Entrepôt de données : Extensions du langage SQL (SQL-3/SQL-99) pour l'OLAP (7.1)

Bernard ESPINASSE
Professeur à Aix-Marseille Université (AMU)
Ecole Polytechnique Universitaire de Marseille

Décembre 2015

- **Evolution de SQL pour OLAP :**
  - **Evolution de la norme SQL : OLAP SQL 3 / SQL 99 extensions**
  - **Nouvelles fonction SQL d'agrégation : Rank, N_tile, …**
  - **Nouvelles fonctions de la clause GROUP BY : CUBE, ROLLUP, GROUPING SETS**
  - **Fonction GROUPING**
  - **Fenêtre glissante**
- **Exploitation en OALP SQL d'un entrepôt sous Oracle**

Plan

Bibliographie

Ouvrages :

- Benitez-Guerrero E., C. Collet, M. Adiba, « Entrepôts de données : Synthèse et analyse », Rapport de recherche IMAG N°IMAG-RR - 99-1017-I, 1999.
- Gardarin G., « Internet/intranet et bases de données », Ed. Eyrolles, 1999, ISBN 2-212-09069-2.
- Han J., Kamber M., « Data Mining: Concepts and Techniques », Morgan Kaufmann Publishers, 2004.
- Kimball R., M. Ross, « Entrepôts de données : guide pratique de modélisation dimensionnelle », 2°édition, Ed. Vuibert, 2003, ISBN : 2-7117-4811-1.
- Gray J. et al., Data Cube : A telationnal Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, Data Mining and Knowledge Discovery 1, 29-53, 1997.
- Zemke F. et al., Introduction to OLAP functions, ISO/IEC JTC1/SC32 WG3 :YGJ-068 – ANSI NCITS H2-99-154r2, 1999.
- …

Cours :

- Cours de F. Bentayeb, O. Boussaid, J. Darmont, S. Rabaseda, Univ. Lyon 2
- Cours de P. Marcel, Univ. de Tours
- Cours de K.J. Han, Kon-Ku University
- Cours de M. Adiba et M.C. Fauvet, Univ. Grenoble
- Cours de H. Garcia-Molina, Stanford Univ
- Cours de J. Akoka, I. Comyn-Wattiau, CNAM, Paris
- ….

1 – Evolution de SQL pour l'OLAP

- **Evolution de la norme SQL : OLAP SQL 3 / SQL 99 extensions**
- **Nouvelles fonction SQL d'agrégation : Rank, N_tile, …**
- **Nouvelles fonctions de la clause GROUP BY : CUBE, ROLLUP, GROUPING SETS**
- **Fonction GROUPING**
- **Fenêtre glissante**

Evolution de la norme SQL

- **SQL/86**
- **SQL/89 (FIPS 127-1)**
- **SQL/89 with Integrity Enhancement**
- **SQL/92** July 92
  - Entry Level (FIPS 127–2)
  - Intermediate Level
  - Full Level
- **SQL CLI** Sept 95
- **SQL PSM** Nov 96

- **SQL/3 (SQL 99)**
  - SQL Framework July 99
  - SQL Foundation July 99
  - SQL Call Level Interface(CLI) Sept 99
  - SQL Persistent Stored Modules(PSM) July 99
  - SQL Language Bindings withdrawal
  - SQL Management of External Data Dec 00
  - SQL Object Language Bindings Aug 00

SQL99 Overview

- **Superset of SQL/92** : Completely upward compatible ("**object-oriented SQL**")
- **Significantly larger than SQL/92**
- **Object-Relational extensions :**
  - User-defined data types
  - Reference types
  - Collection types (e.g., arrays)
  - Large object support (LOBs)
  - Table hierarchies
- **Triggers**
- **Stored procedures and user-defined functions**
- **Recursive queries**
- **OLAP extensions (CUBE, ROLLUP, expressions in ORDER BY)**
- **SQL procedural constructs**
- **Expressions in ORDER BY**
- **Savepoints**
- **Update through unions and joins**

Extensions de SQL-3 / SQL-99 pour OLAP :

- **Nouvelles fonctions SQL d'agrégation:**
  - **N_tile**
  - **rank, dense_rank**
  - **every**
  - **any**
  - **some**
- **Nouvelles fonctions de la clause GROUP BY :**
  - **ROLLUP**
  - **CUBE**
  - **GROUPING**
  - **GROUPING SETS**
- **Fenêtre glissante :**
  - **WINDOWS/OVER/PARTITION, …**

Nouvelles fonctions SQL pour l'OLAP

Fonctions classiques d'agrégation de SQL : Count, Max, Min, Sum, Avg

Ajout de **nouvelles fonctions** pour faire des calculs comme c'est possible dans les tableurs :

N_tile(expression, n):

- On **calcule le domaine de l'expression** en fonction des **valeurs** qui apparaissent dans la colonne.
- Ce domaine est alors **divisé en n intervalles** de **tailles approximativement égales**.
- La fonction retourne alors le **numéro de l'intervalle qui contient la valeur de l'expression**.

Ex1 : si le solde du compte est parmi les 10% les plus élevés, N_tile(compte.solde, 10) retourne 10.

Ex2 : Trouver le min, le max et la moyenne des températures parmi les 10% les plus élevées.

```
SELECT Percentile, MIN(Temp), AVG(Temp), MAX(Temp)
FROM  Weather
GROUP BY N_tile(Temp,10) as Percentile
HAVING  Percentile = 10;
```

Nouvelles fonctions SQL de classement

rank(expression): rang de l'expression dans l'ensemble de toutes les valeurs du domaine.

Par exemple, s'il y a N valeurs dans une colonne, et si l'expression est la plus grande alors N, si c'est la plus petite alors 1

Soit la table : population(country, number)

Ex. : Trouver le classement de chaque pays :

```
SELECT country, rank() OVER (ORDER BY number DESC) AS n_rank
FROM population
```

La clause ORDER BY est nécessaire pour retourner les résultats triés :

```
SELECT country, rank() OVER (ORDER BY number DESC) AS n_rank
FROM population ORDER BY n_rank
```

Une utilisation de **rank** : trouver les n-premiers (syntaxe DB2) :

Ex. : Trouver les 5 premiers pays les plus peuplés :

```
SELECT country, rank() OVER (ORDER BY number DESC) AS n_rank
FROM population
ORDER BY n_rank
FETCH FIRST 5 ROWS ONLY
```


Autres nouvelles fonctions SQL

dense_rank(expression): avec la fonction rank(), s'il y a 2 tuples qui ont les mêmes valeurs et sont classés de rang n, le tuple suivant est de rang n+2.

Avec dense_rank() le tuple suivant a le rand n+1.

Ex. : Trouver le classement de chaque pays :

```
SELECT country, dense_rank() OVER (ORDER BY number DESC) AS n_rank
FROM population
```

Every(expression) : **vrai** ssi la valeur de son **argument (expression) est vraie pour tous les tuples**, **sinon faux** (∀)

Soit la table : Film_v(titre, type_film, année, magasin, qté_stock, qté_vendus)

Ex. : Trouver la quantité maxi et mini de films vendus à plus de 10 exemplaires :

```
SELECT COUNT(*), MAX(qté-vendus), MIN(qté-vendus),
       SUM(qté-vendus), AVG(qté-vendus)
FROM   Film_v
GROUP BY magasin Having EVERY(qté-vendus>=10)
```

Any(expression) ou **Some**(expression) : **vrai** ssi la valeur de son **argument (expression) est vraie pour au moins un tuple**, **sinon faux** (∃)

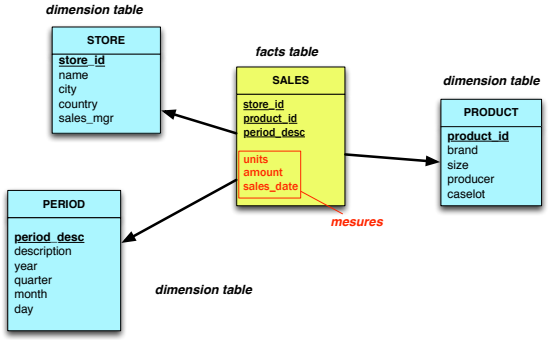
Nouvelles fonctions de la clause GROUP BY

CUBE, **ROLLUP** et **GROUPING SETS** produisent un ensemble de tuples, équivalent à un **UNION ALL** de tuples groupés différemment :

- **ROLLUP** calcule des agrégats (SUM, COUNT, MAX, MIN, AVG) à différents niveaux d'agrégation
- **CUBE** est similaire à **ROLLUP** mais permet de calculer toutes les combinaisons d'agrégations
- **GROUPING SETS** permet d'éviter le calcul du cube, quand il n'est pas globalement nécessaire
- Les fonctions **GROUPING** précise le groupe d'appartenance de chaque tuple pour calculer les sous-totaux et les filtres

Extensions de SQL 99 pour OLAP : un ED

Schéma en étoile:



Création du cube « Sales » :

```
CREATE VIEW Sales AS
     (SELECT ds.*, YEAR(sales_date) AS year, MONTH(sales_date) AS
     month, DAY(sales_date) AS day
FROM (Sales NATURAL JOIN Store NATURAL JOIN Product
     NATURAL JOIN Period) ds
```

GROUP BY ROLLUP (1)

GROUP BY ROLLUP :

- **calcule des agrégats** (SUM, COUNT, MAX, MIN, AVG) **à tous les niveaux de totalisation sur une hiérarchie de dimensions**
- et **calcule le total général :**
  - Selon l'ordre de **gauche à droite dans la clause** GROUP BY
  - S'il y a **n colonnes de regroupements**, GROUP BY ROLLUP génère **n+1 niveaux de totalisation**

Exemples :

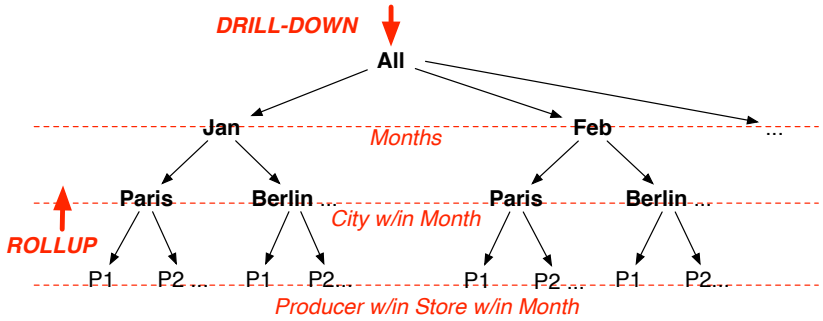
```
GROUP BY ROLLUP (year, month, day)
GROUP BY ROLLUP (country, city)
GROUP BY ROLLUP (month, city, producer)
…
```

Simplifie et accélère la maintenance des tables de synthèse.

GROUP BY ROLLUP (2)

Ex (1) :

```
SELECT month, city, producer, SUM(units) AS sum_units
FROM Sales
WHERE year = 2014
GROUP BY ROLLUP (month, city, producer) ;
```



GROUP BY ROLLUP (3)

Ex 1. : Total des ventes par pays et responsable des ventes pour chaque mois de 2014, avec sous-totaux pour chaque mois, et grand total :

```
SELECT month, country, sales_mgr, SUM(amount)
FROM Sales
WHERE year = 2014
GROUP BY ROLLUP(month, country, sales_mgr)
```

Month	Country	Sales-mgr	SUM(amoun	
April	France	Martin	25000	
April	France	Smith	15000	
April	France	-	40000	*Total France/April*
April	Germany	Smith	15000	
April	Germany	-	15000	*Total Germany/April*
April	-	-	55000	*Tot. April*
May	France	Martin	25000	
May	France	-	25000	*Total France/May*
May	Germany	Smith	15000	
May	Germany	-	15000	*Total France/May*
May	-	-	40000	*Total May*
-	-	-	95000	*Grand Total*

GROUP BY CUBE (1)

GROUP BY CUBE : **calcule des agrégats** (SUM, COUNT, MAX, MIN, AVG) **à différents niveaux d'agrégation** comme ROLLUP mais de plus permet de **calculer toutes les combinaisons d'agrégations :**

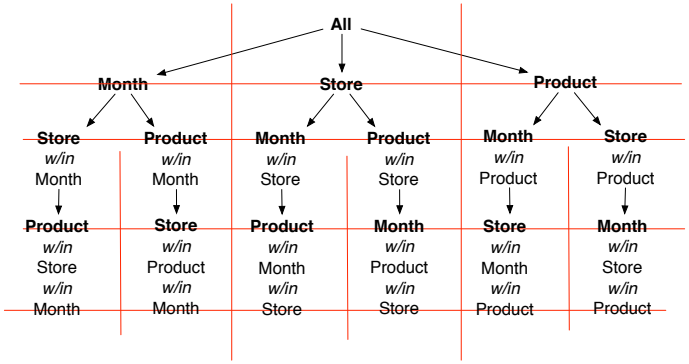
- GROUPE BY CUBE créé des **sous-totaux pour toutes les combinaisons possibles d'un ensemble de colonnes de regroupement**
- Si la clause CUBE contient **n colonnes**, CUBE calcule **2n combinaisons de totaux**
- Intéressant pour des **colonnes** représentant des **dimensions appartenant** à des **hiérarchies différentes**

GROUP BY CUBE est une alternative plus performante au UNION ALL

GROUP BY CUBE (2)

Ex 1. : Tous les totaux et sous totaux des quantités de produits vendus par ville et produit pour chaque mois de 2014 :

```
SELECT month, city, product_id, SUM(units)
FROM Sales
WHERE year = 2014
GROUP BY CUBE (month, city, product.id)
```



GROUP BY CUBE (1)

Ex.2 : Tous les totaux et sous totaux des ventes par pays et responsable des ventes pour chaque mois de 2014

```
SELECT month, country, sales_mgr, SUM(amount)
FROM Sales
WHERE year = 2014
GROUP BY CUBE(month, country, sales_mgr)
```

Month	Country	Sales-mgr	SUM(amount)	
April	France	Martin	25000	
April	France	Smith	15000	
April	France	-	40000	*Total France/April*
April	Germany	Smith	15000	
April	Germany	-	15000	*Total Germany/April*
April	-	Martin	25000	*Total Martin/April*
April	-	Smith	30000	*Total Smith/April*
April	-	-	55000	*Total April*
May	France	Martin	25000	
May	France	-	25000	*Total France/May*
May	Germany	Smith	15000	
May	Germany	-	15000	*Total Germany/May*
May	-	Martin	25000	*Total Martin/May*
May	-	Smith	15000	*Total Smith/May*
May	-	-	40000	*Total May*
-	France	Martin	50000	*Total Martin/France*
-	France	Smith	15000	*Total Smith/France*
-	France	-	65000	*Total France*
-	Germany	Smith	30000	*Total Smith/Germany*
-	Germany	-	30000	*Total Germany*
-	-	Martin	50000	*Total Martin*
-	-	Smith	45000	*Total Smith*
-	-	-	95000	*Grand Total*

GROUP BY GROUPING SETS (1)

GROUP BY **GROUPING SETS** :

- **Utilisé avec les agrégations usuelles** (MAX, MIN, SUM, AVG, COUNT, ...),
- **permet des groupements multiples** (month, country) et (month, sales_mgr) ; les résultats peuvent être restreints par une **clause HAVING**

Ex. : Total des ventes pour chaque mois de l'année 2014, par pays et par responsable des ventes :

```
SELECT month, country, sales_mgr, SUM(amount)
FROM Sales
WHERE year = 2014
GROUP BY GROUPING SETS((month, country),(month, sales_mgr))
```

Month	Country	Sales-mgr	SUM(amount)	
April	France	-	40000	*Total France/April*
April	Germany	-	15000	*Total Germany/April*
April	-	Martin	25000	*Total Martin/April*
April	-	Smith	30000	*Total Smith/April*
May	France	-	25000	*Total France/May*
May	Germany	-	15000	*Total Germany/May*
May	-	Martin	25000	*Total Martin/May*
May	-	Smith	15000	*Total Smith/May*

GROUP BY GROUPING SETS (2)

Tuple de grand total : **Une syntaxe particulière permet d'inclure un tuple « grand total » dans les résultats :**

- les grands totaux » sont générés implicitement avec **ROLLUP** et **CUBE**
- la syntaxe permet aux grands totaux d'être générés sans agrégat supplémentaire

Ex. 2 : Total de ventes par mois, pays, et responsable de vente, et aussi grand total de vente :

```
SELECT month, country, sales_mgr, SUM(amount)
FROM Sales
WHERE year = 2014
GROUP BY GROUPING SETS((month, country), () )
```

Month	Country	Sales-mgr	SUM(amount)
April	France	Martin	25000
April	France	Smith	15000
April	Germany	Smith	15000
May	France	Martin	25000
May	Germany	Smith	15000
-	-	-	95000

La fonction GROUPING

la fonction GROUPING : créé une **nouvelle colonne avec des 0 et des 1**, permettant la **détection de tuples de total** qui sont générés lors de l'exécution de CUBE ou ROLLUP (1 pour les tuples de total et 0 pour les autres)

Ex. : Tous les totaux et sous totaux des ventes par pays et responsable des ventes pour chaque mois de 2014

```
SELECT month, country, sales_mgr, SUM(amount), GROUPING(sales_mgr)
FROM Sales
WHERE year = 2014
GROUP BY ROLLUP (month, country, sales_mgr)
```

Month	Country	Sales-mgr	SUM(amount)	GROUPED
April	France	Martin	25000	0
April	France	Smith	15000	0
April	France	-	40000	1
April	Germany	Smith	15000	0
April	Germany	-	15000	1
April	-	-	55000	1
May	France	Martin	25000	0
May	France	-	25000	1
May	Germany	Smith	15000	0
May	Germany	-	15000	1
May	-	-	40000	1
-	-	-	95000	1

Fenêtre glissante (1)

Permet de répondre à des questions comme :

- *Trouver pour chaque mois, les ventes de chaque magasin en faisant la moyenne entre le mois courant et les deux mois précédents : faire la moyenne sur les 3 derniers mois*
- *Trouver pour chaque magasin, les ventes cumulées pour chaque mois de l'an dernier, ordonnées par mois*
- *Comparer les ventes de chaque mois avec la moyenne des ventes de chaque magasin pendant les 3 mois précédents*

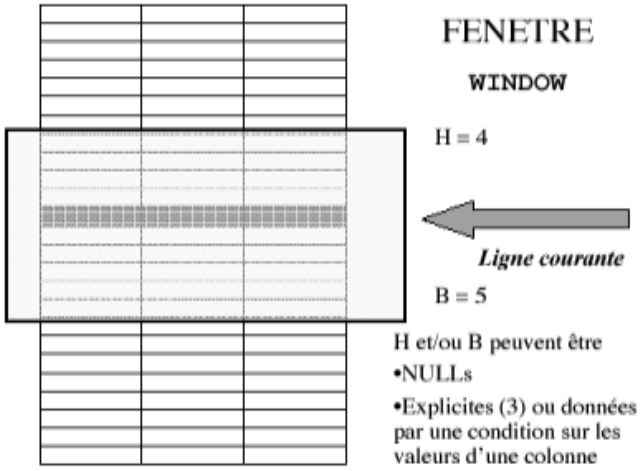
=> Il faut définir une **fenêtre glissante du temps** et y effectuer des cumuls, moyennes, dérivations, …

Notion de FENETRE (WINDOW) :

- Dans une requête, **une fenêtre est une sélection de lignes** (tuples) utilisée pour un certain calcul **par rapport à la ligne courante**
- La taille de la fenêtre est **un nombre de lignes** mais peut être exprimée de différentes façons (nombre entier ou par une condition)
- Une fenêtre se déclare soit explicitement par une clause **WINDOW** ou directement dans le **SELECT** où elle est utilisée

Fenêtre glissante (2)

((source : Adiba & Fauvet, 2005)



Fenêtre glissante (3)

3 paramètres :

- Partitionnement **PARTITION :** Chaque ligne de la partition a des valeurs égales sur un ensemble de colonnes (analogue au GROUP BY)
- Ordre **ORDER :** Ordre des lignes dans la partition comme un tri partiel sur 1 ou plusieurs colonnes (attention aux NULLs)
- Cadre **FRAMING** : Définit la taille de la fenêtre de manière physique (nombre de lignes **ROWS**) ou logique (condition, plage de valeurs **RANGE**) - Ex: ROWS 2 PRECEDING

Ex.1: Soit la table : population(country, year, number)

*Trouver les populations de chaque pays et année, calculer la moyenne du taux de croissance de la population pour chaque pays et année, sur la base des années courante, **précédente** et **suivante** :*

```
SELECT country, year, avg(population)
       OVER (ORDER BY country, year
       ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) AS p_avg
FROM population
ORDER BY country, year, p_avg;
```

Fenêtre glissante (4)

Ex.2 : Soit table Fvm(magasin, anmois, qté_totale) *(source : Adiba & Fauvet, 2005)*

Calculer pour chaque magasin et pour chaque mois du dernier tiers de 2001 le total des ventes, ainsi que la moyenne glissante des quantités vendues avec les 2 mois précédents.

```
SELECT h.magasin, h.anmois, h.qté_totale,
       AVG(h.qté_totale) OVER w AS moygliss
FROM   Fvm h
WHERE  h.anmois BETWEEN 200109 AND 200112
WINDOW w AS
 (PARTITION BY h.magasin ORDER BY h.anmois ROWS 2 PRECEDING)
```

Magasin	anmois	Qté_totale	moygliss
M1	200109	46936	46936
M1	200110	22842	34889
M1	200111	30927	33568
M1	200112	51751	35173
M2	200109	20554	20554
M2	200110	9418	14986
M2	200111	13600	14524
M2	200112	24977	15998
M3	200109	23157	23157
M3	200110	26591	24874
M3	200111	31565	27104
M3	200112	48153	35466

Septembre

Moyenne sept & oct

Sept, oct, nov

Oct, nov, décembre

Et on recommence

Fenêtre glissante – WINDOW : groupe incomplet (6)

Ex.2 : Soit table Fvm(magasin, anmois, qté_totale) *(source : Adiba & Fauvet, 2005)*

Calculer pour chaque magasin et pour chacun des 4 derniers mois 2001, le total des ventes ainsi que la moyenne glissante des quantités vendues sur les 2 mois précédents à condition qu'il y ait bien 2 mois, sinon NULL.

```
SELECT h.magasin, h.anmois, h.qté_totale,
       CASE WHEN Count(*) OVER w < 3 THEN NULL
       ELSE AVG(h.qté_totale) OVER w End AS moygliss
FROM Fvm h  Where h.anmois BETWEEN 200109 AND 200112
WINDOW w AS
 (PARTITION BY h.magasin ORDER BY h.anmois ROWS 2 PRECEDING)
```

Magasin	anmois	Qté_totale	moygliss
M1	200109	46936	NULL
M1	200110	22842	NULL
M1	200111	30927	33568
M1	200112	51751	35173
M2	200109	20554	NULL
M2	200110	9418	NULL
M2	200111	13600	14524
M2	200112	24977	15998
M3	200109	23157	NULL
M3	200110	26591	NULL
M3	200111	31565	27104
M3	200112	48153	35466

Septembre

Moyenne sept & oct

Sept, oct, nov

Oct, nov, décembre

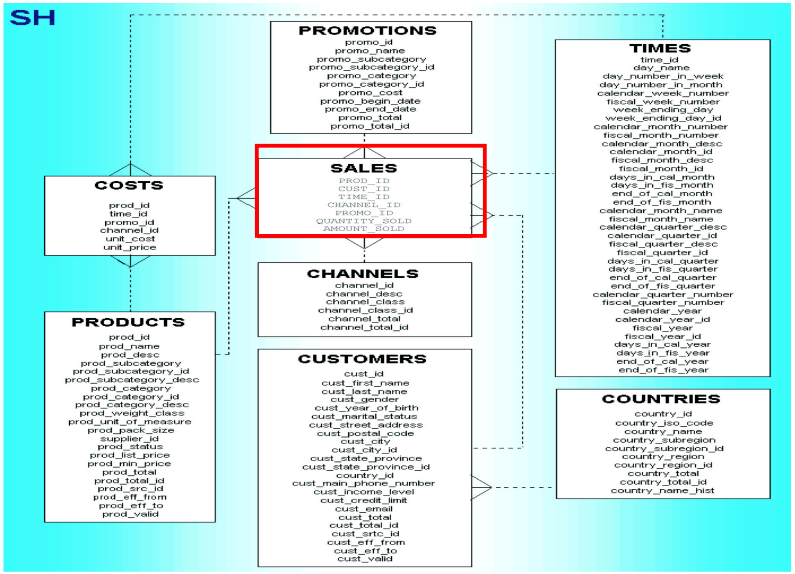
Et on recommence

2 – Exploitation OLAP en SQL d'un entrepôt sous Oracle

- L'entrepôt de données concerné
- GROUP BY ROLLUP
- GROUP BY ROLLUP partiel
- GROUP BY CUBE
- GROUP BY CUBE partiel
- La fonction GROUPING
- La fonction GROUPING_ID
- GROUPING SET
- Colonnes composites

(Sources : cours formation Oracle, et supports J. Akoka & I. Comyn-Wattiau)

L'entrepôt de données concerné (1)



L'entrepôt de données concerné (2)

```
CREATE TABLE channels (channel_id VARCHAR2(1), channel_desc VARCHAR2(20),
channel_class VARCHAR2(20));

CREATE TABLE times (time_id DATE, day_name VARCHAR2(9), day_number_in_week
NUMBER(1), day_number_in_month NUMBER(2), calendar_week_number NUMBER(2),
fiscal_week_number NUMBER(2), week_ending_day DATE, calendar_month_number
NUMBER(2), fiscal_month_number NUMBER(2), calendar_month_desc VARCHAR2(8),
fiscal_month_desc VARCHAR2(8), days_in_cal_month NUMBER, days_in_fis_month
NUMBER, end_of_cal_month DATE, end_of_fis_month DATE, calendar_month_name
VARCHAR2(9), fiscal_month_name VARCHAR2(9), calendar_quarter_desc VARCHAR2(7),
fiscal_quarter_desc VARCHAR2(7), days_in_cal_quarter NUMBER,
days_in_fis_quarter NUMBER, end_of_cal_quarter DATE, end_of_fis_quarter DATE,
calendar_quarter_number NUMBER(1), fiscal_quarter_number NUMBER(1),
calendar_year NUMBER(4), fiscal_year NUMBER(4), days_in_cal_year NUMBER,
days_in_fis_year NUMBER, end_of_cal_year DATE, end_of_fis_year DATE);

CREATE TABLE customers (cust_id NUMBER, cust_first_name VARCHAR2(20),
cust_last_name VARCHAR2(40), cust_gender VARCHAR2(1), cust_year_of_birth
NUMBER(4), cust_marital_status VARCHAR2(20), cust_street_address VARCHAR2(40),
cust_postal_code VARCHAR2(10), cust_city VARCHAR2(30), cust_state_province
VARCHAR2(40), country_id VARCHAR2(2), cust_main_phone_number VARCHAR2(25),
cust_income_level VARCHAR2(30), cust_credit_limit NUMBER, cust_email
VARCHAR2(30));

CREATE TABLE sales (prod_id NUMBER(6), cust_id NUMBER, time_id DATE,
channel_id VARCHAR2(1), promo_id NUMBER(6), quantity_sold NUMBER(3),
amount_sold NUMBER(10,2));
```


L'entrepôt de données concerné (3)

Création d'index et clés :

```
ALTER TABLE channels ADD CONSTRAINT pk_channel PRIMARY KEY (channel_id) USING INDEX
TABLESPACE indx_sml;

CREATE INDEX ix_channels_index_channel_desc ON channels(channel_desc)
TABLESPACE indx_sml;

ALTER TABLE times ADD CONSTRAINT pk_time_pk PRIMARY KEY (time_id) USING INDEX
TABLESPACE indx_sml;

CREATE INDEX ix_times_calendar_month_desc ON times(calendar_month_desc)
TABLESPACE indx_sml;

ALTER TABLE customers ADD CONSTRAINT pk_customers PRIMARY KEY (cust_id) USING INDEX
TABLESPACE indx_sml;

CREATE INDEX ix_customers_country_id ON customers(country_id)
TABLESPACE indx_sml;

CREATE INDEX ix_sales_time_id ON sales(time_id)
TABLESPACE indx_sml;

CREATE INDEX ix_sales_cust_id ON sales(cust_id)
TABLESPACE indx_sml;

CREATE INDEX ix_sales_channel_id ON sales(channel_id)
TABLESPACE indx_sml;
```


Extensions OLAP de SQL dans Oracle : GROUP BY ROLLUP

GROUP BY ROLLUP calcule tous les niveaux de totalisation sur une hiérarchie de dimensions et calcule le total général (selon l'ordre de gauche à droite dans la clause GROUP BY) :

```
GROUP BY ROLLUP(année, mois, jour)
GROUP BY ROLLUP(pays, état, ville)
```

Requête 1:

```
SELECT channel_desc,calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id = channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY ROLLUP(channel_desc, calendar_month_desc,
      country_iso_code);
```

- GROUP BY ROLLUP simplifie et accélère la maintenance des tables de synthèse
- S'il y a n colonnes de regroupements, GROUP BY ROLLUP génère n+1 niveaux de totalisation

GROUP BY ROLLUP (1)

Résultats requête 1 :

CHANNEL_DESC	CALENDAL	COUNTRY	SALES$
Internet	2000-09	GB	16,569
Internet	2000-09	US	124,224
Internet	2000-09		140,793
Internet	2000-10	GB	14,539
Internet	2000-10	US	137,054
Internet	2000-10		151,593
Internet			292,387
Direct Sales	2000-09	GB	85,223
Direct Sales	2000-09	US	638,201
Direct Sales	2000-09		723,424
Direct Sales	2000-10	GB	91,925
Direct Sales	2000-10	US	682,297
Direct Sales	2000-10		774,222
Direct Sales			1,497,646
			1,790,032

GROUP BY ROLLUP (2)

Exemple :

GROUP BY ROLLUP(channel_desc, calendar_month_desc, country_iso_code);

Fonctionnement du GROUP BY ROLLUP :

- **GROUP BY** ***(channel_desc, calendar_month_desc, country_iso_code)***
- **Premier niveau de totalisation,** tous pays confondus pour toutes les combinaisons de ***channel_desc*** et ***calendar_month_desc***
- **Deuxième niveau** de **totalisation par canal de distribution** ***(channel_desc)***
- **Total général**

GROUP BY ROLLUP partiel (1)

Permet de totaliser à certains niveaux sur une dimension :

Ex : GROUP BY expr1, ROLLUP(expr2, expr3) :

- Crée 2+1 = 3 niveaux d'agrégation : (expr1, expr2, expr3), (expr1, expr2) et (expr1)
- Pas de total général

Ex. requête 2 :

Xxx

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY channel_desc,
  ROLLUP(calendar_month_desc, country_iso_code);
```


GROUP BY ROLLUP partiel (2)

Résultats requête 2 :

CHANNEL_DESC	CALENDAR	COUNTRY	SALES$
Internet	2000-09	GB	16,569
Internet	2000-09	US	124,224
Internet	2000-09		140,793
Internet	2000-10	GB	14,539
Internet	2000-10	US	137,054
Internet	2000-10		151,593
Internet			292,387
Direct Sales	2000-09	GB	85,223
Direct Sales	2000-09	US	638,201
Direct Sales	2000-09		723,424
Direct Sales	2000-10	GB	91,925
Direct Sales	2000-10	US	682,297
Direct Sales	2000-10		774,222
Direct Sales			1,497,646

GROUP BY CUBE (1)

GROUP BY CUBE crée des **sous-totaux pour toutes les combinaisons possibles d'un ensemble de colonnes de regroupement :**

Ex.: CUBE sur les dimensions temps, géographie et canal de distribution calcule tous les sous-totaux des ventes pour toutes les combinaisons

- Si la clause CUBE contient n colonnes, CUBE calcule 2n combinaisons de totaux
- Intéressant pour des colonnes représentant des dimensions appartenant à des hiérarchies différentes
- Le GROUP BY CUBE est une alternative plus performante que le UNION ALL

Ex. requête 3 :

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND sales.channel_id= channels.channel_id
       AND customers.country_id = countries.country_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY CUBE(channel_desc, calendar_month_desc,
     country_iso_code);
```

GROUP BY CUBE (2)

Résultat requête 3 :

CHANNEL_DESC	CALENDAR	COUTRY	SALES$
			1,790,032
		GB	208,257
		US	1,581,775
	2000-09		864,217
	2000-09	GB	101,792
	2000-09	US	762,425
	2000-10		925,815
	2000-10	GB	106,465
	2000-10	US	819,351
Internet			292,387
Internet		GB	31,109
Internet		US	261,278
Internet	2000-09		140,793
Internet	2000-09	GB	16,569
Internet	2000-09	US	124,224
Internet	2000-10		151,593
Internet	2000-10	GB	14,539
Internet	2000-10	US	137,054
Direct Sales			1,497,646
Direct Sales		GB	177,148
Direct Sales		US	1,320,497
Direct Sales	2000-09		723,424
Direct Sales	2000-09	GB	85,223
Direct Sales	2000-09	US	638,201
Direct Sales	2000-10		774,222
Direct Sales	2000-10	GB	91,925
Direct Sales	2000-10	US	682,297

GROUP BY CUBE partiel (1)

GROUP BY CUBE partiel

GROUP BY expr1, CUBE(expr2, expr3)

- calcule 4 niveaux de regroupement : (expr1, expr2, expr3), (expr1, expr2), (expr1, expr3), et (expr1)
- Ne calcule pas de total général

Ex. requête 4 :

Xxx

```
SELECT  channel_desc, calendar_month_desc, country_iso_code,
        TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM    sales, customers, times, channels, countries
WHERE   sales.time_id = times.time_id
        AND sales.cust_id = customers.cust_id
        AND customers.country_id=countries.country_id
        AND sales.channel_id = channels.channel_id
        AND channel_desc IN ('Direct Sales', 'Internet')
        AND calendar_month_desc IN ('2000-09', '2000-10')
        AND country_iso_code IN ('GB', 'US')
GROUP BY channel_desc, CUBE(calendar_month_desc,
      country_iso_code);
```


GROUP BY CUBE partiel (2)

Résultats requête 4 :

CHANNEL_DESC	CALENDAR	COUTRY	SALES$
Internet			292,387
Internet		GB	31,109
Internet		US	261,278
Internet	2000-09		140,793
Internet	2000-09	GB	16,569
Internet	2000-09	US	124,224
Internet	2000-10		151,593
Internet	2000-10	GB	14,539
Internet	2000-10	US	137,054
Direct Sales			1,497,646
Direct Sales		GB	177,148
Direct Sales		US	1,320,497
Direct Sales	2000-09		723,424
Direct Sales	2000-09	GB	85,223
Direct Sales	2000-09	US	638,201
Direct Sales	2000-10		774,222
Direct Sales	2000-10	GB	91,925
Direct Sales	2000-10	US	682,297

Remarques générales sur CUBE et ROLLUP

- Attention, les **lignes de totalisation contiennent des valeurs nulles**
- **Problème si la table initiale contient aussi des valeurs nulles**
- On a **besoin** parfois **d'autres fonctions que des sommes**
- Pour exploiter le résultat d'un CUBE ou d'un ROLLUP, on a besoin de savoir :
  - **quelles** sont les **lignes de totalisation** et
  - **combien** de **niveaux de totalisation** comporte **chaque ligne**

La fonction GROUPING (1)

Fonction **GROUPING** **s'insère dans le** SELECT, et a **un argument** qui est **un nom de colonne** :

- Elle renvoie 1 quand elle rencontre une valeur nulle créée par **ROLLUP** ou **CUBE** Sinon elle renvoie 0
- Valeur non nulle et Valeur nulle initialement contenue dans la table

Exemple requête 5:

Xxx

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$,
       GROUPING(channel_desc) AS Ch,
       GROUPING(calendar_month_desc) AS Mo,
       GROUPING(country_iso_code) AS Co
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY ROLLUP(channel_desc, calendar_month_desc,
       country_iso_code);
```


La fonction GROUPING (2)

Résultat requête 5 :

CHANNEL_DESC	CALENDAR	COUNTRY	SALES$	CH	MO	CO
Internet	2000-09	GB	16,569	0	0	0
Internet	2000-09	US	124,224	0	0	0
Internet	2000-09		140,793	0	0	1
Internet	2000-10	GB	14,539	0	0	0
Internet	2000-10	US	137,054	0	0	0
Internet	2000-10		151,593	0	0	1
Internet			292,387	0	1	1
Direct Sales	2000-09	GB	85,223	0	0	0
Direct Sales	2000-09	US	638,201	0	0	0
Direct Sales	2000-09		723,424	0	0	1
Direct Sales	2000-10	GB	91,925	0	0	0
Direct Sales	2000-10	US	682,297	0	0	0
Direct Sales	2000-10 7		74,222	0	0	1
Direct Sale			1,497,646	0	1	1
			1,790,032	1	1	1

La fonction GROUPING (3)

Combinée avec la fonction Oracle **DECODE**, la fonction GROUPING permet d'améliorer la lisibilité :

Ex. requête 6 :

Xxx

```
SELECT DECODE(GROUPING(channel_desc), 1, 'Multi-channel sum',
       channel_desc) AS Channel,
       DECODE (GROUPING (country_iso_code),
       1, 'Multi-country sum',country_iso_code) AS Country,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc= '2000-09'
       AND country_iso_code IN ('GB', 'US')
GROUP BY CUBE(channel_desc, country_iso_code);
```


La fonction GROUPING (4)

Résultats requête 6 :

CHANNEL	COUNTRY	SALES$
Multi-channel sum	Multi-country sum	864,217
Multi-channel sum	GB	101,792
Multi-channel sum	US	762,425
Internet	Multi-country sum	140,793
Internet	GB	16,569
Internet	US	124,224
Direct Sales	Multi-country sum	723,424
Direct Sales	GB	85,223
Direct Sales	US	638,201

La fonction GROUPING (5)

GROUPING avec HAVING permet de **sélectionner les niveaux de regroupement à afficher** (Par exemple, tous les niveaux sauf le moins agrégé)

Exemple requête 7 :

Xxx

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$,
       GROUPING(channel_desc) CH,
       GROUPING (calendar_month_desc) MO,
       GROUPING(country_iso_code) CO
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY CUBE(channel_desc, calendar_month_desc, country_iso_code)
       HAVING (GROUPING(channel_desc)=1
       AND GROUPING(calendar_month_desc)= 1
       AND GROUPING(country_iso_code)=1)
       OR (GROUPING(channel_desc)=1
       AND GROUPING (calendar_month_desc)= 1)
       OR (GROUPING(country_iso_code)=1
       AND GROUPING(calendar_month_desc)= 1);
```


La fonction GROUPING (6)

Résultat requête 7 :

CHANNEL_DESC	COUNTRY	SALES$	CH	MO	CO
	US	1,581,775	1	1	0
	GB	208,257	1	1	0
Direct Sales		1,497,646	0	1	1
Internet		292,387	0	1	1
		1,790,032	1	1	1

La fonction GROUPING_ID (1)

- GROUPING porte sur une seule colonne, pour tester les différents critères d'agrégation, il faut autant de colonnes de GROUPING que de colonnes de regroupement
- La fonction **GROUPING_ID synthétise en une seule fonction l'état d'agrégation sur les ensembles de critères**
- **Fonction GROUPING_ID CUBE(a,b) :**

Niveau d'agrégation	Vecteur de bits	GROUPING_ID
a,b	0 0	0
a	0 1	1
b	1 0	2
Total général	1 1	3

- **Très utile pour rafraîchir les vues matérialisées**

GROUPING SETS

Permet de spécifier des ensembles précis de regroupements

Xxx

```
SELECT manager_id, hire_date, count(*)
FROM employees
GROUP BY GROUPING SETS (manager_id, hire_date);
```

Équivalent à l'union suivante :

Xxx

```
SELECT manager_id, null hire_date, count(*)
FROM employees GROUP BY manager_id, 2
UNION
ALL SELECT null, hire_date, count(*)
FROM employees GROUP BY 1, hire_date;
```

GROUPING SETS

Requête 8 :

Xxx

```
SELECT country_iso_code, SUBSTR(cust_state_province,1,12),
       SUM(amount_sold),
       GROUPING_ID(country_iso_code, cust_state_province)
       GROUPING_ID, GROUP_ID()
FROM   sales, customers, times, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id=countries.country_id
       AND times.time_id= '30-OCT-00'
       AND country_iso_code IN ('FR', 'ES')
GROUP BY GROUPING SETS (country_iso_code,
ROLLUP(country_iso_code, cust_state_province));
```


GROUPING SETS

Résultats de la requête 8 :

COUNTRY	SUBSTR(cust_state_province	amount_sold	GROUPING_ID	GROUP_ID()
ES	Alicante	135,32	0	0
ES	Valencia	4133,56	0	0
ES	Barcelona	24,22	0	0
FR	Centre	74,3	0	0
FR	Aquitaine	231,97	0	0
FR	Rhone-Alpes	1624,69	0	0
FR	Ile-de-France	1860,59	0	0
		12372,05	3	0
ES		4293,1	1	0
FR		8078,95	1	0
ES		4293,1	1	1
FR		8078,95	1	1

GROUPING SETS

Requête 9 :

Xxx

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY GROUPING SETS((channel_desc, calendar_month_desc,
       country_iso_code), (channel_desc, country_iso_code),
       (calendar_month_desc, country_iso_code));
```

Calcule les agrégats pour 3 regroupements :

- (channel_desc, calendar_month_desc, country_iso_code)
- (channel_desc, country_iso_code)
- (calendar_month_desc, country_iso_code)

GROUPING SETS

GROUPING SETS	GROUP BY équivalent
GROUP BY GROUPING SETS (a, b, c)	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY c
GROUP BY GROUPING SETS (a, b, (b, c))	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY b, c
GROUP BY GROUPING SETS ((a, b, c))	GROUP BY a, b, c
GROUP BY GROUPING SETS (a,(b),())	GROUP BY a UNION ALL GROUP BY b UNION ALL GROUP BY ( )
GROUP BY GROUPING SETS (a, ROLLUP(b, c))	GROUP BY a UNION ALL GROUP BY ROLLUP (b, c)

Colonnes composites

- C'est une **collection de colonnes** qui est **traitée comme un tout dans les regroupements**
- Il suffit de les mettre entre parenthèses
- Exemple :

```
ROLLUP (année, (trimestre, mois), jour)
```

Calcule les regroupements suivants :

  - (année, trimestre, mois, jour)
  - (année, trimestre, mois)
  - (année)
  - ( )

- Permet de « sauter » certains niveaux de regroupement

Colonnes composites

Requête 10 :

Xxx

```
SELECT channel_desc, calendar_month_desc, country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM   sales, customers, times, channels, countries
WHERE  sales.time_id=times.time_id
       AND sales.cust_id=customers.cust_id
       AND customers.country_id = countries.country_id
       AND sales.channel_id= channels.channel_id
       AND channel_desc IN ('Direct Sales', 'Internet')
       AND calendar_month_desc IN ('2000-09', '2000-10')
       AND country_iso_code IN ('GB', 'US')
GROUP BY ROLLUP(channel_desc, calendar_month_desc,
     country_iso_code);
```

La requête **calcule les regroupements suivants** :

- (channel_desc, calendar_month_desc, country_iso_code)
- (channel_desc, calendar_month_desc)
- (channel_desc)
- ( )

Si on ne veut pas le 2 ième niveau, on mettra :

GROUP BY ROLLUP(channel_desc, (calendar_month_desc, country_iso_code));