Interrogation de données décisionnelles

V

A. Manipulation des données multidimensionnelles

1. Opérateurs OLAP

Remarque

Cette section est basé sur le support de cours de Pr. Bernard ESPINASSE Professeur à Aix-Marseille Université (AMU) Ecole Polytechnique Universitaire de Marseille. http://www.lsis.org/espinasseb/Supports/DWDM/3-OLAP-4p.pdf

Nous distinguons 3 catégories d'opérateurs OLAP

- Opérateurs de restructuration
- Opérateurs de granularité
- Opérateurs ensembliste

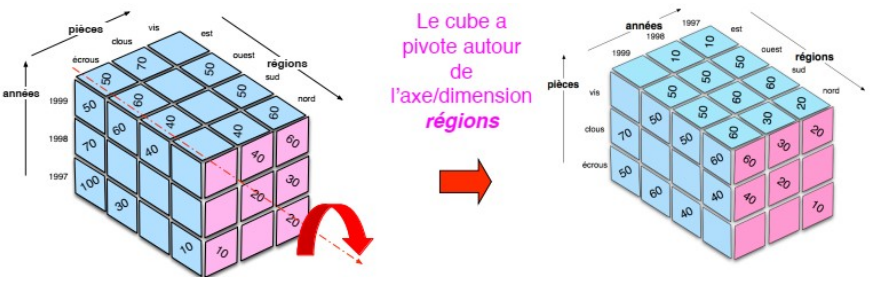
a) Opérateurs de restructuration

Ce sont des opérateurs qui concernent la représentation et permettent un changement de points de vue selon différentes dimensions :

- Rotate/pivot
- Switch
- Split, nest, push, pull

Rotate/Pivot

Effectue au cube une rotation autour d'un de ses axes de façon à présenter un ensemble de faces différent (sélection de faces)



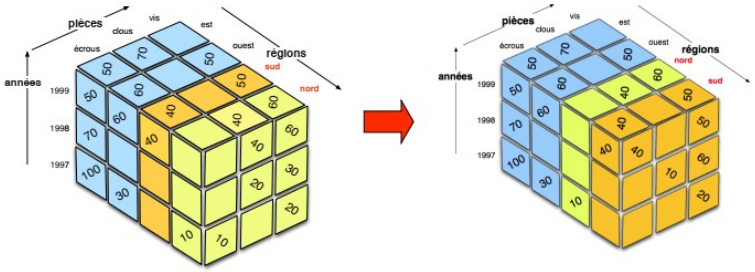
La visualisation résultante 2D :

nord	1999	1998	1997
vis	60	30	20
clous	40	20	
écrous			10

vis	1999	1998	1997
est		10	10
ouest	50	50	50
sud	50	60	60
nord	60	30	20

Switch ou permutation :

Consiste à inter-changer la position des membres d'une dimension :



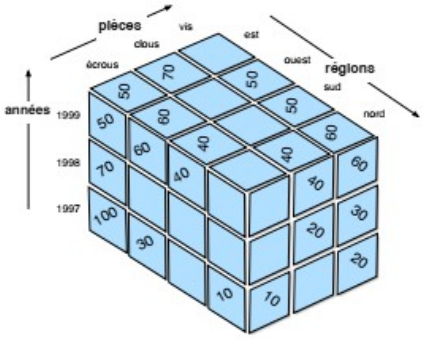
La visualisation résultante 2D :

nord	1999	1998	1997
vis	60	30	20
clous	40	20	
écrous			10

sud	1999	1998	1997
vis	50	60	60
clous		10	
écrous	40	20	

Split ou division

Consiste à présenter chaque tranche du cube et de passer de sa présentation tridimensionnelle à sa présentation sous la forme d'un ensemble de tables.



Un split(region) du cube Ventes conduit aux 4 tables suivantes :

ventes est	1999	1998	1997
écrous	50	70	100
vis		10	10
clous	70	70	100

ventes ouest	1999	1998	1997
écrous		10	30
vis	50	50	50
clous		10	40

ventes sud	1999	1998	1997
écrous	40	20	
vis	50	60	60
clous		10	

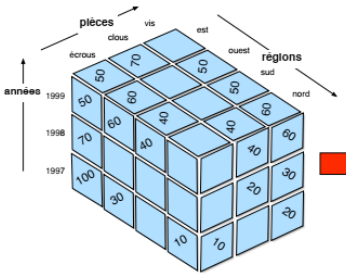
ventes nord	1999	1998	1997
écrous			10
vis	60	30	20
clous	40	20	

Nest ou emboîtement :

Permet d'imbriquer des membres à partir du cube. L'intérêt de cette est qu'elle

permet de grouper sur une même représentation bi-dimensionnelle toutes les informations (mesures et membres) d'un cube quelque soit le nombre de ses dimensions.

nest(pièces, région) :

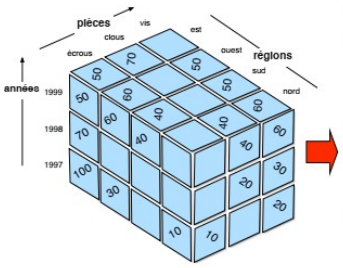


ventes nest		1999	1998	1997
écrous	est	50	70	100
	ouest		10	30
	nord			10
	sud	40	20	
vis	est		10	10
	ouest	50	50	50
	nord	60	30	20
	sud	50	60	60
clous	est	70	70	100
	ouest		10	40
	nord	40	20	
	sud		10	

Push ou l'enfoncement :

Consiste à combiner les membres d'une dimension aux mesures du cube, i.e. de faire passer des membres comme contenu de cellules.

push(année) :



ventes push	est	ouest	nord	sud
écrous	1999 50 1998 70 1997 100	1998 10 1997 30	1997 10	1999 40 1998 20
vis	1998 10 1997 10	1999 50 1998 50 1997 50	1999 60 1998 30 1997 20	1999 50 1998 60 1997 60
clous	1999 70 1998 70 1997 100	1998 10 1997 40	1999 40 1998 20	1998 10

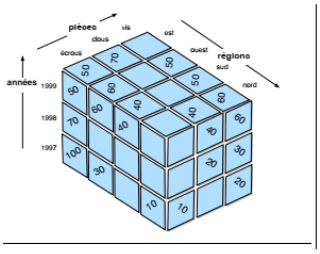
b) Opérateurs de granularité

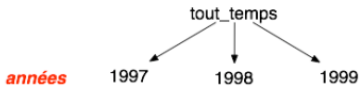
Dans un entrepôt l'information est hiérarchisée en différents niveaux de détails appelés niveaux de granularité. Les opérateurs d'agrégation successives sur ces données permettent de nouveaux points de vue de moins en moins (ou de plus en plus) détaillés de l'information.

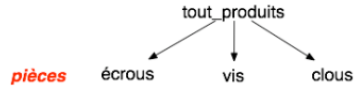
- Roll-up,
- Drill-down

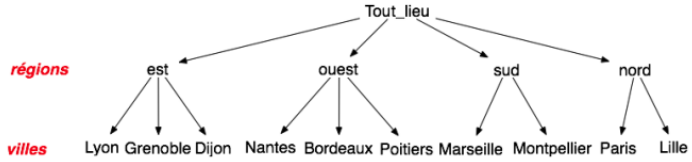
Roll-up ou forage vers le haut

Consiste à représenter les données du cube à un niveau de granularité supérieur conformément à la hiérarchie définie sur la dimension.

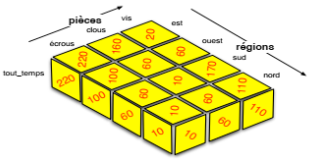


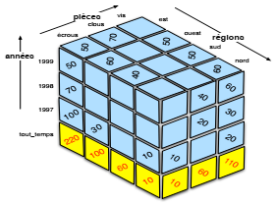






roll-up(annee) : Ventes 97-99





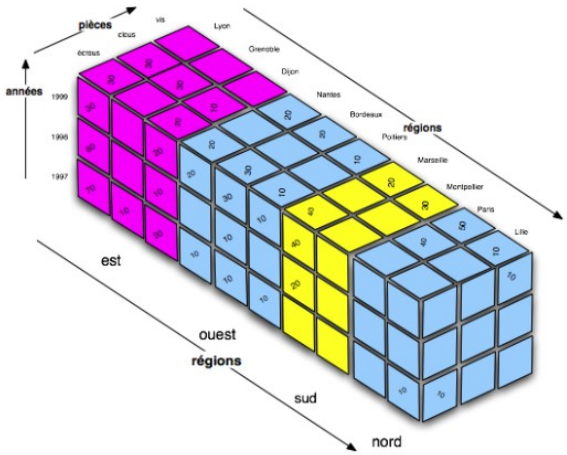
roll-up(annees, pieces) : la visualisation est souvent 2D :

nord	1999	1998	1997	tout_temps
vis	60	30	20	110
clous	40	20		60
écrous			10	10
tout_produit	100	50	30	180

Drill-down ou forage vers le bas

Consiste à représenter les données du cube à un niveau de granularité de niveau inférieur, donc sous une forme plus détaillée.

Drill-down du niveau des régions au niveau villes : Drill-down(regions) :



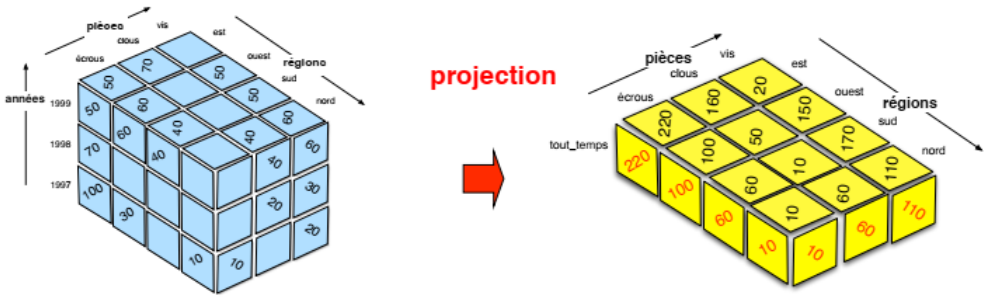
c) Opérateurs ensemblistes

Ils concernent l'extraction et constituent des manipulations classiques.

- Slice et Dice (sélection et projection)

Slice

Correspond à une projection selon une dimension du cube



π piece, region :

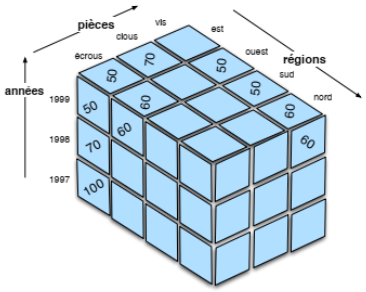
ventes 97-99	est	ouest	sud	nord
écrous	220	100	60	10
clous	160	50	10	60
vis	20	150	170	110

Dice (Sélection)

Correspond à une sélection d'un sous coube.

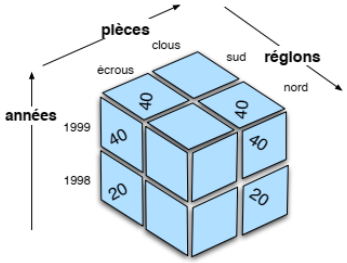
Selection 1

vente ≥ 50



Sélection 2

(regions = nord ou regions = sud) et
(pieces = clous ou pieces = ecrous) et
(annees = 1998 ou années = 1999)



B. SQL analytique

1. GROUPING SETS

GROUPING SETS

GROUP BY multiples en précisant quelles UNION sont souhaitées. L'imbrication d'attributs permet de séparer les GROUP BY simples de l'UNION de GROUP BY

GROUP BY GROUPING SETS ((jour, ville, pièce))	≡	GROUP BY jour, ville, pièce
GROUP BY GROUPING SETS (jour, ville, pièce)	≡	GROUP BY jour UNION GROUP BY ville UNION GROUP BY pièce
GROUP BY GROUPING SETS (jour,(ville,pièce))	≡	GROUP BY jour UNION GROUP BY ville, pièce

Image 20 Grouping sets

2. ROLLUP/CUBE

Les options Rollup et Cube dans un group by permettent d'introduire des sous totaux à différents niveaux.

a) ROLLUP

ROLLUP

Permet à une instruction SELECT de calculer plusieurs niveaux de sous-totaux parmi un groupe de dimensions spécifié. ROLLUP se glisse dans la clause GROUP BY d'une instruction SELECT selon le format suivant :

SELECT . . .

GROUP BY ROLLUP(listeDeColonnes)

ROLLUP crée des sous-totaux qui peuvent porter sur le niveau de détail, jusqu'au niveau du total, en fonction de la liste de colonnes spécifiée dans la clause ROLLUP.

ROLLUP calcule d'abord les valeurs d'agrégat standard spécifiées dans la clause GROUP BY, puis calcule progressivement les sous-totaux des niveaux supérieurs et se déplace parmi les colonnes de la liste jusqu'à aboutir à un total général. 6 [6]

ROLLUP crée ainsi des sous-totaux à n + 1 niveaux, où n est le nombre de colonnes de regroupement. Par exemple, si une requête spécifie un ROLLUP sur les colonnes de regroupement Année, Région et Département (donc, n = 3), alors l'ensemble de résultats comporte des lignes de 4 niveaux d'agrégation. 6 [6]

Exemple

Requête simple, sans sous-totaux :

```
select annee, region, departement, sum(CA) as SommeCA
from ventes
group by (annee, region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	CA
2013	Centre	Dept1	50000
2013	Centre	Dept2	75000
2013	Est	Dept1	55000
2013	Est	Dept2	68000
2013	West	Dept1	62000
2013	West	Dept2	100000
2014	Centre	Dept1	110000
2014	Centre	Dept2	95000
2014	Est	Dept1	88000
2014	Est	Dept2	69000
2014	West	Dept1	57000
2014	West	Dept2	67000

Requête simple

Exemple

Requête impliquant l'extension ROLLUP :

```
select annee, region, departement, sum(CA) as SommeCA
from ventes
group by rollup (annee, region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	SOMMECA
2013	Centre	Dept1	50000
2013	Centre	Dept2	75000
2013	Centre		125000
2013	Est	Dept1	55000
2013	Est	Dept2	68000
2013	Est		123000
2013	West	Dept1	62000
2013	West	Dept2	100000
2013	West		162000
2013			410000
2014	Centre	Dept1	110000
2014	Centre	Dept2	95000
2014	Centre		205000
2014	Est	Dept1	88000
2014	Est	Dept2	69000
2014	Est		157000
2014	West	Dept1	57000
2014	West	Dept2	67000
2014	West		124000
2014			486000
			896000

Rollup

Rollup Partiel

Il est également possible d'avoir seulement les sous-totaux pour certaines colonnes.

Exemple

```
select annee, region, departement, sum(CA) as SommeCA
from ventes
```

```
group by annee, rollup (region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	SOMMECA
2013	Centre	Dept1	50000
2013	Centre	Dept2	75000
2013	Centre		125000
2013	Est	Dept1	55000
2013	Est	Dept2	68000
2013	Est		123000
2013	West	Dept1	62000
2013	West	Dept2	100000
2013	West		162000
2013			410000
2014	Centre	Dept1	110000
2014	Centre	Dept2	95000
2014	Centre		205000
2014	Est	Dept1	88000
2014	Est	Dept2	69000
2014	Est		157000
2014	West	Dept1	57000
2014	West	Dept2	67000
2014	West		124000
2014			486000

Rollup Partiel

b) CUBE

CUBE

CUBE prend l'ensemble de colonnes de regroupement spécifiées et crée des sous-totaux pour chacune des combinaisons possibles. CUBE se place dans la clause GROUP BY d'une instruction SELECT, selon la forme suivante :

```
SELECT . . .
GROUP BY CUBE(listeDeColonnes)
```

Exprimée en termes d'analyse multidimensionnelle, CUBE génère tous les sous-totaux que l'on pourrait calculer pour un cube de données des dimensions spécifiées.

Ainsi, si nous indiquons CUBE(année, région, département), l'ensemble de résultats contient toutes les valeurs que donnerait une instruction ROLLUP équivalente, plus toutes les combinaisons supplémentaires.

Dans l'exemple précédent, les totaux sur département pour les combinaisons de année ne sont pas calculés par la clause ROLLUP(année, région, département), tandis qu'ils le sont par la clause CUBE(année, région, département). Si la clause CUBE comporte n colonnes, alors les résultats renvoient (2 à la puissance n) combinaisons de sous-totaux. L'exemple que nous venons de choisir illustre un cube tridimensionnel.

Exemple

```
select annee, region, departement, sum(CA) as SommeCA
from ventes
group by cube (annee, region, departement)
```

order by annee, region, departement;

ANNEE	REGION	DEPARTEMENT	SOMMECA
2013	Centre	Dept1	50000
2013	Centre	Dept2	75000
2013	Centre		125000
2013	Est	Dept1	55000
2013	Est	Dept2	68000
2013	Est		123000
2013	West	Dept1	62000
2013	West	Dept2	100000
2013	West		162000
2013		Dept1	167000
2013		Dept2	243000
2013			410000
2014	Centre	Dept1	110000
2014	Centre	Dept2	95000
2014	Centre		205000
2014	Est	Dept1	88000
2014	Est	Dept2	69000
2014	Est		157000
2014	West	Dept1	57000
2014	West	Dept2	67000
2014	West		124000
2014		Dept1	255000
2014		Dept2	231000
2014			486000
	Centre	Dept1	160000
	Centre	Dept2	170000
	Centre		330000
	Est	Dept1	143000
	Est	Dept2	137000
	Est		280000
	West	Dept1	119000
	West	Dept2	167000
	West		286000
		Dept1	422000
		Dept2	474000
			896000

Cube

Remarque

CUBE est utilisable dans toute situation qui nécessite l'obtention de tableaux croisés dynamiques. Les données nécessaires pour les tableaux croisés peuvent être générées en un seul SELECT contenant une clause CUBE. Comme ROLLUP, CUBE s'avère utile surtout lorsqu'il est nécessaire d'obtenir des tableaux de synthèse.

CUBE est particulièrement indiqué dans les requêtes qui emploient des colonnes de dimensions multiples et non de colonnes représentant des niveaux différents d'une seule et même dimension. 6 [6]

CUBE partiel

CUBE partiel (partial CUBE) ressemble au ROLLUP partiel (partial ROLLUP) dans le sens que l'on peut la limiter à certaines dimensions. Les sous-totaux de toutes les combinaisons possibles sont limitées aux dimensions qui se trouvent entre parenthèse.

Exemple

select annee, region, departement, sum(CA) as SommeCA

from ventes

```
group by annee, cube (region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	SOMMECA
2013	Centre	Dept1	50000
2013	Centre	Dept2	75000
2013	Centre		125000
2013	Est	Dept1	55000
2013	Est	Dept2	68000
2013	Est		123000
2013	West	Dept1	62000
2013	West	Dept2	100000
2013	West		162000
2013		Dept1	167000
2013		Dept2	243000
2013			410000
2014	Centre	Dept1	110000
2014	Centre	Dept2	95000
2014	Centre		205000
2014	Est	Dept1	88000
2014	Est	Dept2	69000
2014	Est		157000
2014	West	Dept1	57000
2014	West	Dept2	67000
2014	West		124000
2014		Dept1	255000
2014		Dept2	231000
2014			486000

Cube Partiel

3. Fonction GROUPING

Fonction GROUPING

En utilisant un colonne comme argument, GROUPING retourne 1 quand il rencontre une valeur NULL créée par ROLLUP ou CUBE. En effet, si un NULL indique un sous-total, GROUPING retourne un 1. N'importe quelle autre valeur, incluant une valeur stockée NULL, retourne un 0.

Exemple : Requête de type ROLLUP avec une fonction GROUPING

```
select annee, region, departement, sum(CA) as SommeCA,
grouping (annee) as an, grouping (region) as reg, grouping (departement) as dept
from ventes
group by rollup (annee,region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	SOMMECA	AN	REG	DEPT
2013	Centre	Dept1	50000	0	0	0
2013	Centre	Dept2	75000	0	0	0
2013	Centre		125000	0	0	1
2013	Est	Dept1	55000	0	0	0
2013	Est	Dept2	68000	0	0	0
2013	Est		123000	0	0	1
2013	West	Dept1	62000	0	0	0
2013	West	Dept2	100000	0	0	0
2013	West		162000	0	0	1
2013			410000	0	1	1
2014	Centre	Dept1	110000	0	0	0
2014	Centre	Dept2	95000	0	0	0
2014	Centre		205000	0	0	1
2014	Est	Dept1	88000	0	0	0
2014	Est	Dept2	69000	0	0	0
2014	Est		157000	0	0	1
2014	West	Dept1	57000	0	0	0
2014	West	Dept2	67000	0	0	0
2014	West		124000	0	0	1
2014			486000	0	1	1
			896000	1	1	1

Grouping

On peut augmenter la lisibilité en utilisant les fonctions GROUPING et DECODE comme dans l'exemple ci-dessous.

```
select decode (grouping (region), 1, 'Total_Région', region) as region,
decode (grouping (departement), 1, 'Total_Dept', departement) as Dept,
sum(CA) as SommeCA
from ventes
group by rollup (region, departement);
```

REGION	DEPT	SOMMECA
Centre	Dept1	350000
Centre	Dept2	355000
Centre	Total_Dept	705000
Est	Dept1	314000
Est	Dept2	282000
Est	Total_Dept	596000
West	Dept1	247000
West	Dept2	307000
West	Total_Dept	554000
Total_Région	Total_Dept	1855000

Decode

Explication de la ligne Select decode (grouping (region), 1, 'Total_Région', region) as region : La valeur de region est déterminée avec la fonction DECODE qui contient la fonction de groupes GROUPING. GROUPING retourne un 1 si la valeur de la ligne est un agrégat créé par ROLLUP ou CUBE, sinon elle retourne un 0. La fonction DECODE opère ensuite sur le résultat de la fonction de groupe GROUPING. Elle retourne le texte "Total régions" si elle reçoit un 1, et que la valeur de region (de la table) reçoit un 0. La valeur de la table sera une valeur telle que "Centre" ou un NULL.6 [6]

Remarque : Quand faut-il utiliser GROUPING?

La fonction GROUPING est très utile pour identifier les NULLs de la base de données, elle permet aussi de trier, classer les lignes de sous-totaux ainsi que le filtrage des résultats.

Il est également possible d'utiliser la clause HAVING avec une fonction de groupes

GROUPING.6 [6]

Fonction GROUPING_ID

Pour trouver le niveau du GROUP BY pour une ligne particulière, une requête doit retourner une information de la fonction GROUPING pour chaque colonne du GROUP BY. Dans ce cas, avec la fonction GROUPING, chaque colonne du GROUP BY requière une autre colonne utilisant la fonction GROUPING. Il en résulte un gaspillage d'espace de stockage.

Pour résoudre ce problème, Oracle9i introduit la fonction GROUPING_ID. GROUPING_ID retourne un nombre unique qui permet de déterminer le niveau exact du GROUP BY. Pour chaque ligne, GROUPING_ID prend un "set" de 1 et 0.6 [6]

Par exemple, si on utilise l'expression CUBE(a,b), les valeurs possibles seront:

Niveau d'agrégation	Vecteur de bit	GROUPING_ID
a, b	0 0	0
a	0 1	1
b	1 0	2
Grand Total	1 1	3

Fonction GROUPING_ID

Exemple

```
select annee, region, departement, sum(CA) as SommeCA, grouping_id(annee, region, departement) as GID
from ventes
group by rollup (annee, region, departement)
order by annee, region, departement;
```

ANNEE	REGION	DEPARTEMENT	SOMMECA	GID
2013	Centre	Dept1	50000	0
2013	Centre	Dept2	75000	0
2013	Centre		125000	1
2013	Est	Dept1	55000	0
2013	Est	Dept2	68000	0
2013	Est		123000	1
2013	West	Dept1	62000	0
2013	West	Dept2	100000	0
2013	West		162000	1
2013			410000	3
2014	Centre	Dept1	110000	0
2014	Centre	Dept2	95000	0
2014	Centre		205000	1
2014	Est	Dept1	88000	0
2014	Est	Dept2	69000	0
2014	Est		157000	1
2014	West	Dept1	57000	0
2014	West	Dept2	67000	0
2014	West		124000	1
2014			486000	3
			896000	7

Grouping_ID

4. Rank

Rank

La fonction de classement Rank calcule la position d'un enregistrement par rapport aux autres enregistrements d'un ensemble de données, selon les valeurs d'une série de mesures. Un certain nombre de fonctions de classement sont disponibles, comme RANK et DENSE_RANK. Leur syntaxe est la suivante :

Exemple

Select region, sum(CA) as SommeCA, rank() over (order by (sum(CA)) Desc) as classement

from ventes

group by region;

REGION	SOMMECA	CLASSEMENT
Centre	330000	1
West	286000	2
Est	280000	3

Rank

Remarque

Il existe une autre variante : 'DENSE_RANK' qui ne laisse aucun « trou » dans la séquence de classement, quand il y a des ex-æquo dans un classement.

Par exemple si deux régions détiennent ex-æquo la première place DENSE_RANK les identifie en première place et la suivante en deuxième place, alors que la fonction RANK considère la région suivante comme étant à la troisième place.

5. Calculs fenêtrés

Calculs fenêtrés

Les calculs fenêtrés portent ce nom car ils interviennent sur des « fenêtres » de données, prises dans une plage d'enregistrements sélectionnée en fonction de certains critères. La fenêtre de données peut ensuite se déplacer en fonction d'un élément qui représente le temps, présent dans l'une des colonnes du jeu de données, soit de façon explicite (valeurs discrètes), soit implicite (champ dérivé).

Exemple

select annee, sum(CA) as SommeCA,

avg(sum(CA)) over (order by annee rows 1 preceding) as MoyMobile_sur_2_annees

from ventes

group by annee

order by annee;

ANNEE	SOMMECA	MOYMOBILE_SUR_2_ANNEES
2011	467000	467000
2012	492000	479500
2013	410000	451000
2014	486000	448000

Moyenne mobile

Notez que la première ligne des calcul de la moyenne mobile à deux années se base sur un intervalle de plus petite taille que ce que nous avons demandé. Il est donc nécessaire de considérer les différentes tailles de fenêtres trouvées aux frontières des ensembles de résultats.

6. CUME_DIST

CUME_DIST

La fonction CUME_DIST (définie comme l'inverse de percentile dans certains livres de statistiques) calcule la position d'une valeur relative spécifique par rapport à un jeu de valeurs. L'ordre peut être ascendant (par défaut) ou descendant. La rangée de valeurs pour CUME_DIST est plus grande que 0 jusqu'à 1. Pour calculer le CUME_DIST d'une valeur x dans un jeu S de taille N, utiliser la formule suivante:

CUME_DIST(x) = nombre de valeurs dans S venant avant et incluant x, dans l'ordre spécifié par ORDER / N

Syntaxe

CUME_DIST ( ) OVER ([query_partition_clause]

order_by_clause)

Exemple

L'exemple suivant montre une répartition cumulative des ventes par région pour chaque année

```
select annee, region, sum(CA) as SommeCA, cume_dist() over(partition by annee
order by sum(CA)) as Cum_Dist_CA
from ventes
group by annee, region
order by annee, region;
```

ANNEE	REGION	SOMMECA	CUM_DIST_CA
2011	Centre	190000	1
2011	Est	148000	.666666667
2011	West	129000	.333333333
2012	Centre	185000	1
2012	Est	168000	.666666667
2012	West	139000	.333333333
2013	Centre	125000	.666666667
2013	Est	123000	.333333333
2013	West	162000	1
2014	Centre	205000	1
2014	Est	157000	.666666667
2014	West	124000	.333333333

Cume_dist

7. NTILE

NTILE

NTILE permet de facilement calculer des tertiles, quartiles, deciles et autres statistiques récapitulatives. Cette fonction divise une partition ordonnée en un nombre spécifique de groupes appelés buckets, et elle attribue un numéro (bucket number) à chaque ligne de la partition. NTILE permet un calcul très utile parce qu'elle permet aux utilisateurs de diviser un jeu de donnée en quatre, trois ou différents groupes.

Les buckets sont calculés de façon à ce que chaque bucket ait exactement le même nombre de lignes attribuées ou au moins 1 ligne de plus que les autres. Par exemple, si on a 100 lignes dans une partition et que l'on demande la fonction NTILE avec quatre buckets, 25 lignes se feront attribuer la valeur 1, 25 lignes auront la valeur 2, et ainsi de suite.

Syntaxe

NTILE (expr) OVER ([query_partition_clause]
order_by_clause)

Exemple

L'exemple suivant attribue pour chaque total des ventes/année dans 2 buckets

```
select annee, sum(CA) as SommeCA, ntile(2) over(order by sum(CA)) as tile_2
from ventes
group by annee
order by annee;
```

ANNEE	SOMMECA	TILE_2
2011	467000	1
2012	492000	2
2013	410000	1
2014	486000	2

NTile

Remarque

Si le nombre de lignes dans la partition ne se divise pas de manière égale, alors le nombre de lignes attribué pour chaque bucket différera de un tout au plus. Les lignes qui sont en plus, seront distribuées de façon à avoir une ligne pour chaque bucket partant depuis le plus petit nombre (bucket number). Par exemple, s'il y a 103 lignes dans une partition avec une fonction NTILE(5), les premières 21 lignes seront dans le premier bucket, les prochaines 21 dans le second bucket, les prochaines 21 dans le troisième, les 20 prochaines dans le quatrième bucket et les dernières 20 dans le cinquième bucket.

8. Covariance/ Corrélation

Covariance

Soit avgexp1 le résultat de AVG(expression1) et soit avgexp2 le résultat de AVG(expression2),

Alors COVARIANCE(expression1, expression2) = AVG((expression1 - avgexp1) * (expression2 - avgexp2)

Exemple

SELECT COVARIANCE(SALARY, BONUS)
FROM EMPLOYEE WHERE
WORKDEPT = 'A00'

Correlation

CORRELATION(expression1,expression2)=COVARIANCE(expression1,expression2)/(STDDEV(expression1)*STDDEV(expression2))

Exemple

SELECT CORRELATION(SALARY, BONUS)
FROM EMPLOYEE
WHERE WORKDEPT = 'A00'

9. Fonctions de rapport d'agrégat

Fonctions de rapport d'agrégat

Après qu'une requête ait été traitée, les valeurs agrégées, comme le nombre de lignes traitées ou la valeur moyenne pour une colonne peut facilement être calculée dans une partition et être mis à disposition d'autres fonctions de rapport. Les fonctions de rapport d'agrégat retournent la même valeur d'agrégation pour chaque ligne dans une partition.

Syntaxe

{SUM | AVG | MAX | MIN | COUNT | STDDEV | VARIANCE}
([ALL | DISTINCT] {value expression1 | *})
OVER ([PARTITION BY value expression2[,...]])

De plus, les conditions suivantes s'appliquent:

- Un astérisque (*) est seulement autorisé dans COUNT(*)
- DISTINCT est supporté seulement si la fonction d'agrégation le permet
- valeur expression1 et valeur expressio2 peuvent être n'importe quelle expression impliquant des colonnes de références ou d'agrégats.
- La clause PARTITION BY défini les groupes sur lesquels les fonctions de fenêtre (windowing functions) seraient calculées. Si la clause PARTITION BY est absente, alors la fonction est calculée sur tout le jeu de résultat.

Remarque

Les fonctions de rapport peuvent apparaître seulement dans la clause SELECT ou ORDER BY. Le bénéfice majeur des fonctions de rapport est leurs capacités à passer plusieurs données dans un block unique de requête et d'augmenter les performances de la requête. Les requêtes telles que "Compter le nombre de vendeurs avec des ventes de plus de 10% des ventes de la ville" ne requièrent pas de jointures entre les blocs séparés.

Exemple

"Pour chaque département, trouver la région dans laquelle il y a eu le maximum de ventes"

```
Select departement, region,SommeCA
from
(select departement, region,sum(CA) as SommeCA, max(sum(CA)) over(partition by departement) as Max_Reg_CA
from ventes
group by departement, region)
where SommeCA=Max_Reg_CA;
```

DEPARTEMENT	REGION	SOMMECA	MAX_REG_CA
Dept1	Centre	350000	350000
Dept1	Est	314000	350000
Dept1	West	247000	350000
Dept2	Est	282000	355000
Dept2	West	307000	355000
Dept2	Centre	355000	355000

Résultat intermédiaire

DEPARTEMENT	REGION	SOMMECA
Dept1	Centre	350000
Dept2	Centre	355000

Image 21 Rapport d'agrégat

10. RATIO_TO_REPORT

RATIO_TO_REPORT

La fonction RATIO_TO_REPORT calcule le ratio d'une valeur par rapport à la somme d'un jeu de valeurs. Si l'expression valeur expression est évaluée à NULL, alors RATIO_TO_REPORT est évaluée aussi à NULL, mais cela est traité comme un zéro pour calculer la somme des valeurs pour le dénominateur.

Syntaxe

RATIO_TO_REPORT (expr) OVER ([query_partition_clause])

Selon la syntaxe, ceci s'applique pour:

• expr peut être n'importe quelle expression impliquant des colonnes de références ou d'agrégats.

• La clause PARTITION BY définit les groupes dans lesquelles la fonction RATIO_TO_REPORT doit être calculée. Si la clause PARTITION BY est absente, alors la fonction est calculée sur tout le jeu de résultat.

Exemple

```
select departement, sum(CA) as SommeCA, sum(sum(CA)) over() as Total,
ratio_to_report (sum(CA)) over() as Ratio
from ventes
group by departement;
```

DEPARTEMENT	SOMMECA	TOTAL	RATIO
Dept1	911000	1855000	.491105121
Dept2	944000	1855000	.508894879

Ratio_to_report

C. Références Bibliographiques

6. T. Connolly, C.Begg, "Systèmes de base de données", Editions Eyrolles, 2005.