	EG2210	24	
NULL	EG2110	35	*Total par cours*
NULL	EG1020	9	
NULL	NULL	68	*Total général*

Avec « CUBE »

GROUP BY : CUBE OLAP



Auto-Evaluation

N'oubliez pas de prendre le temps d'évaluer le niveau de maîtrise que vous estimez avoir acquis personnellement concernant les notions abordées dans ce module !

Rappel de la signification des lettres dans les tableaux d'auto-évaluation :

- **Parfait (P) :** vous avez parfaitement compris cette notion et vous vous sentez à votre aise
- **Satisfaisant (S) :** vous avez compris de quoi il s'agit mais la pratique vous manque
- **Vague (V) :** vous savez de quoi il s'agit, mais cela reste un peu vague dans votre esprit. Une explication supplémentaire du formateur ou une bonne révision de votre part s'impose
- **Insatisfaisant (I) :** Vous n'avez pas du tout compris la notion abordée, il faut tout faire pour y remédier !

Auto-Evaluation

Notions à évaluer

Notions	P	S	V	I
Clause « GROUP BY … HAVING » et règles d'or				
Différence entre les clauses « WHERE » et « HAVING »				
« GROUP BY » sur plusieurs colonnes				
Clause « ROLLUP »				
Clause « CUBE »				