Master d'Informatique spécialité DAC

BDLE (Bases de Données Large Echelle)

UE 5I852

Bases de données Multidimensionnelles

2017-2018

Anne Doucet

www-bd.lip6.fr

Master d'Informatique spécialité DAC

BDLE (Bases de Données Large Echelle)
UE 5I852

Bases de données Multidimensionnelles
2017-2018

Anne Doucet
www-bd.lip6.fr

Plan

- Aide à la décision
  - Définition
  - Information vs données
  - Entrepôts de données
  - Requêtes décisionnelles
- Données multidimensionnelles
  - Concepts et opérateurs
  - Représentation du cube
  - Optimisation
- Extensions de SQL pour les requêtes décisionnelles
  - Agrégation, rollup, cube, grouping sets
  - Fonctions (classement, statistiques, fenêtrage..)

Prise de décision

- Répondre aux demandes d'analyse de données
  - Qui sont les meilleurs clients ? Quelles sont les meilleures ventes par magasin ? Quelles sont les caractéristiques des clients achetant tel produit ? Analyser la baisse du chiffre d'affaire, etc.
- Dégager des informations qualitatives
- Les décideurs sont des **experts, analystes d'un métier**. Ils ne sont pas informaticiens, ni statisticiens.

Aide à la décision

La prise de décision nécessite une information
- précise
- fiable
- actualisée
- pertinente

L'information est à la base du cycle
- information - analyse - prise de décision

Information

L'information occupe un rôle croissant dans tous les métiers

- Qualité de service
  - traitement personnalisé des clients, offres compétitives
- Gestion
  - réduction des coûts, gestion des profits
- Prospective
  - analyse des comportements des clients, du marché
- Communication
  - informer les individus

Information vs données

- Données
  - montant total des ventes pour région Paris
  - vendeur ayant réalisé le meilleur chiffre ce mois
- Information
  - évolution des ventes pour région Paris au cours des 5 dernières années ?
  - sur quels produits faire des offres promotionnelles ?
  - quelle est la rentabilité d'une activité ?

Les requêtes décisionnelles permettent d'obtenir de l'information à partir des données.

Besoins des décideurs

- Avoir un accès rapide et simple à l'information stratégique
- Donner du sens aux données
- Avoir une vision transversale des données de l'entreprise (intégrer les différentes bases de données de l'entreprise)
- Extraire, grouper, organiser, agréger, résumer les données

Problématique

- Transformer des données de production en informations stratégiques
- Créer des entrepôts de données (systèmes d'information regroupant les données de l'entreprise pour les applications décisionnelles)

Processus de décision

- Définir le problème
- Rassembler les données
- Transformer les données
- Extraire les données
- Analyser les données

Systèmes décisionnels

- Proposer des solutions
- Décider

Besoins

- Un entrepôt de données doit permettre de
  - Synthétiser
  - visualiser
  - analyser

  de très grandes quantités de données de production, très détaillées, provenant de sources hétérogènes.

- Il doit pouvoir être utilisé par des non informaticiens

Gestion des données

- Systèmes « Online Transaction Processing » (OLTP)
  - comptabilité, achats, réservation, télécommunications, …
  - systèmes stratégiques, haute performance et disponibilité

- Multitude de systèmes spécialisés
  - fichiers Excel, bases personnelles, documents, …
  - systèmes autonomes, non stratégiques

Caractéristiques des systèmes OLTP

Priorités	**Performance, forte disponibilité**
Utilisation du processeur	**Prévisible**
Temps de réponse	**quelques secondes**
Modèle de données	**hiérarchiques, réseaux, relationnel, fichiers plats**
Contenu des données	**organisées par applications**
Nature des données	**Dynamiques, changent constamment état courant des affaires**
Traitement	**Très structuré, répétitif**
Utilisateurs	**employés, administrateurs**

Limites des systèmes OLTP

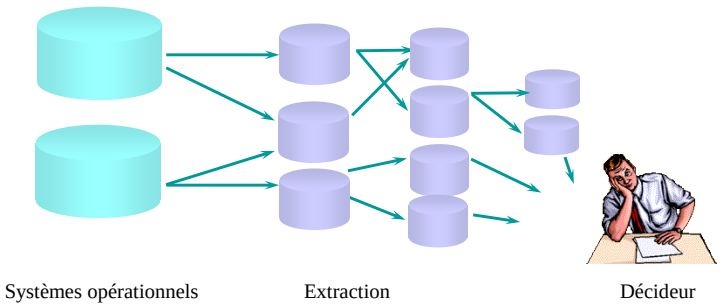
- Les systèmes OLTP sont mal adaptés à la gestion d'information pour l'aide à la décision
- Problèmes :
  - Analyse de données massives (giga, tera octets) stockées dans l'entrepôt pour l'aide à la décision.
  - Requêtes moins fréquentes mais plus complexes, longues, nécessitant une reformulation (agrégation) des données de masse.
  - Extractions de données non productives
  - Qualité des données incertaine

Accès aux données

- Données structurées pour applications
  - tables normalisées (performance transactionnelle)
  - valeurs d'attributs codées
  - attributs spécifiques pour la production
- Données dans des systèmes indépendants
  - systèmes hétérogènes (protocoles réseau, systèmes de gestion, modèles de données)
- Requêtes simples
  - incompatibilité (performance) avec requêtes décisionnelles

Extraction de données

- Extraire les données pour applications décisionnelles



- Problèmes
  - duplication d'effort dans les extractions multiples
  - versions incohérentes, obsolètes

Concept de schéma métier

- Vue idéale pour le décideur
- Décrit les objets métier
- Interrogation «naturelle» pour un spécialiste du métier
- Décision facilitée

- Conception du schéma métier à partir des besoins
- Deux options possibles : les données métier sont matérialisées ou non.

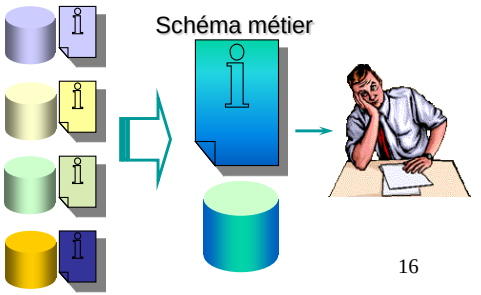
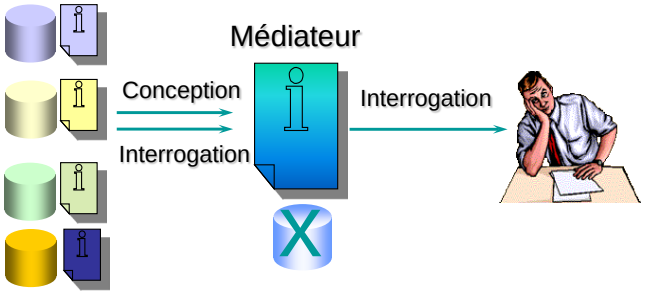


Schéma métier et médiation de données

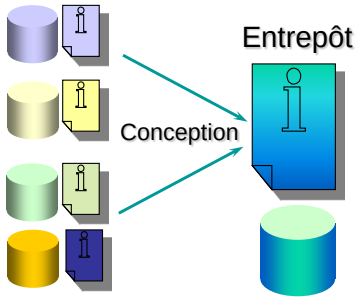
- Schéma de médiation
  définit les correspondances entre le schéma métier et les schémas des sources
- Le médiateur ne stocke pas les données des sources
- Interrogation du médiateur avec accès sous-jacent aux sources

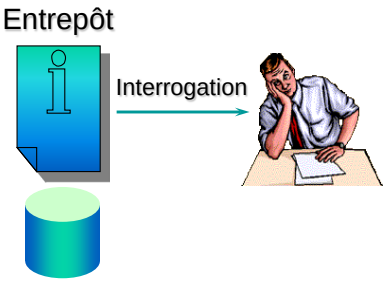


17

Schéma métier et entrepôt de données

- Schéma de l'entrepôt
  défini à partir des schémas des sources
- Matérialisation des données dans l'entrepôt
- Interrogation de l'entrepôt sans accès aux sources

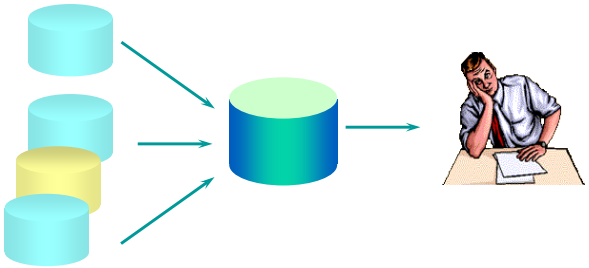




18

Concept d'entrepôt de données

- Définition (B. Inmon) : *un entrepôt de données est une collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour le support d'un processus d'aide à la décision.*



19

Données thématiques (orientées sujet)

Les données sont organisées par sujets métier et non par traitement

Exemples :

client (contrats assurance, prêts, comptes, plans d 'épargne, etc.)

produit (gamme, ventes, achats, coûts de production, etc.)

20

Données intégrées

- Toutes les données relatives à un sujet métier sont présentées de façon *pertinente*, *cohérente* et *non redondante*
- L'intégration s'effectue via des processus de transformation des données :
  - consolidation
  - agrégation
  - interprétation
- Ces processus doivent être documentés (via les méta-données)

Données datées

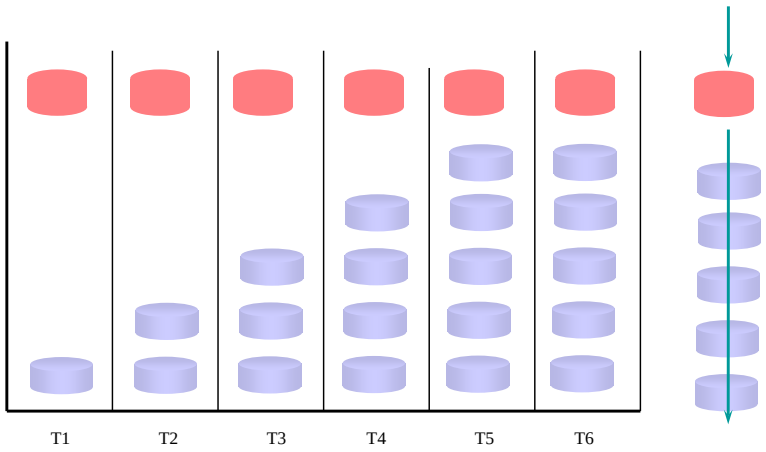
- Les données de l'entrepôt représentent des clichés successifs du monde réel.
  - granularité de temps
  - granularité de rafraîchissement
  - cohérence des clichés

Données historisées

- Les données résident dans l'entrepôt pour une grande période de temps.
- Ajout successif d'incréments de données
  - mises à jour ou suppressions rares
  - chargements successifs
  - archivage des données trop anciennes

Clichés et séries chronologiques

- Les systèmes opérationnels donnent des clichés successifs.
- Les entrepôts offrent une série chronologique.



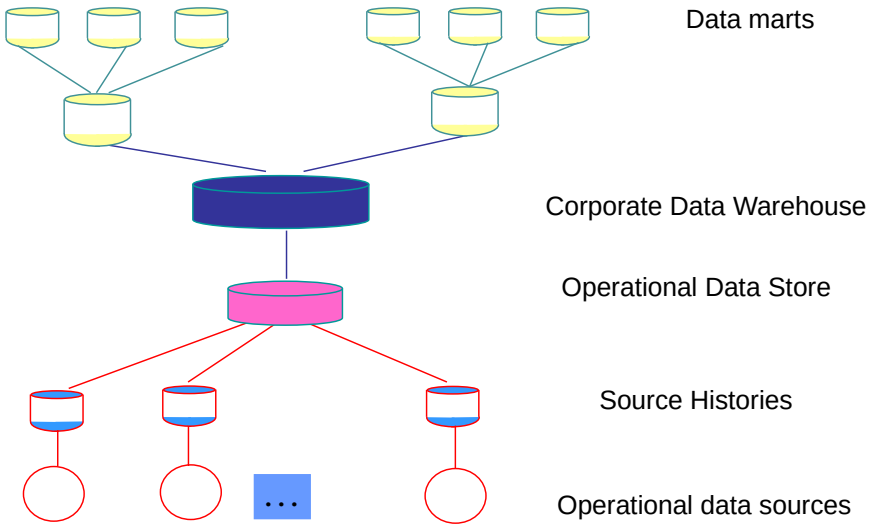
OLTP vs Entrepôt

Propriétés	Opérationnel	Entrepôt
Temps de réponse	en secondes	souvent en heures
Operations	lectures, écritures	Lectures seules
Nature des données	30-60 jours	historiques, de 2 à 10 ans
Organisation des données	Application	Sujet, temps
Taille	de petite à grande	de grande à très grande
Sources de données	Opérationnel, Interne	Opérationnel, Interne, Externe
Activités	Traitement	Analyse

Architecture des entrepôts

- Une architecture à base d'entrepôts met en jeu plusieurs couches de données « entreposées » entre les sources et les applications
  - Operational data store (ODS)
  - Corporate Data Warehouse (CDW)
  - Data marts
- Chaque niveau représente un ensemble de « vues » matérialisées du niveau précédent

Vue générale



Operational Data Store

- Niveau intermédiaire avant l'entrepôt
  - données intégrées, faiblement agrégées
  - support à l'analyse sur des données très actualisées
  - niveau d'entrée possible du nettoyage de données
- Les données sont des données dynamiques, de valeur courante, organisées par sujet et intégrées.
- Facile à utiliser.
- Correspond au support des applications « Infocentre »

Data Mart

- Données fortement agrégées, taillées sur mesure, historisées, organisées par sujet.
- Les données sont relationnelles ou multidimensionnelles

- Data mart indépendant
  - dérivé des sources
  - rapide à développer
- Data mart dépendant
  - dérivé de l'ODS ou du CDW
  - cohérence de l'information
  - transformation factorisée

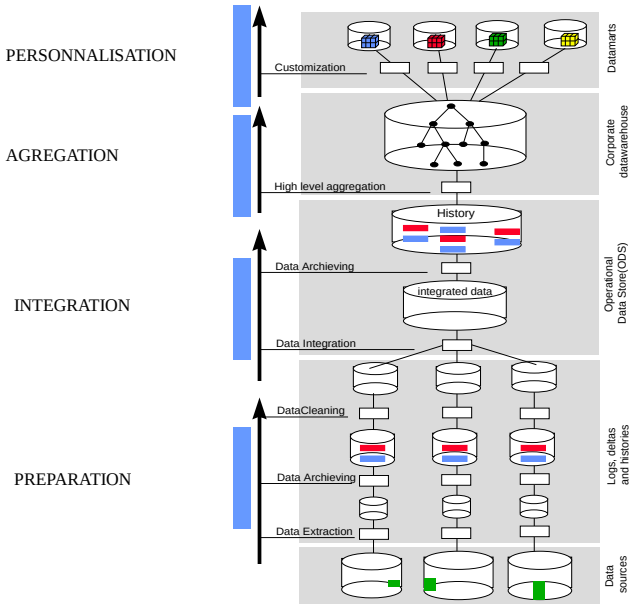
Processus de construction (1)

- Extraction, transformation
  - sélection des données extraites, transformation et formattage de sortie,
  - archivage éventuel
- Nettoyage (« cleaning ») et integration
  - analyse des données (statistiques)
  - dédoublement, élimination des erreurs, consolidation,
  - archivage éventuel

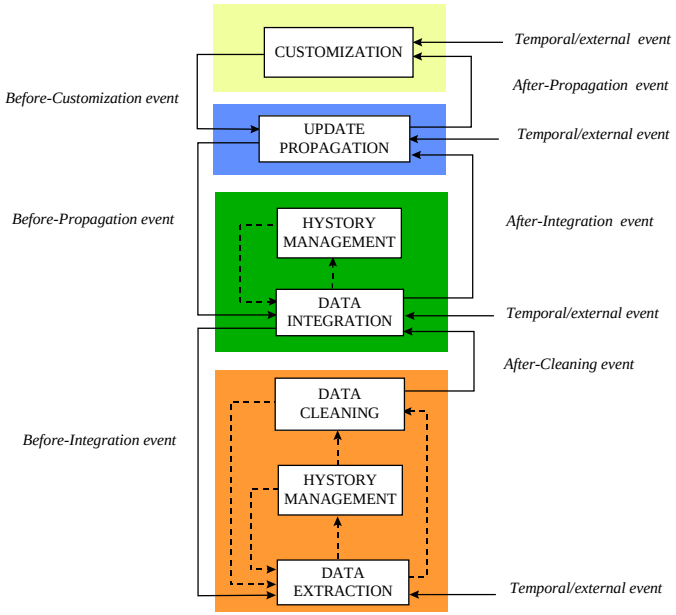
Processus de construction (2)

- Agrégation
  - agrégation des données et chargement dans l'entrepôt global (CDW)
- « Customisation »
  - agrégation des données
  - mise en forme spécifique pour applications OLAP
- Rafraîchissement
  - couvre l'ensemble du processus de construction
  - politiques dépendent des contraintes de qualité des données et des capacités des sources

Chargement des données



Rafraîchissement des données



Méta-données

- Essentielles pour la gestion des couches de données et des processus de construction
- Aide à l'administrateur et au concepteur
  - architecture complexe
  - gros volumes de données
  - nombreux processus évolutifs au cours du temps
- Types de méta-données
  - data dictionary : définitions des schémas des BD
  - rafraîchissement: structure et fréquence de l'alimentation
  - transformations : définition et flux
  - versions : contrôle des changements de méta-données
  - statistiques : profils des données entreposées
  - sécurité : conditions d'accès aux données
  - localisation physique des données

Plan

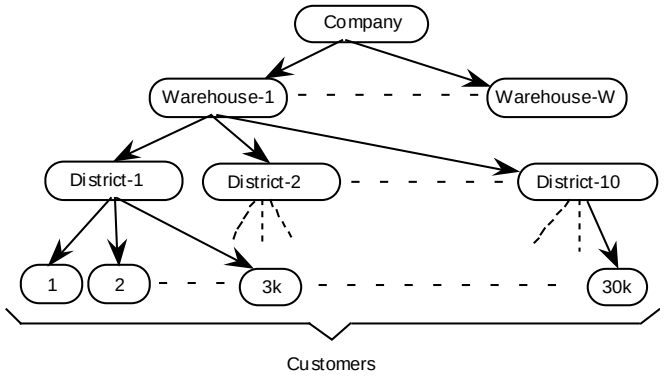
- Aide à la décision
  - Information vs données
  - Entrepôts de données
  - **Requêtes décisionnelles**
- Données multidimensionnelles
  - Concepts et opérateurs
  - Représentation du cube
  - Optimisation
- Extensions de SQL pour les requêtes décisionnelles
  - Agrégation, rollup, cube, grouping sets
  - Fonctions (classement, statistiques, fenêtrage..)

Objectifs du TPC-H

- Benchmark pour requêtes décisionnelles
  - Examiner de très grands volumes de données
  - Exécuter des requêtes complexes
  - Obtenir des réponses aux requêtes décisionnelles critiques
- Les requêtes sont longues, coûteuses, et portent sur de grosses quantités de données à trier, joindre, passer en revue, regrouper.
- La base est remise à jour périodiquement, sans bloquer le système (accès concurrents)

Exemple OLTP: base de données TPC-C

- TPC : Transaction Processing Performance Council
- Application: gestion, vente et distribution de produits ou services (www.tpc.org/bench.descrip.html)



Exemple OLTP : benchmark TPC-C

- Transactions OLTP:
  - *New-order*: enter a new order from a customer
  - *Payment*: update customer balance to reflect a payment
  - *Delivery*: deliver orders (done as a batch transaction)
  - *Order-status*: retrieve status of customer's most recent order
  - *Stock-level*: monitor warehouse inventory
- Les transactions agissent sur une BD de 9 relations.
- Les opérations des transactions sont *update*, *insert*, *delete*, et *abort*;
  Elles font des accès aux clés primaires et secondaires.

Schéma de la base

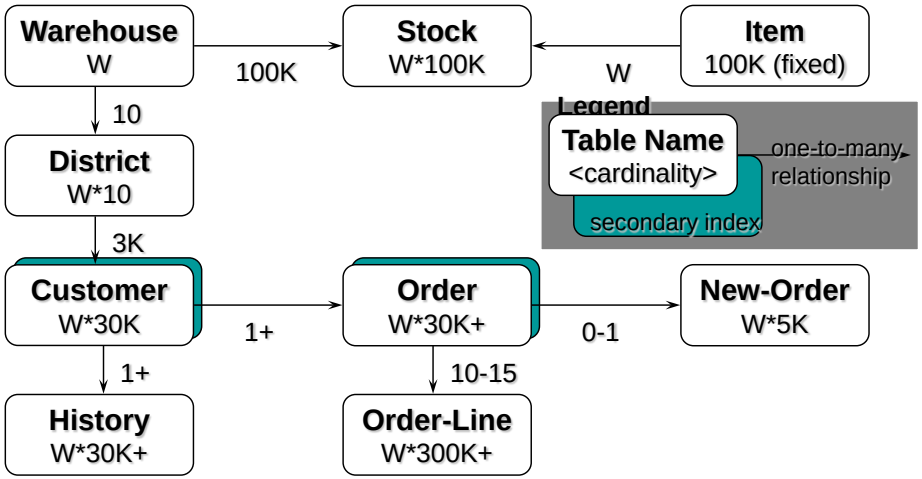
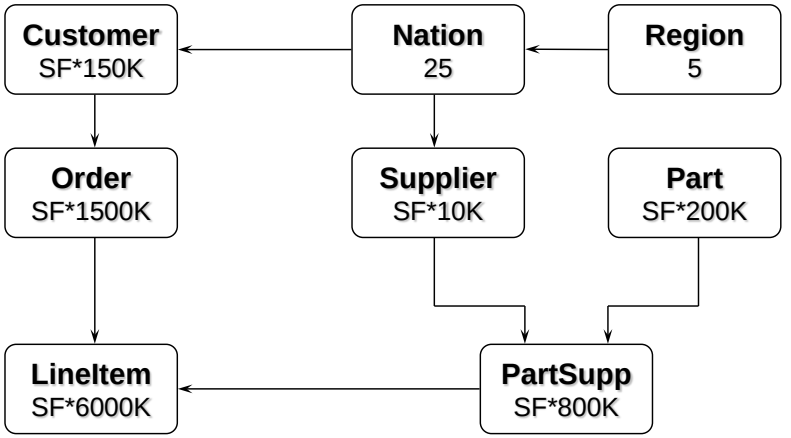


Schéma de la base TPC-H



Légende:
- Les flèches pointent dans la direction d'associations 1:N.
- La valeur sous les noms des tables indique la cardinalité. SF est le facteur d'échelle (Scale Factor).

Schéma TPC-H

Part (p_partkey, p_name, p_mfgr, p_brand, p_type, p_size, p_container, p_retailprice, p_comment)

Supplier (s_suppkey, s_name, s_address, s_nationkey, s_phone, s_acctbal, s_comment)

PartSupp (ps_partkey, ps_suppkey, ps_availqty, ps_supplycost, ps_comment)

Customer (custkey, name, address, nationkey, phone, acctbal, mktsegment, comment)

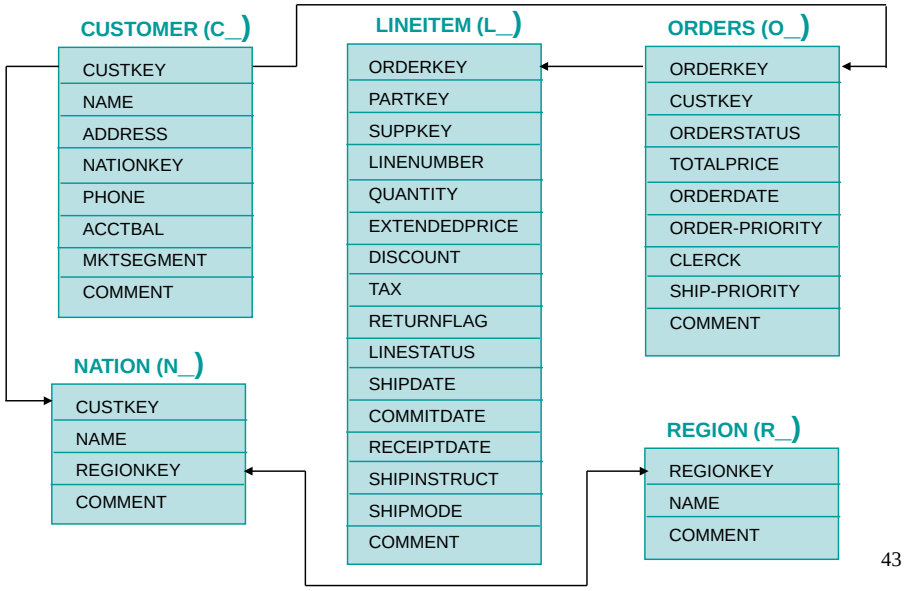
Orders (o_orderkey, o_custkey, o_orderstatus, o_totalprice, o_orderdate, o_orderpriority, o_clerk, o_shippriority, o_comment)

LineItem (l_orderkey, l_partkey, l_suppkey, l_linenumber, l_quantity, l_extendedprice, l_discount, l_tax, l_returnflag, l_linestatus, l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment)

Nation (n_nationkey, n_name, n_regionkey, n_comment)

Region (r_regionkey, r_name, r_comment)

Détails du schéma



Exemples de Requêtes TPC-H

- *Q2: Minimum cost supplier query*
  - This query finds which supplier should be selected to place an order for a given part in a given region
- *Q3 : Shipping priority query*
  - This query retrieves the 10 unshipped orders with the highest value.
- *Q4: Order priority checking query*
  - This query determines how well the order priority system is working and gives an assessment of customer satisfaction
- *Q8 : National market share query*
  - This query determines how a market share of a given nation within a given region has changed over two years for a given part type.
- *Q13 : Customer distribution query*
  - This query seeks relationships between customers and the size of their orders.

Définition des requêtes

- Chaque requête est définie par les composants suivants :
  - Business question (contexte)
  - Functional query definition (définit en SQL la fonction à effectuer)
  - Substitution parameters (décrit comment générer les valeurs nécessaires pour compléter la syntaxe de la requête)
  - Query validation (décrit comment valider la requête)

Exemple Q4 : Order priority checking query

This query determines how well the order priority system is working and gives an assessment of customer satisfaction

Business question :

The Order Priority Checking Query counts the number of orders ordered in a given quarter of a given year in which at least one lineitem was received by the customer later than its committed date. The query lists the count of such orders for each order priority sorted in ascending priority order.

Functional query definition :

```
Select o_orderpriority,
count(*) as order_count
from orders
where
   o_orderdate >= date '[DATE]'
   and o_orderdate < date '[DATE]' + interval '3' month
   and exists ( select * from lineitem
        where l_orderkey = o_orderkey and l_commitdate <
   l_receiptdate)
group by o_orderpriority
order by o_orderpriority;
```


Exemple Q4 : Order priority checking query

Substitution parameters :

DATE is the first day of a randomly selected month between the first month of 1993 and the 10th month of 1997.

Query validation :

Pour être validée, la requête doit utiliser cette valeur de substitution

DATE = 1993-07-01

Et doit avoir comme résultat :

O_ORDERPRIORITY	ORDER_COUNT
1-URGENT	10594
2-HIGH	10476
3-MEDIUM	10410
4-NOT SPECIFIED	10556
5-LOW	10487

Modélisation des données multidimensionnelles

Données relationnelles

Table des ventes :

Produit	Région	Chiffre
Soda	Ouest	300
Vins	Est	250
Entretien	Centre	150
Soda	Centre	400
Vins	Centre	300

Calcul du total des ventes de soda ?

requêtes, tables redondantes, attribut redondant

Données pour l'analyse

grille des ventes : les axes correspondent à des attributs ayant des associations m-n

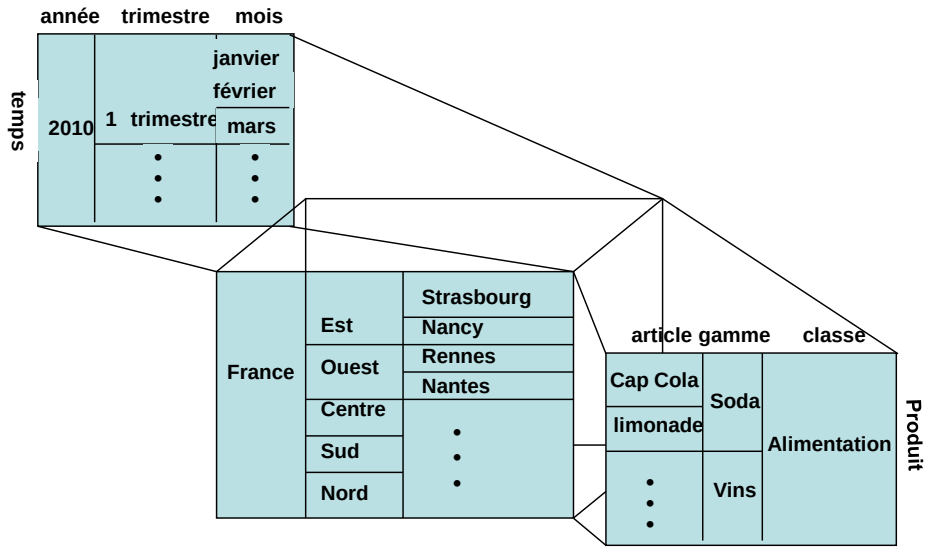
	Est	Ouest	Centre	Sud	**Total**
Soda	150	300	400	450	1300
Vins	250	200	300	150	900
Entretien	90	100	150	80	420
Total	**490**	**600**	**850**	**680**	**2620**

- facilité de lecture et de navigation
- intégration des calculs

Concepts du modèle multidimensionnel

- *mesure* = critère d'évaluation du processus décisionnel (ex : chiffre d'affaires, quantité en stock)
- *dimension* = axe d'analyse associé à un indicateur, représentant un sujet d'intérêt (ex : temps, produit, localisation)
- *hiérarchie* = décomposition d'une dimension en une arborescence de *niveaux* (ex : temps décomposé en mois, trimestre, année, ...)
- *attribut* = caractéristique d'un niveau d'une hiérarchie (ex : prix d'un article)
- *agrégat* = résultat d'un indicateur par rapport à des niveaux (ex : chiffre d'affaires du mois)

Cube multidimensionnel



Interrogation OLAP

Principes:

- Affichage d'une « face projetée » du cube multi-dimensionnel
- Opérations de manipulations d'un cube
- Visualisation des résultats

Exemple

Projection :

slicing : classe = alimentation et année=2012

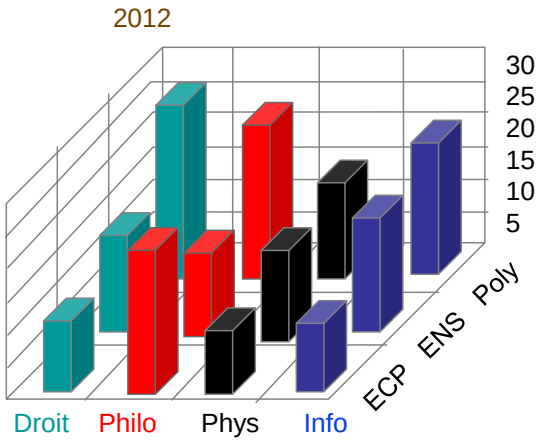
Produit : classe alimentation				
Temps : année 2012				
	trimestre 1	trimestre 2	trimestre 3	trimestre 4
Soda	1900	2000	2200	1300
Vins	....			
....				

Exemple

Pivotement : ajout de localisation
Slicing : trimestre=4 et pays=France et année = 2012

Produit : alimentation					
Temps : trimestre 4, année 2012					
Localisation : pays France					
	Est	Ouest	Centre	Sud	Total
Soda	150	300	400	450	1300

Autre visualisation



Avantage visuel

Problème de multiplicité des diagrammes

Nombre de succès par matière (droit, philo, phys, info) et par école (Centrale ECP, Normale Sup ENS, Polytechnique)

Choix des niveaux de hiérarchies

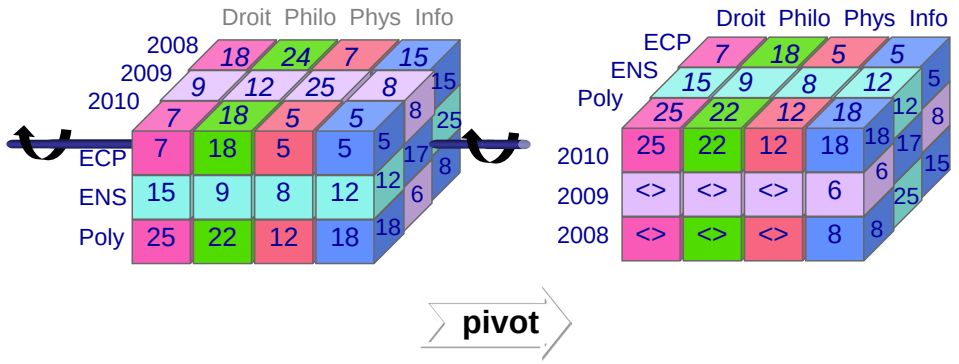
- Le choix des niveaux affecte
  - le volume des données représentées
  - le niveau de détail de l'information
  - la formulation des requêtes
- Exemple:
  - facturation détaillée : 200 appels/mois, 50 octets/appel, sur 2 mois on a 20 000 octets/abonné
  - sans facturation détaillée : 50 octets/abonné

Opérations sur le cube

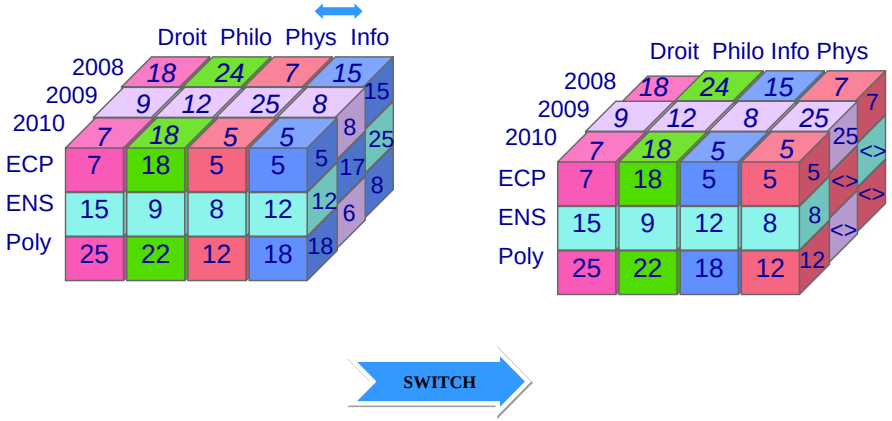
- Sur la structure
  - Rotate (pivot)
  - Switch
  - Split
  - Nest/unnest
  - Push/pull
  - Slice
- Sur le contenu
  - Dice (sélection)
  - Roll-up (grain supérieur)
  - Drill-down (grain inférieur)
- Entre cubes
  - Jointure
  - Opérations ensemblistes (union, intersection, différence)

Rotation (slicing)

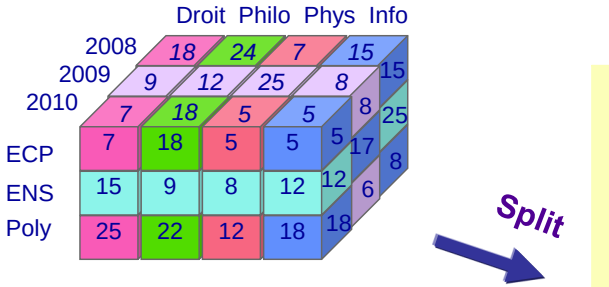
- Rotation (Pivot) : rotation par rapport à l'un des axes de dimension



Permutation de valeurs de dimensions (switch)



Décomposition (split)



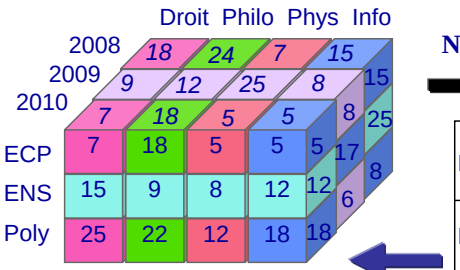
Droit	2010	2009	2008
ECP	7	9	18
ENS	15	<>	<>
Poly	25	<>	<>

Info	2010	2009	2008
ECP	5	8	15
ENS	12	17	25
Poly	18	6	8

Phys	2010	2009	2008
ECP	5	25	7
ENS	8	<>	<>
Poly	12	<>	<>

Philo	2010	2009	2008
ECP	18	12	24
ENS	9	<>	<>
Poly	22	<>	<>

Imbrication/désimbrication (nest/unnest)

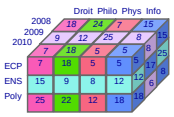


NEST →

	Droit			Philo			Phys			Info		
	2008	2009	2010	2008	2009	2010	2008	2009	2010	2008	2009	2010
ECP	18	9	7	24	12	18	7	25	5	15	8	5
ENS	<>	<>	15	<>	<>	9	<>	<>	8	<>	<>	12
Poly	<>	<>	25	<>	<>	22	<>	<>	12	<>	<>	18

← UNNEST

Concaténation (push/pull)



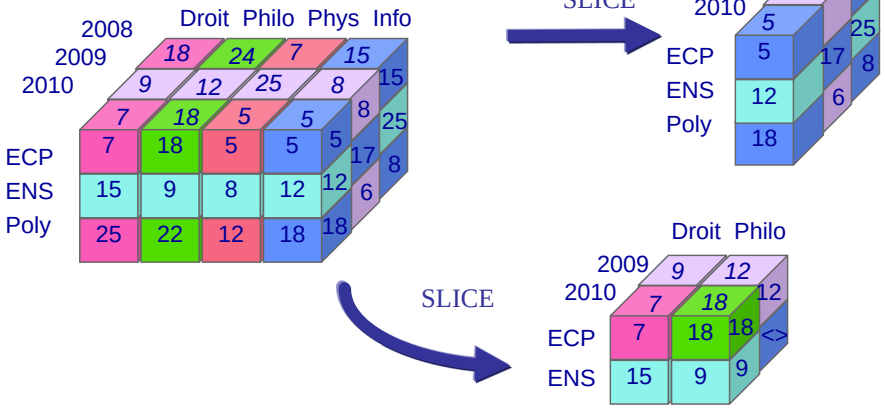
Push →

← Pull

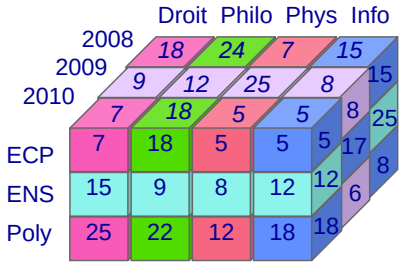
	Droit			Philo			Phys			Info		
ECP	2008 18	2009 9	2010 7	2008 24	2009 12	2010 18	2008 7	2009 25	2010 5	2008 15	2009 8	2010 5
ENS	2008<>	2009 <>	2010 15	2008 <>	2009 <>	2010 9	2008 <>	2009 <>	2010 8	2008 <>	2009 <>	2010 12
Poly	2008 <>	2009 <>	2010 25	2008 <>	2009 <>	2010 22	2008<>	2009 <>	2010 12	2008 <>	2009<>	2010 18

Projection (slice)

S'applique sur les valeurs des dimensions

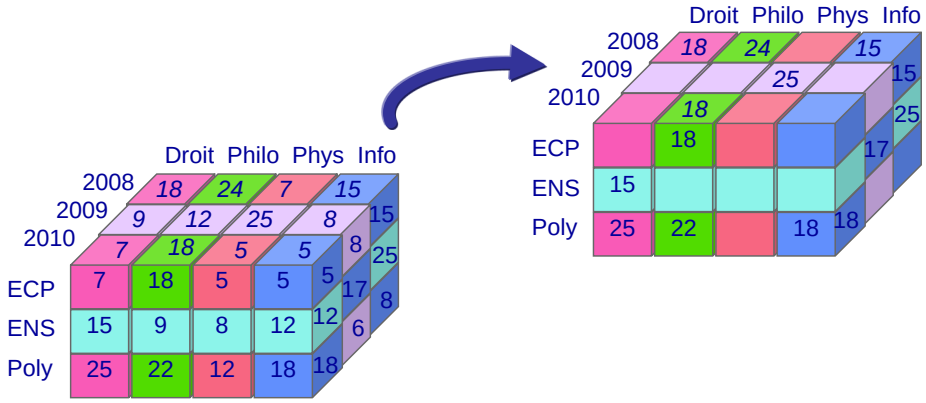


Projection agrégative

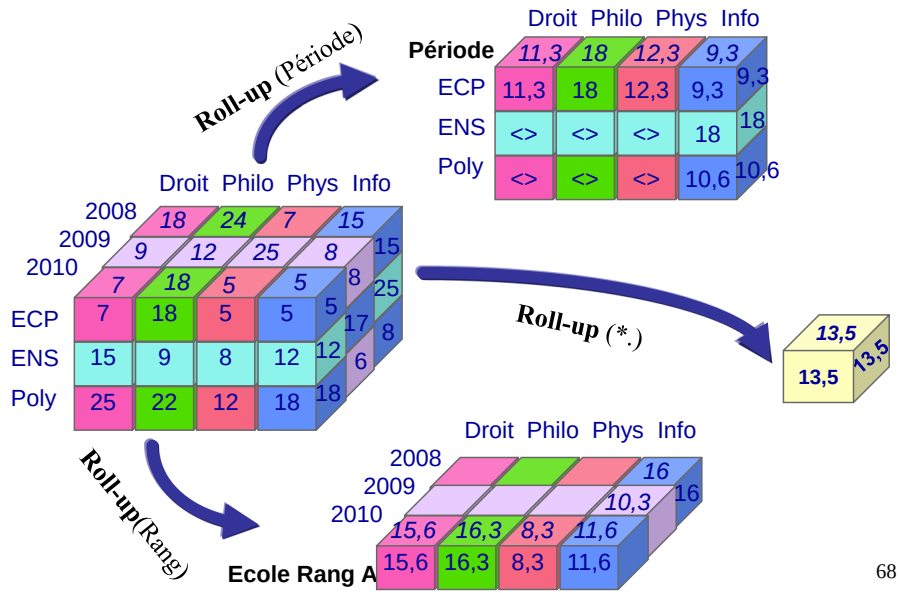


Sélection (Dice)

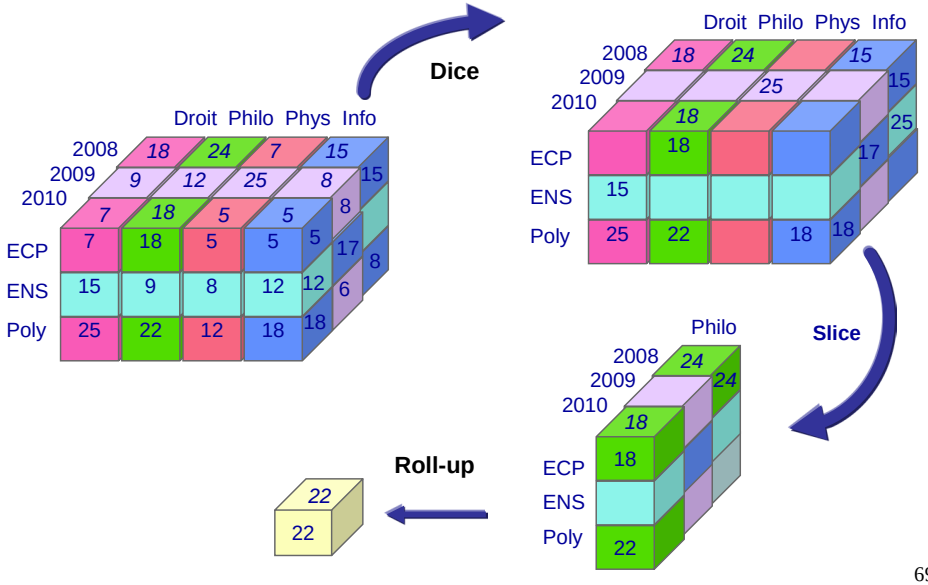
S'applique sur les valeurs des cellules



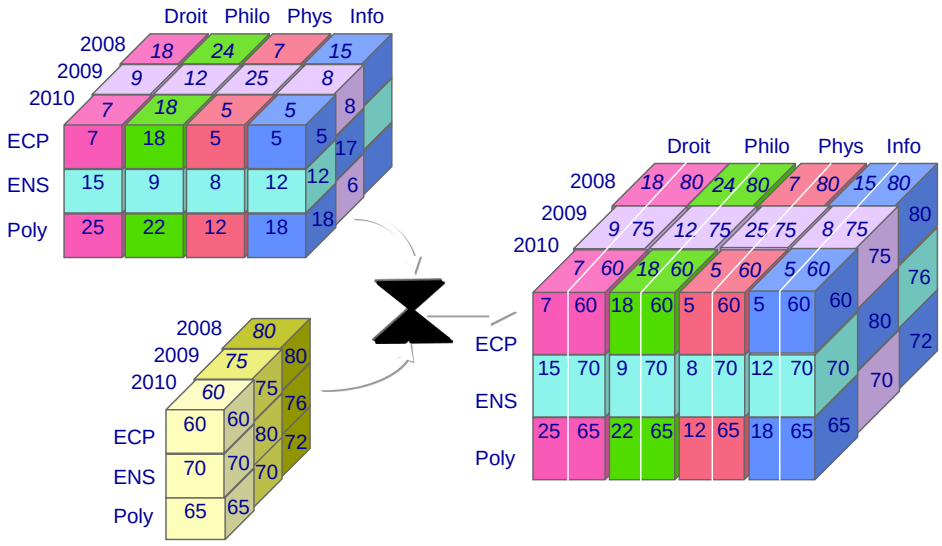
Roll up/Drill-down



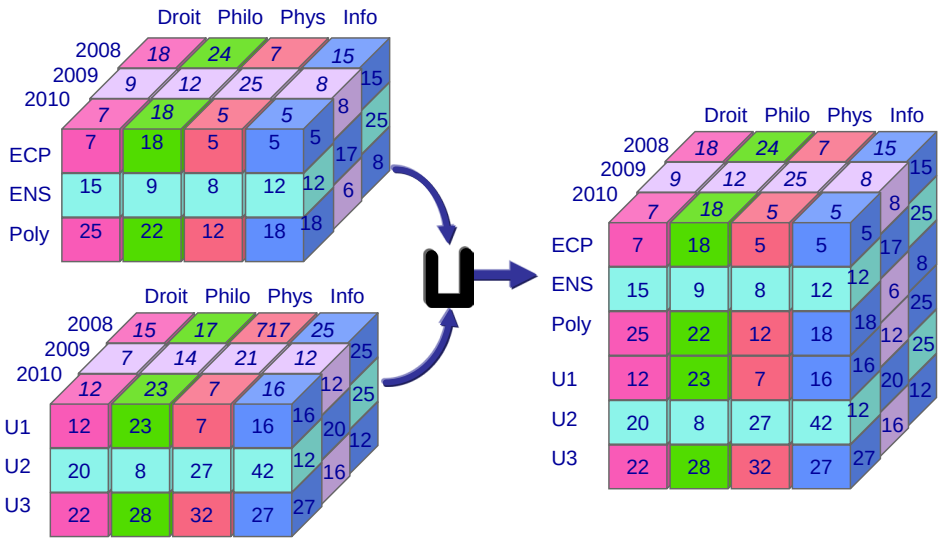
Composition d'opérations



Jointure



Union

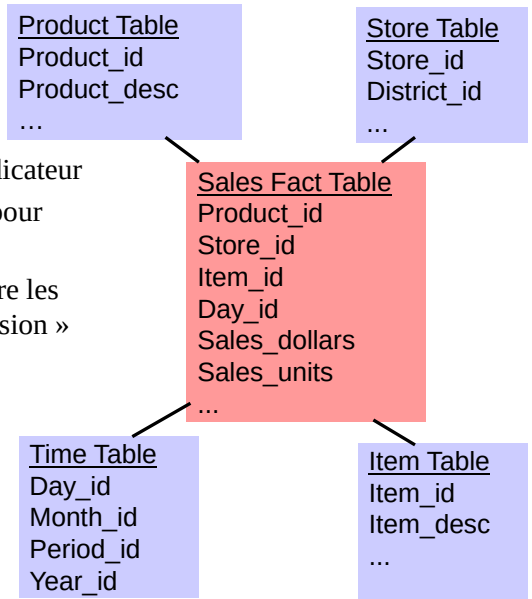


Représentation du cube

- Deux façons de représenter le cube :
  - Le modèle ROLAP : le serveur OLAP traduit les opérations sur le cube en opérations relationnelles.
  - Le modèle MOLAP stocke la BD multidimensionnelle dans des structures non relationnelles.
- La majorité des serveurs OLAP suivent le modèle ROLAP.
- Objectifs :
  - Dénormaliser (minimiser les jointures)
  - Résumés (effectuer des précalculs)
  - Partitionnement vertical (diminuer la taille des tables)
- Trois modèles : étoile, flocon, constellation

Modèle en étoile

- Une table « fact » par indicateur
- Une table dénormalisée pour chaque dimension
- associations en étoile entre les tables « fact » et « dimension »



Choix des tables « fact » et « dimension »

- Analyse des requêtes
  - attributs « group-by » indiquent les dimensions
  - attributs agrégés indiquent les mesures
  - attributs « where » sont les attributs des tables « dimension »

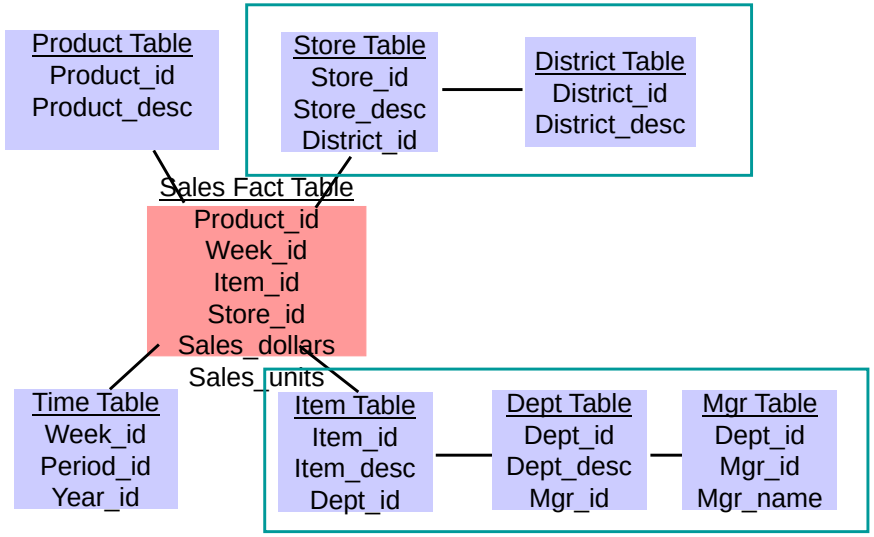
Exemple :

```
select sale.store_id,sale_product_id, sum(sale.price)
from product, sale
where product.product_id=sale.product_id and
product.product_desc = « clothes »
group by store_id, product_id
```

Choix des tables

- Analyse des procédés métier
  - La table fait est dérivée d'une activité liée à un procédé métier
  - les attributs de la table fait sont des attributs clés qui identifient l'activité plus des mesures
  - les dimensions sont dérivées des attributs clés des tables faits

Modèle en flocon



Avantages/Inconvénients

- Modèle en étoile :
  - évite jointures par dénormalisation
- Modèle en flocons
  - chargement/rafraîchissement plus rapide

Optimisation

- Bien choisir le schéma (étoile, flocon, …)
- Indexer les données
- Créer des vues matérialisées
- Paralléliser

On peut optimiser à tout niveau…

Choix du schéma

	Espace	Jointures	hiérarchie
Relation universelle	- - -	+++	- - -
étoile	+	+	- -
flocon	++	- -	+
3emeFN	+++	- - -	++

Dans l'idéal :

pour économiser les jointures : relation universelle
pour économiser l'espace : 3eme FN

Meilleur compromis : schéma en étoile

Indexation

- Pourquoi indexer ?
  - Tables de faits avec des millions de lignes
  - Nombreuses tables de dimension
  - Requêtes complexes impliquant beaucoup de données et comportant de nombreuses jointures

⇨ Nécessité d'accéder rapidement aux données pertinentes.

Différents types d'index :

arbre B, index bitmap, index de jointure

Vues matérialisées

Une vue est une requête nommée.
Elle ne contient pas de nuplets (même si de par son utilisation elle ressemble à une table)
La requête définissant la vue est exécutée à chaque utilisation

Une vue matérialisée est une table contenant les résultats d'une requête.

Elle peut améliorer l'exécution des requêtes en précalculant les opérations les plus coûteuses (jointure, agrégation, etc).
Elle peut servir pour précalculer le résultat de requêtes fréquentes

Pbs: Comment choisir les vues à matérialiser ?
Comment maintenir ces vues matérialisées ?

Sélection des vues

Suivant le modèle de données, une vue à matérialiser sera:

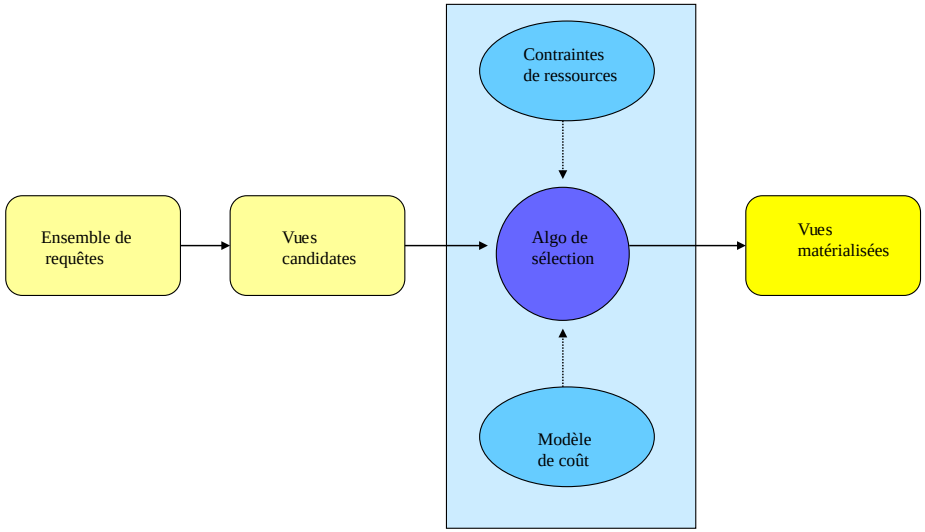
- une cellule du cube de données, dans le cas MOLAP où on a essentiellement des requêtes d'agrégations pour obtenir les cellules du cube
- un nœud de l'arbre algébrique d'exécution d'une requête dans le modèle ROLAP

Problématique

- dans un cube, il existe des dépendances entre cellules (une cellule calculée à partir d'autres)
- en ROLAP, certains nœuds sont partagés entre des requêtes

essayer de matérialiser ces cellules/nœuds partagés

Problématique



Complexité

- Nombre de sélections de vues matérialisables possibles
  dans un cube si n agrégations: 2^n (card. ens. des parties)
  si d est le nombre de dimensions et qu'il n'y a pas de hiérarchie, alors $n = 2^d$ (toutes les combinaisons de dimensions possibles)
  ⇨ 2^{2^d} sélections de vues possibles!! (problème NP-difficile)
- On considère généralement deux catégories de problèmes de sélection:

  1. connaissance a priori de l'ensemble des requêtes (statique)
  2. non-connaissance ⇨ sélection dynamique

Sélection des vues

Contexte: modèle ROLAP statique.

Principes:

1. sélection de l'arbre d'exécution optimal (d'après un modèle de coût) de chaque requête
2. recherche des expressions communes à ces arbres
3. fusion des différents arbres en un graphe unique:
   les feuilles = les tables,
   le niveau 1 = les sélections et projections,
   les niveaux supérieurs = les opérations ensemblistes (jointure, union, etc),
   le dernier niveau = le résultat
4. chaque nœud est étiqueté par le coût de l'opération et le coût de maintenance
5. recherche l'ensemble des vues dont la matérialisation minimise le coût total

Sélection des vues

Contexte MOLAP statique.

Principes de l'heuristique:

1. construction du treillis des vues en s'appuyant sur la hiérarchie de certaines dimensions: un nœud représente une vue, et on a un arc entre 2 nœuds si le résultat de l'une peut être évalué seulement avec le résultat de l'autre

2. une vue *Vi* est candidate si elle est associée à plusieurs requêtes et si il existe 2 vues *Vj* et *Vk* candidates telles que *Vi* soit le plus petit ancêtre commun

3. pour chacune des vues candidates, on calcule le bénéfice apporté par la matérialisation, et on choisit finalement celle de bénéfice maximal

Maintenance des vues

- Un entrepôt contient un ensemble de vues matérialisées
- Les tables ayant servi à construire ces vues changent (**insertions**, voire mises à jour ou suppressions)
- Si les changements ne sont pas répercutés sur les vues, leur contenu devient obsolète et donc elles deviennent inutiles (voire dangereuses)

stratégie de maintenance des vues matérialisées

Quand ?

Snapshot

Les vues sont mises à jour périodiquement. Une vue est une photographie de la base à un instant donné.

Transaction

Les vues sont mises à jour à la fin de chaque transaction.

Attente

Les vues sont mises à jour de manière différée, uniquement lorsqu'elles sont utilisées par une requête.

Comment ?

- Recalculer la totalité du contenu de la vue à partir des sources

  ⇨ très coûteux!!
- Propager les changements réalisés sur les tables sans recalculer complètement leur contenu.

  Différentes techniques :
  - maintenance incrémentale: on met à jour l'ancien résultat en manipulant seulement les nouvelles données qui ont été modifiées
  - maintenance autonome des vues: technique afin de permettre de maintenir la vue seulement en connaissant les changements (au prix d'info stockées en plus)
  - maintenance en batch: on traite un ensemble de changements en même temps, en différé

Extensions de SQL pour OLAP

Agrégation avec SQL : GROUP BY

Regroupement suivant n dimensions

GROUP BY d1, … , dn

Un groupe contient tous les faits qui ont les mêmes valeurs pour (d1, … , dn). Les groupes sont disjoints.

Clause SELECT : fonction d'agrégation

sum, avg, count, min, max, …

Résultat: un n-uplet par groupe

Exemple : Sales(StoreID, ItemID, CustID, qty, price)

```
SELECT StoreID, ItemID, CustID, SUM(price)
FROM Sales
GROUP BY StoreID,ItemID,CustID
```


Opérateur ROLLUP

Syntaxe: **GROUP BY [D] ROLLUP(D')**

D et D' sont des listes de dimensions d1 … dn

Agrégation sur n+1 niveaux de regroupements

Niveau 1 : group by d1 … dn-1 dn
Niveau 2 : group by d1 … dn-1
…
Niveau n : group by d1
Niveau n+1 : un seul groupe = la table des faits toute entière

Rollup partiel : moins de niveaux

GROUP BY D ROLLUP(D')

Ex: group by e1 rollup (e2, e3) crée les sous-totaux **(e1,e2,e3), (e1, e2) et (e1)**

Le ROLLUP est utile lorsque les Di sont les niveaux d'une même hiérarchie (ROLLUP (jour, mois, année))

Exemple de ROLLUP(1)

Sales(StoreID, ItemID, CustID, qty, price)

Requête

```
SELECT StoreID, ItemID, CustID, SUM(price)
FROM Sales
GROUP BY ROLLUP(StoreID,ItemID,CustID)
```

Résultat [s, i, c, p] ∈ **Résultat**

[s1, i1, c1, 2] [s1,i1,c3, 1] …
[s1, i1,null, 100] [s1, i2, null, 250] … **total par mag. par art.**
[s1, null, null, 4000] … **total par magasin**
[null, null, null, 100 000 000] **total général**

mais pas les n-uplets suivants :

[s1, null, c1, 30] … **total par mag. par client**
[null, null, c1, 200] … **total par client**

Exemple de ROLLUP (2)

Sales(RegionID, StoreID, ClerkID, hourlyPay)

Requête

```
SELECT RegionID, StoreID, ClerkID, AVG(hourlyPay)
FROM Sales
GROUP BY RegionID, ROLLUP(StoreID, ClerkID),
```

Résultat

Calcule les agrégations au niveau RegionID, au niveau regionID, StoreID et au niveau RegionID, StoreID, ClerkID.

Pas de total sur l'ensemble

Exemple de ROLLUP (3)

```
SELECT Type, Store, SUM(Number)
FROM Pets
GROUP BY type,store
```

Type	Store	Number
Dog	**Miami**	**12**
Cat	**Miami**	**18**
Turtle	**Tampa**	**4**
Dog	**Tampa**	**14**
Cat	**Naples**	**9**
Dog	**Naples**	**5**
Turtle	**Naples**	**1**

Résultat **avec ROLLUP**

Type	Store	Number
Cat	**Miami**	**18**
Cat	**Naples**	**9**
Cat	NULL	27
Dog	**Miami**	**12**
Dog	**Naples**	**5**
Dog	**Tampa**	**14**
Dog	NULL	31
Turtle	**Naples**	**1**
Turtle	**Tampa**	**4**
Turtle	NULL	5
NULL	NULL	63

Opérateur CUBE

Syntaxe: **GROUP BY [D] CUBE(D')**

D et D' sont des listes de dimensions d1 … dn

Agrégation sur tous les niveaux de regroupements par face, arrête, sommet du cube (2^n groupes)

group by d_1 group by d_2 group by d_3 …

group by d_1, d_2 group by d_1, d_3 …

…

group by d_1, …d_n

Cube partiel : moins de niveaux

GROUP BY D CUBE(D')

ex: group by e1 cube(e2,e3)

Calcule des sous-totaux pour

(e1, e2, e3), (e1, e2), (e1, e3), (e1)

Exemple de CUBE

```
SELECT Type, Store,
    SUM(Number)
FROM Pets
GROUP BY CUBE  (type, store)
```

Cat	**Miami**	**18**
Cat	**Naples**	**9**
Cat	NULL	27
Dog	**Miami**	**12**
Dog	**Naples**	**5**
Dog	**Tampa**	**14**
Dog	NULL	31
Turtle	**Naples**	**1**
Turtle	**Tampa**	**4**
Turtle	NULL	5
NULL	NULL	63
NULL	Miami	30
NULL	Naples	15
NULL	Tampa	18

GROUPING

- Les sous-totaux calculés par Rollup et Cube sont souvent utilisés dans les procédures pour effectuer d'autres calculs. On ne peut pas déterminer quels sont les tuples de sous-totaux, ni leur niveau d'agrégation.
- On ne peut pas distinguer les valeurs nulles de la base (notées NULL) de celles qui sont créées par rollup et cube (notées NULL aussi).
- GROUPING est une fonction qui renvoie 1 s'il y a un NULL créé par Rollup ou cube, 0 sinon

Syntaxe : SELECT …[GROUPING(dimension)…]
…
GROUP BY …[CUBE | ROLLUP|GROUPING SETS] (D)

GROUPING

EX: R(A, B, C, D)

```
SELECT A, B, SUM(D) as Total, GROUPING(A) as A1, GROUPING(B) as B1
FROM   R
WHERE …
GROUP BY ROLLUP(A, B)
```

A	B	Total	A1	B1
a1	b1	500	0	0
a1	b2	300	0	0
a1		800	0	1
a2	b1	200	0	0
a2	b2	400	0	0
a2		600	0	1
		1400	1	1

GROUPING SETS

- Permet de définir l'ensemble de groupes sur lesquels on veut calculer des agrégations. Evite de calculer tout le cube.
- Se définit dans la clause GROUP BY :

EX :

```
SELECT A, B, C, SUM(D)
FROM  R
WHERE…
GROUP BY GROUPING SETS ((A,B), (A,C), ())
```

Calcule les sous-totaux pour les groupes (A,B), (A,C), et le total global

CUBE(a,b,c) est équivalent à
GROUPING SETS ((a,b,c), (a,b), (a,c), (b,c), (a), (b), (c) , ())

GROUPING SETS

- La clause GROUPING SETS est l'union de plusieurs GROUP BY

Ex:

```
GROUP BY GROUPING SETS (a, b, c) est équivalent à
    GROUP BY a UNION ALL
    GROUP BY b UNION ALL
    GROUP BY c

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


Fonctions de classement

`rank()` : classement par position
`dense-rank()` : classement par valeur

Syntaxe : **`rank() over (order by ..)`**
`Order by` indique la mesure sur laquelle faire le classement.

Classement sur des groupes :
`Rank() over (partition by X Order by Y)`
Le classement de Y s'effectue sur chaque partition de X

Fonctions statistiques

CUME_DIST (): calcule la position d'une valeur par rapport à un ensemble de valeurs. Résultat dans]0,1].

PERCENT_RANK() : similaire à CUME_DIST, mais donne le pourcentage. Classement du rang d'une valeur par rapport à l'ensemble. Résultat dans [0,1]

NTILE(x) : partitionne un ensemble ordonné en x groupes de taille égale (à un n-uplet près), et numérote chaque paquet.

Syntaxe : `fct() over (order by ..)`

ROW_NUMBER : assigne un numéro unique, à partir de 1, à chaque n-uplet.

Fonctions de fenêtrage

```
analytic_function([ arguments ]) OVER (analytic_clause)

analytic_clause =
       [query_partition_clause]
       [order_by_clause[windowing_clause]]

query_partition_clause =
       PARTITION BY
       { value_expr [,value_expr]…
       |
       (value_expr [,value_expr]…)
       }
```


Fonctions de fenêtrage

```
windowing_clause =
   { ROWS | RANGE }
   { BETWEEN
       { UNBOUNDED PRECEDING
    | CURRENT ROW
    | value_expr { PRECEDING | FOLLOWING }
       }
   AND
   { UNBOUNDED FOLLOWING
    | CURRENT ROW
    | value_expr { PRECEDING | FOLLOWING }
     }
   | {UNBOUNDED PRECEDING | CURRENT ROW | value_expr PRECEDING
     }
   }
```


Exemple

Montant total des ventes effectuées pendant la dernière semaine (7 jours, incluant le jour courant)

```
select d.la_date ,
       sum(sum(v.prix)) over (order by d.la_date
  range between interval '7' day preceding and
  current row) as CA_semaine_glissante …
```

Densification

Principe

Obtenir un cube sans 'trous'. Ajouter les agrégats dont la valeur est nulle.

Cube + jointure externe sur chaque dimension

Syntaxe

PARTITION BY D

[LEFT | RIGHT] OUTER JOIN table ON pred

D, table : dimensions

pred : prédicat de jointure

Avantages: jointure plus efficace, moins de tri.

Exemple

- PURCHASE (Product-id, purchase-price, time-key)
- PRODUCT(product-id, product-name, category)
- TIME(time-key, day, month, year)

Q1. Vente des produits par mois

```
SELECT t.month, p.product-name, SUM(f.purchase-price) as
   sales
FROM PURCHASE f, TIME t, PRODUCT p
WHERE f.time-key = t.time-key
AND f.product-id = p.product-id
GROUP BY p.Product-name, t.month
```

S'il n'y a eu aucune vente du produit p1 au mois de février 2005, le n-uplet (02-2005, p1, …) n'apparaîtra pas dans le résultat.

Pb. Moyenne des ventes par trimestre ?

Prendre en compte les produits non vendus certains mois, en générant un n-uplet avec la valeur 0 pour les ventes dans ce cas.

Définitions

- INNER join : pour qu'un n-uplet soit dans le résultat, il faut que la valeur de l'attribut de jointure apparaisse dans les 2 tables.
- OUTER join : tous les n-uplets apparaissent dans le résultat, avec une valeur nulle pour les autres attributs lorsque les valeurs ne joignent pas.
- RIGHT OUTER JOIN : tous les n-uplets de la table de droite apparaissent dans le résultat.
- LEFT OUTER JOIN : tous les n-uplets de la table de gauche apparaissent dans le résultat.

Exemple (suite)

Génération d'un n-uplet pour les produits n'ayant pas été vendus certains mois, en mettant la valeur 0 pour les ventes dans ce cas.

```
Select v2.month, v1.product-name, nvl(v1.sales, 0)
FROM
(SELECT t.month, p.product-name, SUM(f.purchase-price) as sales
FROM PURCHASE f, TIME t, PRODUCT p
WHERE f.time-key = t.time-key
AND f.product-id = p.product-id
GROUP BY p.Product-name, t.month) v1 PARTITION BY (product-
   name)
RIGHT OUTER JOIN
(SELECT distinct t.month
FROM TIME t) v2
ON v1.month = v2.month)
```

Exemple (suite)

v1 : vente des produits par mois

v2 : valeurs distinctes des mois

PARTITION BY (product-name) : prend le résultat de v1 et partitionne par nom de produit.

RIGHT OUTER JOIN : fait la jointure externe en prenant toutes les valeurs des mois (RIGHT)

nvl(v1.sales, 0): insère la valeur 0 (au lieu de NULL) pour les n-uplets n'ayant pas de valeur de jointure avec la table de gauche.