Entrepôts de données

Responsable : Patricia Serrano Alvarado

Organisation du cours OLAP

- Cours (3 séances)
  1. Introduction et modèle OLAP
  2. Requêtes multidimensionnelles
  3. Matérialisation des vues

- TD (3 séances)
  - Conception et interrogation d' entrepôts de données

Plan

- OLTP vs OLAP
- Architecture type
- Construction d'entrepôts de données
  - Extraction,
  - Transformation,
  - Stockage,
  - Rafraîchissement
- Modélisation multidimensionnelle
  - Opérateurs OLAP
  - Architectures
  - Indexation
- Implémentation
  - Schéma en étoile
  - Schéma en flacon de neige
  - Constellation de faits
- Interrogation OLAP

Pourquoi a-t-on besoin des entrepôts de données ?

OLTP (*On-Line Transaction Processing*)

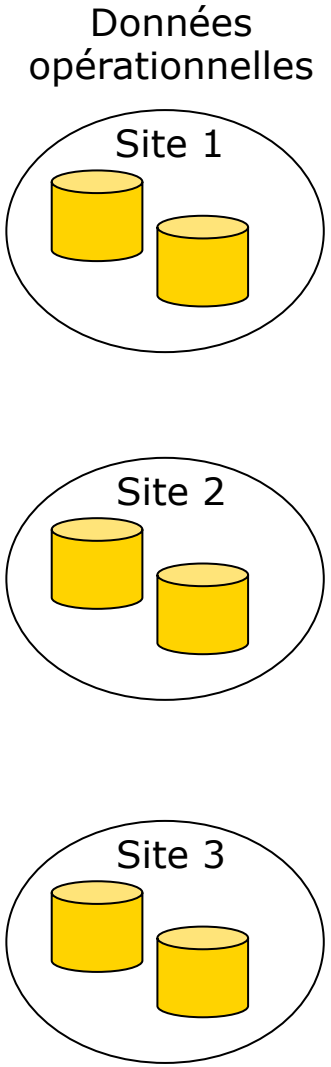
- Applications commerciales
- Importants volumes de données
- Données détaillées et à jour
- Processus transactionnel en ligne
- Transactions
  - Courtes/simples
  - Accès à un nombre restreint de tuples
  - Mises à jour et lectures
  - Nombreuses
  - Répétitives
  - Concurrentes
  - Recouvrables

Exemple

- Chaîne de supermarchés enregistrant ses ventes
- Transactions
  - Pour chaque vente
  - Consultation de stocks
  - Etc.
- Bases de données évolutives et grandissantes

Est-il possible de traiter des requêtes pour l'analyse des données de tous les sites simultanément ?

« Est-ce que les ventes d'un tel produit en septembre ont augmenté par rapport à l'année dernière ? »



Intégration de données (1)

Paresseuse (*query-driven*) : interrogation des sources OLTP

- Avantages
  - Pas besoin d'un support de stockage dédié à l'analyse de données
  - Données à jour
  - Intéressant si les sources changent souvent
  - Adéquat pour des analyses imprédictibles

- Désavantages
  - Délai dans le traitement des requêtes
    - Sources lentes ou indisponibles
    - Filtres et intégration complexe
  - Inefficient et potentiellement coûteux pour des requêtes fréquentes
  - Concurrence avec le traitement local sur les sources

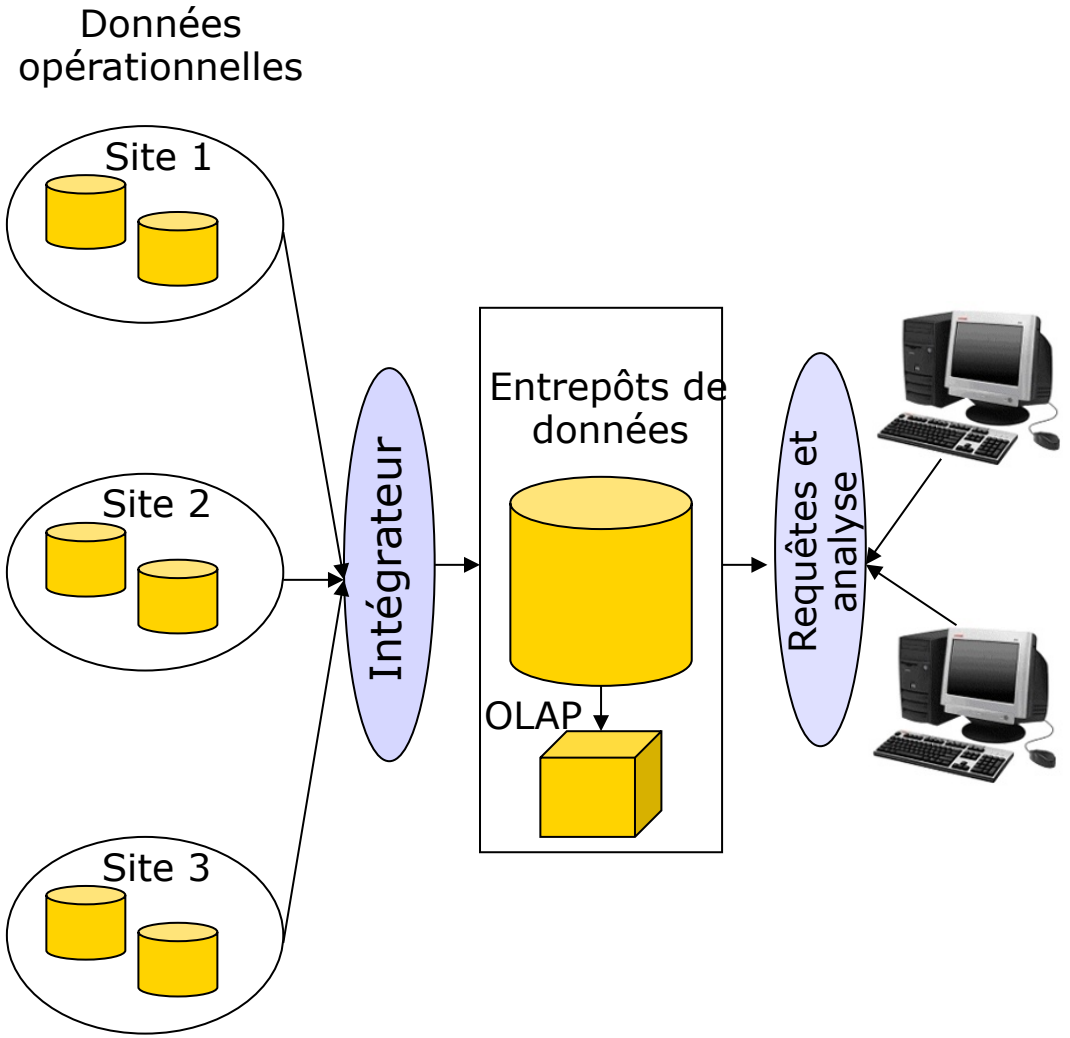
Intégration de données (2)

Par avance (*update-driven*) : création d'une base de données dédiée à partir des sources

- Avantages
  - Indépendance vis à vis des sources
  - Performances de l'évaluation de requêtes
- Désavantages
  - Intégration par avance des sources
  - Support de stockage dédié
- Évolution de ce type d'intégration
  - Bases de données fédérées
  - Architecture multibases
  - Médiation
  - Entrepôt de données et OLAP

Exemple

- Données orientées sujet
- Opérations focalisées sur l'aide à la décision
- Ex: ventes au cours du temps
  - Par produit
  - Par client
  - Par magasin
  - Par région
  - Etc.
- Beaucoup d'agrégations et de jointures



Entrepôts de données (1)

- Collection de données pour la prise de décision
  - Homogène
  - Exploitable
- Réalisés à partir de sources de données
  - Hétérogènes
  - Nombreuses
  - Réparties
- Fonctions importantes
  - Extraction
  - Transformation
  - Stockage
  - Rafraîchissement

Entrepôts de données (2)

«Le Data Warehouse est une collection de données orientées sujet (métiers), intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision.»

Bill Inmon (1996)

- Données sources combinées dans un schéma global
- Données copiées dans une BD de l'entrepôt
- Mise à jour sur les sources exclusivement
- Rafraîchissement périodique de la BD de l'entrepôt

OLAP (*On-Line Analytical Processing*)

- Exploration et analyse de données historiques
- Énormes volumes de données (Tera octets)
- Données consolidées et synthétiques
- Processus analytiques en ligne
  - Selon plusieurs axes d'analyse
  - Selon différents niveaux de détail
  - Peu nombreux (pas besoin de gestion de concurrence)
  - Accédant en lecture à un nombre important de tuples (pas besoin d'opérations recouvrables)
  - Complexes
- Facilité
  - Changer les axes d'analyse, le niveau de détail, etc.
  - Réaliser les opérations OLAP classiques (requêtes)

Applications

- Domaines : commercial, financier, transport, télécommunications, santé, services, ...
  - Prévisions (de ventes, de commandes, de stocks, etc.)
  - Comparaison de performances
  - Surveillance
  - Définition de profil utilisateur
  - Analyse de transactions (de ventes, bancaires, etc.)
  - Détection de fraudes
  - Etc.

OLAP et Tedd Codd

- Le terme OLAP a été définit par Ted Codd en 1993 dans son article :
  E. F. Codd. Providing OLAP to user-analysts: An IT mandate. E.F. Codd and Associates, 1993.

Les 12 règles de Codd :

1. Vue conceptuelle multidimensionnelle
2. Transparence
3. Accessibilité
4. Constance des temps de réponses
5. Architecture Client/Serveur
6. Indépendance des dimensions
7. Gestion des matrices creuses
8. Accès multi-utilisateurs
9. Pas de restrictions sur les opérations inter et intra dimensions
10. Manipulation des données aisée
11. Simplicité des rapports
12. Nombre illimité de dimensions et nombre illimité d'éléments sur les dimensions

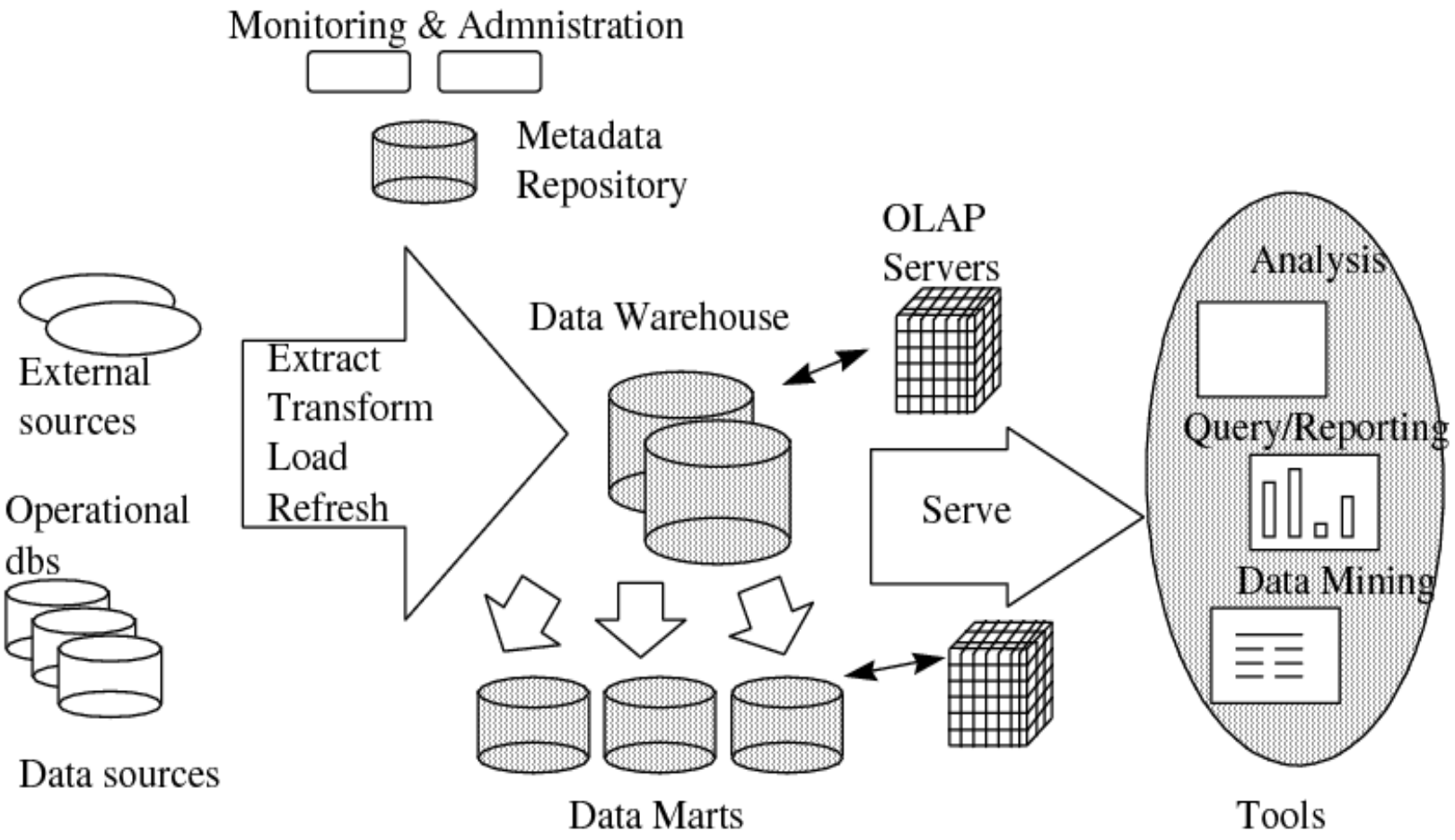
OLTP/OLAP

	OLTP	OLAP
orientation	transaction	analyse
utilisateur	DBA	analyste
conception	E/R	star/snowflake
granularité	détail	résumé
vue	relationnelle	multidimensionnelle
unité de travail	transactions simples	requêtes complexes
accès	lecture/écriture	lecture
nombre de tuple accédés	dizaine	millions
nombre d'utilisateurs	milliers	centaines
unité	Mo/Go	Go/To
métrique	débit de transactions	temps de réponse

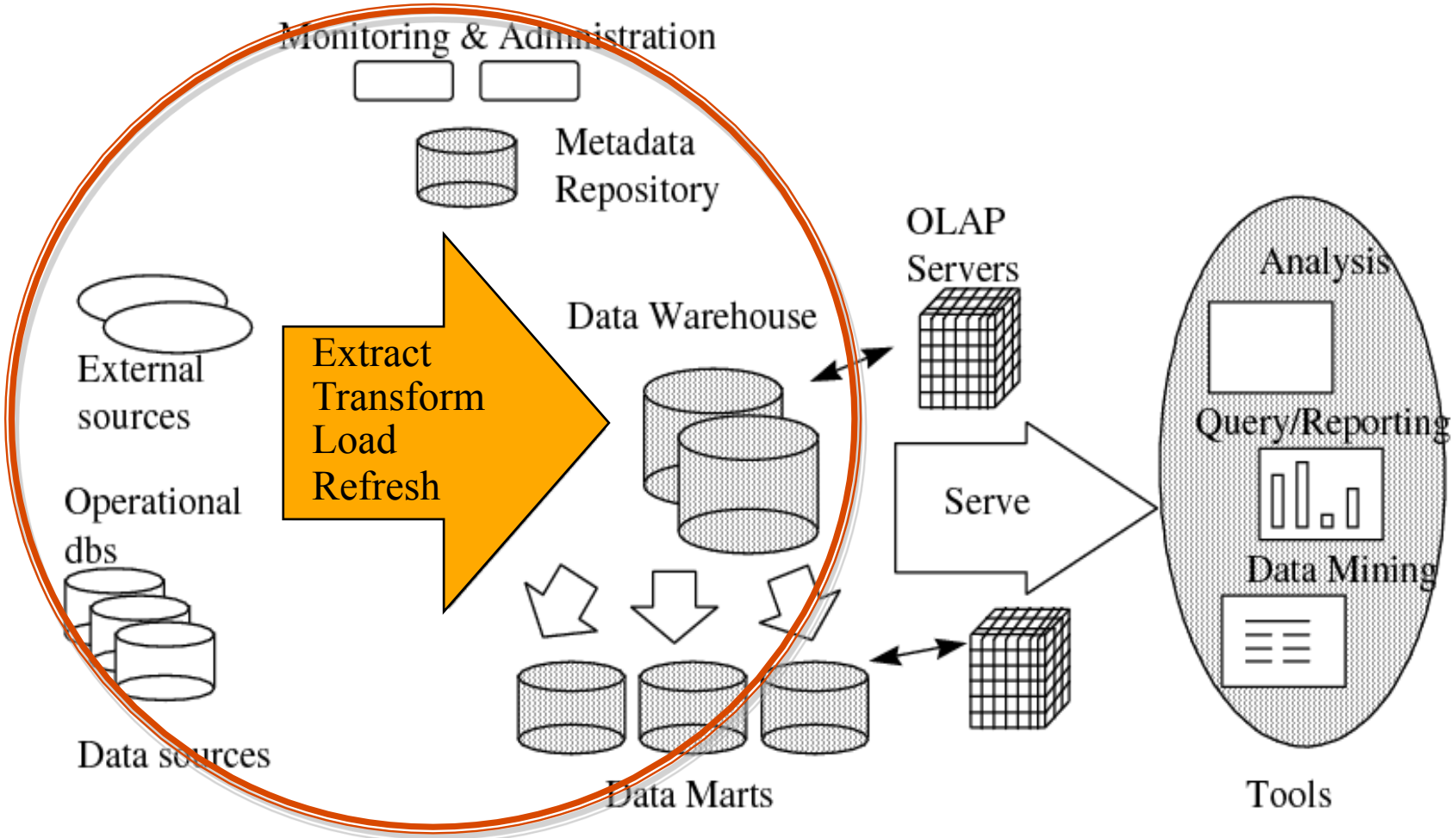
Besoin des deux systèmes

- Hautes performances pour les deux types de systèmes
  - **SGBD** (Système de Gestion de Bases de Données) – orienté applications OLTP : méthodes d'accès, indexation, contrôle de concurrence, récupération, etc.
  - **Entrepôts de données** – orienté applications OLAP : requêtes OLAP complexes, vues multidimensionnelles, données consolidées, etc.

Architecture type



Architecture type – l'intégration



4 niveaux de construction

1. Préparation des données
2. Intégration des données
3. Agrégation des données
4. Personnalisation des données

Jusqu'à 80 % du temps de développement d'un entrepôt préparation, intégration et agrégation. La définition des indexes est cruciale

4 niveaux de construction

1. Préparation des données	• Identification des données à extraire des sources • Extraction de ces données • Nettoyage des données extraites • Archivage éventuel
2. Intégration des données	• Transformation de données • Mise à un format commun • Stockage
3. Agrégation des données	• Calcul de vues agrégées • Stockage
4. Personnalisation des données	• Construction des présentations requises par les utilisateurs • Construction de cubes de données • Construction de magasin de données (data marts)

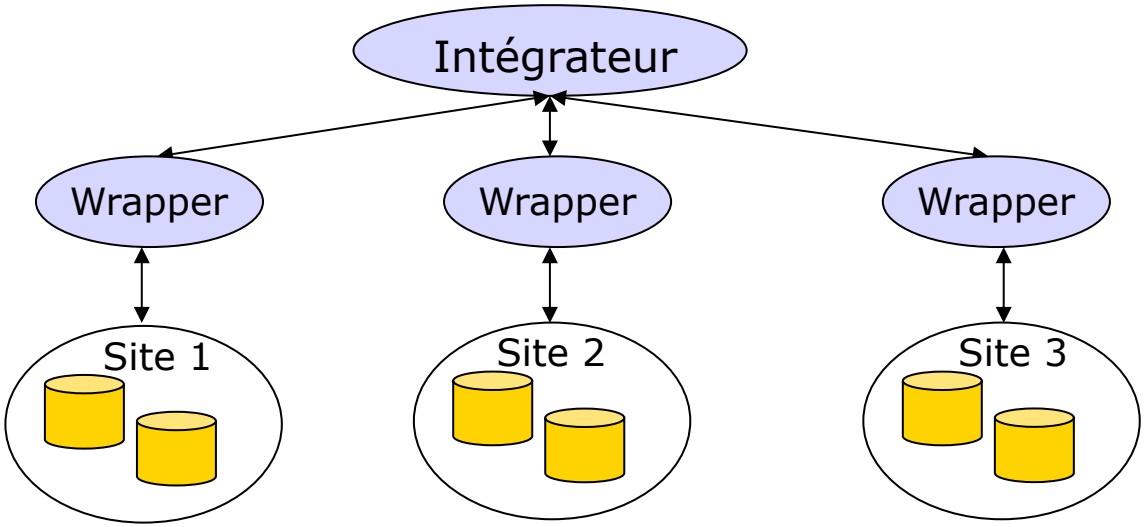
ETL tools (Extract Transform Load)

- Support et/ou automatisation de :
  - Extraction -> accès aux différentes sources
  - Nettoyage -> recherche et résolution des inconsistances dans les sources
  - Transformation entre différents formats, langages, etc.
  - Chargement des données dans l'entrepôt

  - Duplication des sources dans l'entrepôt
  - Analyse e.g., détection de valeurs non valides ou inattendues
  - Transfert de données haut débit pour les très grands entrepôts
  - Test de qualité e.g., pour correction et complétude
  - Analyse des métadonnées aide à la conception

Extraction

- Un *wrapper* (extracteur) est associé à chaque source
- Fonctions de
  - Sélection et extraction de données
  - Formatage de données dans un format commun
- Utilisation d'interfaces de type ODBC, JDBC, OCI
- Le format cible est majoritairement le modèle relationnel



Nettoyage (1)

- Résoudre le problème de consistance de données au sein de chaque source
- 5 à 30% des données des BD commerciales sont erronées
- Une centaine de types d'inconsistances ont été répertoriées
  - Présence de données fausses dès leur saisie
    - Faute de frappe
    - Différent format dans une même colonne
    - Texte masquant de l'information (e.g., "N/A")
    - Valeur nulle
    - Incompatibilité entre la valeur et la description de la colonne
    - Duplication d'information
  - Persistance de données obsolètes
  - Confrontation de données sémantiquement équivalentes mais syntaxiquement différentes

Nettoyage (2)

- Un outil de nettoyage comprend
  - Des fonctions de normalisation
  - Des fonctions de conversion
  - Des dictionnaires de synonymes ou d'abréviations

- Définition de table de règles

remplacer	valeur	par
	Mr	M
	monsieur	M
	mnsieur	M
	masculin	M
	M	M
	Msieur	M
	M.	M
	Monseur	M

- Nettoyage = jointure + update

Transformation (1)

- Faire converger sémantiquement les sources (schémas et données) vers l'entrepôt de données

- Problème de sources hétérogènes

  Pour un même concept
  - Schémas différents
  - Noms d'attribut différents
  - Types de données différents
  - Valeurs différentes
  - Sémantiques différentes

Transformation (2)

- Sources hétérogènes

- Exemple : chaîne de concessionnaires
  - Concession 1

    véhicules(série, modèle, couleur, autoradio, ...)

    ex : véhicules(“1234”, “206 xrp”, “rouge”, “abs.”, ...)
  - Concession 2

    automobiles(numserie, modèle, couleur)

    options(numserie, option)

    ex : automobiles(1234, “206”, “r”)

    ex : automobiles(1233, “206”, “r”)

    ex : options(1234, “abs.”)

Intégration de schémas

- Demande une solide connaissance de la sémantique des schémas
- Peu traité par les produits du marché
- Nombreux travaux de recherche
- Généralement faite à la main...

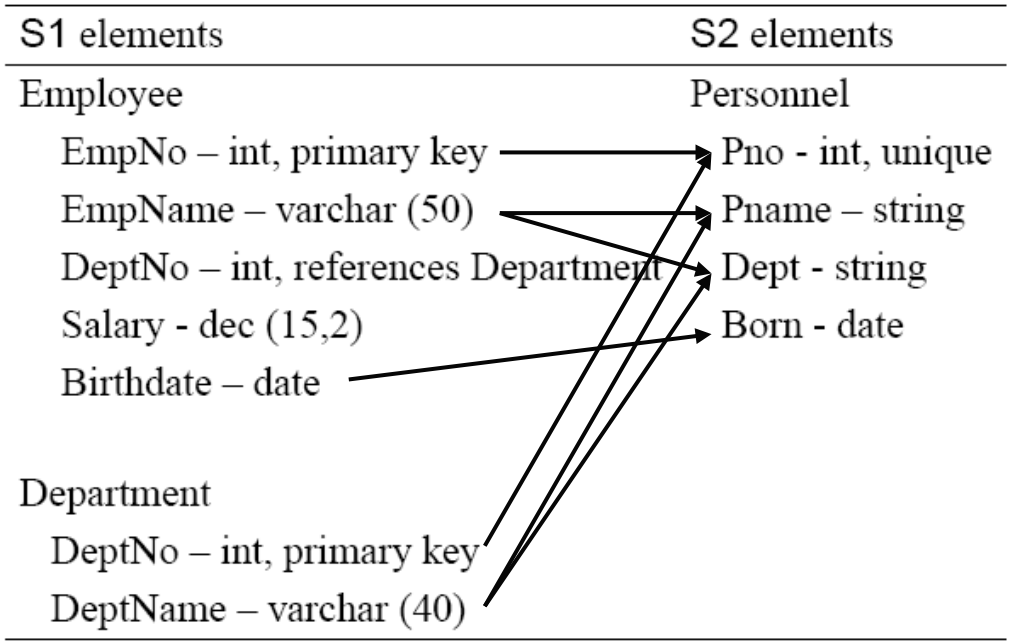
- Les heuristiques de réconciliation de schémas se basent sur l'existence de similarités entre :
  - Structure des schémas
  - Noms de tables et d'attributs
  - Types de données
  - Contraintes d'intégrité
  - Instances

Classification of schema matching

- Instance vs schema
  - Matching approaches can consider instance data (i.e., data contents) or only schema-level information.
- Element vs structure matching
  - Match can be performed for individual schema elements, such as attributes, or for combinations of elements, such as complex schema structures.
- Language vs constraint
  - A matcher can use a linguistic based approach (e.g., based on names and textual descriptions of schema elements) or a constraint-based approach (e.g., based on keys and relationships).
- Matching cardinality
  - The overall match result may relate one or more elements of one schema to one or more elements of the other, 1:1, 1:n, n:1, n:m.
- Auxiliary information
  - Most matchers rely not only on the input schemas S1 and S2 but also on auxiliary information, such as dictionaries, global schemas, previous matching decisions, and user input

Exemple 1

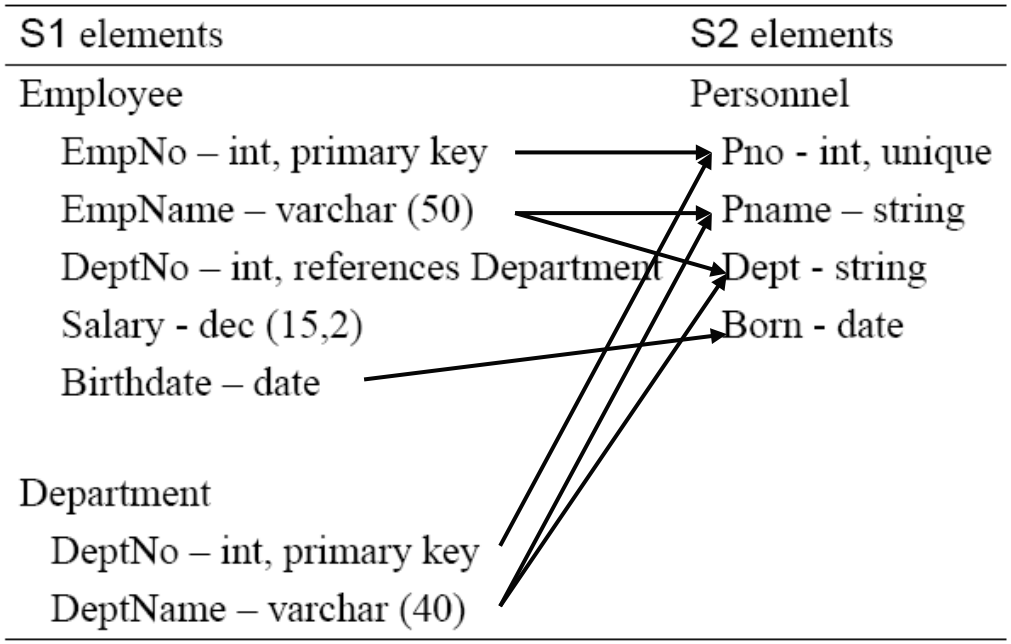
S1 elements	S2 elements
Employee	Personnel
EmpNo – int, primary key	Pno - int, unique
EmpName – varchar (50)	Pname – string
DeptNo – int, references Department	Dept - string
Salary - dec (15,2)	Born - date
Birthdate – date	
Department	
DeptNo – int, primary key	
DeptName – varchar (40)	



Synonym table	
Primary key	unique
Varchar()	string

- The type and key information suggest that
  - Born matches Birthdate and Pno matches either EmpNo or DeptNo. Pname and Dept are strings and thus likely match EmpName or DeptName.

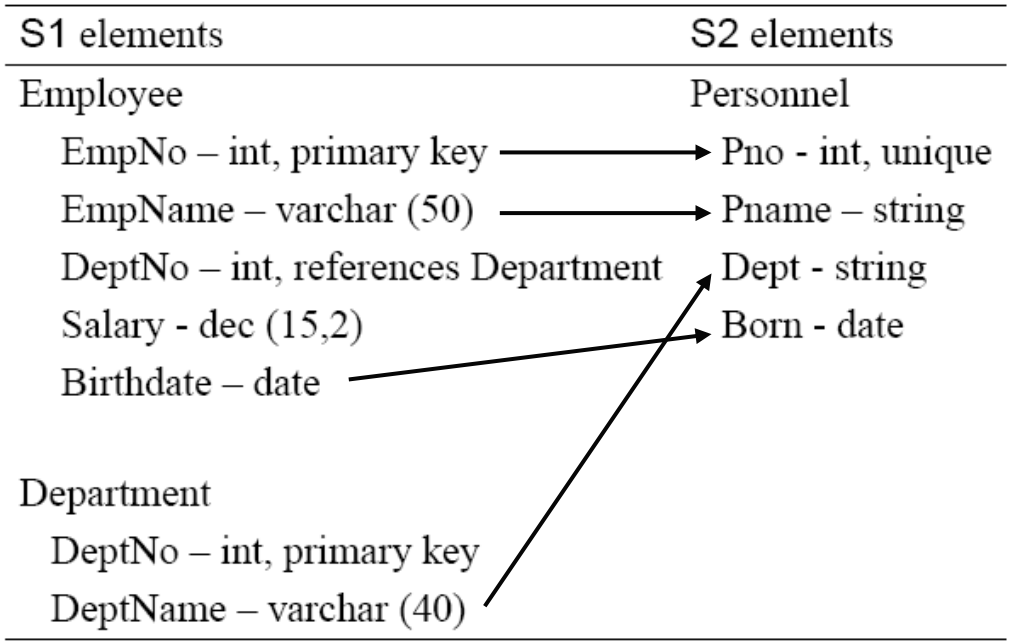
Exemple 2



Synonym table	
Primary key	unique
Varchar()	string

- The type and key information plus the instance information suggest that
  - Pname matches EmpName, Dept matches DeptName, Pno matches EmpNo, Born matches Birthdate.

Exemple 3



Synonym table	
Primary key	unique
Varchar()	string

Table of canonical names	
S1	
EmpNo	Employee number
EmpName	Employee name
DeptNo	Department number
DeptName	Department name
S2	
Pno	Personnel number
Pname	Personnel name
Dept	Department name

- Combining the type, the key and the name (using a table of canonical names and synonymous dictionaries) candidate elements suggest that
  - Pno matches EmpNo, Pname matches EmpName, Dept matches DeptName, Born matches Birthdate.

Chargement

- Une fois que les données sont intégrées elles peuvent être chargées dans l'entrepôt

- Les données sont alors disponibles pour les différents outils d'analyse et de présentation
  - Analyse multidimensionnelle OLAP,
  - Data Mining,
  - Analyses géographiques,
  - Etc.

- Chargement réalisé
  - À la création de l'entrepôt ou
  - Lors du rafraîchissement

Quelques produits :
http://www.informatica.com
http://www.sas.com/

Rafraîchissement (1)

- Détection de changements dans les sources et propagation vers l'entrepôt
- Deux approches
  - Reconstruction (plus simple, plus longue)

    Élimination de l'ancien contenu de l'entrepôt et chargement à nouveau
    - Périodique
    - Longue période d'indisponibilité
  - Incrémental (plus complexe, plus rapide)

    Mise à jour de l'entrepôt en utilisant les changements dans les sources détectés par le *wrapper* et traités par l'intégrateur
    - Périodique

      Volume de données limité
    - Instantanée

      Nécessite de nombreuses communications

Rafraîchissement (2)

- La détection des changements dépend des sources
  - Triggers utilisés pour déclencher la mise à jour
  - Exploitation des journaux (*logs*) des changements
  - Extractions des changements pertinents par requêtes
  - Comparaison de différentes images de la sources

- Dans les bases de données relationnelles utilisation du rafraîchissement incrémental
  - Contexte -> vues matérialisées représentant des agrégats pré calculés
  - Maintenance d'une vue
    - la source signale les mises à jour
    - L'entrepôt questionne la source
    - La source envoie les données concernées
    - L'entrepôt met la vue à jour

Rafraîchissement (3)

La collecte et l'intégration de données restent encore des processus intuitifs, réalisés à l'aide d'outils semi-automatiques

En général, ces processus sont mal documentés et ils ne peuvent pas être facilement évalués

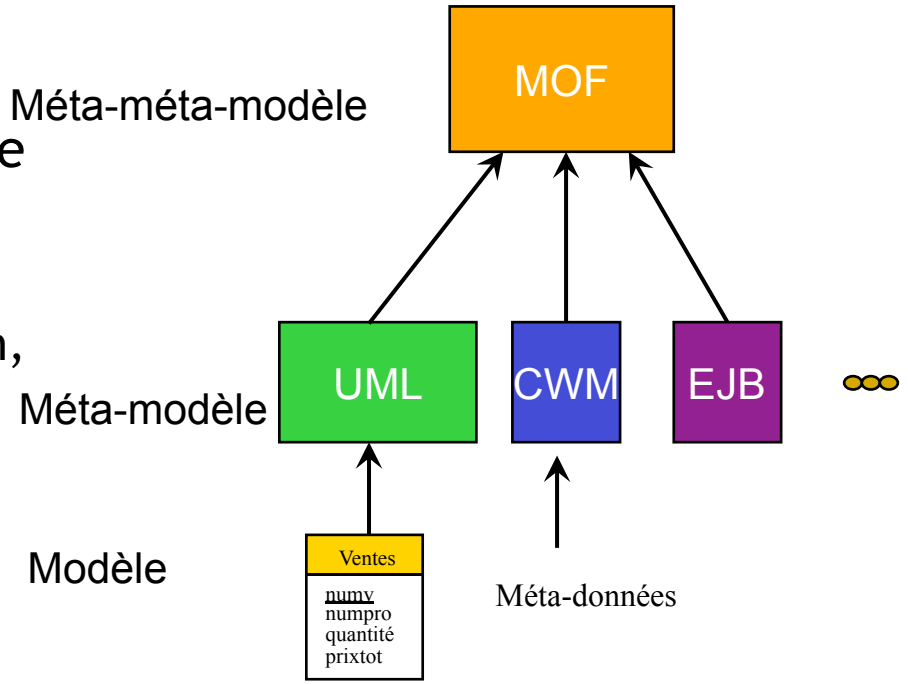
Métadonnées (1)

Sont les données qui décrivent l'entrepôt de données

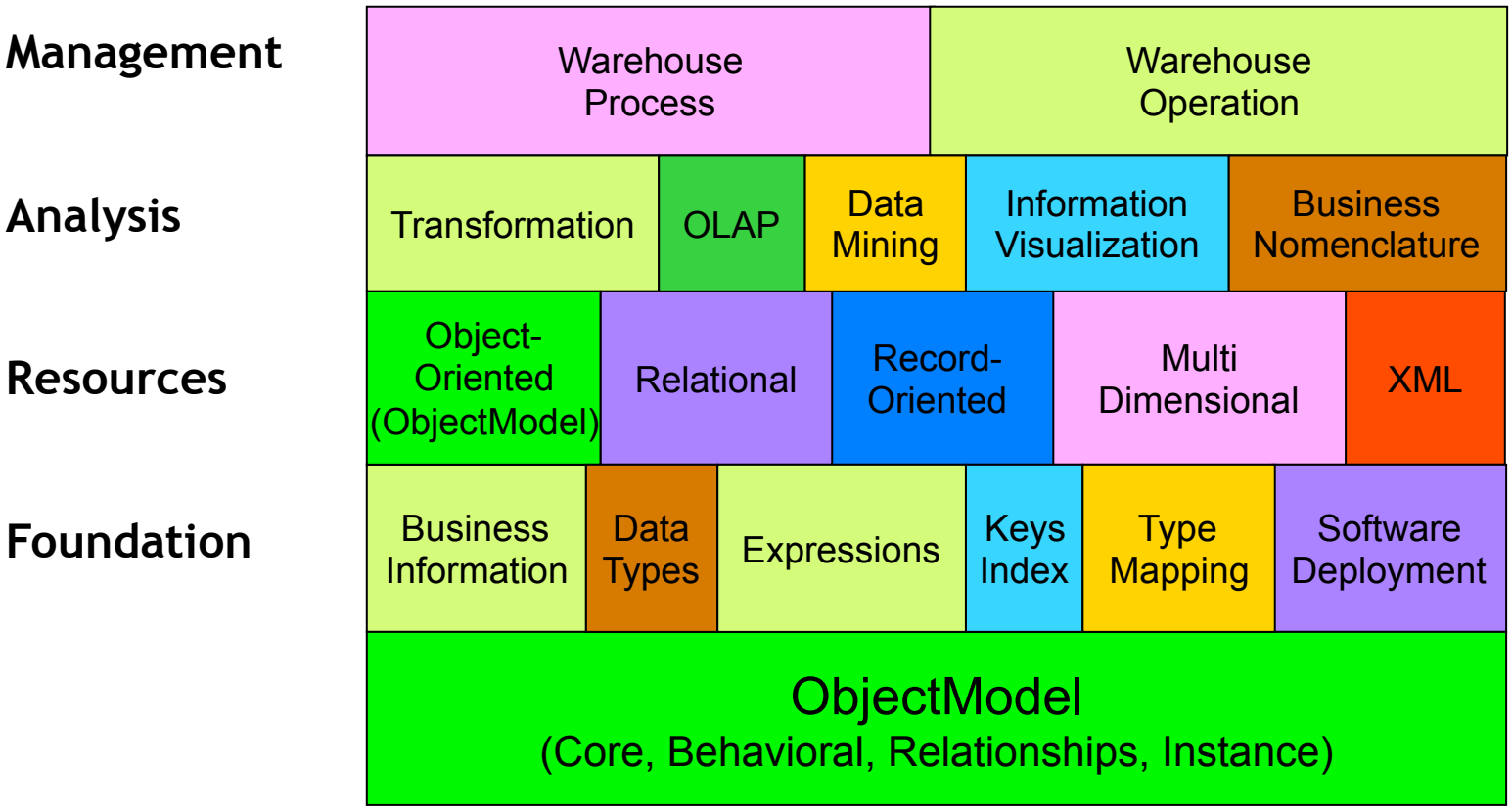
- Structure
  - Schéma, vues, dimensions, hiérarchies, les données dérivées, localisation et contenu des data mart
- Données opérationnelles
  - Historique des données migrées, processus de transformation, état des données, information de monitorage (statistiques, rapports d'erreurs, etc.)
- Algorithmes utilisés pour les résumés
- Mapping de l'environnement opérationnel vers l'entrepôt de données
- Données liées aux performances du systèmes
- Données économiques
  - Terminologie, définitions, propriétaires des données, politiques de chargement, etc.

Métadonnées (2)

- CWM (Common Warehouse Meta-model)
  - Standard proposé par l' OMG (Object Management Group)
- Basé sur le méta-modèle objet de l'OMG (MOF)
  - Constructions de base: classe (attribut, operation), association, package, type de données, contraintes
  - Extensions: métaclasses, métarelations
  - Défini en UML
- Echange en XML (XMI)



Packages CWM



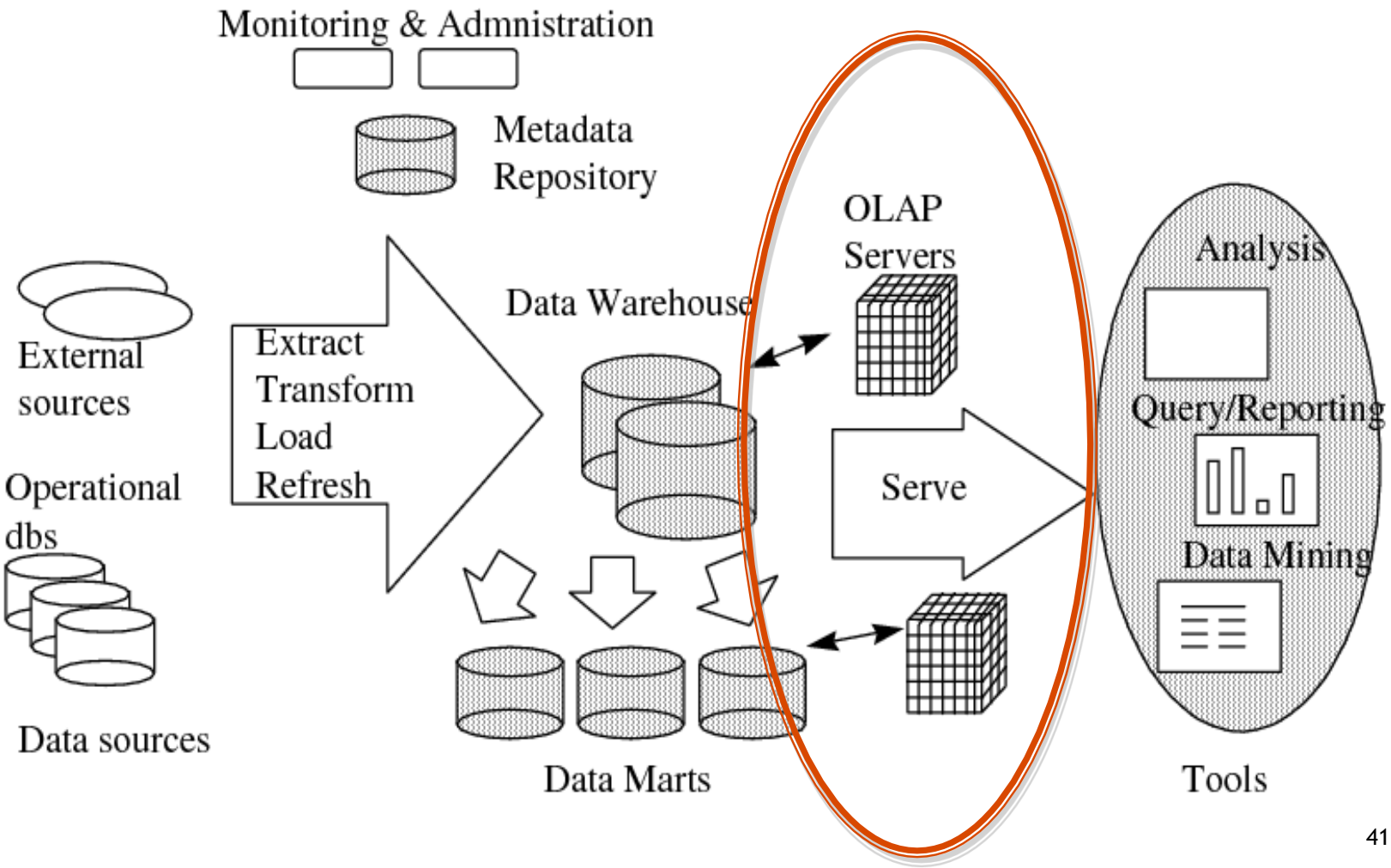
Chaque package est défini en UML ...

Data Marts

- Appelé magasin de données : entrepôt plus petit
- C'est comme un entrepôt mais il se spécialise dans un domaine « métier » ou sujet particulier (e.g., gestion de la relation client ou gestion de la chaîne logistique)
- En général, il se situe en aval d'un entrepôt et est alimenté à partir de cet entrepôt, dont il constitue un extrait
- Plusieurs data marts peuvent être conçus pour répondre chacun à un besoin différent

Modélisation dimensionnelle

Architecture type – Architectures OLAP



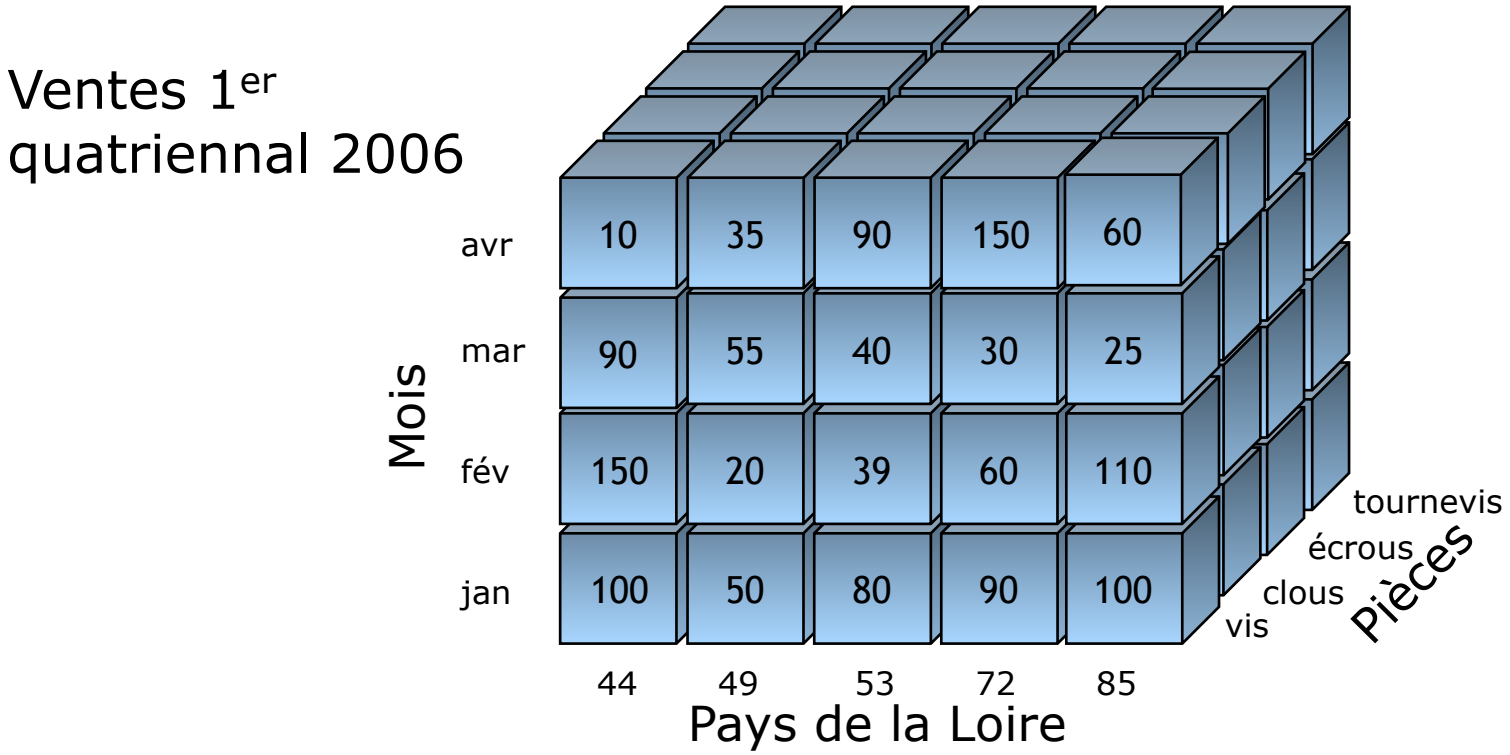
Modélisation et opérateurs

- Modèle multidimensionnel
  - Structure
  - Opérateurs OLAP
- Architectures OLAP
  - ROLAP (Relational OLAP)
  - MOLAP (Multidimensional OLAP)
  - HOLAP (Hybrid OLAP)

Structure multidimensionnelle

- En OLAP l'information est organisée sous une forme conceptuelle proche de la perception des analystes

- Une donnée est un point (cellule) dans un espace multidimensionnel



Faits et dimensions (1)

- Une structure multidimensionnelle (fréquemment appelée cube ou hypercube) est composée de faits et dimensions

- Fait ou métrique
  - Mesure ou indicateur d'analyse composée de dimensions
  - Composée d'attributs souvent servant à faire de calculs (numériques)

Ventes	Dép	Mois	Pièces	Qté
	44	Jan	Vis	100
	49	Jan	Vis	50

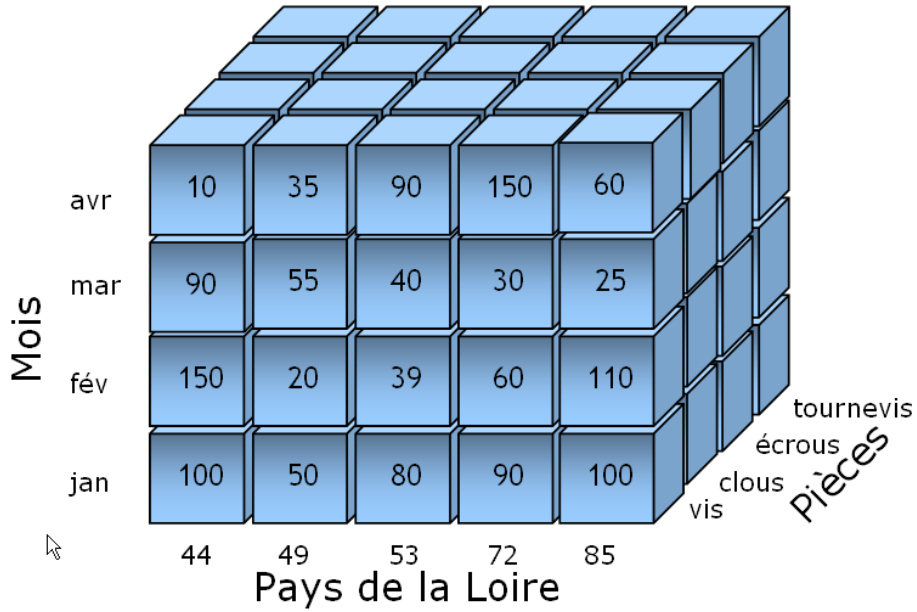
- Dimension ou axe
  - Point d'entrée à un ou plusieurs faits
  - Composée d'attributs souvent servant à décrire (textuels)

  Dans l'exemple 3 dimensions :

  Départements

  Mois

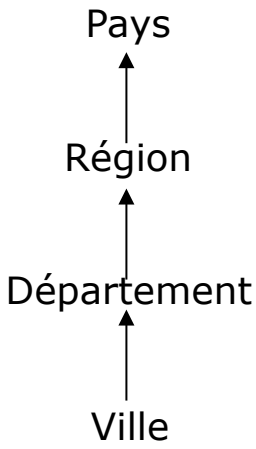
  Pièces



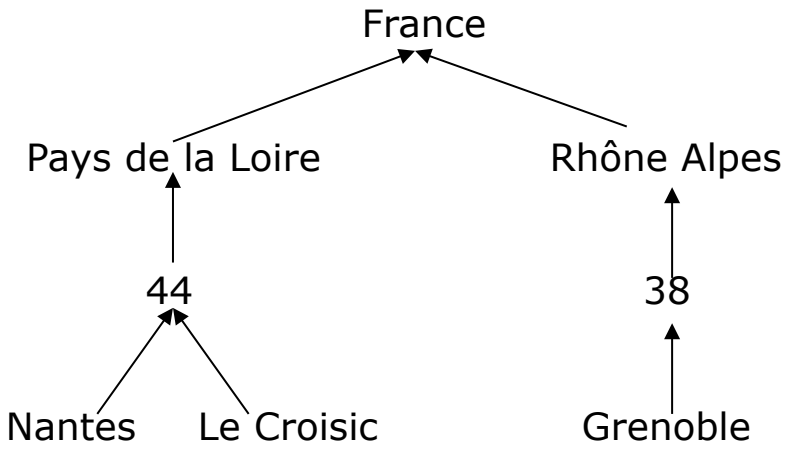
Faits et dimensions (2)

- Chaque dimension fournit un ensemble de propriétés
  - Mois : Nom, quatriennal, nombre de jours, etc.
  - Pièces : Nom, code, catégorie, rayon, etc.
  - Département : Nom, numéro, région, villes, etc.

- Les dimensions peuvent être structurées hiérarchiquement selon des niveaux d'agrégation



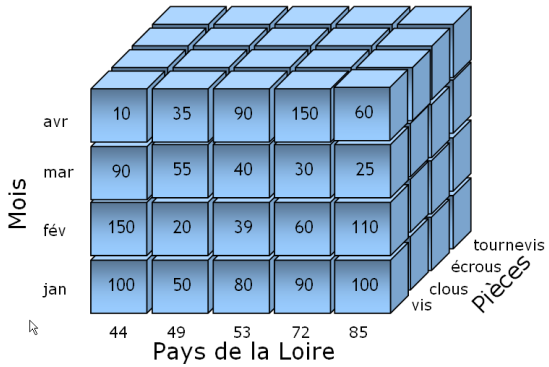
Hiérarchie associée à la dimension département



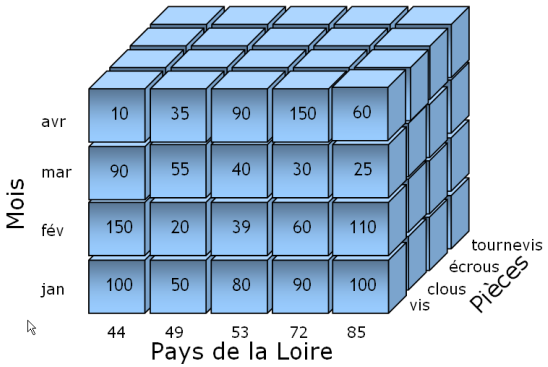
Instance de la hiérarchie

Faits et dimensions (3)

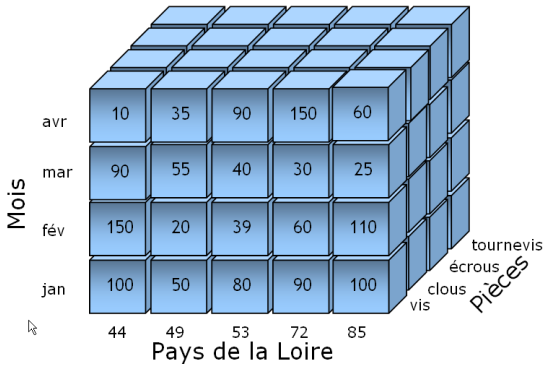
- Un exemple à quatre dimensions
  - Département
  - Mois
  - Pièces
  - Fournisseurs



Fournisseur A



Fournisseur B



Fournisseur C

Opérateurs OLAP

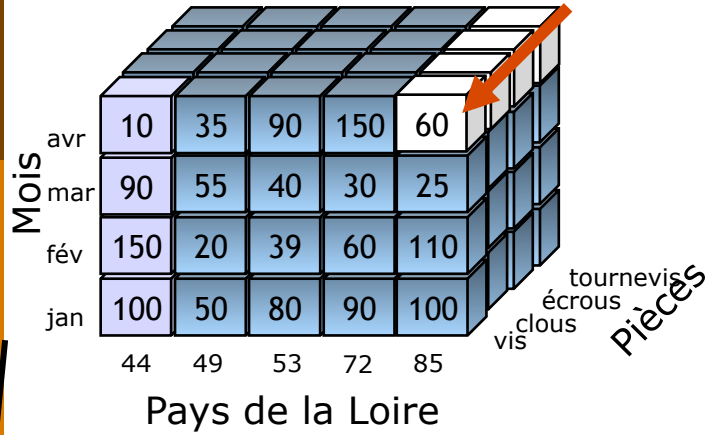
- Opérations pour naviguer dans les cubes afin d'extraire de l'information pour visualisation, analyse ou traitement

- Opérateurs de restructuration
  - Pivot/rotate,
  - Switch

- Opérateurs de granularité
  - Drill-up/Roll-up,
  - Drill-down

- Opérateurs ensemblistes
  - Slice
  - Dice

Rotate/pivot



- Changement de point de vue ou
- Rendre visible de nouvelles faces de ce cube

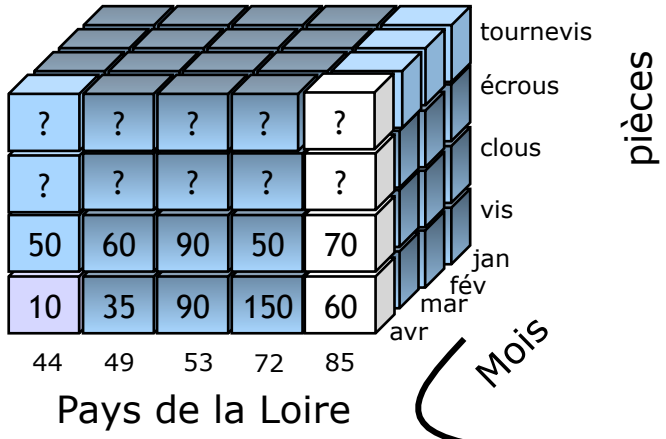
exemple 1 :

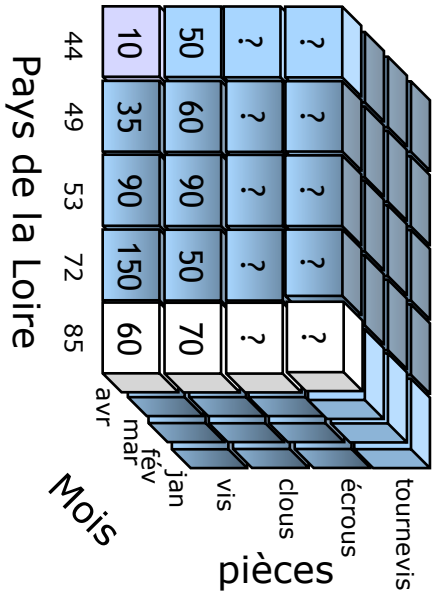
- Du Mois et Pays de la Loire
- À Pièces et Pays de la Loire

exemple 2

- Mêmes facettes mais différent point de vue

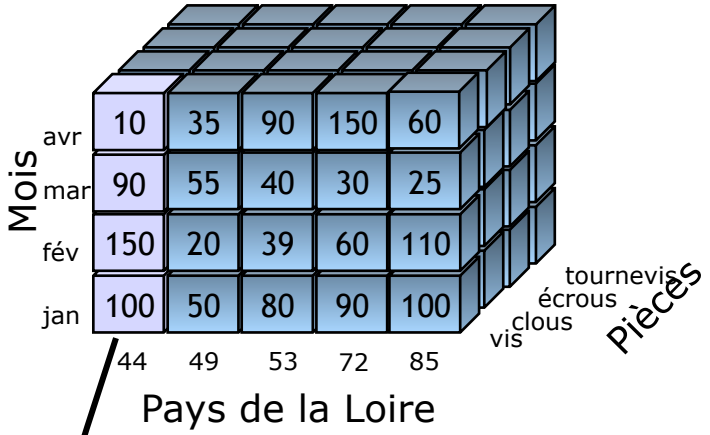
Exe 1





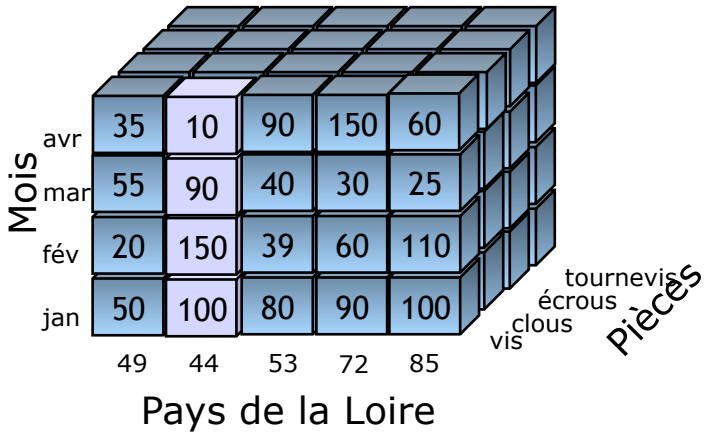
Exe 2

Switch

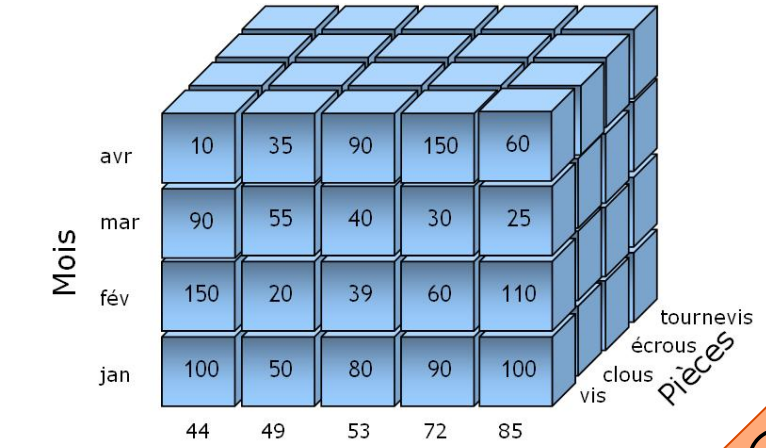


- Changement de positions de deux membres d'une même dimension
- Pas de changement de faces à déployer

  exemple
  - Échange des département 44 et 49



Drill-up/roll-up

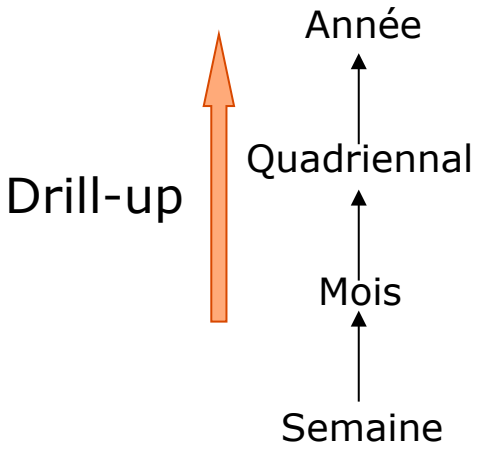


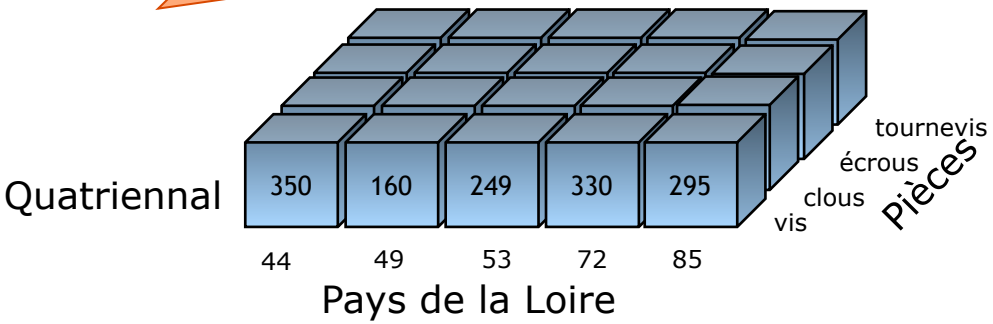
Départements de la région Loire Atlantique

- Représentation du cube en un niveau plus élevé de granularité
- Moins de détails

  exemple
  - Passage du mensuel au quatriennal

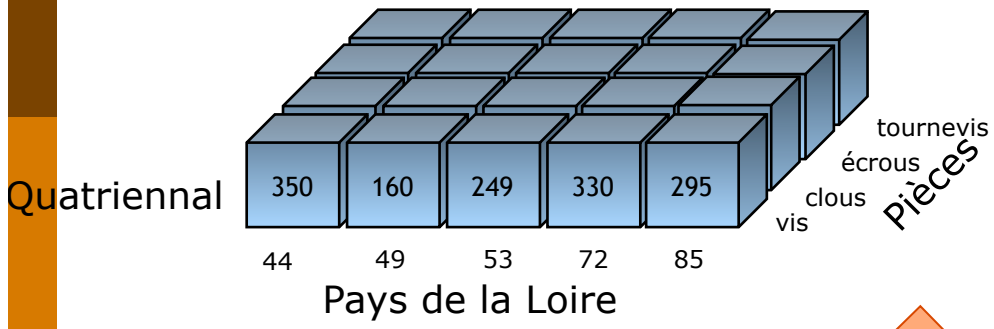
Drill-up (sur le temps)





Pays de la Loire

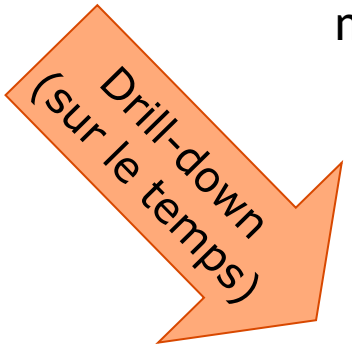
Drill-down

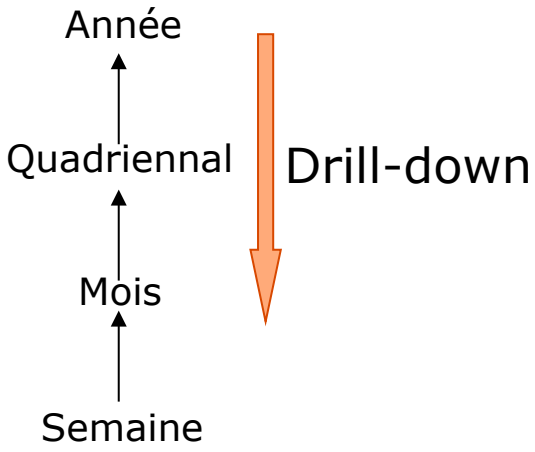


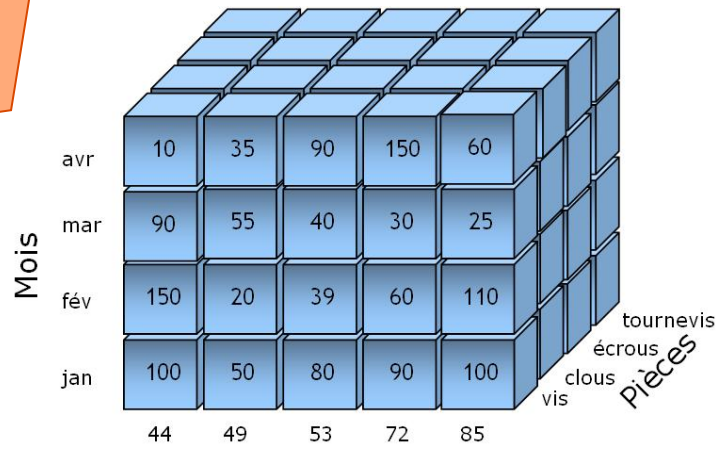
- Représentation du cube en un niveau mois élevé de granularité
- Plus de détails

exemple

- Passage du quatriennal au mensuel

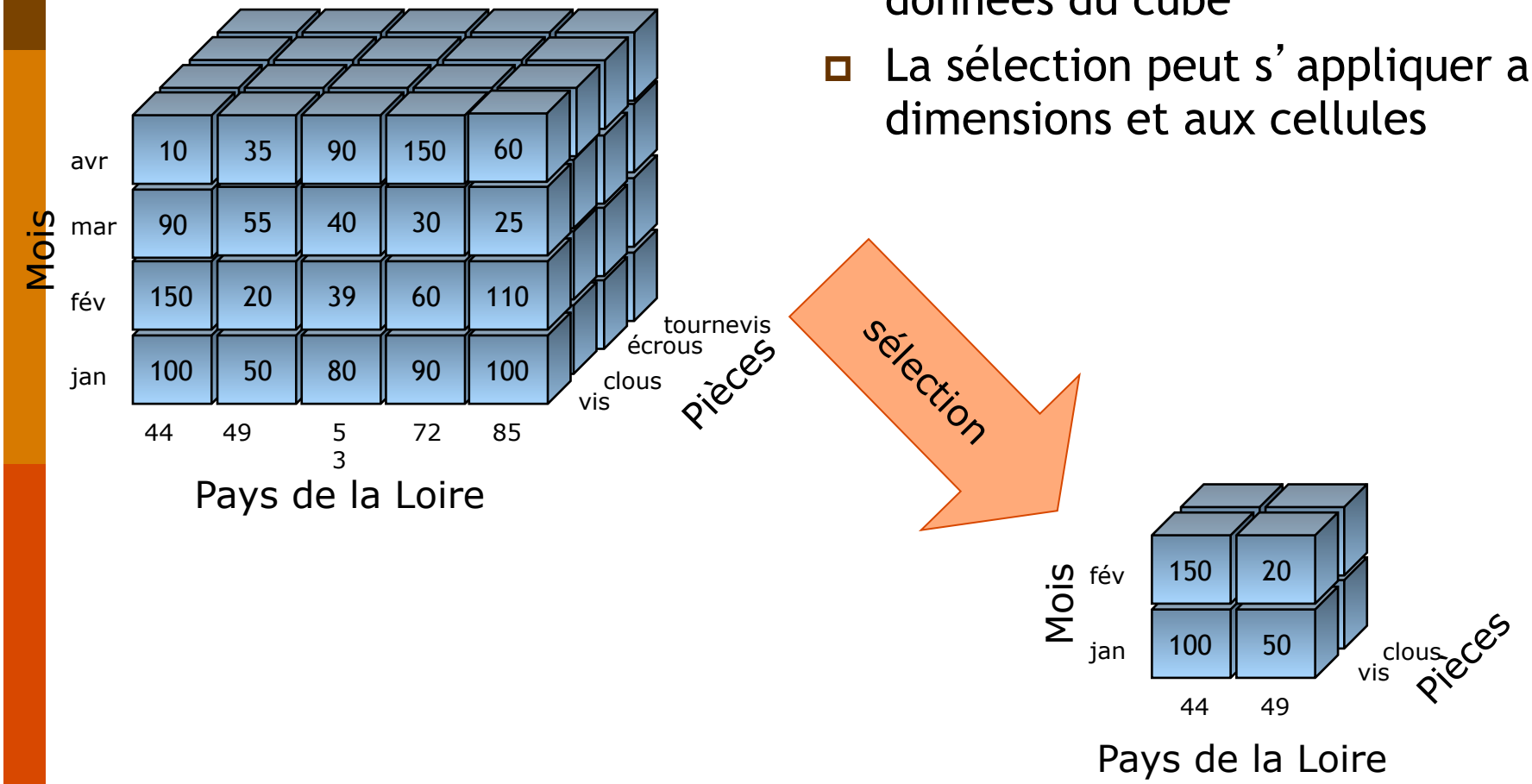






Sélection

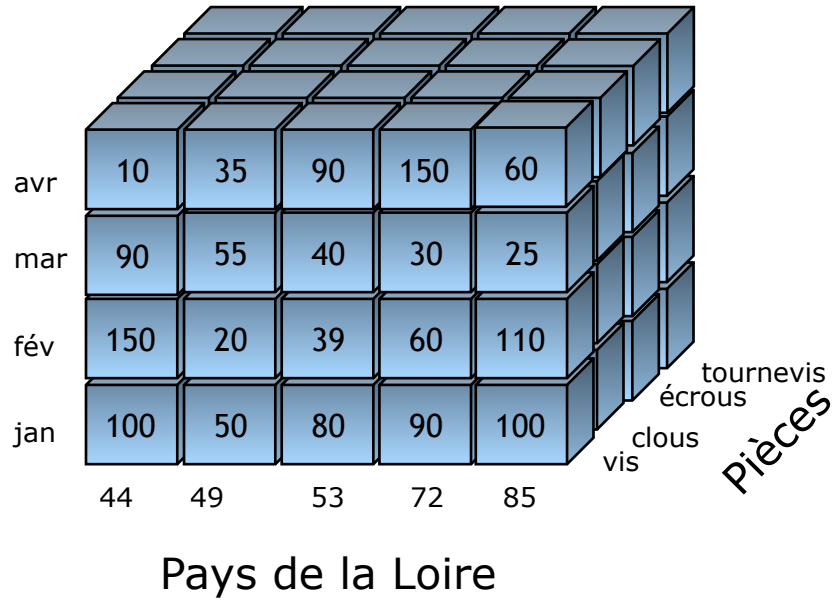
- Sélectionne un sous-ensemble des données du cube
- La sélection peut s'appliquer aux dimensions et aux cellules

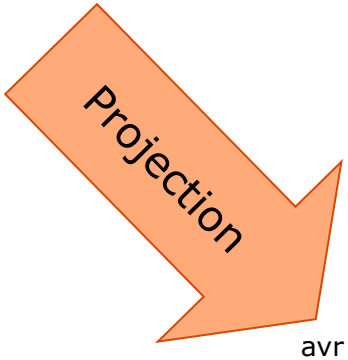


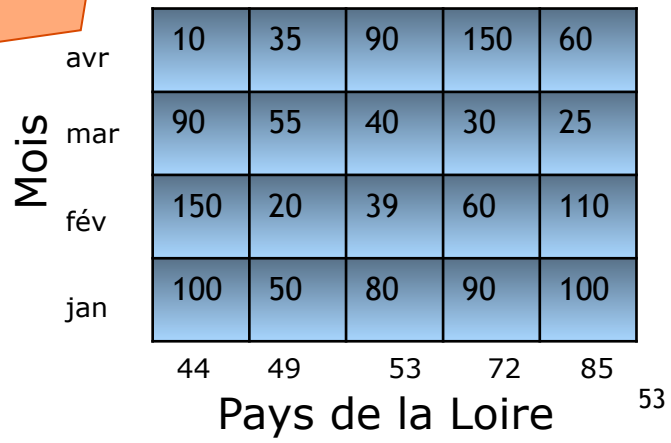
Projection

- Restreint l'ensemble de dimensions

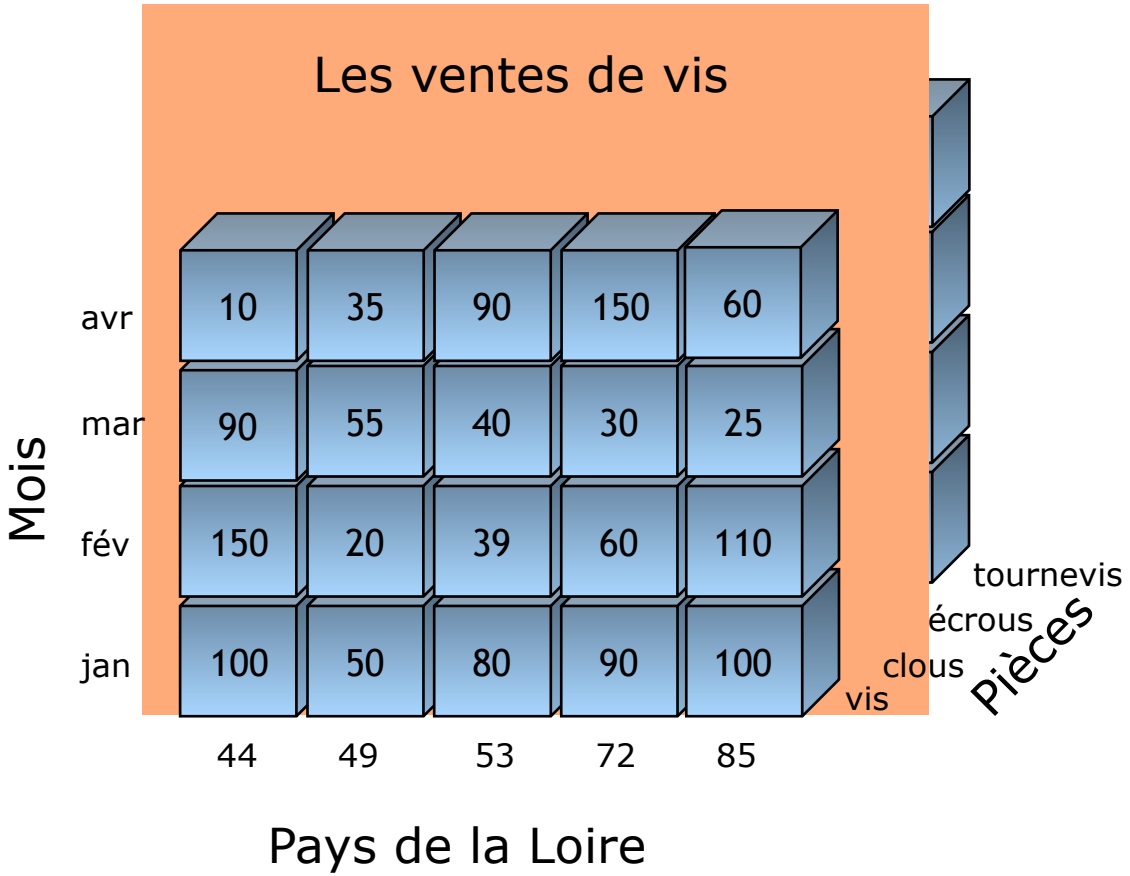
  exemple
  - De (Mois, Pay de la Loire, Pièces) à (Mois, Pays de la Loire)



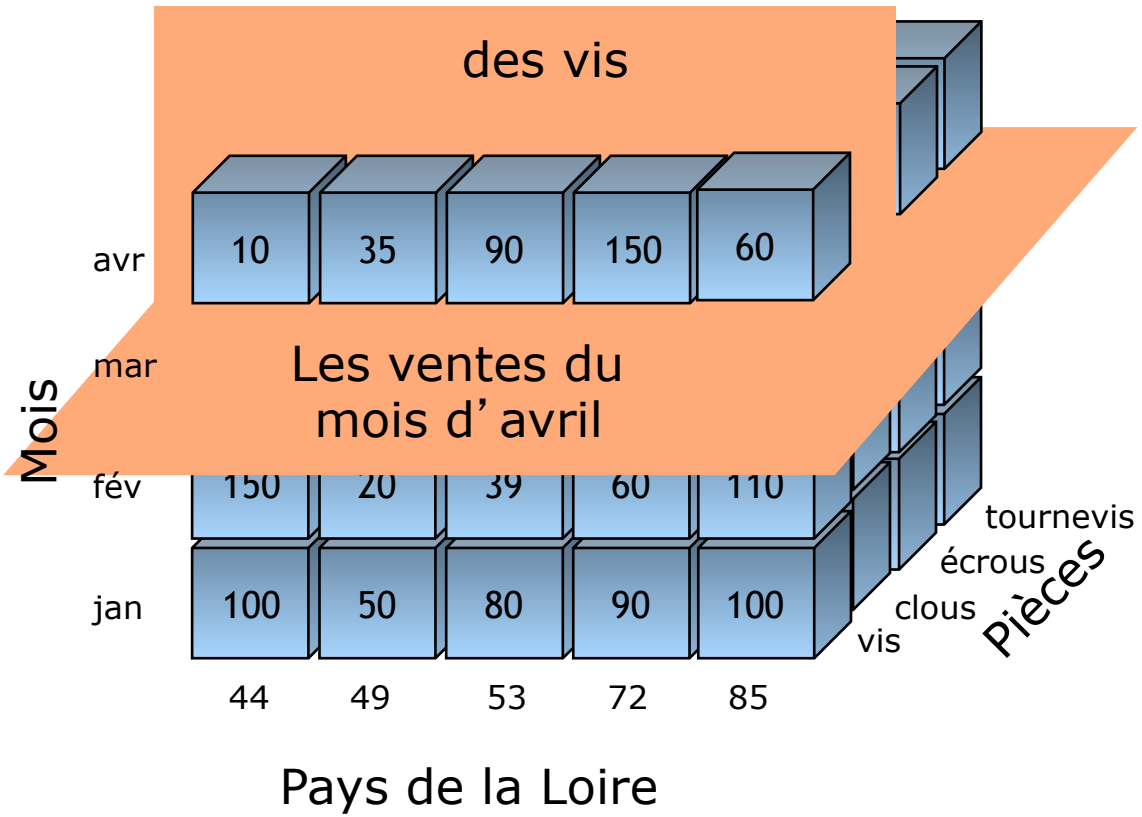




Slice


Les ventes de vis
Mois
avr
mar
fév
jan
10 35 90 150 60
90 55 40 30 25
150 20 39 60 110
100 50 80 90 100
44 49 53 72 85
Pays de la Loire
vis
clous
écrous
tournevis
Pièces

Slice and dice



Architectures OLAP

- MOLAP (*Multidimensional OLAP*)
  - Stockage basé sur des vecteurs multidimensionnels
- ROLAP (*Relational OLAP*)
  - Utilise les bases de données relationnelles et les technologies middleware pour le support des pièces manquantes
  - Inclut l'optimisation de SGBD, l'implémentation d'une logique de navigation à base d'agrégats
  - Passage à l'échelle
- HOLAP (*Hybrid OLAP*)
  - Flexibilité d'utilisation, e.g., bas niveau : relationnel, haut niveau : vecteurs
  - Serveurs SQL spécialisés
  - Support spécialisé pour les requêtes SQL sur les schémas en étoile et flocon de neige

MOLAP

- Les données sont stockées en vecteurs multidimensionnels

- Avantages
  - Bonne performance : données construites pour fournir des réponses rapides

    Les opérations slice/dice sont particulièrement performantes
  - Calcul complexe : grâce au fait que tous les calculs ont été pré générés lors de la création de l'hypercube

- Désavantages:
  - Quantité limitée de données manipulable : à cause de la réalisation de calculs au moment de la construction du cube. Cependant, on peut inclure de l'information agrégée dans le cube
  - Stockage non optimisé : le vecteur multidimensionnel peut être creux (certaines cellules n'ont pas de valeur associée)
  - Demande d'un investissement additionnel : les technologies MOLAP sont souvent propriétaires et souvent elles n'existent pas dans l'organisation de l'entreprise

ROLAP

- Les données sont stockées en bases de données (BD) relationnelles
  - Schémas en étoile, flocon de neige et constellation de faits
- Avantages
  - Maturité de la technologie relationnelle
  - Manipulation de grandes quantités de données : la limite est la taille de la BD
  - Utilisation de fonctionnalités inhérentes aux BD relationnelles
- Désavantages:
  - Probable mauvaise performance : les opérations ROLAP sont des requêtes dont l'exécution dépend de la taille de la BD
  - Fonctionnalités limitées par SQL

HOLAP

- MOLAP fonctionne bien sur les cubes denses
- ROLAP fonctionne bien sur les cubes peu denses

- Observations :
  - La plupart des cubes sont peu dense
  - Les sous-cubes sont denses
  - Les données agrégées sont les plus denses

- HOLAP : combinaison des technologies ROLAP et MOLAP
  - Données détaillées dans BDR
  - Données agrégées dans BDMD
  - Granularité moins fine
  - Index en mémoire centrale

Techniques d'indexation

- Les indexes basés sur arbres B ne sont pas adéquats pour l'analyse de données
  - Volumineux
  - Lent
- Index bitmap
  - Microsoft SQL server
  - Sybase IQ
  - Oracle 8
- Index join
  - Oracle 8

Index bitmap (1)

- Idéal pour
  - Des attributs qui ont peu de valeurs distincts (sexe, situation familiale, etc.)
  - Des colonnes avec un nombre important de lignes (entrepôt)
  - Des données qui ne sont pas modifiées fréquemment (pas performant dans les systèmes OLTP)
- Une valeur d'index est associée à une suite de bits
  - Chaque bit correspond à un enregistrement
  - Si le bit est à 1, l'enregistrement contient la valeur d'index
  - L'accès s'effectue par la lecture de l'index
- Avantages
  - Meilleures performances (que les arbres B) du fait de l'utilisation d'opérations binaires (AND, OR, XOR, NOT)
  - Plus compact que les arbres B
  - Structure qui se prête bien à la compression (ceci facilite une lecture rapide)

Index bitmap (2)

Exemples simples

- La table Personnes

Ligne	Nom	Sexe	Age
1	Martin	Homme	30
2	Monroe	Femme	25
3	Sosa	Homme	22
4	Zola	Femme	18

- Index à 2 bits pour l'attribut sexe

Ligne	Homme	Femme
1	1	0
2	0	1
3	1	0
4	0	1

- La table Géographie de France

Row	City	Region
1	Lille	North
2	Nancy	East
3	Nantes	West
4	Rennes	West
5	Montpellier	South
6	Dunkerque	North

- Index à 4 bits pour l'attribut région

Row	North	East	West	South
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0

Index bitmap (3)

Exemples avec des valeurs continues

- Plusieurs valeurs d'un attribut correspondent à un colonne dans le bitmap

- La table Ménagères

Ligne	Ménagère	Produits	Coût
1	A	{P1,P3,P5}	120
2	B	{P2,P3}	70
3	C	{P4}	150
4	D	{P2,P5}	110
5	E	{P3,P4,P6}	220

- Index à 3 bits pour l'attribut coût

Ligne	0-100	100-200	200-300
1	0	1	0
2	1	0	0
3	0	1	0
4	0	1	0
5	0	0	1

- Index à 6 bits pour l'attribut produits

Ligne	P1	P2	P3	P4	P5	P6
1	1	0	1	0	1	0
0	0	1	1	0	0	0
3	0	0	0	1	0	0
4	0	1	0	0	1	0
5	0	0	1	1	0	1

Index bitmap (4)

- Création d'indexes
  - Sur des tables très grandes (milliers de lignes)
  - Sur des attributs qui risquent de se trouver fréquemment dans la clause WHERE

- Exemple

```
CREATE BITMAP INDEX
bitmap_Index_Personnes_Sexe
ON Personnes(Sexe);
```

```
SELECT Nom
FROM Personnes
WHERE Sexe= 'Homme' ;
```

```
SELECT Count(*)
FROM Personnes
WHERE Sexe= 'Femme'
```

Index join (1)

- Idéal pour
  - Des tables qui vont être jointes fréquemment
  - Rendre plus rapide les requêtes de jointure
  - Des tables très grandes
- Les indexes sont pré calculés selon une condition de jointure
  - Les jointures peuvent être entre 2 ou plus tables, sur un ou plusieurs attributs
  - L'index stocke d'après les valeurs des jointures
- Désavantages
  - Les indexes peuvent croître rapidement si plusieurs colonnes de chaque dimension sont considérées dans la jointure (voir bitmap join index)

Index join (2)

- On suppose que chaque transaction représente la vente d'un produit
- Table Ventes

Ligne1	Trans_id	P_id
50	10	P1
51	20	P3
52	30	P1
53	40	P2
54	50	P3

- Table Produits

Ligne2	P_id	Produit	Prix
100	P1	A	120
101	P2	B	70
102	P3	C	150

- Index join des tables Ventes et Produit sur les attributs P_id

Ligne	Ligne 1	Ligne2
1	50	100
2	51	102
3	52	100
4	53	101
5	54	102

Bitmap join index

- Idéal pour
  - Réduire la taille des index join
  - Rendre plus rapide l'accès des index join
- Création des bitmap join index
  1. D'abord l'index join est créé
  2. Ensuite il est définit comme un index bitmap
- Avantages
  - Celles des indexes bitmap plus
  - Celles des indexes join
- Exemple

```
CREATE BITMAP INDEX cust_sales_bji
ON sales(customers.state)
FROM sales, customers
WHERE sales.cust_id = customers.cust_id;
```

Conception

Niveau conceptuel

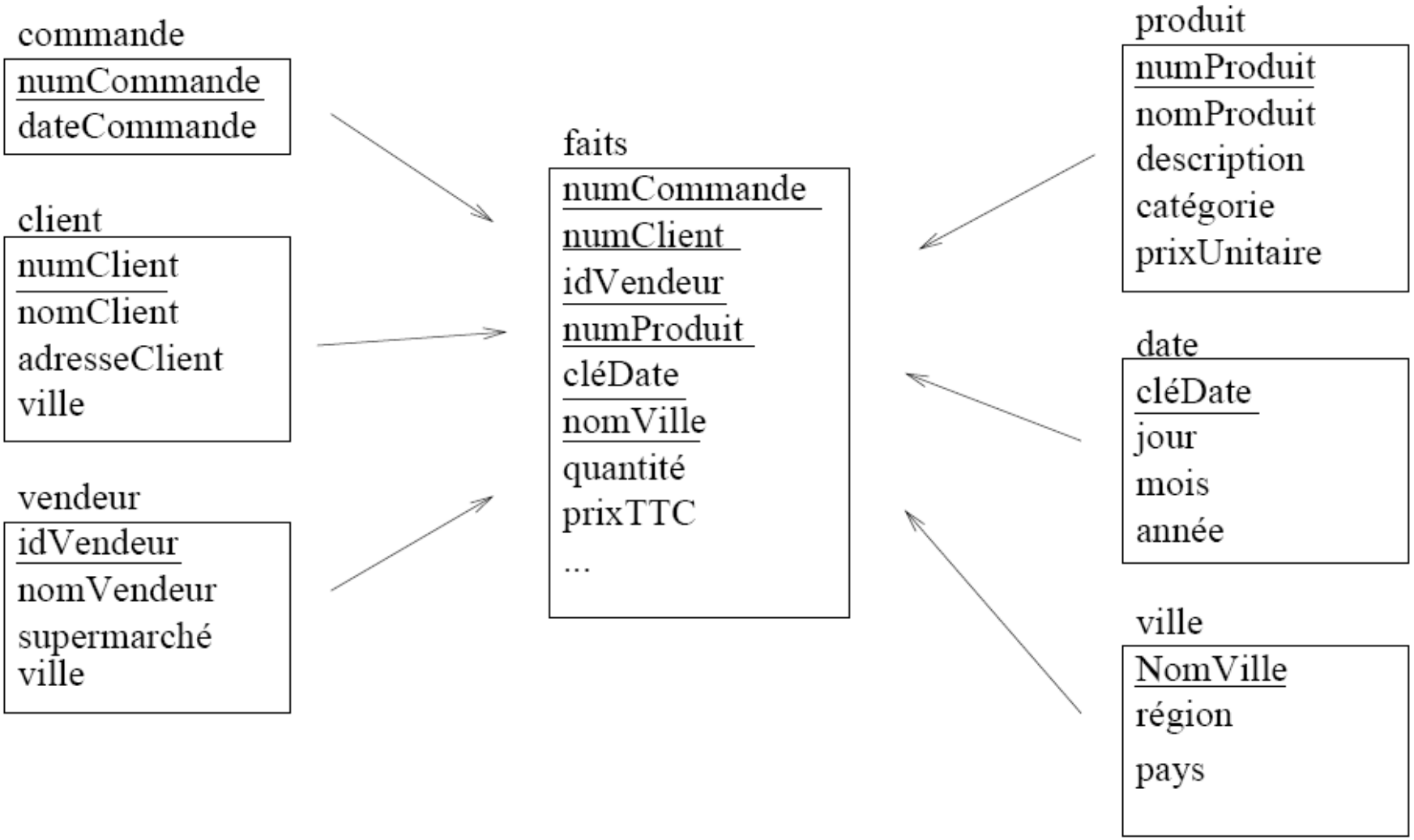
- Schéma de BD relationnelle reflétant la vue de l'analyste :
  - Multidimensionnelle
  - Hiérarchisée

- Schéma en étoile (*star schema*)
- Schéma en flocon (*snowflake schema*)
- Constellation de faits (*fact constellation*)

Schéma en étoile

- Structure simple utilisant le modèle entité relation
  - Une entité centrale (table des faits)
    - Objets de l'analyse
    - Taille très importante (cardinalité très grande)
    - Beaucoup d'attributs
  - Des entités périphériques (tables de dimensions)
    - Dimensions de l'analyse
    - Taille peu importante
    - Peu de champs

Schéma en étoile



Normalisation

Tables de dimensions

- Contiennent le plus souvent des informations textuelles descriptives
- Représentent une ou plusieurs hiérarchies
- Contiennent des données redondantes
- Pas/peu de mises à jour des dimensions
  - La perte d'espace n'est donc pas significative

- Les tables de dimensions sont dénormalisées en 2FN
  - 1FN + tout attribut qui n'est pas une clé dépend de la clé primaire entière (complète)

Pas 2FN

exemple:

produit	fournisseur	adresse fournisseur
A	dd	1, rue machin
B	dd	1, rue machin
A	ff	5, avenue bidule

2FN

solution:

produit	fournisseur
A	dd
B	dd
A	ff

fournisseur	adresse fournisseur
dd	1, rue machin
ff	5, avenue bidule

Normalisation

Tables des faits

- La table des faits constitue l'essentiel du stockage
- Les faits les plus utiles d'une table des fait sont numériques et additifs
  - Clés étrangères formant une clé primaire composée
  - Des mesures associées à chaque clé primaire

- Les tables des faits sont en 3FN
  - 2FN + pas de dépendance entre deux attributs qui ne sont pas des clés (primaires ou non)

Pas 3FN

exemple:

élève	nationalité	écolage
ww	belge	1500
xx	belge	1500
yy	français	1500
zz	australien	1850

3FN

élève	nationalité
ww	belge
xx	belge
yy	français
zz	australien

nationalité	écolage
belge	1500
français	1500
australien	1850

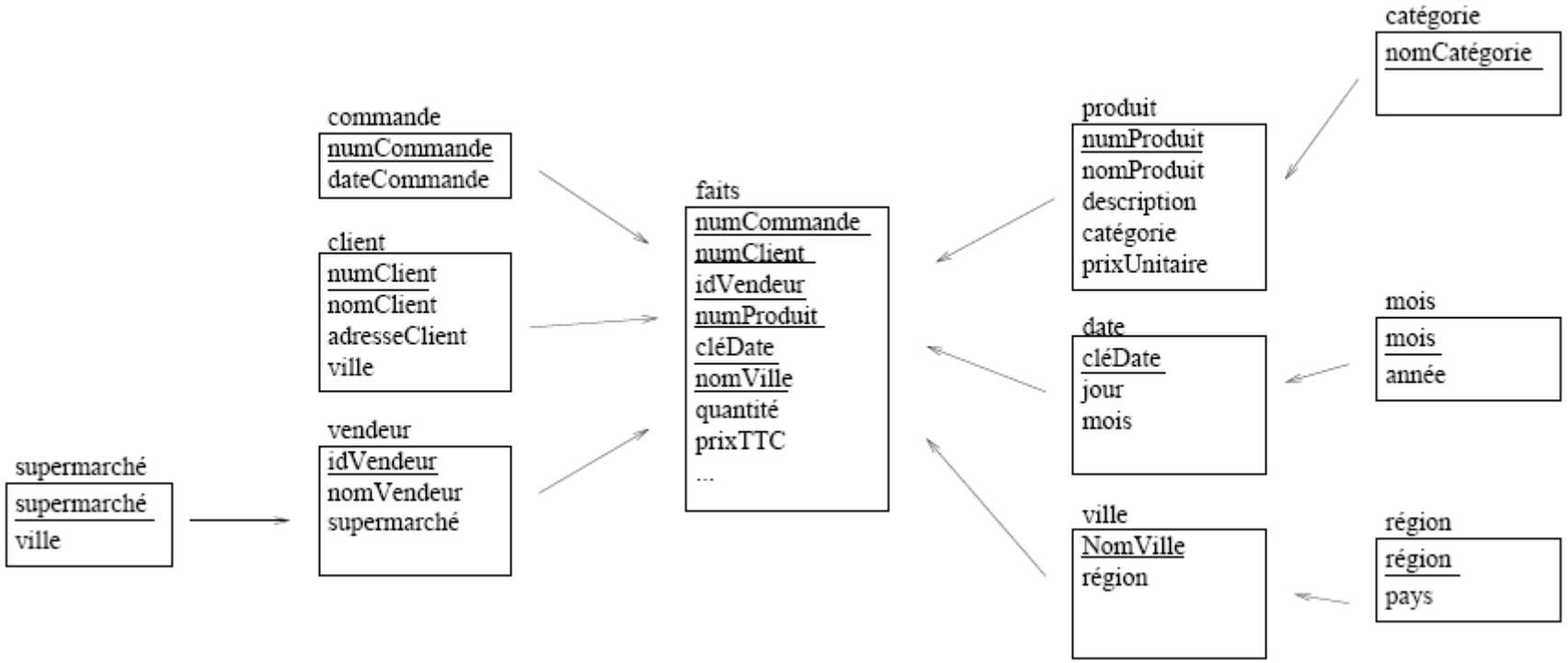
Schéma en flocon de neige

- Évolution du schéma en étoile
- Normalisation des tables de dimensions
- Structure hiérarchique des dimensions
- Un niveau inférieur identifie un niveau supérieur

- Avantages
  - Maintenance des tables de dimensions simplifiée
  - Réduction de la redondance

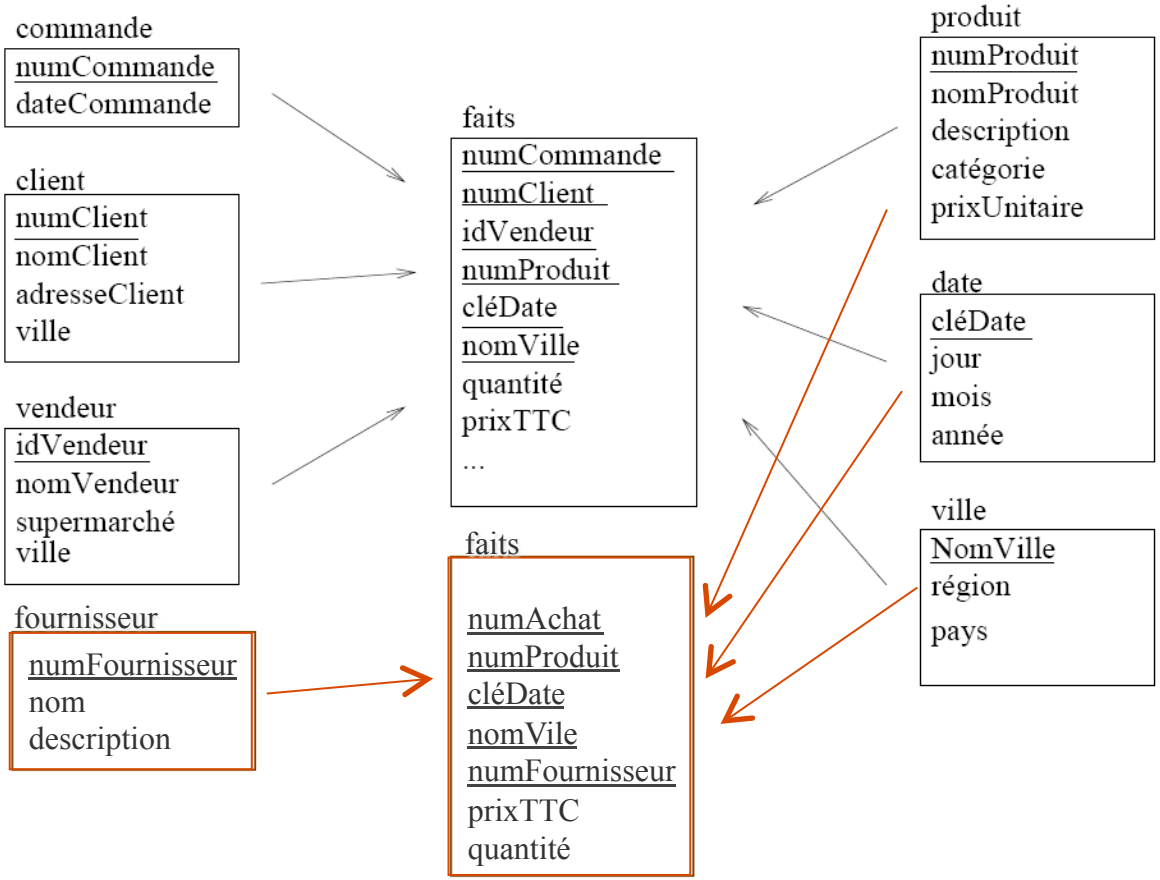
- Inconvénients
  - Navigation complexe, coûteuse et lente
  - Économie de espace disque insignifiante (inférieur à 1%)
  - Les flocons de neige empêchent l'utilisation des indexes bitmap

Schéma en flocon de neige



Constellation de faits

- Généralisation du schéma en étoile
  - Plusieurs tables des faits
  - Partage de tables de dimensions



Conception en 4 étapes

1. Sélectionner le processus d'entreprise à modéliser
2. Déclaration du grain du processus
3. Choix des dimensions
4. Identification des faits

Cas d'étude de la Distribution

- Siège d'une chaîne de magasins d'alimentation
- 100 magasins
- 5 régions

- Chaque magasin
  - +-60 000 produits/unités de stock (55 000 viennent de fabricants extérieurs)

- Management du magasin
  - Commandes
  - Stockage
  - Ventes
  - Profit maximal

1ère étape

Sélectionner le processus d'entreprise à modéliser

- Achats de clients aux Terminaux Point de Vente (TPV)

- Quels produits se vendent
  - Dans quels magasins,
  - Quels jours
  - Dans quelles conditions de promotion

2ème étape

Choisir le grain du processus -> décider de ce que représente une ligne de table de faits

- Grain le plus fin -> atomique (il n'est pas possible de le subdiviser davantage
- Les données atomiques sont hautement dimensionnelles
- Plus le niveau de détail des mesures est fin plus les analyses sont souples

- Grain
  - Ligne individuelle d'une transaction saisie au TPV (date, produit, etc.)

- Cela permet de connaître
  - Différences entre les ventes de lundi et de vendredi
  - Différentes marques pour un même produit
  - Les ventes sur une promotion particulière

3ième étape

Identifier les dimensions qui s'appliquent aux lignes de la table de faits

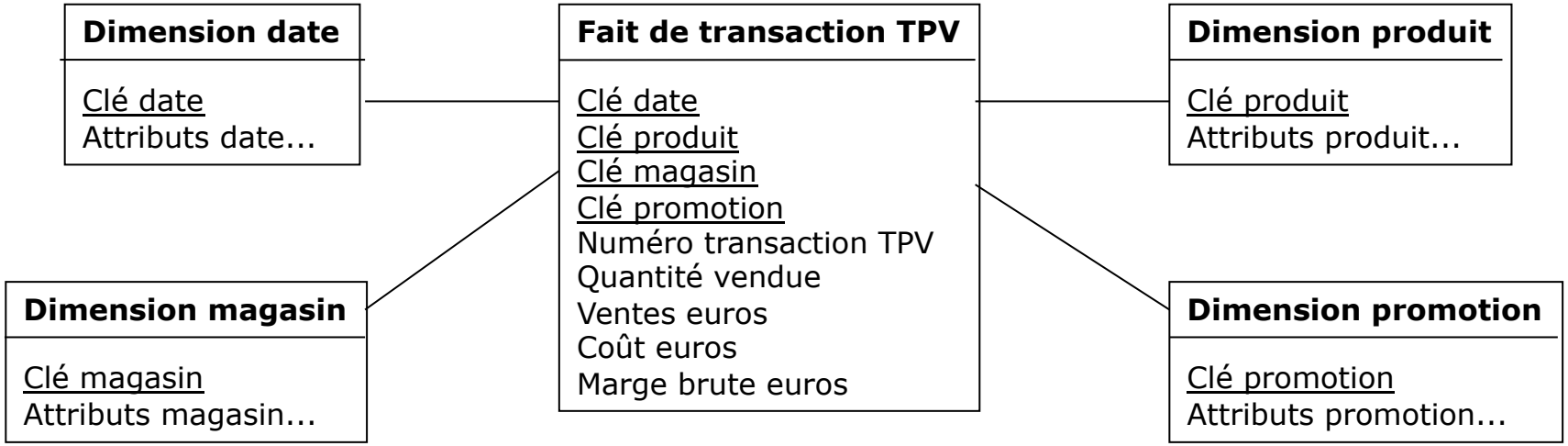
- Naturellement s' imposent les dimensions
  - Temps
  - Produit
  - Magasin
- D' autres dimensions peuvent être rajoutées sans pour autant détruire le grain choisi initialement
  - Promotion
  - Fournisseur
  - Etc.

4ième étape

- **Identifier les faits** numériques qui renseignent la table de faits

- D' après le processus TPV
  - Quantité vendue
  - Prix unitaire
  - Montant de la vente
  - Coût standard (tel qu' il est livré par le fournisseur)

Faits mesurés du schéma vente



- Ce schéma permet de calculer par exemple le bénéfice brut (en soustrayant le coût du montant de la vente)
  - Pour toute combinaison de produits vendus dans tous les magasins pour toute intervalle de jours

Dimension date

- Dimension très importante
- Habituellement la première dimension dans l'ordre de tri de la BD
- Elle peut être construite à l'avance
- Possibilité de prévoir cinq ou dix ans de lignes (dix années ->3 650 lignes)

Dimension date
Clé date Date Jour de la semaine Numéro de jour Numéro de semaine Numéro de mois Semestre Trimestre Semaine de l'exercice Mois de l'exercice Trimestre exercice Semestre de l'exercice Année de l'exercice Saison de vente Etc.

Dimension produit

- Décrit chacun des articles
- Chaque magasin +- 60 000 articles en stock
  - Plus différences de marchandises entre magasin
  - Plus produits pas disponibles
  - Cette dimension peut atteindre plus de 150 000 lignes

Dimension produit
Clé produit
Description produit
Numéro US
Marque
Catégorie
Sous-catégorie
Rayon
Type emballage
Taille emballage
Contenu matières grasses
Type de régime
Poids
Type de stockage
Largeur sur l'étagère
Hauteur sur l'étagère
Etc.

Dimension magasin

- Décrit chaque magasin de la chaîne de distribution alimentaire
- Dimension géographique
  - Code postal, province, état, région, commune, district, etc.

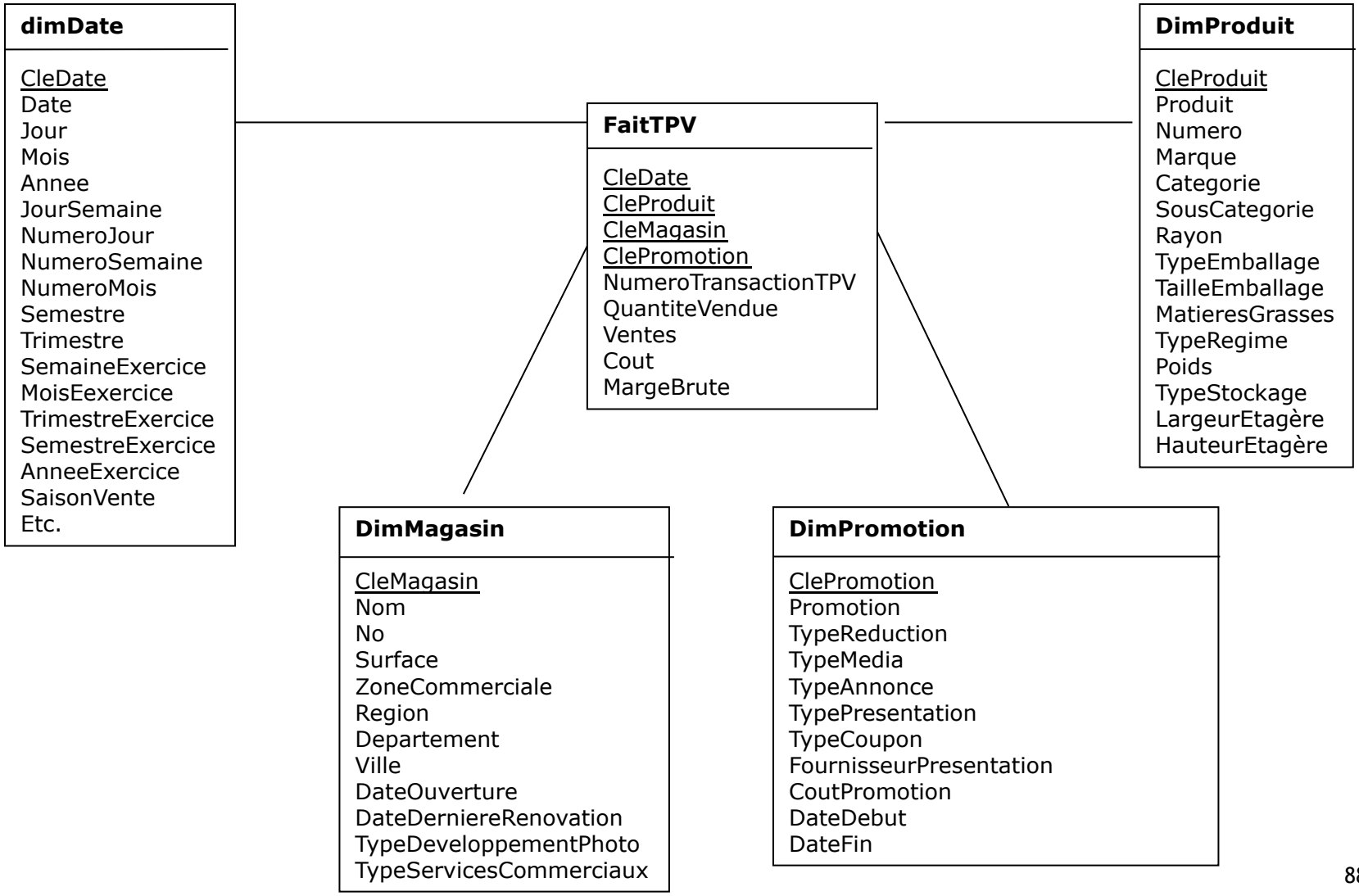
Dimension magasin
Clé magasin
Nom du magasin
No du magasin
Surface du magasin
Zone commerciale
Région
Département
Ville du magasin
Date première ouverture
Date dernière rénovation
Type développement photo
Type services commerciaux
Etc.

Dimension promotion

- Décrit chacune des modalités de promotion
  - Réductions temporaires, présentation en tête de gondole, annonces dans les journaux, coupons, etc.
- L'objectif -> savoir si une promotion est efficace ou non
  - Y a-t-il eu un gain?
  - Baisse de ventes avant ou après ?
  - Y a-t-il eu cannibalisation
  - Croissance du marché

Dimension promotion
Clé promotion Nom de la promotion Type de réduction de prix Type de media de la promotion Type d'annonce Type de présentation Type de coupon Fournisseur de la présentation Coût de la promotion Date de début de la promotion Date de la fin de la promotion Etc.

Schéma sur la Distribution



Quelques analyses typiques (1)

- Quelles ont été les ventes du mois de septembre par marque de produit ?

- Quelles ont été les ventes des promotions du mois de septembre par promotion ?

Quelques analyses typiques (2)

- Quelles ont été les ventes de « frites » à Nantes en septembre 2006 ?

- Quelles sont les différentes marques de « pâtes » vendues dans la région des Pays de la Loire ?

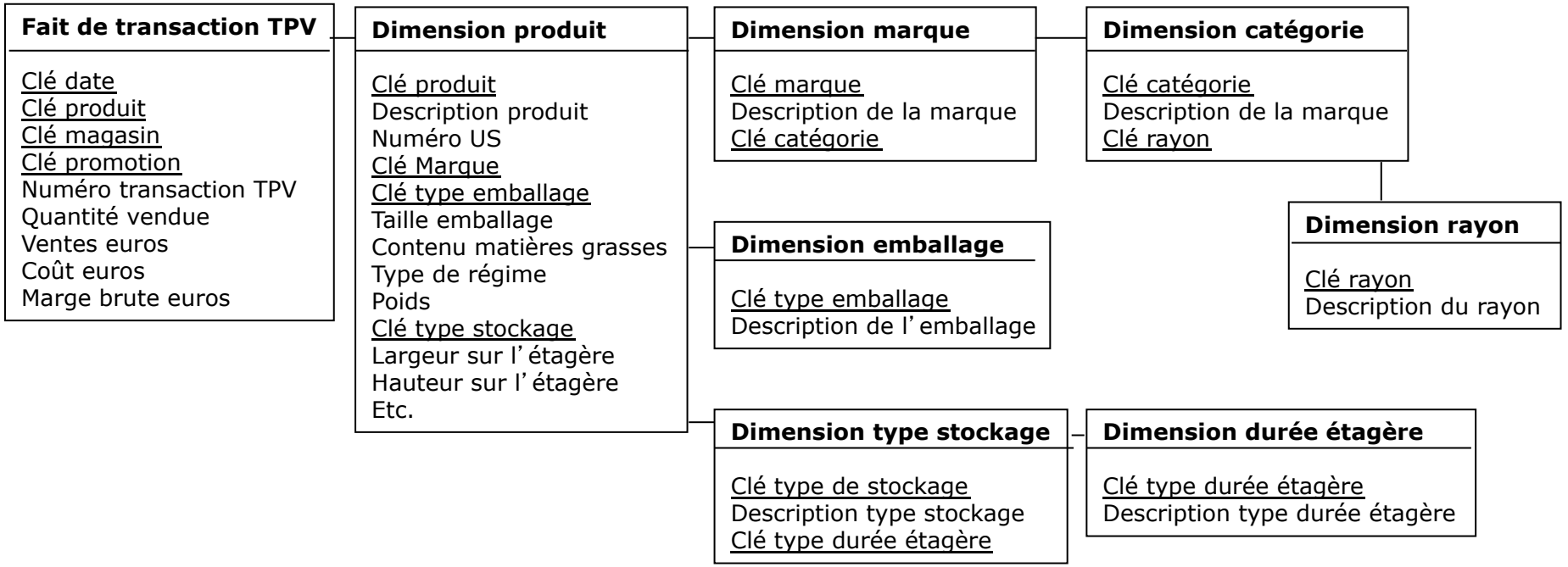
Quelques analyses typiques (3)

- Quelles sont les ventes de la marque « Bidon » lors de la promotion « Petite faim » ?

- Quelles ont été les ventes de l'année dernière sur la période de la promotion « Petite faim » de cette année ?

Le schéma Dimension en flocon

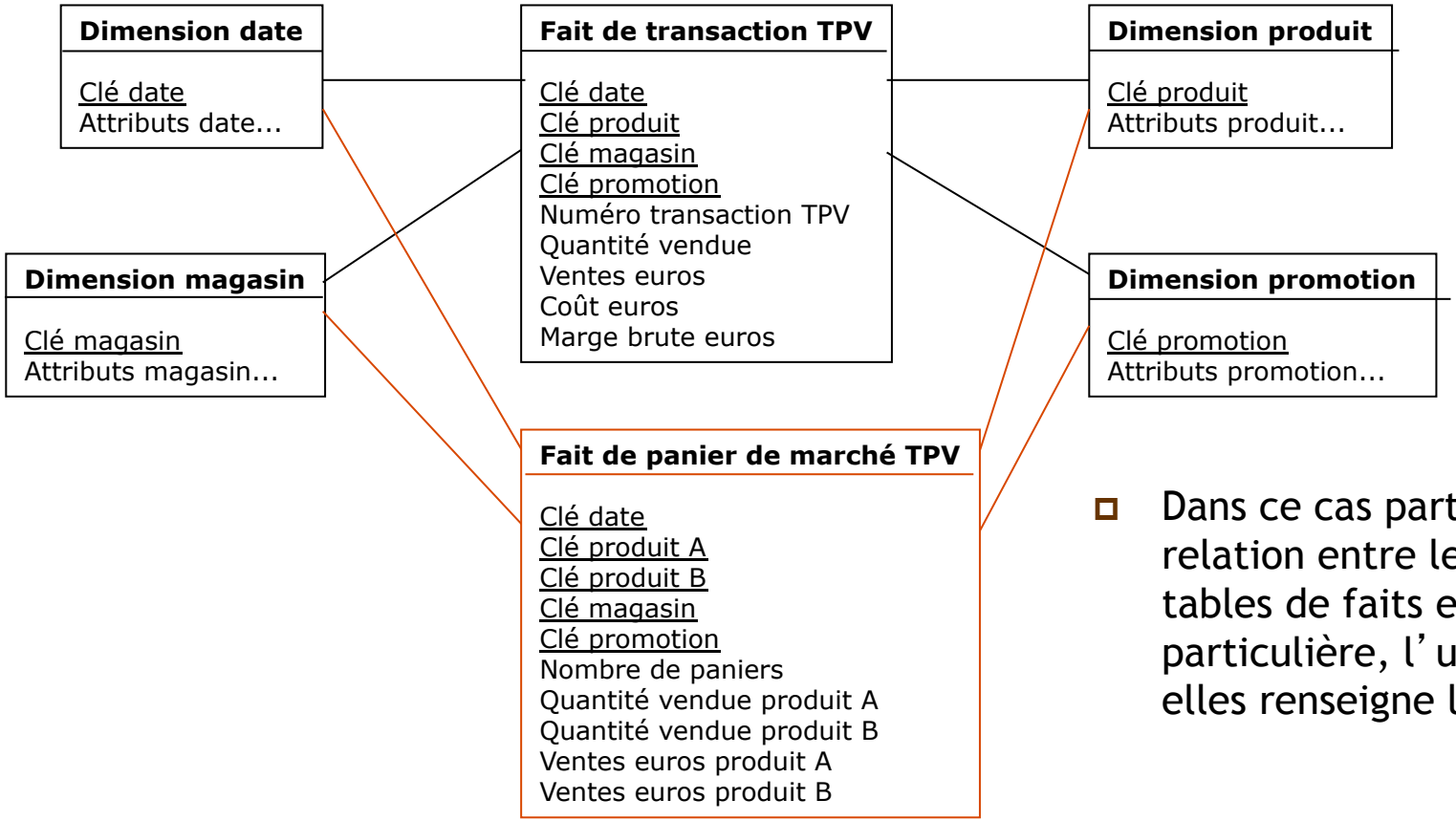
- La dimension Produit partiellement en flocon de neige



- A toute évidence l'analyse de l'entrepôt deviens plus compliquée

Le schéma dimension en Constellation de faits

- On peut construire une constellation de faits en rajoutant une table de faits qui permet d'analyser les couples de produits achetés ensemble (dans la même transaction TPV)



- Dans ce cas particulier, la relation entre les deux tables de faits est particulière, l'une d'entre elles renseigne l'autre

Taille de l'entrepôt de données (1)

- Dimensionner l'entrepôt
  - Choix des granularités
  - Choix d'une machine/SGBD cible (benchmark)

- Rappel

kilo-	k or K **	10^3	2^{10}
mega-	M	10^6	2^{20}
giga-	G	10^9	2^{30}
tera-	T	10^{12}	2^{40}
peta-	P	10^{15}	2^{50}
exa-	E	10^{18} *	2^{60}
zetta-	Z	10^{21} *	2^{70}
yotta-	Y	10^{24} *	2^{80}

Taille de l'entrepôt de données (2)

- Exemple : Supermarché
  - Dimensions
    - Temps : 4 ans * 365 jours = 1460 jours
    - Magasin : 300
    - Produit : 20,000 références GENCOD (10% vendus chaque jour)
    - Promotion : un article est dans une seule condition de promotion par jour et par magasin
  - Fait
    - 1460 * 300 * 20,000 * 1 = 8,76 milliards d'enregistrements
    - Nb de champs de clé = 4
    - Nb de champs de fait = 4
    - Table des Faits = 8,76.10^9 * 8 champs * 4 octets = 280 Go

Taille de l'entrepôt de données (3)

- Exemple : Suivi d'appels téléphoniques
  - Temps : 3 ans * 365 jours = 1095 jours
  - Nombre d'appel par jour = 100 000 000
  - Nb de champs de clé = 5
  - Nb de champs de fait = 3
  - Table des Faits = 1095.10^8 * 8 champs * 4 octets = 3,49 To

- Exemple : Suivi d'achats par carte de crédit
  - Temps : 3 ans * 12 mois = 36 mois
  - Nombre de compte carte = 50 000 000
  - Nombre moyen d'achat par mois par carte = 50
  - Nb de champs de clé = 5
  - Nb de champs de fait = 3
  - Table des Faits = 54.10^9 * 8 champs * 4 octets = 1,73 To

Interrogation OLAP

Extensions SQL

- SQL
  - Projection -> Slice
  - Sélection -> Dice
  - Agrégation -> Roll-up

- Ajouts OLAP par rapport à SQL-92
  - Extension de GROUP BY
    - CUBE
    - ROLLUP
    - GROUPING
  - Windowing : cumuls ou moyenne glissante
  - Ranking : position d'une ligne dans une partition

  - Supporté notamment par DB2 et Oracle (à partir de 8i)

OLAP vs SQL (1)

- Fréquemment, une opération OLAP conduit à plusieurs requêtes étroitement liées
- Opérations
  - Agrégation
  - Groupage
- Exemple
  - Donner les ventes par année et région avec en plus les totaux sur chaque année et chaque région
  - Ceci est appelé tabulation croisée

	44	49	Total
2000	100	150	250
2001	80	160	240
2002	90	170	260
Total	270	480	750

OLAP vs SQL (2)

Cet exemple a besoin des requêtes suivantes

- Pour le résultat entouré en gras

```
SELECT dt.annee, dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and
    f.cleMagasin=dm.cleMagasin
GROUP BY dt.annee, dm.dep
```

- Pour la colonne à droite

```
SELECT dt.annee, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt
WHERE f.cleDate=dt.cleDate
GROUP BY dt.annee
```

- Pour la dernière ligne

```
SELECT dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimMagasin dm
WHERE f.cleMagasin=dm.cleMagasin
GROUP BY dm.dep
```

- Pour la cellule de la dernière colonne et dernière ligne

```
SELECT SUM(f.ventes)
FROM faitsTPV f, dimMagasin dm
WHERE f.cleMagasin=dm.cleMagasin
```

- Cela représente un roll-up sur chaque dimension
- D dimensions = 2^d requêtes

GROUP BY avec CUBE

- SQL:1999 étends la fonction GROUP BY afin de fournir un meilleur support des fonctions roll-up d'OLAP et des tabulations croisées

- Il crée 2^d niveaux d'agrégats, d étant le nombre de colonnes de groupage

- Exemple

```
SELECT dt.annee, dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and f.cleMagasin=dm.cleMagasin
GROUP BY CUBE (dt.annee, dm.dep)
```

- Cette requête donne comme résultat la représentation tabulaire de la tabulation croisée correspondante (4 niveaux d'agrégats)

dt.annee	dm.dep	SUM(f.ventes)
2000	44	100
2000	49	150
2000	*Null*	250
2001	44	80
2001	49	160
2001	*Null*	240
2002	44	90
2002	49	170
2002	*Null*	260
Null	44	270
Null	49	480
Null	*Null*	750

GROUP BY avec ROLLUP

- SQL:1999 propose également un GROUP BY ROLLUP avec l'objectif de fournir un sous ensemble du résultat du GROUP BY CUBE

- Il crée d+1 niveaux d'agrégats, d étant le nombre de colonnes de groupage

- Exemple

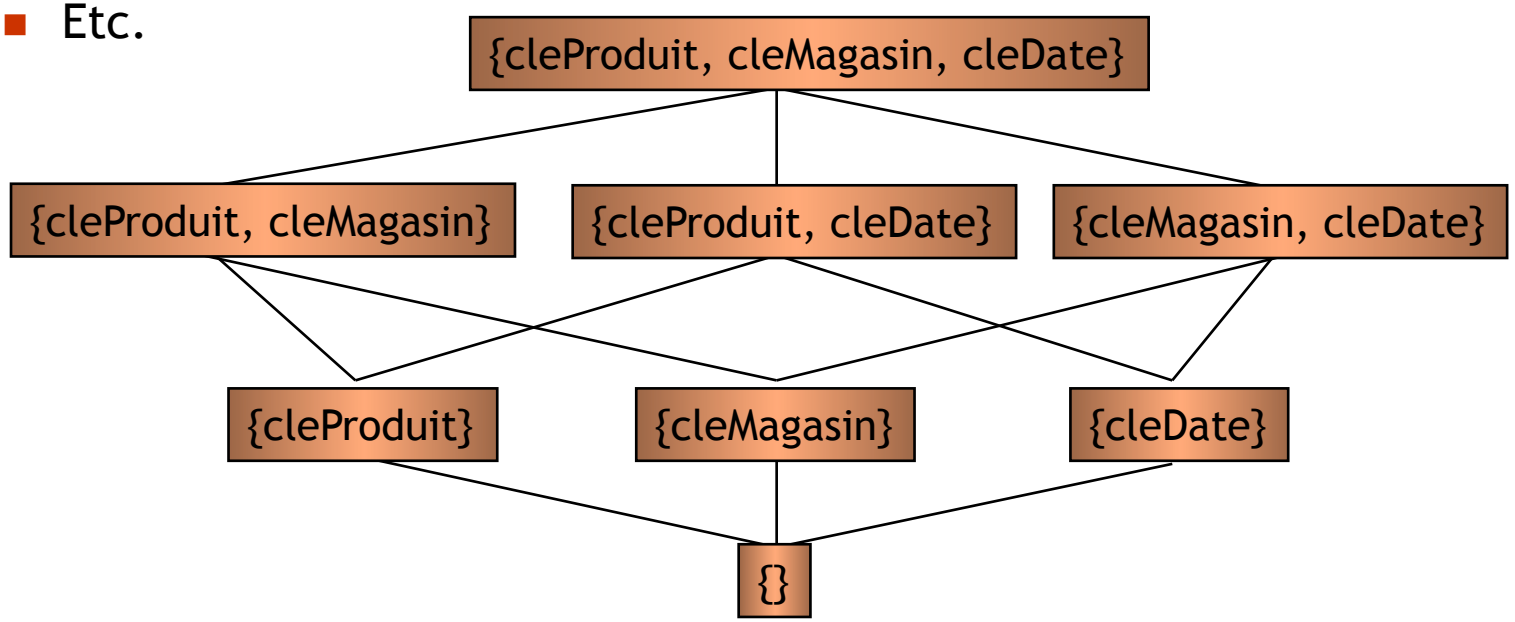
```
SELECT dt.annee, dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and f.cleMagasin=dm.cleMagasin
GROUP BY ROLLUP (dt.annee, dm.dep)
```

- Cette requête donne comme résultat la représentation tabulaire (2+1 nivaux d'agrégats)

dt.annee	dm.dep	SUM(f.ventes)
2000	44	100
2000	49	150
2000	*Null*	250
2001	44	80
2001	49	160
2001	*Null*	240
2002	44	90
2002	49	170
2002	*Null*	260
Null	*Null*	750

GROUPING SETS (1)

- Les fonctions ROLLUP et CUBE peuvent produire une grande quantité de tuples
- GROUPING SETS permet de sélectionner de combinaisons de groupage
- Les combinaisons de toutes les dimensions (CUBE) = (2^d)
  - 2 dimension = 4 combinaisons
  - 3 dimensions = 8 combinaisons
  - 4 dimension = 16 combinaisons
  - Etc.



GROUPING SETS (2)

- Exemple 1

```
SELECT dt.annee, dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and f.cleMagasin=dm.cleMagasin
GROUP BY GROUPING SETS((dt.annee), (dm.dep))
```

- Exemple 2

```
SELECT dt.annee, dm.dep, SUM(f.ventes)
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and f.cleMagasin=dm.cleMagasin
GROUP BY GROUPING SETS ((dt.annee, dm.dep),(dt.annee),())
```

Dans cet exemple le résultat est le même qu'en utilisant :

```
GROUP BY ROLLUP (dt.annee, dm.dep)
```

Exemple 1

dt.annee	dm.dep	SUM(f.ventes)
2000	*Null*	250
2001	*Null*	240
2002	*Null*	260
Null	44	270
Null	49	480

Exemple 2

dt.annee	dm.dep	SUM(f.ventes)
2000	44	100
2000	49	150
2000	*Null*	250
2001	44	80
2001	49	160
2001	*Null*	240
2002	44	90
2002	49	170
2002	*Null*	260
Null	*Null*	750

GROUPING (1)

- Cette fonction aide à faire la distinction entre les lignes agrégées et les lignes normales
- Facilite la lecture des sorties (valeurs *nulles*) des fonctions CUBE et ROLLUP
- La fonction rajoute la valeur 1 lorsqu' il s' agit d' un agrégat ou sous total, autrement, la valeur est 0

GROUPING (2)

- Exemple

```
SELECT dt.annee, dm.dep, SUM(f.ventes),
       GROUPING(dt.annee) as year,
       GROUPING(dm.dep) as dep
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and
      f.cleMagasin=dm.cleMagasin
GROUP BY ROLLUP (dt.annee, dm.dep)
```

dt.annee	dm.dep	SUM(f.ventes)	year	dep
2000	44	100	0	0
2000	49	150	0	0
2000	*Null*	250	0	1
2001	44	80	0	0
2001	49	160	0	0
2001	*Null*	240	0	1
2002	44	90	0	0
2002	49	170	0	0
2002	*Null*	260	0	1
Null	*Null*	750	1	1

GROUPING_ID

- Cette fonction concatène les colonnes du GROUPING et rend une valeur binaire
  00 -> 0
  01 -> 1
  10 -> 2
  11 -> 3

- Exemple

```
SELECT dt.annee, dm.dep, SUM(f.ventes),
    GROUPING_ID(dt.annee,dm.dep) as grp
FROM faitsTPV f, dimTemps dt, dimMagasin dm
WHERE f.cleDate=dt.cleDate and
    f.cleMagasin=dm.cleMagasin
GROUP BY ROLLUP (dt.annee, dm.dep)
```

dt.annee	dm.dep	SUM(f.ventes)	grp
2000	44	100	0
2000	49	150	0
2000	*Null*	250	1
2001	44	80	0
2001	49	160	0
2001	*Null*	240	1
2002	44	90	0
2002	49	170	0
2002	*Null*	260	1
Null	*Null*	750	3

WINDOW (1)

- Également appelé fenêtre mobile
- Permet de partitionner une table de manière horizontale
- Identifie un ensemble de lignes (fenêtre) « autour » d'un tuple dans une relation
- Exemple

```
select t.month, sum(sales),
       sum(sum(sales))
       over (order by t.month
       rows unbounded preceding)
from wsales s, times t
where t.timeid=s.timeid
group by t.month;
```

t.month	Sum(sales)	Cumul
1	8	8
2	40	48
3	70	118
4	30	148
5	18	166

- Les clauses FROM et WHERE sont exécutées comme d'habitude : génération d'une table intermédiaire nommée Temp. Les fenêtres sont créées sur cette relation Temp

WINDOW (2)

- Trois pas pour définir une fenêtre
  - Définir la fonction d'agrégation et la déclaration de la fenêtre
    - Dans l'exemple SUM(SUM(sales))
    - OVER définit une fenêtre
  - Spécifier l'ordonnancement des lignes dans la fenêtre avec ORDER BY
    - Dans l'exemple on ordonne chaque partition par T.month
  - Définir les limites de la fenêtre
    - Définir les limites de la fenêtre associées à chaque ligne
    - Dans l'exemple la fenêtre d'une ligne inclut la ligne en question + les lignes précédentes

WINDOW (3)

- La définition de limites de la fenêtre peut être faite de deux manières
  - Directement
    - L'ordonnancement est utilisé directement en spécifiant combien de lignes avant et/ou après doivent se trouver dans la fenêtre

    Quelques exemples :

    SUM(SUM(sales)) OVER (ORDER BY t.month ROWS UNBOUNDED PRECEDING)

    AVG(SUM(sales)) OVER (ORDER BY t.month ROWS 2 FOLLOWING)

    AVG(SUM(sales)) OVER (ORDER BY t.month ROWS BETWEEN 2 PRECEDING AND 2 FOLLOWING)

  - Avec la clause RANGE
    - Exemple

    AVG(SUM(sales)) OVER (ORDER BY t.month RANGE BETWEEN 2 PRECEDING AND 2 FOLLOWING)

RANK

- RANK
  - Rend la position d'une ligne dans sa partition
  - Exemple : la première ligne a le rang de 1 la deuxième de 2, etc. La valeur des rangs dépend de la valeur de l'attribut ordonné

    Janvier -> 1 ; février -> 2; février -> 2, mars-> 4
- Exemple

```
select pid,sum(sales),
        rank() over (order by sum(sales) desc)
from wsales
group by pid;
```

Pid	sum(sales)	Rank
12	191	1
11	115	2
13	93	3

- DENSE_RANK
  - Comme RANK sauf qu'il n'y a pas d'intervalles vides

    Janvier -> 1 ; février -> 2; février -> 2, mars-> 3

PARTITION BY

- Cette fonction permet de diviser le résultat d'une requête en partitions
- La différence entre une partition et un GROUP BY c'est que dans la partition on a un ensemble de lignes par partition et dans le GROUP BY on a uniquement une par group

- Exemple 1

```
select p.category, p.pid, sum(sales),
      dense_rank() over (partition by p.category
            order by sum(sales) desc) rank
from wsales s, products p
where s.pid=p.pid
group by p.category,p.pid,p.price;
```

Top n (1)

- Dans les entrepôts de données, une requête peut avoir comme réponse un nombre important de lignes
- Comment faire lorsqu'on a besoin de savoir par exemple « quelles sont les top 10 produits vendus l'année dernière ? »

- SQL Server

```
SELECT TOP 10 product, descr, email
FROM products
```

- ORACLE

```
SELECT product, descr, email
FROM products
WHERE ROWNUM <= 10
```

- MySQL:

```
SELECT product, descr, email
FROM products
LIMIT 10
```

Cette fonctionnalité ne fait pas partie de SQL:1992 ou SQL:1999 !!

Top n (2)

- Même si cette opération doit faire le calcul pour toutes les valeurs possibles dans le résultat, elle permet de faite une optimisation, par exemple, dans l'ordonnancement des lignes

- Une manière d'optimiser cette opération est de demander le calcul uniquement des produits qui risquent de se trouver dans les top n

NTILE(n)

- Cette fonction
  - Permet de calculer de tiers, quarts, cinquièmes, etc.
  - Découpe une partition ordonnée en un nombre n de groups et donne un nombre de group à chaque tuple de la partition

- Exemple 1

```
select pid, sales,
        NTILE(4) over(order by sales desc)
from wsales;
```

Pid	Sales	Ntile(4)
12	50	1
12	45	1
13	40	1
11	35	1
12	30	1
12	26	2
11	25	2
11	22	2
12	20	2
13	20	2
12	20	3
11	15	3
13	10	3
…	…	…

ROW_NUMBER

- Cette fonction désigne un numéro unique à chaque ligne dans une partition
- La numérotation est séquentielle et commence par 1
- Exemple

```
select pid, sales,
    row_number() over (order by sales) row_number
from wsales;
```

Plus de fonctions

- Dans Oracle10g
  - Ranking Functions
    - CUME_DIST
    - PERCENT_RANK
  - Windowing Aggregate Functions
    - FIRST_VALUE and LAST_VALUE
    - RATIO_TO_REPORT
    - LAG/LEAD
    - FIRST/LAST
  - Linear Regression Functions
    - REGR_COUNT
    - REGR_AVGY and REGR_AVGX
    - REGR_SLOPE and REGR_INTERCEPT
    - REGR_R2
    - REGR_SXX, REGR_SYY, and REGR_SXY
  - Etc.

http://download-uk.oracle.com/docs/cd/B14117_01/server.101/b10736/part5.htm

Vues matérialisées

Définition de nouvelles relations en utilisant des expressions SQL

Précalcul des agrégats

- Afin de réduire le temps de réponse des opérations, il est possible de pré calculer les agrégats

- 2 possibilités
  - Stocker tous les agrégats possibles
    - Coûteux en espace
  - Ne stocker qu'une partie des agrégats -> vues matérialisées
    - Difficulté : quels agrégats ?

Vues

- Afin de réduire le temps de réponse des requêtes sur de grandes quantités de données utilisation du pré calcul de vues
- Par exemple, la fonction CUBE pourrait répondre rapidement si préalablement un sous ensemble des agrégats est calculé à l'aide de vues
- Problèmes
  - Comment stocker sur support permanent les vues ?
  - Comment maintenir à jour les vues ?

Rappel

- Une vue est un ensemble de tuples déduit d'une base de données, par composition de relations de la base
- Table virtuelle calculée par une requête
- Création d'une vue

```
CREATE VUE RegionalSales (category, sales, state)
AS SELECT (P.category, S.sales, L.state
FROM Products P, Sales S, Locations L
WHERE P.pid=S.pid AND S.locid=L.locid
```

- Interrogation d'une vue

```
SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales R
GROUP BY R.category, R.state
```

Utiliser les vues

- Comment profiter de l'utilité des vues ?
- Exemple, considérons deux requêtes

```
SELECT P.categorie, SUM(S.sales)
FROM Products P, Sales S
WHERE P.pid=S.pid
GROUP BY P.category
```

```
SELECT L.state, SUM(S.sales)
FROM Locations L, Sales S
WHERE L.locid=S.locid
GROUP BY L.state
```

- La relation Sales peut être très grande, ce qui ralentira les deux requêtes
- Solution : définir une vue sur Sales

```
CREATE VUE TotalSales (pid, locid, total)
AS SELECT S.pid, S.locid, SUM(S.sales)
FROM Sales S
GROUP BY S.pid, S.locid
```

- Cette vue peut alors être utilisée dans les deux requêtes

```
SELECT P.categorie, SUM(T.total)
FROM Products P, TotalSales T
WHERE P.pid=T.pid
GROUP BY P.category
```

```
SELECT L.state, SUM(T.total)
FROM Locations L, TotalSales T
WHERE L.locid=T.locid
GROUP BY L.state
```

Vues matérialisées

- Vues matérialisées (snapshots, vues concrètes) -> copies calculées à partir d'une requête et stockées sur disque
- Elles peuvent être modifiables ou uniquement en lecture
- Dans les entrepôts de données, les vues matérialisées sont souvent basées sur des agrégats ou des jointures
- Exemple

```
CREATE MATERIALIZED VIEW vueSales
AS SELECT S.pid, S.locid, SUM(S.sales)
    FROM Sales S
    GROUP BY S.pid, S.locid
```

Problématique

- Quelles vues matérialiser
  - Question importante et abordée partiellement par quelques SGBD
- Indexation des vues
  - L'indexation d'attributs dans les vues est traitée de la même manière que dans les relations
- Maintenance des vues et indexes (mise à jour)
  - Cela dépend de la manière dont la vue a été créée (sélection, projection, jointures, etc.)
  - Une relation de base peut être utilisée dans la création de plusieurs vues, cela complexifie le processus de rafraîchissement
  - Pas de consensus, encore beaucoup de travail à faire (recherche et industrie)

Maintenir les vues matérialisées

- Lorsque les relations de base sont modifiées
  - INSERT, DELETE, UPDATE

- From scratch
  - Recalculer les vues périodiquement ou à chaque fois qu' il y a une modification
    - Utile lorsque les tables sont dans une BD lointaine
    - Avantage -> les tables ne sont pas dupliquées dans l' entrepôt

- Incrémentale
  - Utilisation d' algorithmes de rafraîchissement
    - Utile lorsque le coût de rafraîchissement est inférieur au coût du recalcule des vues

Mise à jour incrémentale (1)

- La mise à jour s'effectue si possible en différentiel (pas toujours possible)
- Différents algorithmes selon la nature de la vue (sélection, projection, agrégat, etc.)
- Dans la suite on analyse des algorithmes de maintenance pour les vues générées à partir d'opérations de projection, jointure binaire, agrégats
- L'approche présentée peut être étendue à d'autres opérations telles la sélection, l'union, l'intersection, la différence ainsi que des expressions contenant plusieurs opérateurs

Mise à jour incrémentale (2)

- Vues à partir de projections
  - V est une vue faite à partir d'une projection sur la relation R $V=\pi(R)$
    chaque tuple v dans V a un compteur c associé (le nombre de tuples de R qui ont dérivé v)
  - Insertion d'un ensemble de tuples Ri dans R
    - calculer π(Ri) et le rajouter à V
    - si π(Ri) contient un tuple r avec un compteur c et r n'existe pas dans V alors on rajoute r et c à V
  - Suppression d'un ensemble de tuples Rd dans R
    - calculer π(Rd) et le supprimer de V
    - si π(Rd) contient un tuple r avec un compteur c, r doit également exister dans V avec un compteur supérieur à c ; soustraire r du compteur c de V
  - La mise à jour peut être considérée comme une insertion suivie d'une suppression

Mise à jour incrémentale (3)

- Vues à partir de jointures binaires
  - V est une vue faite à partir de la jointure de deux tables, R X S
  - Insertion d'un ensemble de tuples Ri
    - Calculer Ri X S
    - Rajouter le résultat à V
  - Suppression d'un ensemble de tuples Rd
    - Calculer Rd X S
    - Supprimer le résultat de V. Noter que si r existe en Rd X S avec comptage c, il doit également exister dans V avec un comptage supérieur

Mise à jour incrémentale (4)

- Vues à partir d'agrégats
  - V est une vue faite à partir R en utilisant GROUP BY sur la colonne G et une opération d'agrégation sur la colonne A
  - Chaque tuple v dans V résume un group de tuples dans R <g, résumé> où
    - g est la valeur de la colonne de groupage G et
    - résumé dépends de la fonction d'agrégation
  - Afin de maintenir la vue, il est nécessaire de garder plus d'information, cela dépend de l'opération d'agrégation
  - COUNT -> garder un comptage c pour chaque v dans V

    v <g, c>
    - Insertion d'un tuple r dans R
      - S'il n'y a pas de tuple v dans V où v.G=r.G alors on rajoute une nouvelle ligne dans V <r.G, 1>
      - S'il y a un tuple v dans V où v.G=r.G alors on incrément le comptage c de v
    - Suppression d'un tuple r dans R
      - Le tuple v dans V où v.G=r.G est modifié en décrémentant le comptage c de v

      v peut être supprimé si le comptage devient 0

Mise à jour incrémentale (5)

- SUM -> garder en plus une addition s et un comptage c pour chaque v

  v <g, s, c>

  - Insertion d'un tuple r dans R
    - S'il n'y a pas de tuple v dans V où v.G=r.G alors on rajoute une nouvelle ligne dans V ; <r.G, a, 1>
    - S'il y a un tuple v dans V où v.G=r.G alors on la remplace par <r.G, s+a, c+1>
  - Suppression d'un tuple r dans R
    - Le tuple v <r.G, s, c> dans V où v.G=r.G est remplacé par <r.G, s-a, c-1>

    v peut être supprimé si le comptage devient 0

- AVG -> garder en plus une addition s, un comptage c et la moyenne mo pour chaque v

  v <g, s, c, mo>

  - L'addition et le comptage sont maintenus comme pour l'opération SUM et la moyenne est calculée par s/c

Mise à jour incrémentale (6)

- MIN -> garder en plus la valeur minimale m pour la colonne A dans le group g et un comptage c pour chaque v où r.G = g and r.A = m (le nombre de tuples dans R qui ont la valeur minimale)

  v <g, m, c>

  - Insertion d' un tuple r dans R
    - Si r.G=g et r.A > m pour le groupe g alors r est ignoré
    - Si r.G=g et r.A = m pour le groupe g alors le tuple v est replacé par <g, m, c+1>
    - Si r.G=g et r.A < m pour le groupe g alors le tuple v est replacé par <g, r.A, 1>
  - Suppression d' un tuple r dans R
    - Si r.G=g et r.A = m pour le groupe g alors

      si c > 0 v est remplacé par <g, m, c-1>

      si c = 0 cela signifie que le dernier tuple avec la valeur minimale A a été supprimée de R et qu' l est nécessaire de récupérer la valeur minimale A de du groupe r.G de R. Cela revient à récupérer tous les tuples de R du groupe r.G

Quand synchroniser les vues ? (1)

- Politique de maintenance de vue -> décision sur le moment de rafraîchir les vues (indépendant du type de rafraîchissement)
  - Maintenance immédiate -> avec la transaction qui modifie les relations de base
    - Inconvénient -> la transaction de mise à jour est ralentie par le processus de rafraîchissement, l'impact augment avec le nombre de vues qui dépendent de la table modifiée
  - Différé -> les modifications peuvent être capturées dans un journal (log) et appliquées ultérieurement

Quand synchroniser les vues ? (2)

- Il existe plusieurs politiques de maintenance de vues différées
  - Lazy
    - La vue est rafraîchie lors d'une requête la concernant si elle n'est pas cohérente
    - Cette approche ralentie les requêtes et pas les mise à jour
  - Périodique
    - La vue est rafraîchie périodiquement, par exemple, une fois par jour
    - Avec cette approche les vues ne sont pas cohérentes en permanence
  - Forcée
    - La vue est rafraîchie après qu'un certain nombre de mise à jour sont faites sur les tables de base
    - Comme dans l'approche périodique, les vues ne sont pas cohérentes en permanence

Ré écriture de requêtes (1)

- Les vues matérialisées peuvent être utilisées de manière transparente pour l'utilisateur
- Les vues sont utilisées de manière analogue aux indexes à la différence que les vues peuvent être accédées directement avec un SELECT
- Amélioration automatique des performances (*tuning*)
  - L'optimisateur de requêtes vérifie si des vues matérialisées existantes peuvent être utilisées dans une requête
  - La requête est ré écrite automatiquement

Ré écriture de requêtes (2)

- Utilisation sur un calcul d' agrégats
- On considère la vue

```
CREATE MATERIALIZED VIEW monthly_sales_mv
ENABLE QUERY REWRITE
AS
SELECT t.month, p.product_id, SUM(ps.purchase_price) as sum_of_sales,
  COUNT (ps.purchase_price) as total_sales
FROM time t, product p, purchases ps
WHERE t.time_key = ps.time_key AND
  ps.product_id = p.product_id
GROUP BY t.month, p.product_id;
```

- Cette vue peut être utilisée dans la requête

```
SELECT t.month, p.product_id, AVG(ps.purchase_price) as avg_sales
FROM time t, product p, purchases ps
WHERE t.time_key = ps.time_key AND
  ps.product_id = p.product_id
GROUP BY t.month, p.product_id;
```

Ré écriture de la requête en utilisant la vue :

```
SELECT t.month, p.product_id,
sum_of_sales/total_sales as avg_sales
FROM monthly_sales_mv;
```

Bibliographie

- E. F. Codd. Providing OLAP to user-analysts: An IT mandate. E.F. Codd and Associates, 1993.
- http://www.olapreport.com/fasmi.htm
- Raghu Ramakrishnan and Johannes Gehrke. Database Management Systems. McGraw-Hill International Edition, 2003.
- Ralph Kimball, Margy Ross. Entrepôts de données, Guide pratique de modélisation dimensionnelle. Vuibert Edition, 2003
- Oracle Database Data Warehousing Guide
  http://download-uk.oracle.com/docs/cd/B14117_01/server.101/b10736/toc.htm
- Documentation Oracle
  http://www.oracle.com/technology/documentation/index.html
- Oracle Database SQL Reference 10g Release 2 (10.2)
  html : http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14200/toc.htm
  pdf : http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14200.pdf

En particulier regarder « Part V Data Warehouse Performance »
http://download-uk.oracle.com/docs/cd/B14117_01/server.101/b10736/part5.htm

Les data trucs

Data warehouse	⇨ Entrepôt de données
Decision Support Systems	⇨ Systèmes d' aide à la décision
Business Intelligence	⇨ Intelligence Économique
OLAP On-Line Analytical Processing	⇨ Analyse en ligne de données
Knowledge Discovery	⇨ Extraction de Connaissances
in Databases	⇨ Dans les Données
Data Mining	⇨ Fouille de données
Customer Relationship Management	⇨ Gestion de la Relation Client

Les data trucs

- OLTP On-line transaction processing ⇨ Gestion transactionnelle en ligne
- Datamart ⇨ Magasin de données