

# Business Intelligence

## Chapitre 3

### Modélisation multi-dimensionnelle avancée

# Partie 1 : Généralités

Points traités :

- Gestion des faits

- Gestions des clés

- Types d'attributs de dimension

- Slow Changing Dimension

# Tables de faits avec faits et sans faits

- En général la table des faits comporte un ou plusieurs attributs représentant des faits, que l'on va sommer sur les dimensions lors de l'analyse
  - Par exemple une table des faits représentant des achats de produits pourra contenir la quantité de produits achetés, le chiffre d'affaire de la vente, ...
  - Cette table pourra être consulté par la requête suivante :

```
SELECT sum(v.quantite)
FROM ventes v JOIN date d
ON fk_date=pk_date
GROUP BY d.week
```

# Tables de faits avec faits et sans faits

- Dans certains cas, on mesure directement dans la table des faits des événements unitaires. Un fait est donc juste un enregistrement dans cette table, on parle alors d'une table de faits sans fait (Factless Fact Table)
  - La table des faits **ne contient que des clés étrangères**, et **aucune mesure** en tant que telle (c'est l'enregistrement qui est le fait)
  - Par exemple, on peut enregistrer des ventes de produits qui sont toujours vendus à l'unité, en vue d'une analyse en quantité (où le prix de vente n'intervient pas)

```
SELECT count (*)  
FROM ventes v JOIN date d  
ON fk_date=pk_date  
GROUP BY d.week
```

# Tables de faits avec faits et sans faits

- Pour analyser une table de faits sans faits, on utilise la fonction COUNT (on analyse le nombre de faits enregistrés)
  - Attention : on ne peut pas utiliser la fonction SUM car il n'y a rien à sommer
- Comme alternative, on peut ajouter un attribut avec la valeur constante 1
  - On peut ajouter une colonne qui contient la valeur 1 pour toutes les lignes
  - On matérialise ainsi le fait par une valeur, même si elle est toujours la même, et il est de nouveau possible de travailler avec SUM

# Clé artificielles

- « Every join between dimension and fact tables in the data warehouse should be based on meaningless integer **surrogate keys**. You should avoid using the natural operational production codes » (Kimball, Ross, 2008)

	AddressID	StreetNumber	StreetName	City	State	ZipCode	ModifiedDate
1	1	123	6th St.	Melbourne	FL	32904	2018-04-11 20:02:41.637
2	2	71	Pilgrim Avenue	Chevy Chase	MD	20815	2018-04-11 20:02:41.647
3	3	70	Bowman St.	South Windsor	CT	06074	2018-04-11 20:02:41.647
4	4	4	Goldfield Rd.	Honolulu	HI	96815	2018-04-11 20:02:41.647
5	5	44	Shirley Ave.	West Chicago	IL	60185	2018-04-11 20:02:41.650
6	6	514	S. Magnolia St.	Orlando	FL	32806	2018-04-11 20:02:41.650

# Clé artificielles

- L'usage de clés naturelle est plus simple au début, mais plus coûteux sur le long terme : les clés artificielles assurent l'indépendance aux évolutions futures du système opérationnel
  - Les clés artificielles sont plus performantes (entiers compressés)
  - Les clés artificielles permettent aussi de gérer les valeurs nulles (date...)
  - Elles sont aussi appelées clés de substitution ou clés surrogatoires

# Gestion des valeurs nulles

- « You must avoid null keys in the fact table. A proper design includes a row in the corresponding dimension table to identify that the dimension is not applicable to the measurement » (Kimball, Ross, 2008)
- Exemple
  - Lorsqu'un client qui ne possède pas de carte de fidélité achète un produit, il n'est pas possible de lier le fait à un client
  - On évite de mettre une valeur nulle en ajoutant une valeur "Client sans carte de fidélité" à la dimension



# Gestion des erreurs

- Il est souvent utile d'ajouter des valeurs (num ou text) dans une dimension afin de gérer les cas d'erreur dans les données
- Ces valeurs rendent compte d'une typologie comme par exemple :
  - Format invalide
  - Valeur tronquée
  - Référence inconnue
- Si l'on dispose de suffisamment d'information sur la cause de l'erreur, celle-ci peut être utilisée

# Faits semi-additifs

- Un fait est semi-additif s'il est additif sur une partie seulement des dimensions du modèle
- « All measures that record a static level (inventory levels, financial account balances, and measures of intensity such as room temperature) are inherently nonadditive across date dimension and possibly other dimensions. In these cases, the measure may be aggregated usefully across time, for example, by averaging over number of time periods » (Kimball, Ross, 2008)
  - Pour analyser les faits semi-additifs sur les dimensions sur lesquelles ils ne sont pas additifs, il faut faire des moyennes

# Types d'attributs de dimension

## 1. Attributs d'analyse

- La majorité des attributs d'une dimension servent aux analyses (ils sont utilisés dans les GROUP BY). Ils sont aussi appelés '**Attributs de regroupement**'

## 2. Attributs de description

- Certains attributs ne sont pas utiles à l'analyse, mais peuvent être conservés dans le modèle, afin d'améliorer la qualité des états, souvent parce qu'ils sont plus explicites pour identifier un enregistrement d'une dimension
- Par exemple, si l'on dispose d'un numéro de département pour l'analyse, le nom peut néanmoins être conservé à des fins d'amélioration des rapports
- Les attributs de description sont notés en *italique* dans le modèle dimensionnel et/ou annotés de la mention (d)

# Types d'attributs de dimension

## 3. Attributs d'agrégation de faits

- Certaines analyses requièrent de regrouper les faits en fonctions de valeurs elles-mêmes issues des faits
- Par exemple, les **produits les plus vendus** ou **les clients "high spender"** (qui dépensent le plus)
- Dans ce cas des attributs d'agrégation des faits sont pré-calculés au sein des dimensions concernées
- L'attribut est annoté d'un (a) dans le modèle dimensionnel

# La dimension Date

- Quasiment tous les DWh possèdent une dimension Date

Date Dimension
Date Key (PK)
Date
Full Date Description
Day of Week
Day Number in Epoch
Week Number in Epoch
Month Number in Epoch
Day Number in Calendar Month
Day Number in Calendar Year
Day Number in Fiscal Month
Day Number in Fiscal Year
Last Day in Week Indicator
Last Day in Month Indicator
Calendar Week Ending Date
Calendar Week Number in Year
Calendar Month Name
Calendar Month Number in Year
Calendar Year-Month (YYYY-MM)
Calendar Quarter
Calendar Year-Quarter
Calendar Half Year
Calendar Year
Fiscal Week
Fiscal Week Number in Year
Fiscal Month
Fiscal Month Number in Year
Fiscal Year-Month
Fiscal Quarter
Fiscal Year-Quarter
Fiscal Half Year
Fiscal Year
Holiday Indicator
Weekday Indicator
Selling Season
Major Event
SQL Date Stamp
... and more


# Slow Changing Dimension (SCD)

- La gestion des changements des attributs à changement lent dans les dimensions, est un enjeu de l'historisation dans le DWh
- Il y a 5 grands types de solution (Kimball, Ross, 2008) :
  - Type 1 : Remplacer la valeur (pas de gestion d'historique)
  - Type 2 : Ajouter une nouvelle ligne (multiplication du nombre de lignes)
  - Type 3 : Ajouter un attribut (gestion d'un seul niveau d'historique)
  - Type 3+ : Ajouter plusieurs attributs (changements prévisibles)
  - Type 6 (1+2+3) : Combiner les type 1, 2 et 3

# Slow Changing Dimension (SCD)

- Examples :
  - SCD type 1

Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z



Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	ABC922-Z

- SCD type 2

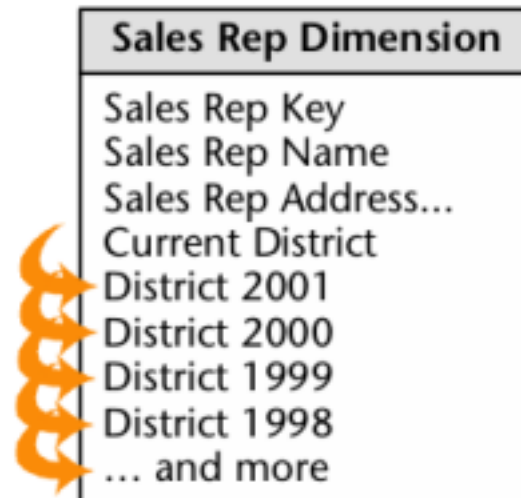
Product Key	Product Description	Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	ABC922-Z

# Slow Changing Dimension (SCD)

- SCD type 3

Product Key	Product Description	Department	Prior Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	➔ Education	ABC922-Z

- SCD type 3+







# Slow Changing Dimension (SCD)

- SCD type 6

Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Education	Education	ABC922-Z



Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Strategy	Education	ABC922-Z
25984	IntelliKidz 1.0	Strategy	Strategy	ABC922-Z



Product Key	Product Description	Current Department	Historical Department	SKU Number (Natural Key)
12345	IntelliKidz 1.0	Critical Thinking	Education	ABC922-Z
25984	IntelliKidz 1.0	Critical Thinking	Strategy	ABC922-Z
31726	IntelliKidz 1.0	Critical Thinking	Critical Thinking	ABC922-Z

# Partie 2 : Niveau logique

Points traités :

- ROLAP, MOLAP et HOLAP

- Représentation et manipulation du cube

- Exemples d'opérations OLAP

# Niveau logique

- Comment stocker les données multi-dimensionnelles ?
  - Dans des tables de la base de données relationnelle
  - Dans les espaces de travail analytiques des bases de données multidimensionnelle
- Description de la base multi-dimensionnelle selon la technologie utilisée :
  - ROLAP (Relational-OLAP)
  - MOLAP (Multidimensional-OLAP)
  - HOLAP (Hybrid-OLAP)

# ROLAP & MOLAP

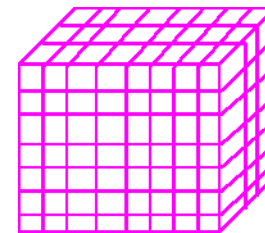
- ROLAP: Relational On-Line Analytical Processing
  - Les données sont stockées dans des tables relationnelles de fait et de dimensions
  - Exploiter l'expérience des modèles relationnels (SQL)
  - Utilise un SGBD relationnel pour stocker les données ainsi qu'un middleware pour implémenter les opérations spécifiques d'OLAP
- MOLAP: Multidimensional On-Line Analytical Processing
  - Utilisation de la notion de cube pour stocker les données multidimensionnelles
  - Les données sont stockées comme un tableau multidimensionnel
  - A ce jour, pas encore de cadre technologique commun pour le développement de tels systèmes : chaque produit est spécifique

# HOLAP

- HOLAP combines ROLAP et MOLAP
  - Les données sont partitionnées en deux sous ensembles :
    - Les données peu utilisées sont stockées dans des tables relationnelles
    - Les données fréquemment utilisées sont stockées dans des tableaux multidimensionnels
- La séparation est transparente pour l'utilisateur final
  - Données brutes dans ROLAP
  - Données agrégées dans MOLAP
- Exemple
  - Si le DW stocke des données de 1990 à ce jour, les années les plus récentes sont stockées dans des tableaux multi-dimensionnels alors que les autres années dans des tables relationnelles

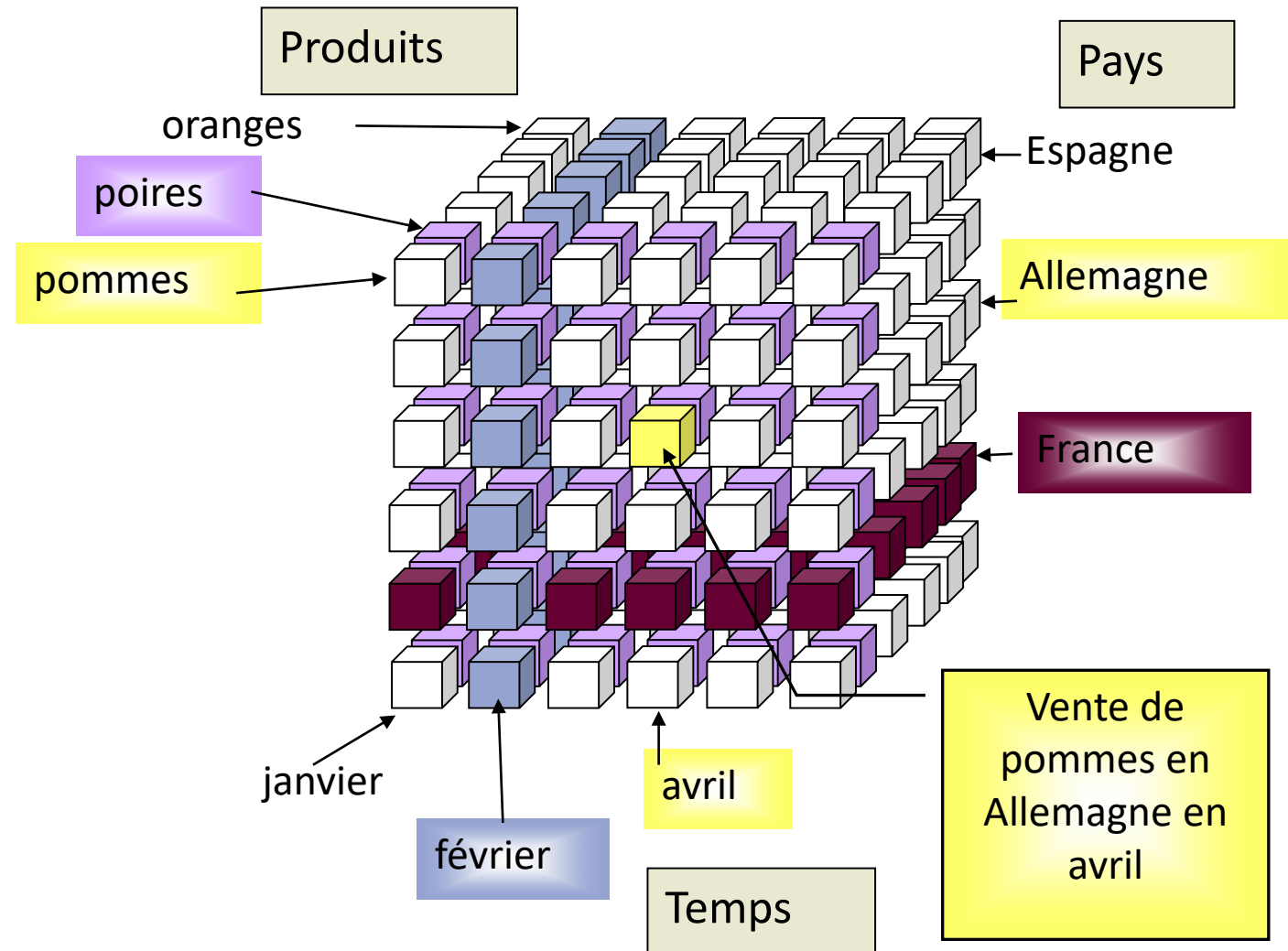
# Cube OLAP

- Modélisation multi-dimensionnelle des données facilitant l'analyse d'une quantité (mesure) selon différentes dimensions : temps, produit, localisation géographique, ....
  - Permet d'obtenir des informations déjà agrégées selon les besoins des utilisateurs
  - La représentation de l'information se fait par un cube à N dimensions
- Les calculs sont réalisés lors du chargement ou de la mise à jour du cube



# Cube OLAP

- Exemple



# Représentation et manipulation du cube

- Le cube de données est traditionnellement représenté sous forme de tables multi-dimensionnelles
- Le cube table est manipulée via différents opérateurs OLAP (SQL/MDX)
  - Fonctionnalités qui servent à faciliter l'analyse multi-dimensionnelles
  - Opérations réalisables sur le cube



# Représentation et manipulation du cube

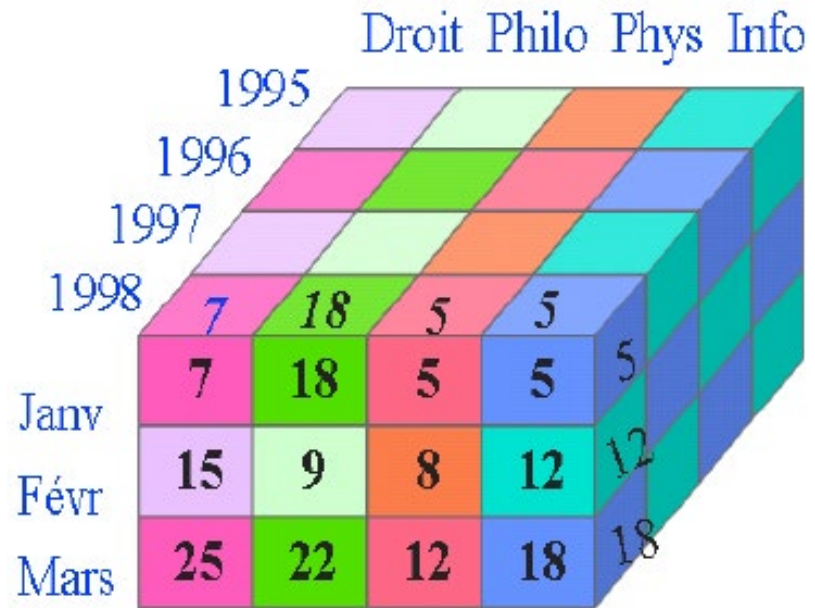
- La table multi-dimensionnelle
  - Correspond à une tranche du cube multi-dimensionnel
  - Présente les valeurs des mesures d'un fait en fonction des valeurs des paramètres des dimensions représentées en lignes et en colonnes étant données des valeurs des autres dimensions
  - Les lignes et les colonnes sont les axes selon lesquels le cube est exploré et chaque cellule contient la (ou les) mesure(s) calculée(s)

# Représentation et manipulation du cube

- Il existe plusieurs types d'opérateurs OLAP de manipulation du cube
  1. Changement de la granularité des données (forage)
  2. Sélection/Projection sur les données du cube
  3. Restructuration/Réorientation du cube
  4. Opérations entre cubes

# Représentation et manipulation du cube

- Exemple d'un cube simple sur lequel nous allons dérouler des opération OLAP

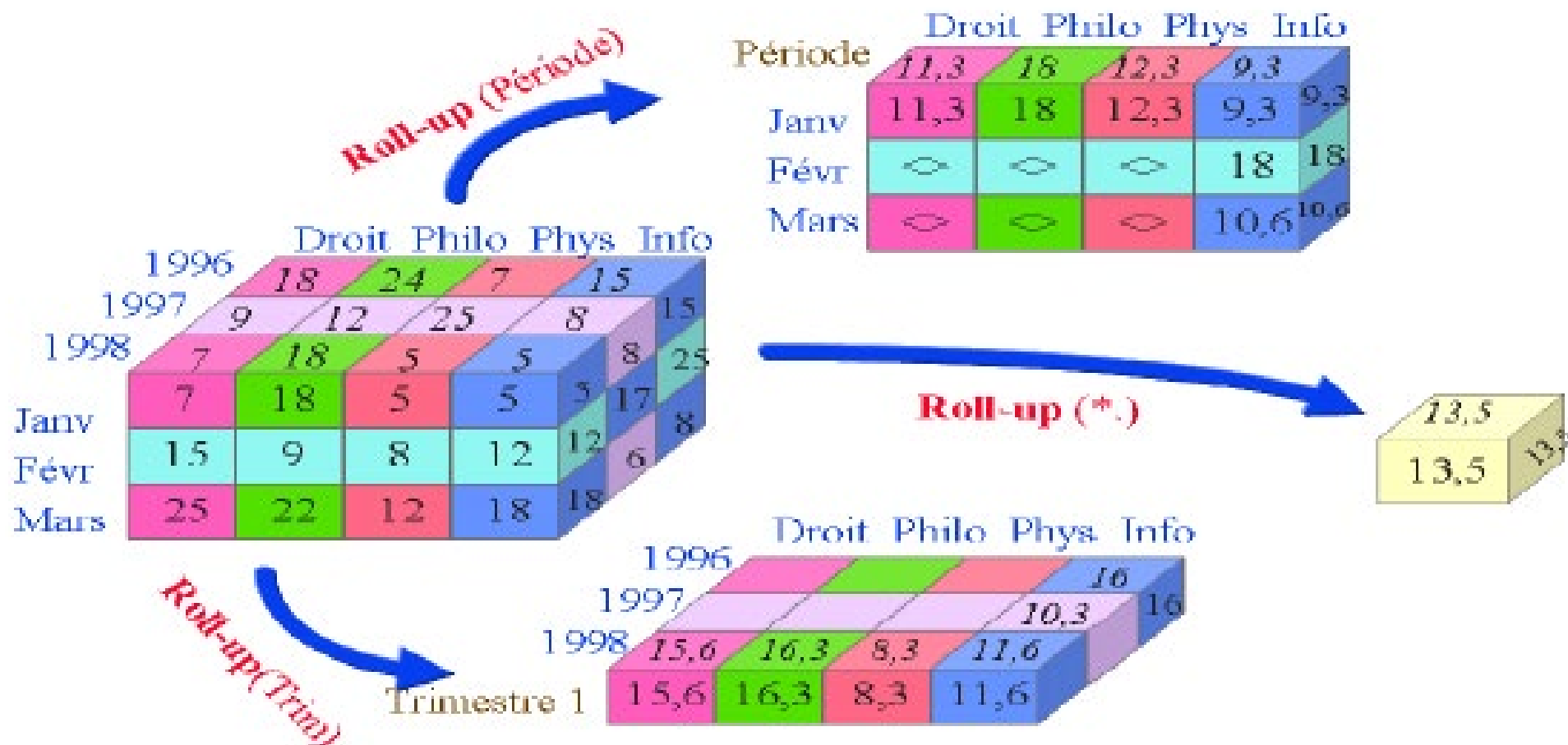


# Opérations de forage (granularité)

- Roll-up (passage au grain supérieur) :
  - Représente les données à un niveau de granularité supérieur selon la hiérarchie de la dimension désirée
  - Agréger selon une dimension
  - Exemple : Semaine -> Mois
- Drill-down (passage au grain inférieur) :
  - Inverse du roll-up
  - Représente les données à un niveau de granularité Inférieur
  - Détailler selon une dimension
  - Exemple : Mois -> Semaine

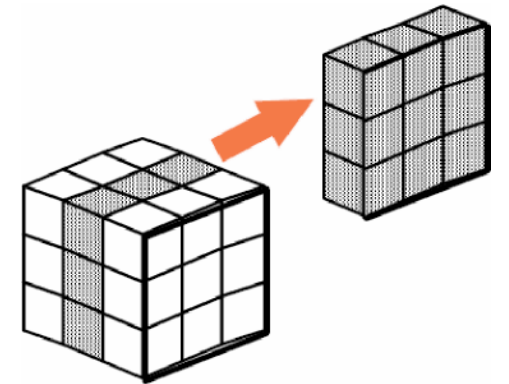
# Opérations de forage (granularité)

- Exemples de Rollup



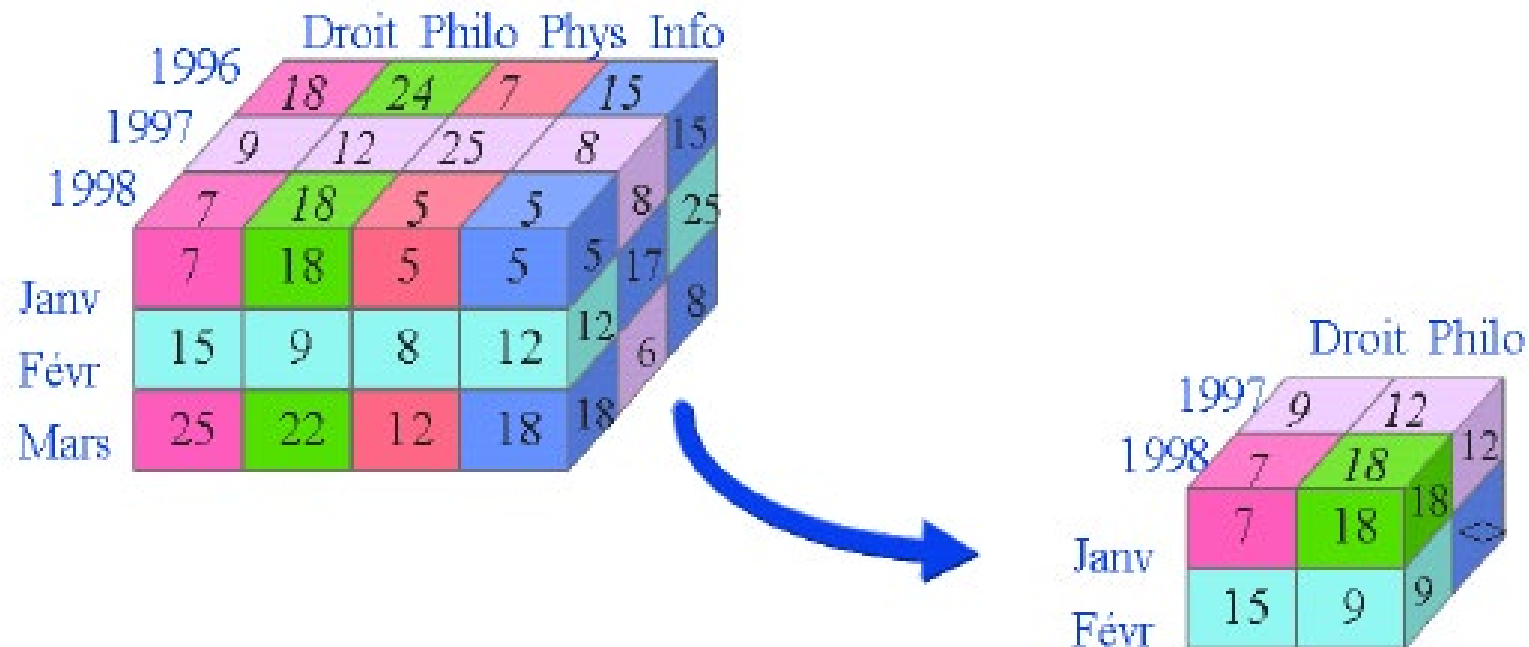
# Opérations de sélection / projection

- Slice :
  - Sélection (restriction) de la tranche du cube obtenue par prédicats selon une plusieurs dimensions
  - Exemple : Mois = « Avril 2004 »
- Dice :
  - Projection selon un ou plusieurs axes
  - Sorte de cumuls de sélection
  - Exemple : Projeter(Région, Produit)



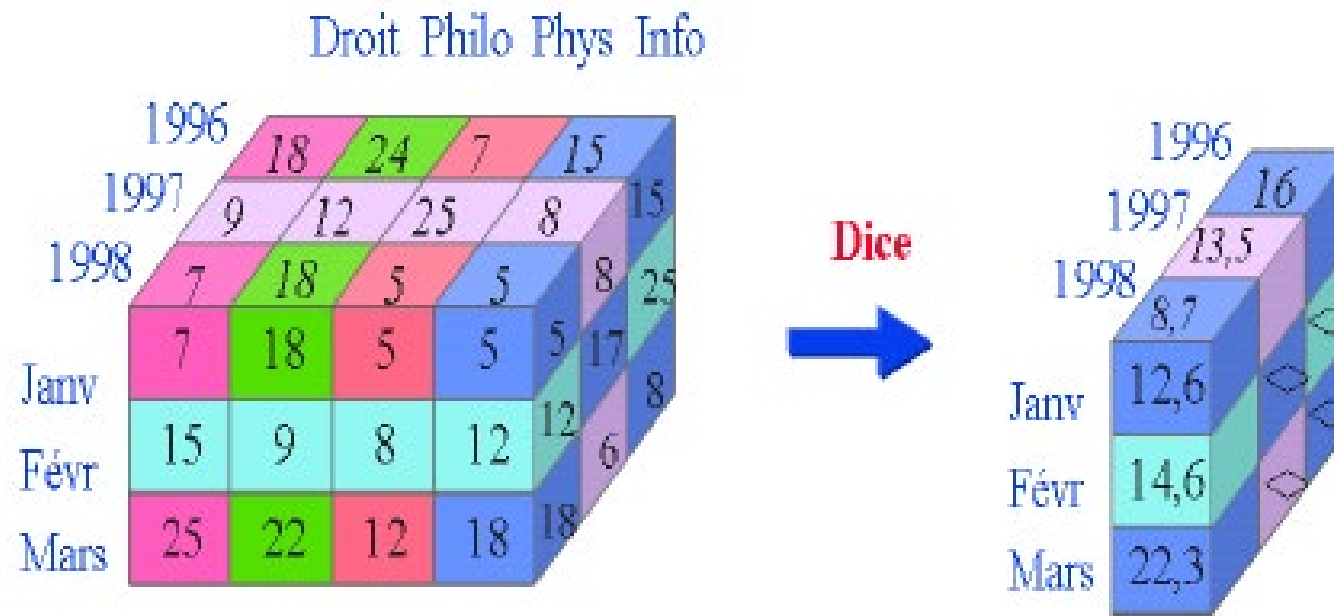
# Opérations de sélection / projection

- Exemple de Slice
  - Département in (Droit,Philo) et Mois in (Janv,Févr)



# Opérations de sélection / projection

- Exemple de Dice
  - Sur les mois et les années



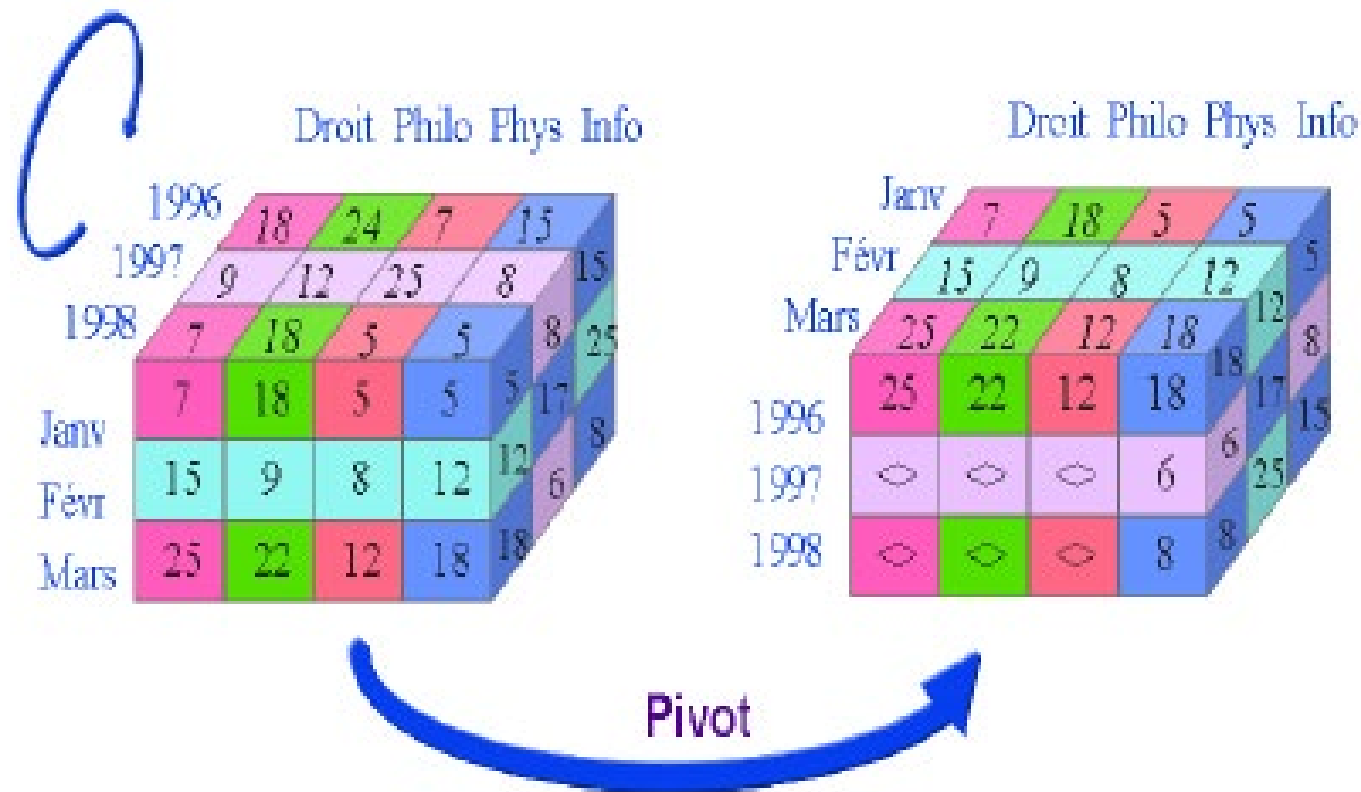


# Opérations de restructuration / réorientation

- Pivot (ou Rotate)
  - Tourne le cube pour visualiser une face différente (Région, Produit) -> (Région, Mois)
- Switch (ou Permutation)
  - Inter-change la position des membres d'une dimension
- Split
  - Eclate le cube en tableaux croisés
- Nest
  - Imbrique des membres issus de dimensions différentes
- Push (ou Enfoncement)
  - Combine les membres d'une dimension aux mesures
- AddM, DelM
  - Pour l'ajout et la suppression de mesures à afficher

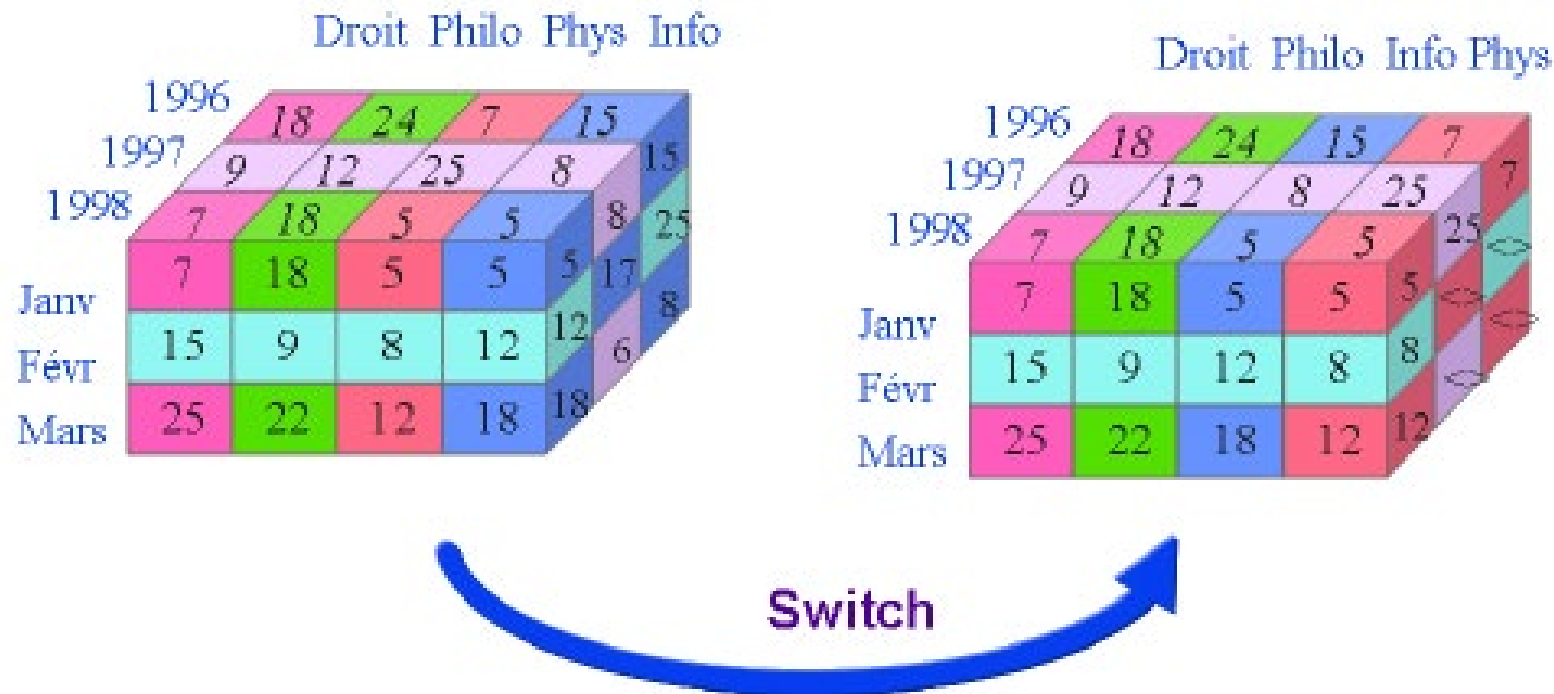
# Opérations de restructuration / réorientation

- Exemple d'opération Pivot



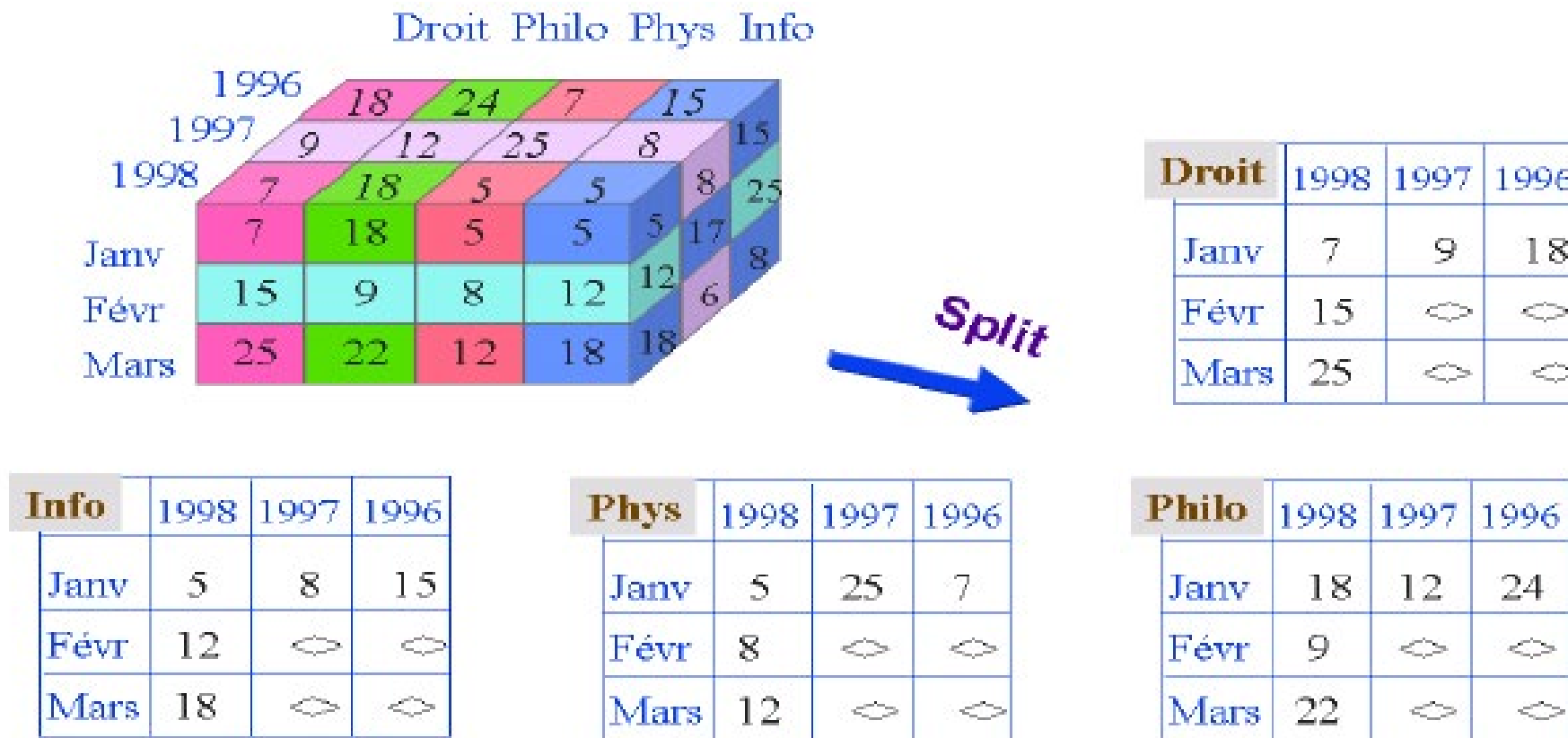
# Opérations de restructuration / réorientation

- Exemple d'opération Switch



# Opérations de restructuration / réorientation

- Exemple d'opération Split

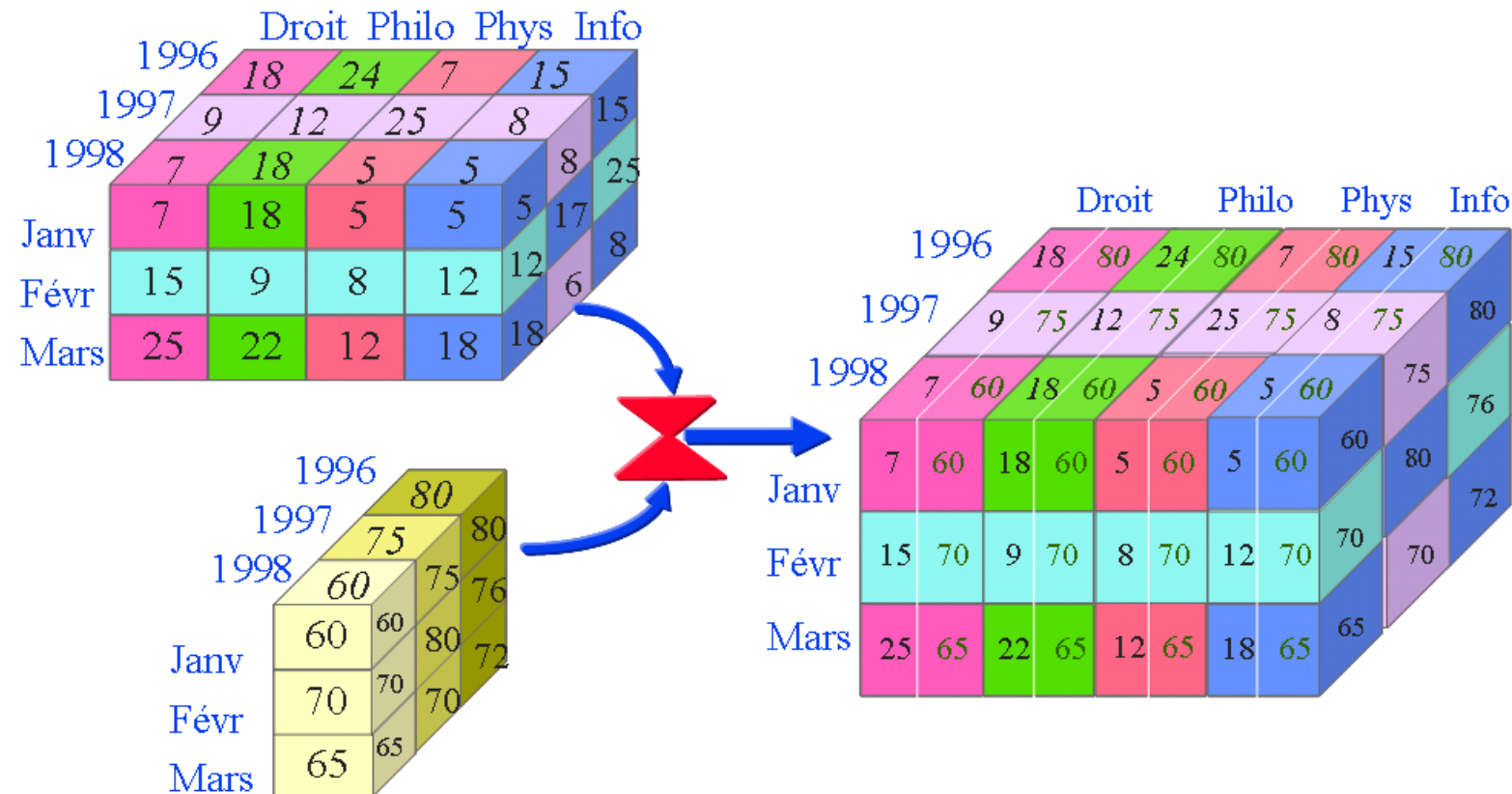


# Opérations entre cubes

- Jointure
  - Entre un cube et un autre cube ou tranche de cube
- Union
  - Entre deux cubes

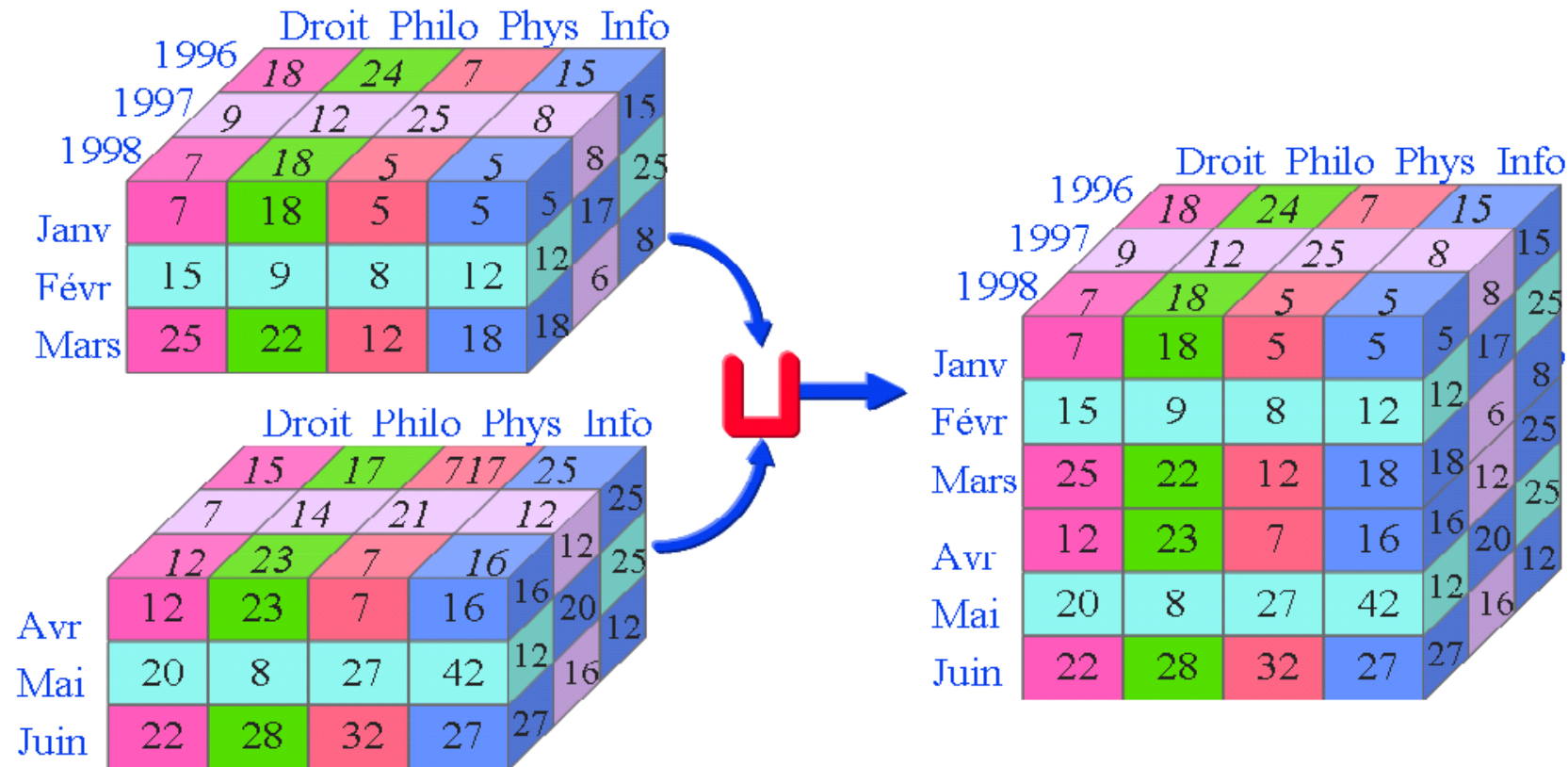
# Opérations entre cubes

- Exemple de jointure



# Opérations entre cubes

- Exemple d'union



# Partie 3 : SQL pour OLAP

Points traités :



# SQL et OLAP

- Certains SGBDR dont Oracle, proposent les extensions OLAP au langage SQL, pour faciliter l'exploration de données
  - Exemple : ROLLUP et CUBE dans l'instruction GROUP BY
- Il existe aussi un certain nombre d'extensions au LMD SQL pour faciliter l'exploitation de données du cube

# Exploration multi-niveaux avec ROLLUP

- GROUP BY ROLLUP (a, b, c, ...) permet de créer tous les sous totaux selon les attributs ordonnés de groupement (ici a, b et c) en allant du plus général (a) au plus détaillé (c)
  - Cette clause est typiquement utilisée pour parcourir une hiérarchie
  - L'ordre (a, b, c) est important et doit être du plus gros grain au plus fin
- Exemple :  
`GROUP BY ROLLUP (YEAR, MONTH, DAY)` donnera :  
YEAR, MONTH, DAY  
YEAR, MONTH  
YEAR  
( )

# Exploration multi-niveaux avec ROLLUP

- Exemple

```
SELECT d.trim, d.month, count(*)  
FROM ventes v, date d  
WHERE v.dat_id=d.dat_id  
GROUP BY ROLLUP (d.trim, d.month);
```

1	1	7159
1	2	7630
1	3	9300
1		24089
2	4	7317
2	5	8017
2	6	8977
2		24311
3	7	7397
3	8	8328
3	9	9042
3		24767
4	10	7448
4	11	7764
4	12	12845
4		28057
		101224

# Exploration multi-dimensions avec CUBE

- GROUP BY CUBE permet de créer tous les sous-totaux possibles pour toutes les combinaisons des attributs de groupement
  - Cette clause est typiquement utilisée pour faire des analyses croisées
- Exemple :

GROUP BY CUBE (YEAR, MONTH, DAY) donnera :

YEAR, MONTH, DAY

YEAR, MONTH

YEAR, DAY

YEAR

MONTH, DAY

MONTH

DAY

()

# Exploration multi-dimensions avec CUBE

- Exemple

```
SELECT p.bs Livre_BS, m.bs Mag_BS, count(*) Nb_Ventes
FROM ventes v, f_dw_produit p, f_dw_mag m
WHERE v.pro=p.isbn AND v.mag=m.mag
GROUP BY CUBE (p.bs, m.bs);
```

Livre BS	Mag BS	Nb Ventes		
		476394		
	0	137368		
	1	339026	71.17%	<i>Mag BS</i>
0		324027	68.02%	<i>Ventes des Livres Non BS en général</i>
0	0	108736	79.16%	<i>Ventes des Livres Non BS dans les Mag Non BS</i>
0	1	215291	63.50%	<i>Ventes des Livres Non BS dans les Mag BS</i>
1		152367	31.98%	<i>Ventes des Livres BS en général</i>
1	0	28632	20.84%	<i>Ventes des Livres BS dans les Mag non BS</i>
1	1	123735	36.50%	<i>Ventes des Livres BS dans les Mag BS</i>

# Projection de classement avec RANK

- `RANK() OVER (PARTITION BY a ORDER BY b DESC)` permet de projeter le classement de l'enregistrement par rapport au critère "b" (qui peut être un agrégat), partitionné par le critère "a".

```
SELECT RANK() OVER ([PARTITION BY a] ORDER BY b DESC), ...  
FROM ...  
GROUP BY a ...
```

- `PARTITION BY` est optionnelle
- On notera que l'attribut de partitionnement du classement "a" est forcément un attribut de regroupement, il doit donc être déclaré dans le `GROUP BY`

# Projection de classement avec RANK

- Exemple

```
SELECT * FROM
(
    SELECT p.titre AS titre, count(*) AS ventes,
           RANK() OVER (ORDER BY count(*) DESC) AS rank
    FROM ventes v, produit p
    WHERE p.isbn=v.pro
    GROUP BY p.titre
)
WHERE rank <= 100;
```

# Projection de classement avec RANK(N)

- Classement sur plusieurs expressions
  - Il est possible de réaliser des classements sur plusieurs expressions, en spécifiant plusieurs critères dans la clause ORDER BY (traités alors de droite à gauche)
- Extrapolation de classement
  - RANK(n) WITHIN GROUP (ORDER BY b) permet de projeter le classement d'un enregistrement hypothétique dont la valeur de "b" serait "n"

```
SELECT RANK(n) WITHIN GROUP (ORDER BY b) FROM ...
```



# Total cumulé

- `SUM(SUM(a)) OVER (ORDER BY b ROWS UNBOUNDED PRECEDING)` permet de calculer la somme cumulée des "a" selon l'évolution "b"

```
SELECT SUM(SUM(a)) OVER (ORDER BY b ROWS UNBOUNDED PRECEDING)
FROM ...
GROUP BY b ...
```

- On notera que "b" doit être un attribut de regroupement
- `SUM(COUNT(a))` est également valide

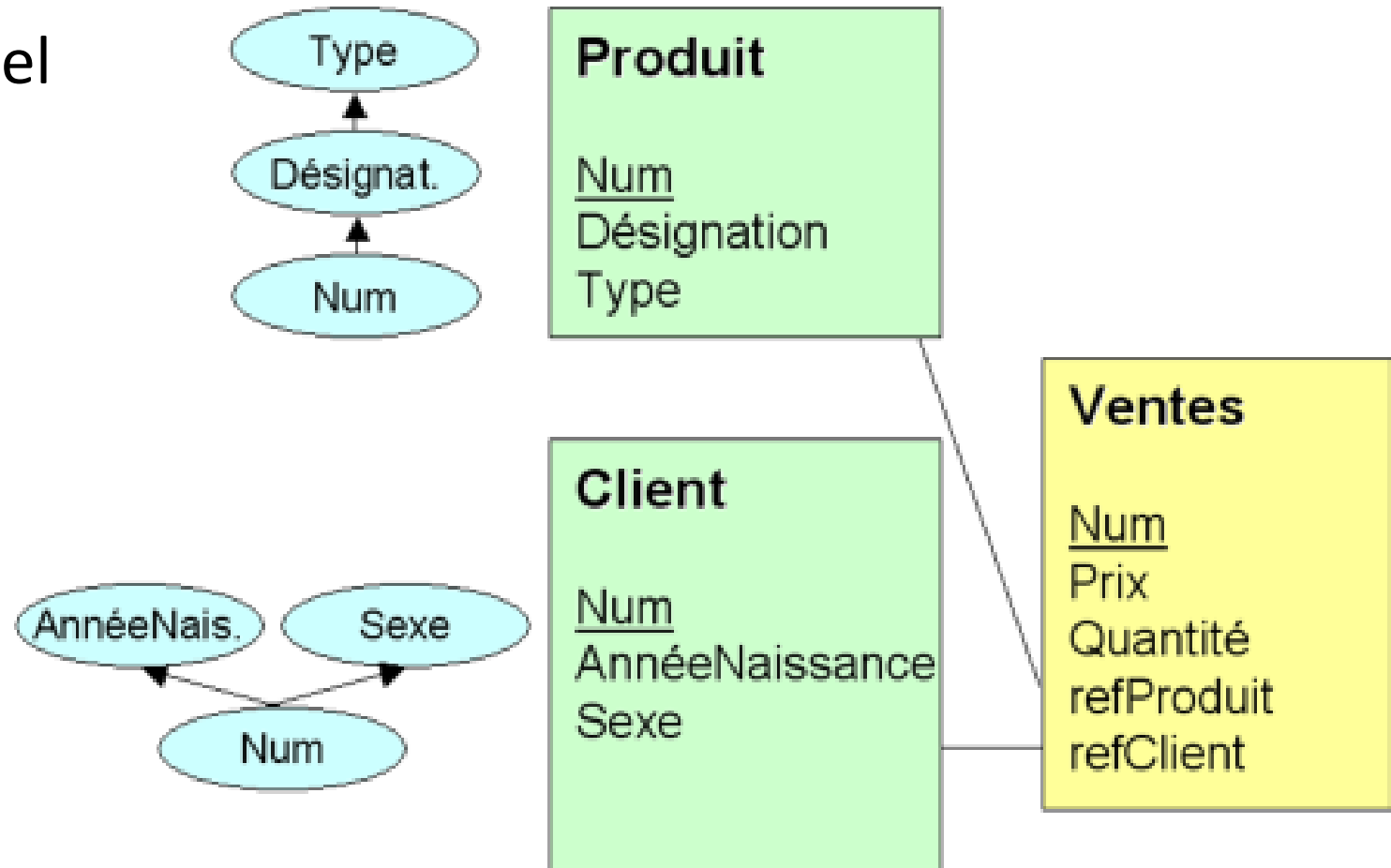
- Ex : 

```
SELECT d.sem, SUM(COUNT(*)) OVER (ORDER BY d.sem
    ROWS UNBOUNDED PRECEDING)
FROM ventes v, date d
WHERE v.dat=d.dat
GROUP BY d.sem;
```

# Partie 4 : Exemple

# Exemple général d'analyse de données

- Modèle dimensionnel



# Exemple général d'analyse de données

- Création des tables

```
CREATE TABLE t_produit (  
    pk_num number,  
    a_designation varchar(50),  
    a_type char(3)  
);  
CREATE UNIQUE INDEX idx_produit_num ON t_produit (pk_num);  
ALTER TABLE t_produit  
    ADD CONSTRAINT cstr_produit_num PRIMARY KEY (pk_num)  
    ADD CONSTRAINT cstr_produit_type CHECK (a_type in ('CD','DVD'));
```

# Exemple général d'analyse de données

- Création des tables de dim

```
CREATE TABLE t_client (  
    pk_num number,  
    a_anneenaiss number,  
    a_sexe char(1)  
);  
CREATE UNIQUE INDEX idx_client_num ON t_client (pk_num);  
ALTER TABLE t_client  
    ADD CONSTRAINT cstr_client_num PRIMARY KEY (pk_num)  
    ADD CONSTRAINT cstr_client_sexe CHECK (a_sexe in ('M', 'F'));
```

# Exemple général d'analyse de données

- Création des tables fait

```
CREATE TABLE t_ventes (  
    pk_num number,  
    a_prix number,  
    a_qte number,  
    fk_produit number,  
    fk_client number  
);  
CREATE UNIQUE INDEX idx_ventes_num ON t_ventes (pk_num);  
ALTER TABLE t_ventes  
    ADD CONSTRAINT cstr_ventes_num PRIMARY KEY (pk_num)  
    ADD CONSTRAINT cstr_ventes_produit FOREIGN KEY (fk_produit)  
REFERENCES t_produit(pk_num)  
    ADD CONSTRAINT cstr_ventes_client FOREIGN KEY (fk_client)  
REFERENCES t_client(pk_num);
```

# Exemple général d'analyse de données

- Insertion dans les tables

```
INSERT INTO t_produit VALUES (1, 'Pink Martini', 'CD');
INSERT INTO t_produit VALUES (2, 'Souad Massi', 'CD');
INSERT INTO t_produit VALUES (3, 'Souad Massi', 'DVD');
INSERT INTO t_produit VALUES (4, 'Raul Paz', 'CD');
INSERT INTO t_produit VALUES (5, 'Star wars', 'DVD');
INSERT INTO t_produit VALUES (6, 'Star wars BO', 'CD');
```

```
INSERT INTO t_client VALUES (1, 1980, 'M');
INSERT INTO t_client VALUES (2, 1980, 'M');
INSERT INTO t_client VALUES (3, 1970, 'F');
INSERT INTO t_client VALUES (4, 1970, 'M');
INSERT INTO t_client VALUES (5, 1985, 'F');
```

PK_NUM	A_DESIGNATION	A_TYPE
1	Pink Martini	CD
2	Souad Massi	CD
3		
4		
5		
6		

PK_NUM	A_ANNEENAISS	A_SEXE
1	1980	M
2	1980	M
3	1970	F
4	1970	M
5	1985	F

# Exemple général d'analyse de données

- Insertion dans les tables

```
INSERT INTO t_ventes VALUES (1,15,1,1,1);
INSERT INTO t_ventes VALUES (2,20,3,1,2);
INSERT INTO t_ventes VALUES (3,30,1,1,3);
INSERT INTO t_ventes VALUES (4,10,2,2,1);
INSERT INTO t_ventes VALUES (5,2,5,2,1);
INSERT INTO t_ventes VALUES (6,10,1,3,1);
INSERT INTO t_ventes VALUES (7,20,1,4,2);
INSERT INTO t_ventes VALUES (8,30,1,4,3);
INSERT INTO t_ventes VALUES (9,10,1,5,3);
INSERT INTO t_ventes VALUES (10,40,4,6,4);
INSERT INTO t_ventes VALUES (11,30,1,6,5);
INSERT INTO t_ventes VALUES (12,10,100,6,5);
```

PK_NUM	A_PRIX	A_QTE	FK_PRODUIT	FK_CLIENT
1	15	1	1	1
2	20	3	1	2
3	30	1	1	3
4	10	2	2	1
5	2	5	2	1
6	10	1	3	1
7	20	1	4	2
8	30	1	4	3
9	10	1	5	3
10	40	4	6	4
11	30	1	6	5
12	10	100	6	5



# Exemple général d'analyse de données

- Jointure des 3 tables et CA

PK_NUM	A_PRIX	A_QTE	FK_PRODUIT	FK_CLIENT	A_DESIGNATION	A_TYPE	A_ANNEE	A_SEXE	CA
1	15	1	1	1	Pink Martini	CD	1980	M	15
2	20	3	1	2	Pink Martini	CD	1980	M	60
3	30	1	1	3	Pink Martini	CD	1970	F	30
4	10	2	2	1	Souad Massi	CD	1980	M	20
5	2	5	2	1	Souad Massi	CD	1980	M	10
6	10	1	3	1	Souad Massi	DVD	1980	M	10
7	20	1	4	2	Raul Paz	CD	1980	M	20
8	30	1	4	3	Raul Paz	CD	1970	F	30
9	10	1	5	3	Star wars	DVD	1970	F	10
10	40	4	6	4	Star wars BO	CD	1970	M	160
11	30	1	6	5	Star wars BO	CD	1985	F	30
12	10	100	6	5	Star wars BO	CD	1985	F	1000

# Exemple général d'analyse de données

- Opération ROLLUP

- Cette requête permet de calculer les chiffres d'affaire (quantité multipliée par le prix de vente) selon la dimension "Produit", c'est à dire pour chaque produit, mais aussi pour chaque type de produit (granularité plus grossière dans la hiérarchie de la dimension produit)

```
SELECT SUM(v.a_prix * v.a_qte) as CA, p.pk_num as P,  
       p.a_type as T  
FROM t_ventes v, t_produit p  
WHERE v.fk_produit=p.pk_num  
GROUP BY ROLLUP (p.a_type, p.pk_num);
```

# Exemple général d'analyse de données

- Opération ROLLUP

- Comme pour un GROUP BY classique, la première ligne nous donne le chiffre d'affaire du produit 1, la seconde celui du produit 2, etc. Mais la cinquième ligne (une fois tous les produits de type CD traités) propose une consolidation et donne le chiffre d'affaire pour tous les produits de type CD. De même la huitième ligne donne le chiffre d'affaire pour tous les produits de type DVD et enfin la dernière ligne donne le chiffre d'affaire global

CA	P	T
105	1	CD
30	2	CD
50	4	CD
1190	6	CD
1375		CD
10	3	DVD
10	5	DVD
20		DVD
1395		

# Exemple général d'analyse de données

- Opération CUBE
  - Cette requête permet de calculer le "cube" des chiffres d'affaire selon les types de produit et le sexe des clients

```
SELECT SUM(v.a_prix * v.a_qte) as CA, p.a_type as T,  
       c.a_sexe as S  
FROM t_ventes v, t_produit p, t_client c  
WHERE v.fk_produit=p.pk_num AND v.fk_client=c.pk_num  
GROUP BY CUBE (p.a_type, c.a_sexe);
```

# Exemple général d'analyse de données

- Opération CUBE

- La première ligne renvoie le chiffre d'affaire global, la seconde celui pour les sexe=F (donc les femmes) seulement, la troisième pour les sexe=M seulement, la quatrième pour les type=CD, la cinquième pour les type=CD et sexe=F (CD achetés par des femmes), etc ...

CA	T	S
1395		
1100		F
295		M
1375	CD	
1090	CD	F
285	CD	M
20	DVD	
10	DVD	F
10	DVD	M

# Exemple général d'analyse de données

- Opération RANK
  - Cette requête renvoie le "top 3" des produits vendus (en quantité) pour les CD et pour les DVD

```
SELECT * FROM (  
  SELECT  
    RANK() OVER (  
      PARTITION BY p.a_type  
      ORDER by sum(v.a_qte) DESC  
    ) as R,  
    SUM(v.a_qte) as V,  
    p.a_type as T,  
    p.a_designation as P  
  FROM t_ventes v, t_produit p  
  WHERE v.fk_produit=p.pk_num  
  GROUP BY p.a_type,  
  p.a_designation  
)  
WHERE R<=3;
```

# Exemple général d'analyse de données

- Opération RANK
  - On notera que pour les DVD, étant donné qu'il n'y a eu que deux ventes et chacune d'une unité, ces deux ventes apparaissent premières ex-aequo et il n'y a pas de troisième vente

R	V	T	P
1	105	CD	Star wars BO
2	7	CD	Souad Massi
3	5	CD	Pink Martini
1	1	DVD	Star wars
1	1	DVD	Souad Massi

# Exemple général d'analyse de données

- Opération RANK
  - La même requête sans la clause PARTITION BY donnera les trois premières positions sur l'ensemble des produits

R	V	T	P
1	105	CD	Star wars BO
2	7	CD	Souad Massi
3	5	CD	Pink Martini

```
SELECT * FROM (
SELECT
  RANK() OVER (
    -- PARTITION BY p.a_type
    ORDER by sum(v.a_qte) DESC
  ) as R,
  SUM(v.a_qte) as V,
  p.a_type as T,
  p.a_designation as P
FROM t_ventes v, t_produit p
WHERE v.fk_produit=p.pk_num
GROUP BY p.a_type,
p.a_designation
)
WHERE R<=3;
```



# Exemple général d'analyse de données

- Opération RANK(N)
  - Cette requête permet de se demander combien serait classée une vente de trois unités dans le domaine des CD ainsi que dans celui des DVD

```
SELECT
  p.a_type as P,
  RANK(3) WITHIN GROUP
    (ORDER BY v.a_qte DESC) as Position
FROM t_ventes v, t_produit p
WHERE v.fk_produit=p.pk_num
GROUP BY p.a_type;
```

# Exemple général d'analyse de données

- Opération RANK(N)
  - On voit qu'une telle vente serait la quatrième meilleure vente pour des CD et la première pour des DVD

P	HYPOTHETICALRANK
CD	4
DVD	1

# Exemple général d'analyse de données

- Opération SUM(SUM(...))
  - Cette requête permet de faire le calcul cumulé des chiffres d'affaire sur les années de naissance des clients

```
SELECT
  c.a_anneenaiss as D,
  -- sum(v.a_prix * v.a_qte) as CA,
  sum(sum(v.a_prix * v.a_qte))
    OVER
      (ORDER BY c.a_anneenaiss ROWS UNBOUNDED PRECEDING)
    as CA_cumul
FROM t_ventes v, t_client c
WHERE v.fk_client=c.pk_num
GROUP BY c.a_anneenaiss;
```

# Exemple général d'analyse de données

- Opération SUM(SUM(...))
  - Le résultat montre ainsi que les clients nés en 1970 ont permis un chiffre d'affaire de 230, ceux de 1980 et moins de 365 et ceux de 1985 et moins de 1395 (le total)

D	CA	CA_CUMUL
1970	230	230
1980	135	365
1985	1030	1395