Systèmes d’information décisionnels et entrepôts de données

Giuseppe Berio

giuseppe.berio@univ-ubs.fr

2023

Organisation/Contenu

- Intervenants : Giuseppe Berio (GB- UBS), Michel Dubois (MD-UBS), Intervenant Extérieur (AL - **Jems Group**)
- Partie I (GB) : Introduction, Modélisation, Intégration
- Partie II (MD) :
  - Schéma en Etoile avec Oracle, optimisations ROLAP, Solution HOLAP
  - MDX (requêtage) avec la plateforme Pentaho BI
  - Solutions ROLAP et HOLAP avec un serveur SAS BI, MDX avec SAS OLAP Server ; Solutions in memory de SAS (SAS Visual Analytics) comme alternative à un cube dans SAS OLAP Server
  - Introduction à l'ETL TALEND
- Partie III. MSD (AL) : SQL Server et services d'intégration (SSIS) + SSAS (stockage+ requêtage)
- Partie III. INFO (MD) : TALEND Approfondi
- Évaluation : projet d'envergure par ? (soutenance) + 2 épreuves individuelles
- Plateforme pédagogique : 2 pages distinctes pour GB et MD

Organisation Partie I

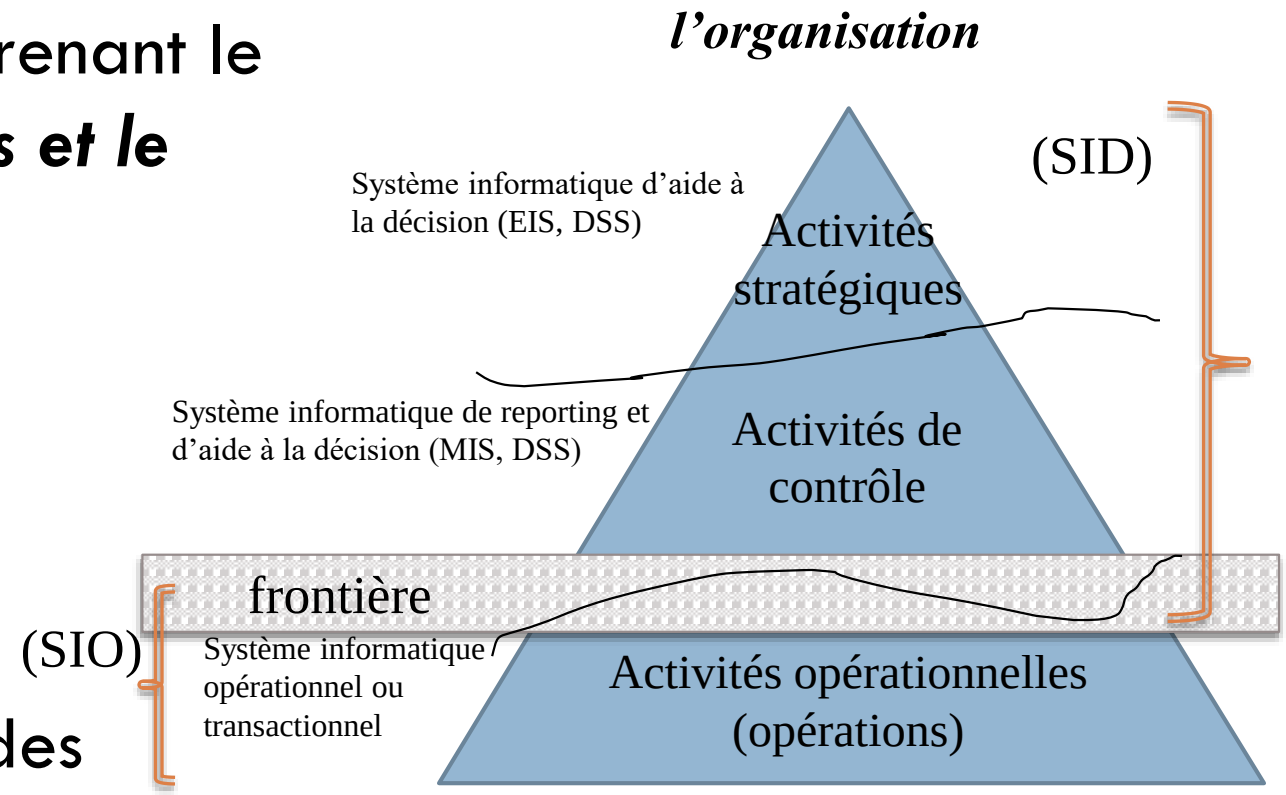
- 8 séances CM et 3,5 TD = 11,5 séances
- CM souvent en salle info
- 2/3 séances : Introduction (CM)
- 3,5 séances : Modélisation de l'entrepôt (CM+TD) → SQL Developer
  - 0,5 séance de présentation de la modélisation dans SQL Developer
- Reste des séances : Intégration de données (CM+TD)
  - Modélisation zone de staging → SQL Developer
  - Traitements ETL → outil à voir ou manuel

Concepts

Définition de système d'information (SI)

- Le système d'information d'une organisation est le ***système*** comprenant le ***matériel, le logiciel, les procédures et le personnel,…,*** nécessaire pour
  - ***Collecter***
  - ***Archiver***
  - ***Élaborer***
  - ***Échanger***

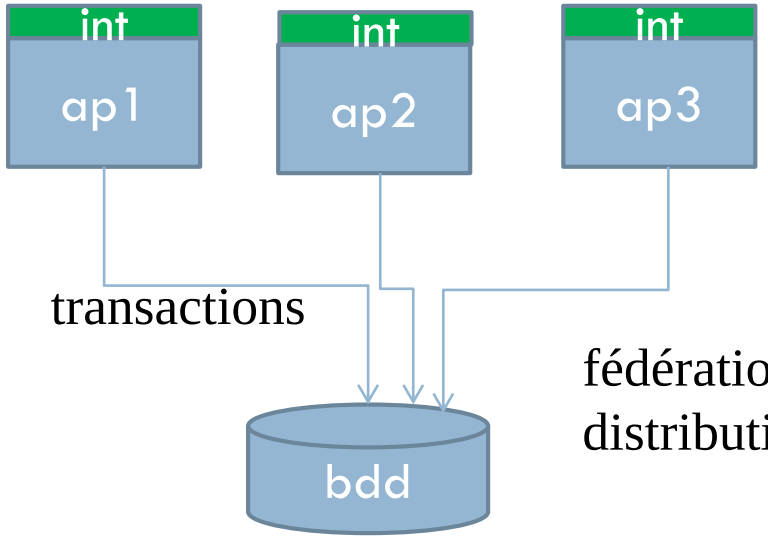
  les informations utilisées au sein des activités de ***l'organisation***



Système informatique ⊆ matériel + logiciel

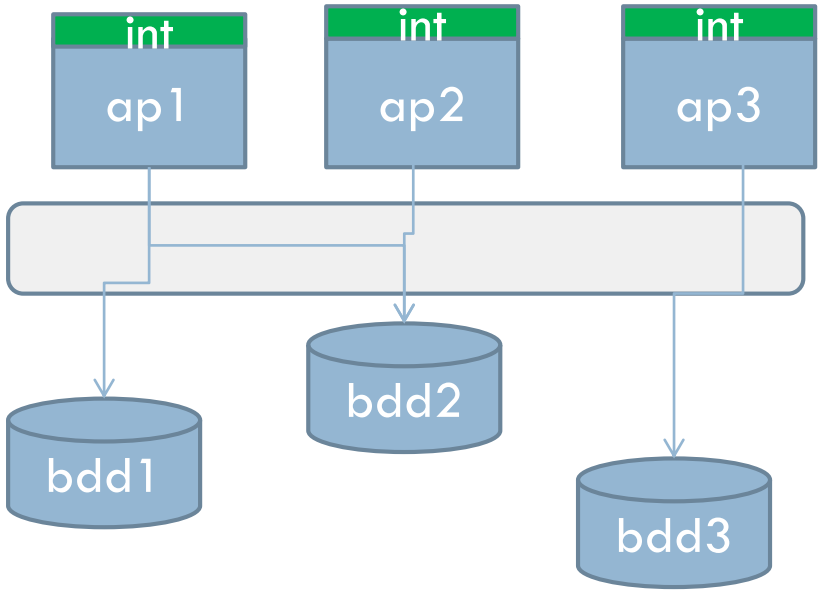
Le système informatique opérationnel (SIO) théorique (architecture)

système centralisé



UE : bddr (bddor,etc.)

système distribué



- ***Nombre élevé de transactions par unité de temps*** : maximiser le nombre d'utilisateurs travaillant en parallèle sur les mêmes bases
- ***Intégrité de données (sous concurrence)*** : maximiser la qualité de données stockées, ne pas avoir un comportement/résultat erroné

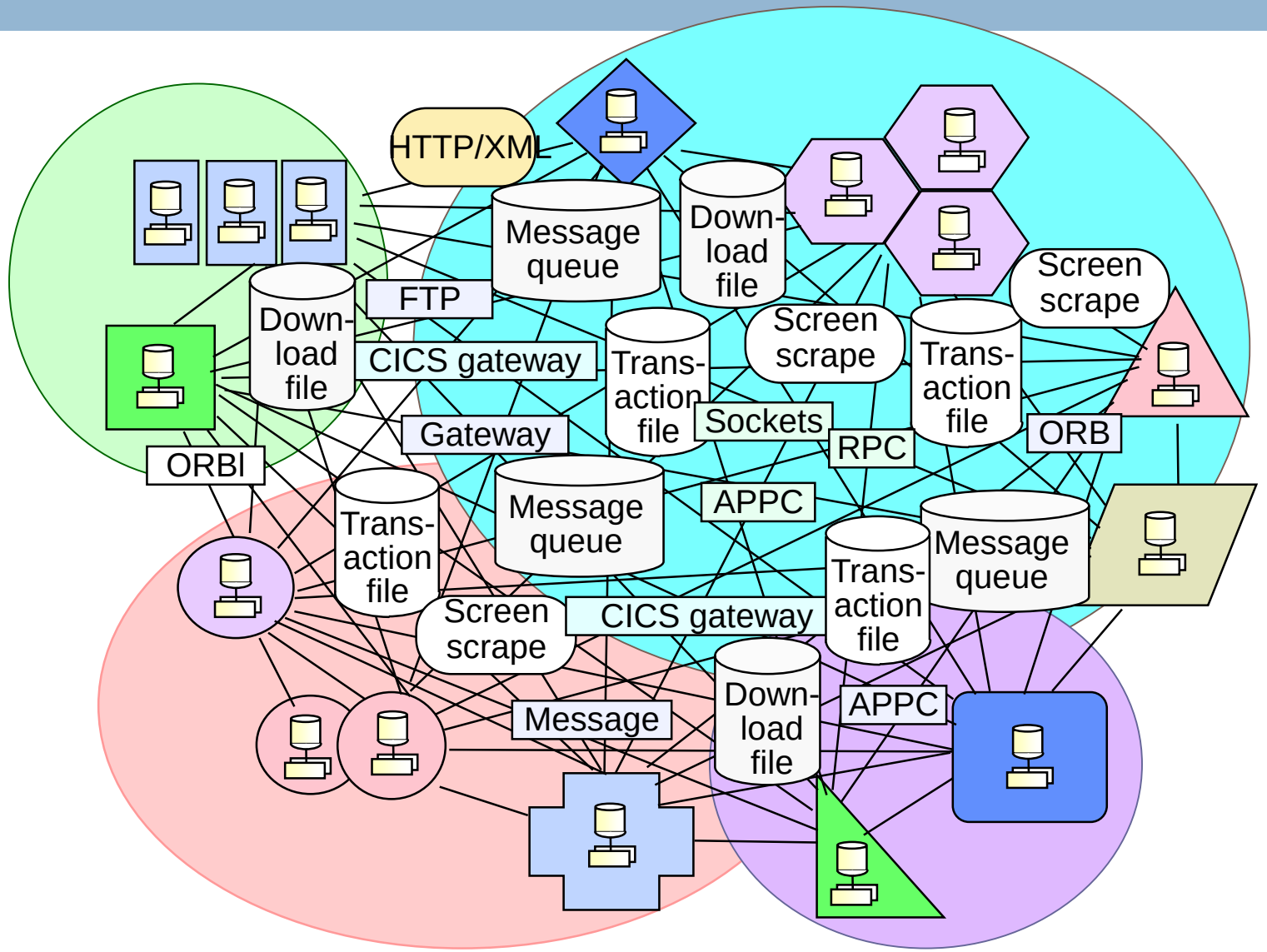
Structure et caractéristiques d'une application SIO

- Un **ensemble d'objets** qui interagissent permettant d'un coté d'interagir avec l'utilisateur et d'autre coté de manipuler (créer, modifier, supprimer, lire) les données en conséquence
- Chaque exécution d'une application manipule **peu de données** mais en concurrence
- **Diagrammes UML clés :** séquence et classe (le diagramme de cas est utilisé pour construire les 2 autres)

Architecture réelle du SIO

Architecture réelle du SIO = ensemble d'applications autonomes et communicantes

données éparpillées, non synchronisées, non standardisées, copies non maitrisées, possédant formats disparates etc.



Idée

Sources

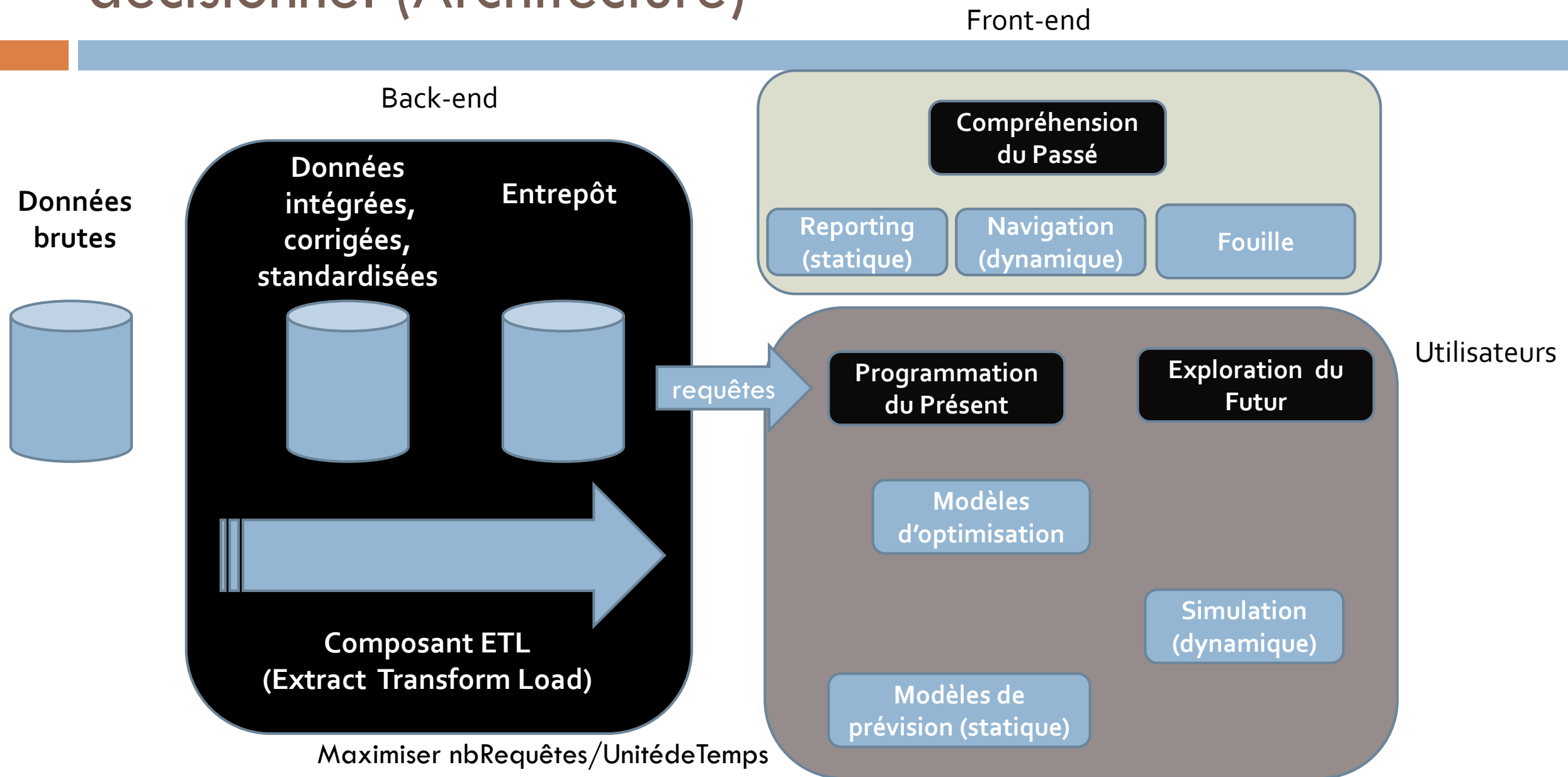
GRC	GC	C	L	SAV
Client-produits d'intérêt	Client-produit commandés (suivi de commandes)	Client-adresse de facturation, facturation (comptabilité)	Client-adresse d'expédition, livreur, état de la livraison (suivi de livraison)	Client-produits rendus

Requêtes

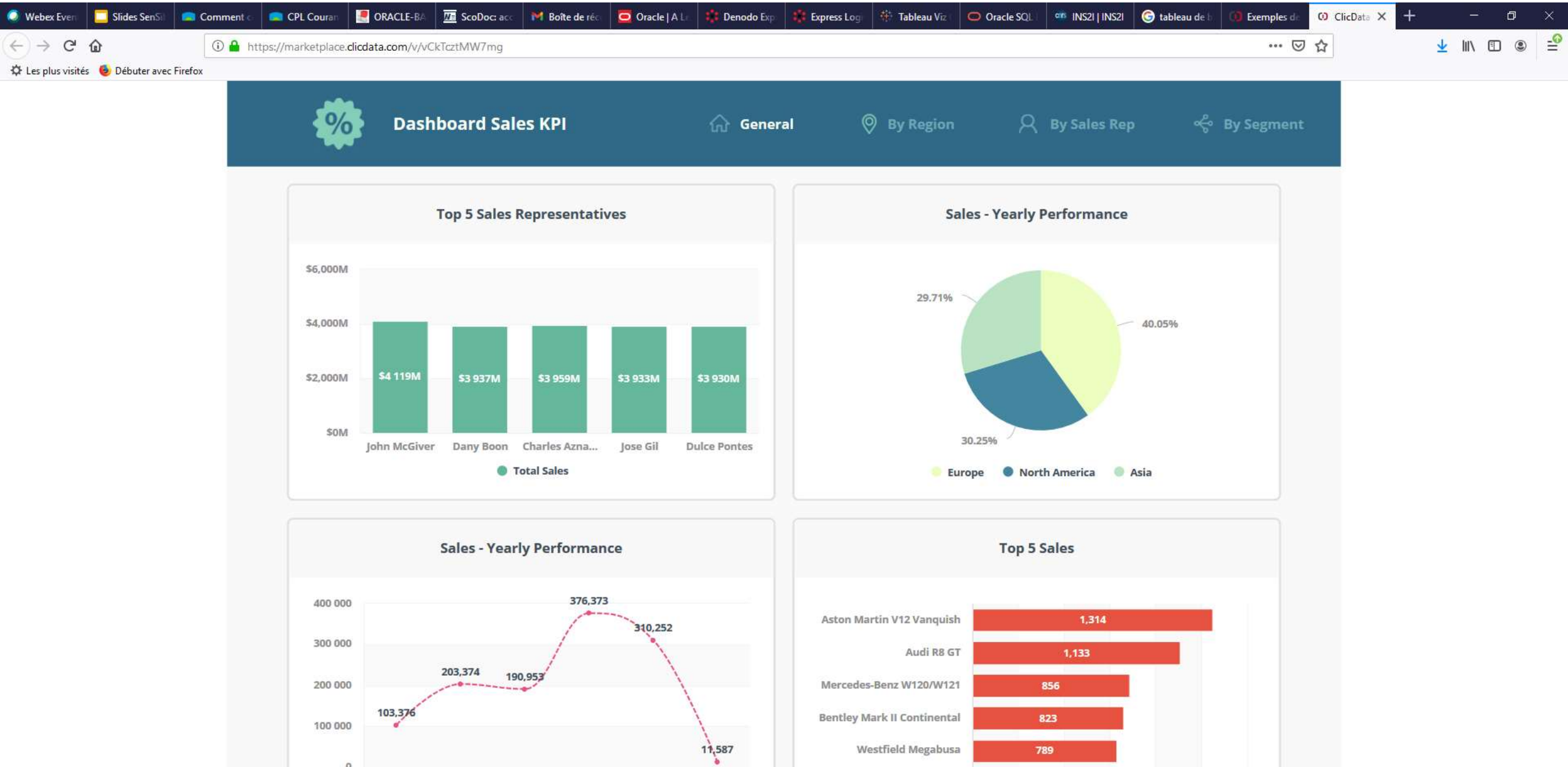
Front-end

- Délai livraison par profil client sur une période fourni (GC+L+GRC)
- Nombre d'impayés par produit (GC+C)
- Nombre de rendus par produit, livreur, adresse d'expédition (SAV+L)

Composants typiques d'un système informatique décisionnel (Architecture)



Front-end : Tableau(x) de bord statiques



Front-end : Tableau(x) de bord dynamiques

futur

OTHER LEVEL'S FINANCE STATUS DASHBOARD

Last Update 10 Sep 2020, Thursday

Main Dashboard
Human Resource
Status Report

Year
2019
2020
2021

Department
CCP | E2E
ICCD | IMC
PAN | SBN
SSBD | SSMC

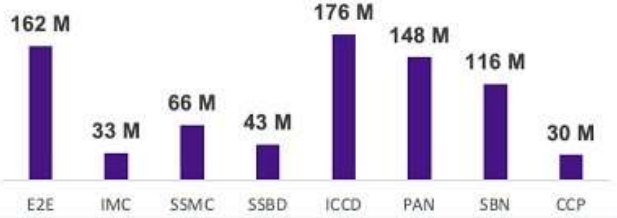
Project Status
Cancelled
Closed
Current

Total Project's	C APEX	Lockers	Popular	Saving	OPEX
99 PO	774,111,895	427,057,284	503,392,307	▼ 15% 657,920,172	61% (100%)

Current 57 | Closed 40 | Canceled 2

InIntroduscuie Project's 20 | Standard Project's 79

LOI Issued by Departments



Total Targets For September 9,817 | Total Actuals For September 9,113 | GAP ▼ -7% -704

GAP 1 Bdgt vs SoW -80,415,428

GAP 2 SoW vs WO -46,538,100

GAP 3 MOP vs As Built -86,969,084

GAP 4 As Built vs Invoiced -27,049,456

	GAP 1	GAP 2	GAP 3	GAP 4
E2E	-40,483,403	-28,340,156	-35,601,280	-23,297,998
CMI	-435,642		-593,591	-2,110,376
DCCI	-34,858,340	-3,323,138	-47,217,444	-1,123,075
NAP	-4,402,315	-3,316,241	-427,132	-210,279
PCC	-166,759	-11,486,148	-3,060,668	133,776

68 Total Project's | 558,037,890 Total Budget's | 31 Canceled

Mommercial Negiation

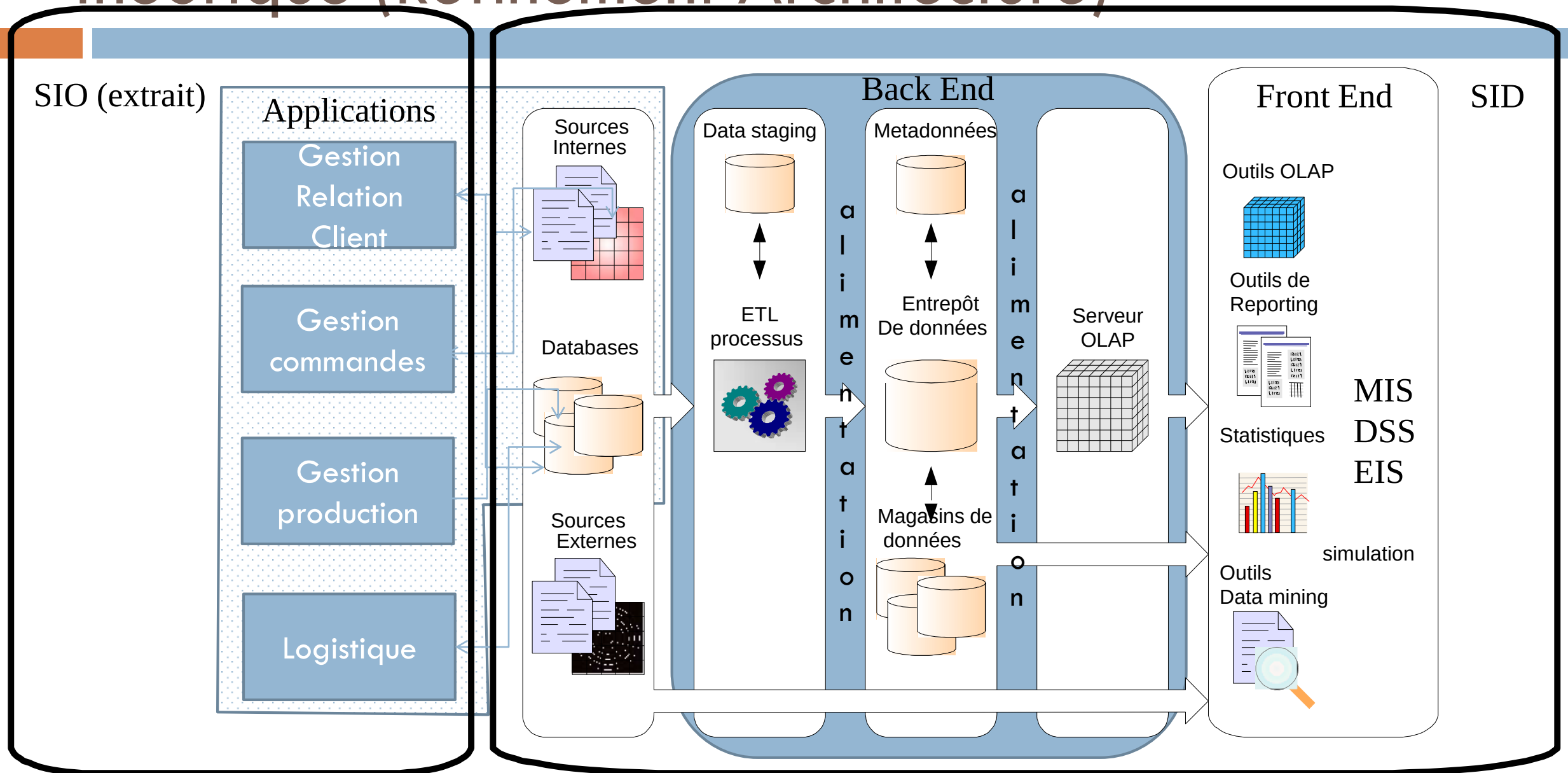
- porttitor congue massa. Fusce posuere
- malesuada libero, sit amet
- porttitor congue massa. Fusce posuere
- malesuada libero, sit amet

"Copyright @ Other Level's Youtube Channel © Any illegal reproduction of this content in any form will result in immediate action against the person concerned."

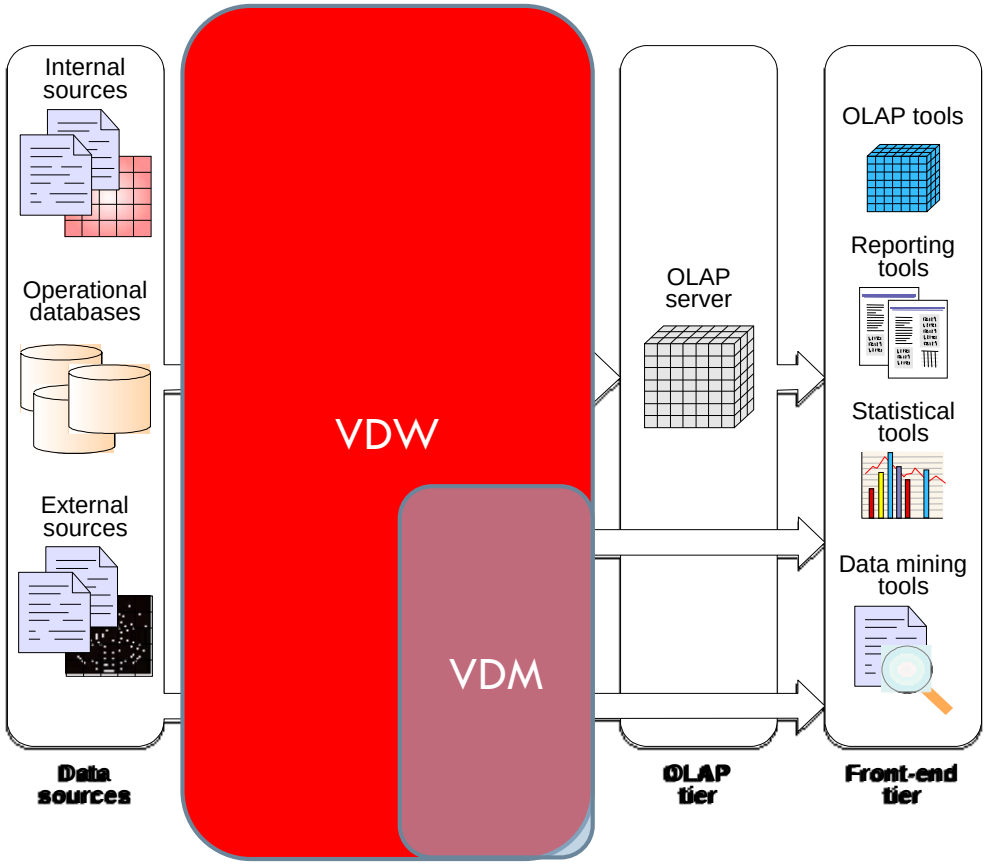
Typologies d'application décisionnelle (front end)

- Management information systems (reporting – exploration du passé) – MIS
- Decision support systems (prévision – optimisation - simulation) - DSS
- Executive information systems (reporting – prévision) - EIS

Le système informatique décisionnel (SID) théorique (Refinement Architecture)



Architectures simplifiées



VDW : Virtual DataWarehouse

VDM : Virtual Data Marts

Structure d'une application SID

- Visualisation d'un ensemble d'informations présentées pour permettre à l'organisation la **prise de décision de pilotage efficace, efficiente et réactive**
  - **Back-end** : fournit une organisation de données pour permettre la lecture (requêtage) efficiente de toutes les **données disponibles, corrigées et intégrées,** peu importe leurs sources, **réactif aux changements** dans le SIO et aux sources externes
  - **Front-end** : chaque exécution d'une application demande la visualisation efficace et efficiente d'un ensemble **d'informations élaborées** à partir d'une grande masse de données stockées dans le back-end – ces informations n'ont pas besoin d'être réélaborées si les données correspondantes ne se modifient pas dans le SIO – une application ne modifie pas ces informations

Entrepôt

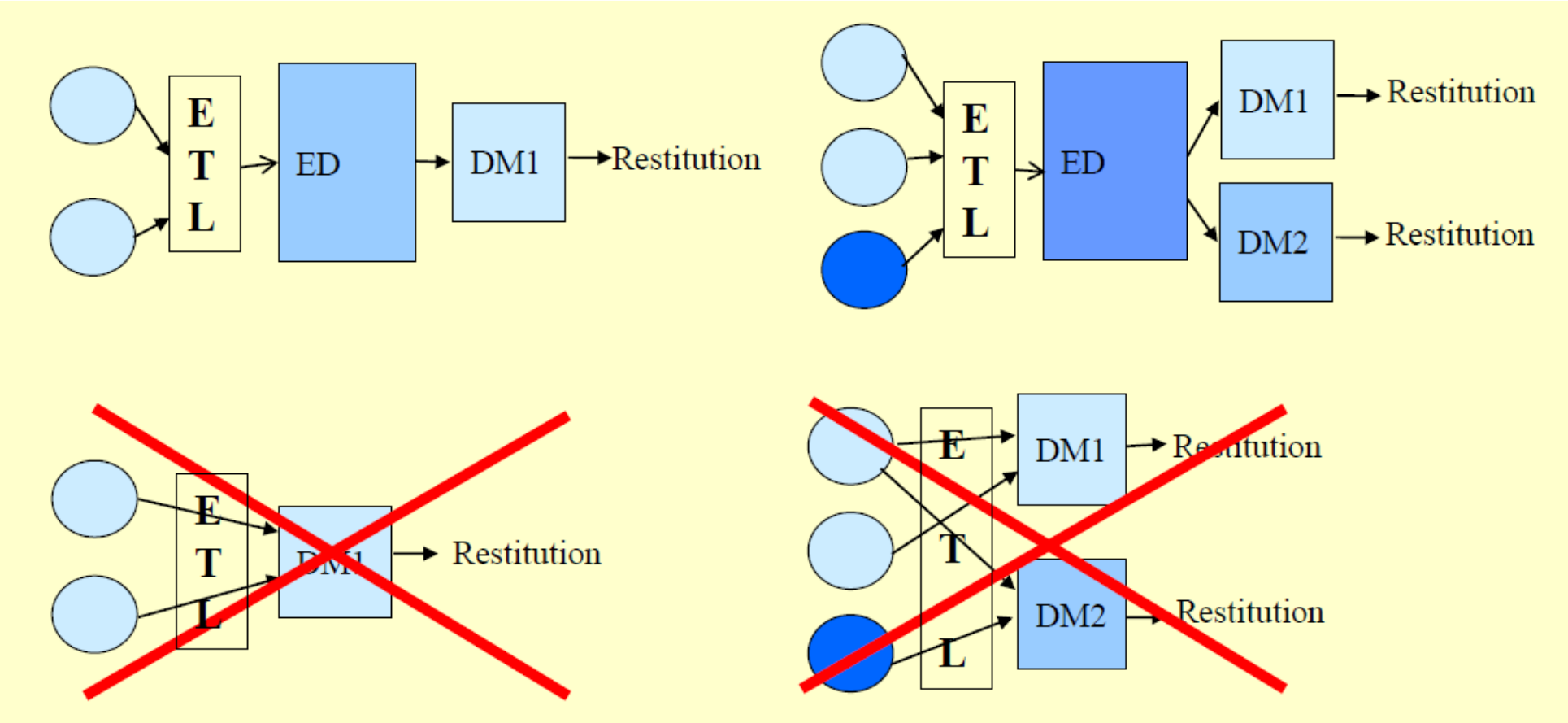
- Un ***Entrepôt de données (Datawarehouse)*** est une collection de données
  - Orientées à tout sujet (d'analyse)
  - Intégrées
  - Historisées
  - Non-volatiles

  principalement utilisées pour l'aide à la décision *W.H. Inmon (1996)*

Magasin de données

- Orienté vers une étude particulière, pour un traitement spécifique
  - Comportement de la clientèle → quels produits sont achetés le plus fréquemment par certains clients (types), pourquoi certains clients n'achètent plus certains produits, etc.
  - Impact des promotions sur les ventes d'un produit → est ce que la promotion fait vendre plus d'un produit, est ce qu'elle attire de nouveaux clients, etc.
- Sous-ensemble de données, normalement dérivées de l'entrepôt

Architectures alternatives



BI (front-end) : Évolution du marché 2013 → 2020

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms

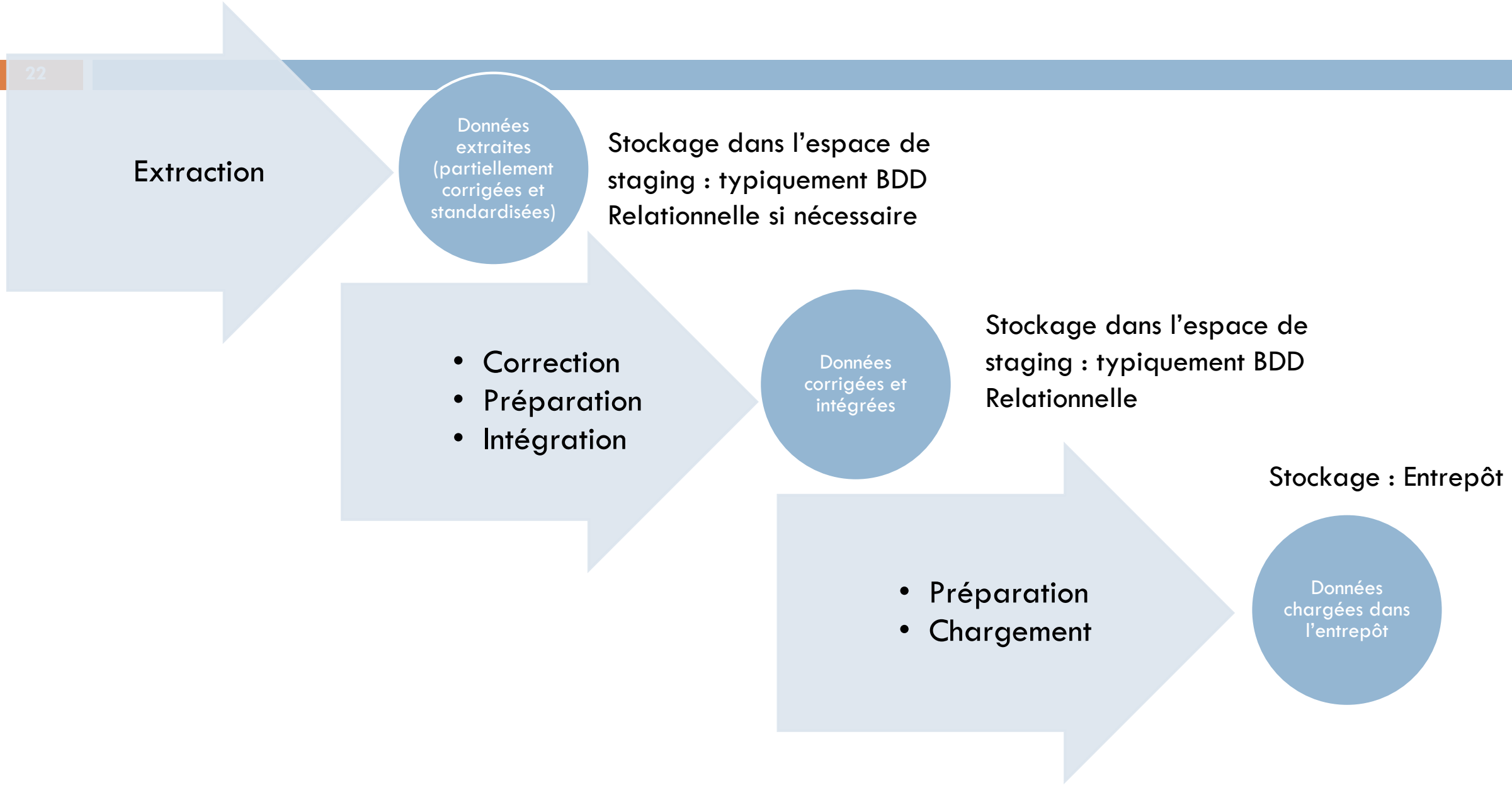
Source: Gartner (February 2020)

Données (back-end) : marché 2019

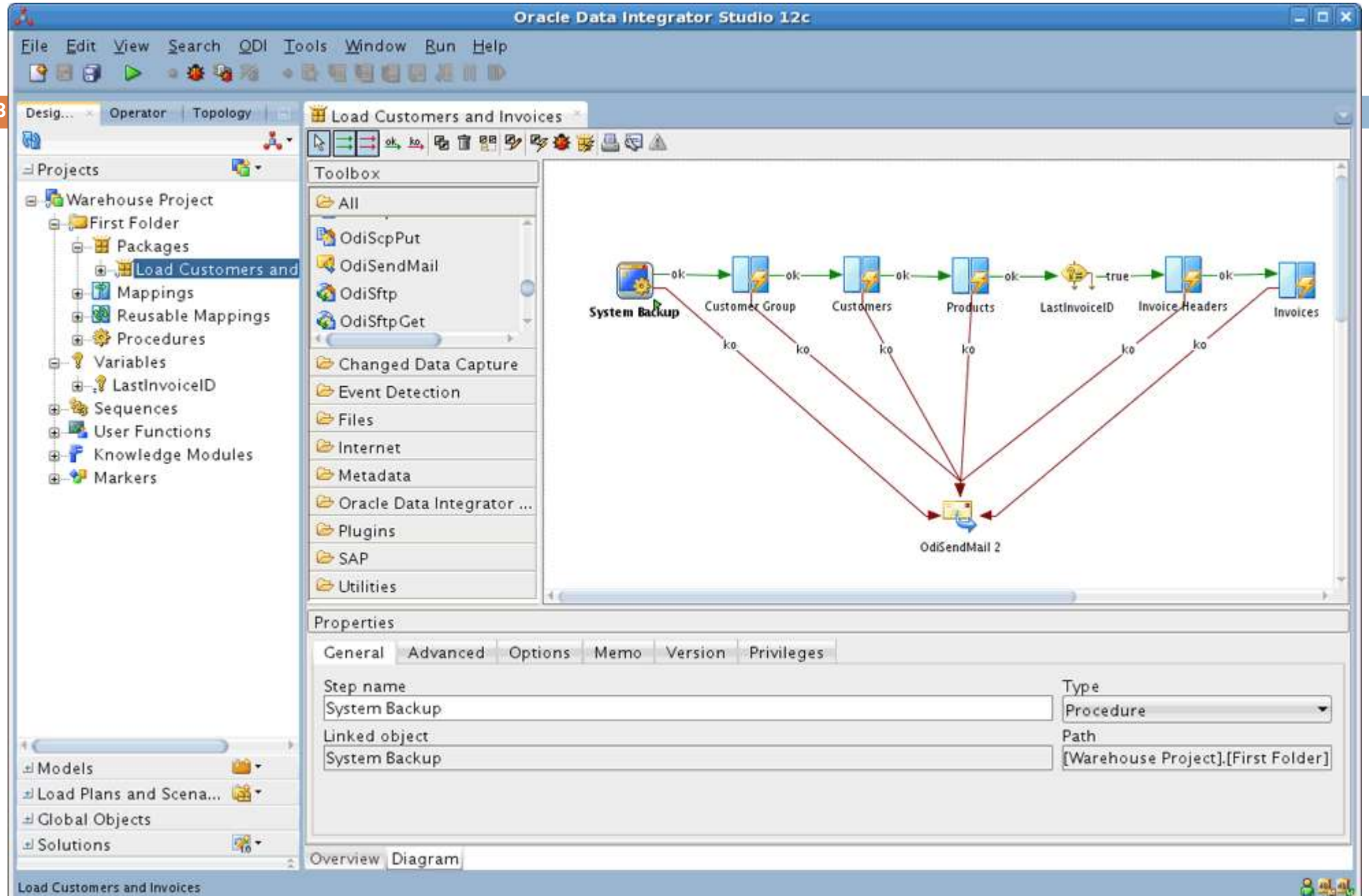
Figure 1. Magic Quadrant for Operational Database Management Systems

Source: Gartner (November 2019)

Processus ETL : décomposition en traitements



Oracle DI


Oracle Data Integrator Studio 12c
File Edit View Search ODI Tools Window Run Help
Desig... Operator Topology
Load Customers and Invoices
Projects
Warehouse Project
First Folder
Packages
Load Customers and
Mappings
Reusable Mappings
Procedures
Variables
LastInvoiceID
Sequences
User Functions
Knowledge Modules
Markers
Models
Load Plans and Scena...
Global Objects
Solutions
Toolbox
All
OdiScpPut
OdiSendMail
OdiSftp
OdiSftpGet
Changed Data Capture
Event Detection
Files
Internet
Metadata
Oracle Data Integrator ...
Plugins
SAP
Utilities
System Backup
Customer Group
Customers
Products
LastInvoiceID
Invoice Headers
Invoices
OdiSendMail 2
ok
ko
true
Properties
General Advanced Options Memo Version Privileges
Step name
System Backup
Type
Procedure
Linked object
System Backup
Path
[Warehouse Project].[First Folder]
Overview Diagram
Load Customers and Invoices

Prepare Customer Target TRAIN data *

Up | Run | Stop | Continue | Step | Run From Selected Transformation | Run Selected Transformations

WEB_CUSTO... (WEB_CUST...)

Target Summe... (WEB_SUMM...)

WEBPATH (WEBPATH)

SQL Join

SQL Join

Web Customer... (WEBSAMP)

Sort

Frequency

Transpose

Transpose_O... (WSIJZNNP)

Diagram | Code | Log | Output

Details

Status | Warnings and Errors | Statistics | Control Flow

Node	Name	Status
0	Precode	Completed successfully
1	Sort	Completed successfully
2	Frequency	Completed successfully
3	Transpose	Completed successfully
4	Postcode	Completed successfully
	Prepare Customer Target TRAIN ...	Completed successfully

Completed successfully

View Data: Transpose_OUTPUT2 (290 rows) (Browse)

#	customer_id	_NAME_	COL1	COL2	COL3	
19	cxx115	requested_file	/Cookie_C...	/Departme...	/Email.jsp ...	/Home
20	cxx116	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Cooki
21	cxx117	requested_file	/Product.js...	...	...	
22	cxx118	requested_file	/CDMA/Err...	...	...	
23	cxx119	requested_file	/Home.jsp ...	/Product.js...	/Search.js...	/Site_
24	cxx12	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Depa
25	cxx120	requested_file	/Catalog_...	/Catalog_...	/Cookie_C...	/Depa
26	cxx121	requested_file	/Home.jsp ...	...	...	
27	cxx122	requested_file	/MiniDisc/E...	/Product.js...	/Retail_Sto...	
28	cxx123	requested_file	/Cookie_C...	/Departme...	/Product.js...	/Subca

Traitements

- Des exemples de traitements nécessaires atteindre les objectifs fonctionnels d'un ETL sont indiqués ci-dessous ; la combinaison de ce traitements forme un processus ETL ; l'ETL par son interface de programmation simplifie l'écriture de chaque traitement ainsi que la combinaison de plusieurs traitements
- Transformation
  - Recodage, changement d'unité de mesure
  - Changement de clés
  - Fusion, split d'une donnée
  - Agrégation
  - Modification d'une donnée
- Identification
  - Recherche de similarité
  - Recherche de données de qualité insuffisante
- Intégration
  - Fusion
- L'écriture de ces traitements est simplifiée par un ETL mais la conception reste complexe

ETL/Intégration : marché 2020

Figure 1. Magic Quadrant for Data Integration Tools

Source: Gartner (August 2020)

Approfondissement des concepts

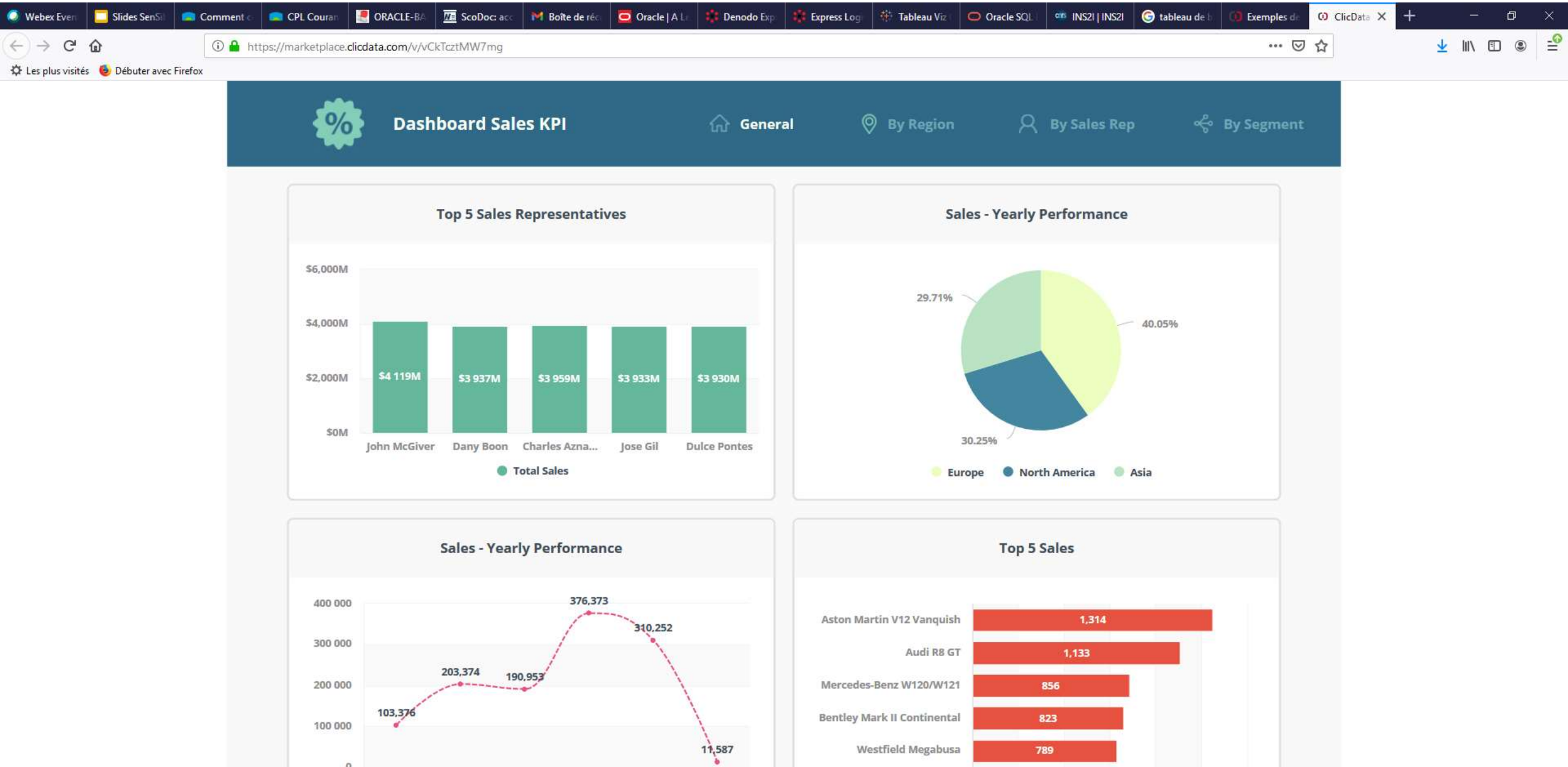
Points à développer

- Compréhension des éléments centraux pour la modélisation de l'entrepôt et de l'espace de staging (données standardisées, corrigées et intégrées)
- Modélisation de l'entrepôt
- Modélisation de l'espace de staging
  - Zone d'extraction
  - Zone d'intégration
- Principes de modélisation du processus ETL

Compréhension des éléments centraux pour la modélisation de l'entrepôt et de l'espace de staging

- Exemples : front-end (ce qu'on souhaite obtenir)
- Exemples : modélisation logique-physique (ce que l'on doit avoir)

Exemple : Rendu visuel pour un utilisateur



Modèles de données (rappel)

- Modèle conceptuel de données 2
  - Eléments pour la représentation (précise, souvent dans un langage formalisé) de l'information sous-jacente les données, indépendants de toute mise en œuvre

- Modèle logique de données 1
  - Eléments pour l'organisation de données suivant une mise en œuvre informatique, mais indépendants de l'outil permettant cette mise en œuvre (par exemple, le modèle relationnel)
- Modèle physique de données
  - Un modèle de stockage du modèle logique augmenté avec des paramètres propres au stockage (par exemple, index, cluster) visant l'optimisation

Exemple I (modèle logique de données)

Données brutes (sources)

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	76	02/02/2013

1, 100a, 54, 01/02/2013; 2, 200, 56, 02/01/2013

Données intégrées

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	76	02/02/2013
3	**100**	**54**	**01/02/2013**
2	200	56	02/01/2013

ClientID	
1	
3	
2	

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	76	02/02/2013
3	100	54	01/02/2013
2	200	56	02/01/2013

	Produit
	54
	76
	56

	Mois
	01/2013
	02/2013

	Année
	2013

Entrepôt (modèle logique ROLAP)

Démo fonctionnement 1 (ROLAP)

Données brutes

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	78VGVS	02/02/2013

1, 100a, 54, 01/02/2013; 2, 200, 56, 02/01/2013

1: Composant ETL (intégration et fiabilisation de données) – Staging

1

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
3	**100**	**54**	**01/02/2013**
2	200	56	02/01/2013

2: Composant ETL (alimentation de la table de faits/agrégation)

2

Mesure précalculée

Mesure

Entrepôt

ΣQuantité	ClientID
300	1
100	3
200	2

ClientID	ΣQuantité	Produit	Date	mois
1	100	54	01/01/2013	01/2013
1	200	76	02/02/2013	02/2013
3	100	54	01/02/2013	02/2013
2	200	56	02/01/2013	01/2013

Produit	ΣQuantité
54	200
76	200
56	200

Une ligne=Un fait, à savoir une *observation directe ou inférée (mesure)* sur le passé → table de faits

ΣQuantité	Mois
300	01/2013
100	02/2013

ΣQuantité	Année
600	2013

Modèle logico-physique dimensionnel ROLAP (type flocon de neige)

Dimension et hiérarchie

Démo fonctionnement 2 (ROLAP)

Données brutes

Non typé

1, 100a, 54, 01/02/2013; 2, 200, 56, 02/01/2013

correction/typage

Typé

Composant ETL (Données extraites) – Staging

1, 100, 54, 01/02/2013; 2, 200, 56, 02/01/2013

produit inexistant

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	78VGVS	02/02/2013

suppression

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013

normalisation

ClientID	Quantité	Produit	Date	ville
1	100	54	01/01/2013	Nantes
1	200	78	02/02/2013	Nantes

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	78	02/02/2013

ClientID	ville
1	Nantes

ClientID	Quantité	Produit	Date	Ville d'achat
1	100	54	01/01/2013	Nantes
1	200	78	02/02/2013	Nantes

Les propriétés de données au sein d'un entrepôt

- Données complètes
- Données fiables
- Données assemblées
- Données représentantes un historique étendu (masses de données)
- Données suffisamment actualisées (données fraiches)
- Données agrégées
- Données précalculées

Fiabilité de la décision : garantie par le processus ETL

Rapidité des requêtes : garantie par l'organisation de l'entrepôt

36

Exemple II (modèle logique)

Données brutes (SIO)

ClientID	Quantité	Produit	Date
1	100	54	01/01/2013
1	200	76	02/01/2013

1, 100a, 54, 01/02/2013, 10; 2, 100, 56, 02/01/2013 11AM, 5; 2, 100, 56, 02/01/2013 5PM, 6,

Composant ETL

ClientID	Quantité	Produit	Date	Prix
1	100	54	01/01/2013	Null
1	200	76	02/01/2013	Null
3	100	54	01/02/2013	10
2	100	56	02/01/2013 11AM	5
2	100	56	02/01/2013 5PM	6

Composant ETL

ClientID	
1	
3	
2	

ClientID	ΣQuantité	Produit	Mois
1	100	54	01/2013
3	100	54	02/2013
2	200	56	01/2013
1	200	76	01/2013

	Produit
	54
	76
	56

	Année
	2013

Entrepôt

Granularité des faits : mois, client, produit (au lieu de jour, client, produit)

Faits inférés !

Exemple III (modèle logique)

ClientID		
1		
3		
2		

ClientID	ΣQuantité	Produit	Mois	Σ(quantité*prix) as CA
1	100	54	01/2013	Null
3	100	54	02/2013	1000
2	200	56	01/2013	1100
1	200	76	01/2013	null

ΣQuantité		
200		
200		
200		

	Année	CA
	2013	null

Entrepôt : table de faits avec plusieurs faits

Exemple IV (modèle logique)

Table 1 - ventes

ClientID		
1		
3		
2		

ClientID	ΣQuantité	Produit	Mois	Σ(quantité*prix) as CA	Promo
1	100	54	01/2013	Null	Null
3	100	54	02/2013	1000	3
2	200	56	01/2013	1100	7
1	200	76	01/2013	null	null

Produit		
54		
56		
76		

Entrepôt : modèle logique en constellation

ΣQuantité	Année	CA
600	2013	null
....	2020	....

Promo
1
2
3
5
6
7
8
0

ID	Date
1	1/1/2020
2	2/1/2020
3	3/1/2020
5	4/1/2020
6	5/1/2020
7	6/1/2020
8	7/1/2020
9	8/1/2020

Produit	Promo	De	A
54	1	1	3
54	3	5	6
56	7	2	4
76	1	4	9

Table 2 - factless (produit-promo)

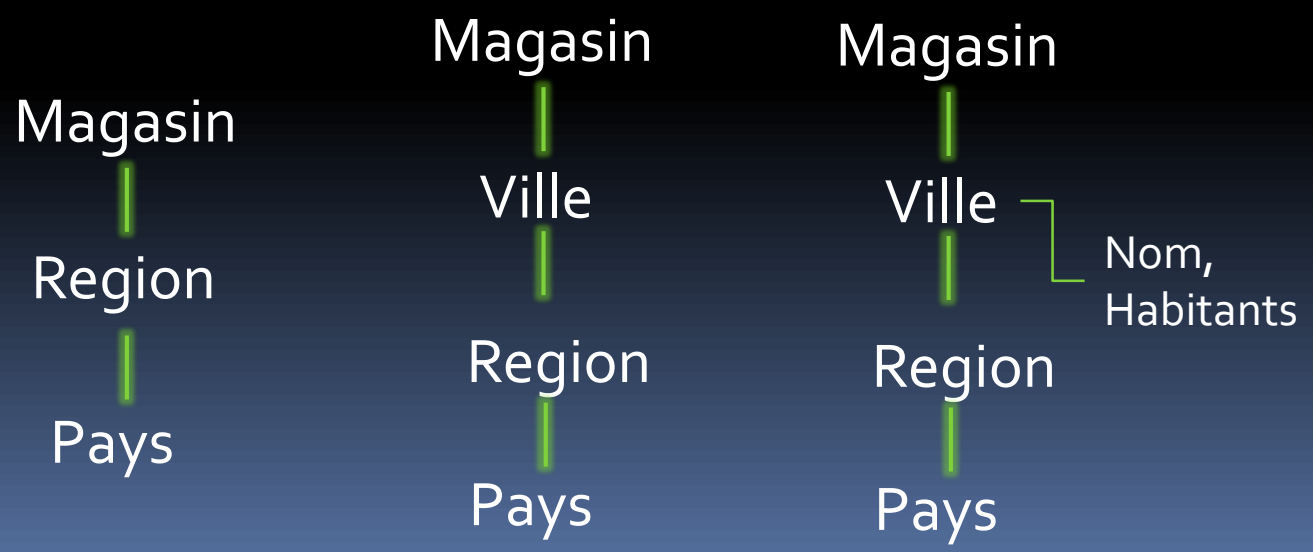
Principes de modélisation conceptuelle d’un entrepôt/magasin de données

- Modélisation de dimensions et hiérarchies
- Modélisation des mesures

Hiérarchies

- Chaque dimension est organisée en ***hiérarchie*** pour permettre des ***agrégations significatives*** de données ; une hiérarchie représente des ***niveaux de détail significatifs*** pour les données, ***du plus agrégé au moins agrégé***
- Chaque niveau pouvant présenter des ***informations additionnelles***
- Ces informations additionnelles peuvent être ***explicatives du phénomène sous-jacent***

Example de 2 hiérarchies distinctes pour une Dimension Magasin, et informations additionnelles



Hiérarchies

Client

Tranche d'âge

Client

Ville

Region

Pays

Ville

Nom,
Habitants

Region

Pays

Hiérarchies temporelles

- Une *dimension et une* ***hiérarchie temporelle*** sont généralement présentes

Exemples de hiérarchies typique pour une Dimension « Temps »

heure
jour
semaine

jour
mois
année

jour
semaine
jours ouvrables
semaine

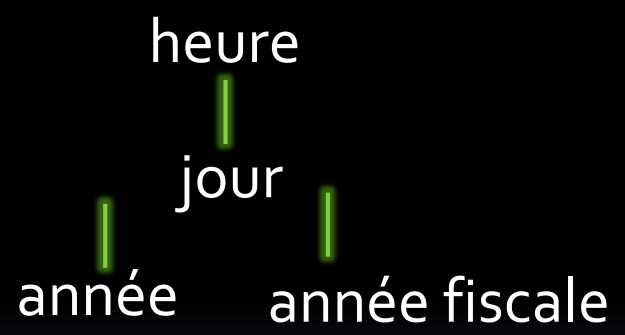
mois
trimestre
année

semaine
mois

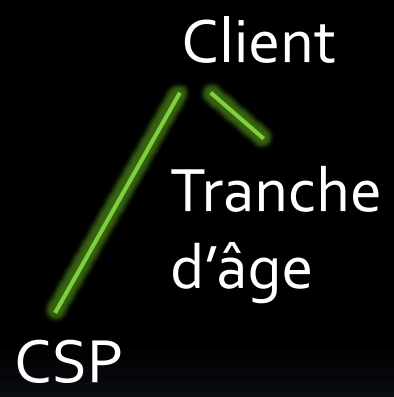
1 semaine → plusieurs mois

Dimensions et hiérarchies

- Pour une dimension, plusieurs hiérarchies peuvent être définies
- Exemples :



Définition : une zone de ventes regroupes plusieurs quartiers de plusieurs villes proches du magasin ; cependant une zone de vente est toujours dans la même région



Dimensions conformes

- Dimensions pouvant être utilisées pour plusieurs tables de faits (ou cubes) distinctes La dimension typique est la *dimension temporelle*, typiquement pré-réalisée dans les systèmes disponibles
- Ces dimensions contiennent tout détail possible, tout information additionnelle pouvant être utile

Dimensions changeantes

- Toute information dimensionnelle représentée par les dimensions pourrait se modifier dans le temps
  - Par exemple, un client change son adresse
- Il y a 2 types de dimension changeante :
  - Slowly changing dimension (SCD)
  - Rapidly changing dimension (RCD)
- Au niveau conceptuel, la modélisation approfondie de cette information n'est pas requise car il s'agit d'un ***constat pour la modélisation logique***

Mesures/table de faits

- Plusieurs mesures peuvent être regroupées dans une même table de faits pour un même ensemble de dimensions ; ces mesures permettent d'analyser, pour un même sujet, plusieurs faits
  - Pour un même client, on analyse les quantités achetées, le CA, le délai de livraison, les produits typiquement achetés
  - Même analyse pour une tranche d'âge de ces clients
- Mais il est tout à fait possible que certaines mesures ne soient par re-groupables dans une même table de faits

Objectif et définition d'une mesure

- L'objectif d'une mesure est de
  - Spécifier comment alimenter la table de faits à partir de données sources ou traitées par l'ETL donc de constituer les faits observés
  - D'établir (d'inférer) les faits à un (ou plusieurs) niveau(x) à partir des faits observés, si possible
- Plusieurs moyens d'inférence peuvent définir une mesure, selon les cas
  - Operateur d'agrégation SQL (+where éventuel)
  - Formule libre utilisant d'autres mesures
- **Mesure calculée** : seule la formule est donnée, nécessaire pour inférer les faits un (ou plusieurs) niveau(x) plus agrégé(s)
- **Mesure dérivée** : elle nécessite à la fois une formule et un operateur d'agrégation utilisé pour inférer les faits

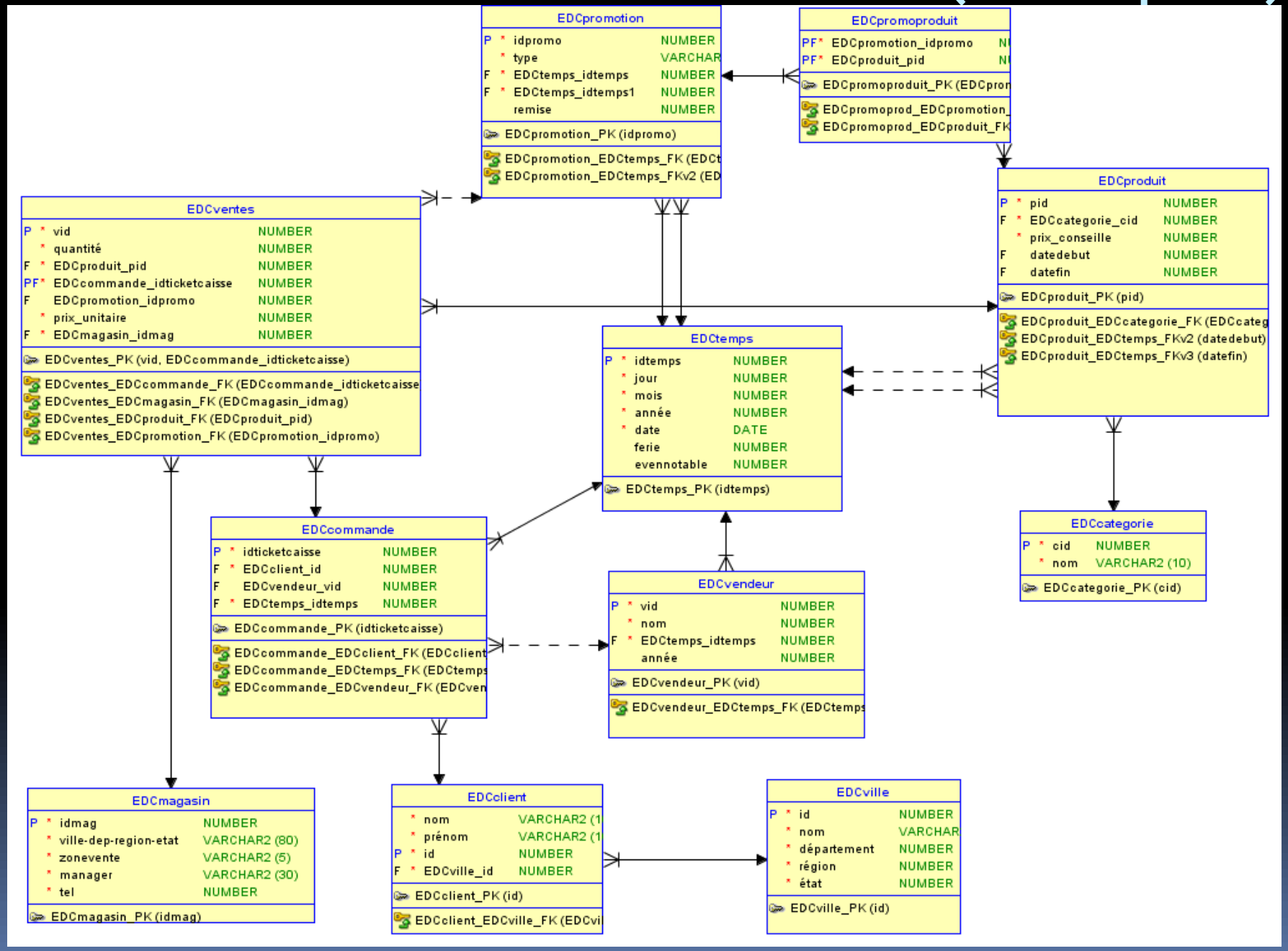
Objectif et définition d’une mesure

- Il est toute à fait possible qu’une mesure n’utilise pas le même operateur d’agrégation pour toutes les dimensions (une mesure → plusieurs operateurs)
- Le cas standard est celui d’une mesure et un seul operateur d’agrégation pour toute (ou certaines) dimension(s)(une mesure → un seul operateur)

Propriétés de mesures : additivité

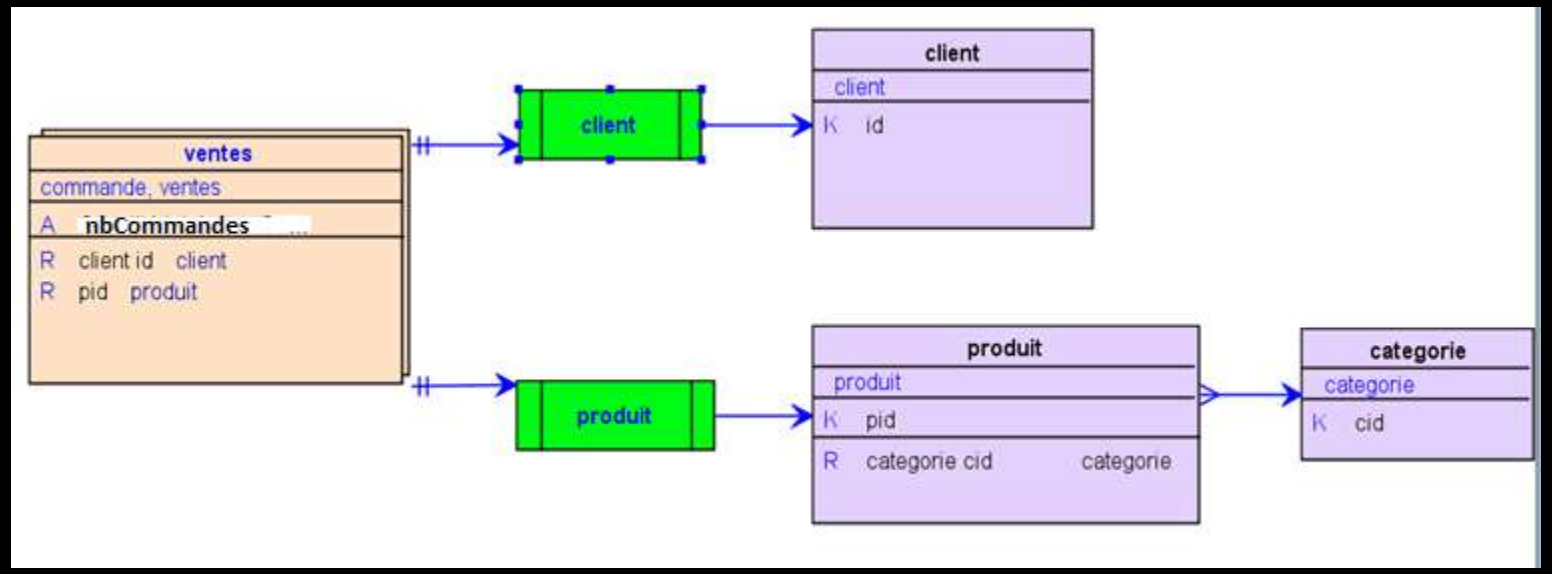
- Mesures additives
  - La ***somme* de la mesure à un niveau** permet d'obtenir la valeur correcte de la mesure à un niveau hiérarchique plus agrégé (et elle est significative), étant cette valeur correcte par rapport à celle que la mesure aurait utilisant les données disponibles (dans les sources et après traitement par l'ETL)
- Mesures semi-additives
  - La **somme de la mesure à un niveau** ne fournit la valeur correcte de la mesure à un niveau hiérarchique plus agrégé que pour certaines hiérarchies et dimensions (ou la somme n'est significative que pour certaines dimensions)
- Mesures non additives
  - La **somme de la mesure à un niveau** ne fournit jamais la valeur correcte de la mesure (ou la somme n'est jamais significative)

Mesures semi-additives (exemple)



Source ou données transformées (schéma) au sein de l'ETL

Mesures semi-additives (exemple)



Modèle dimensionnel

Extrait de la table de faits

PID	Client.ID	NbCommandes
1	1	10
1	2	3
2	1	6
2	2	9

Mesure pour nbCommandes :
Count(Distinct idticketcaisse)

Table de faits présentée sous forme de table pivot (agrégation par SUM)

Somme de NbCommandes	Client.ID		
PID	1	2	Total général
1	10	3	**13**
2	6	9	**15**
Total général	**16**	**12**	28

OK (car 1 commande 1 seul client)

NOK car 1 commande plusieurs produits

Propriétés de mesures : agrégabilité

- Une mesure est « agrégable » par rapport à une dimension, s'il existe un operateur d'agrégation (SQL) fournissant la valeur escomptée lorsqu'il est utilisé avec la dimension
- Une mesure est agrégable si elle est agrégable pour toute dimension
- Autrement la mesure n'est pas agrégable
- Si le mesure n'est pas agrégable, il peut être nécessaire de rajouter des agrégats lors de la modélisation physique

Operateurs vs propriétés

Operateur utilisé	Agrégabilité	Additivité	Calculable
SUM	OUI	OUI	NON
MAX/MIN	OUI	NON	NON
AVG	OUI/NON	NON	OUI (SUM/COUNT)
COUNT	NON	OUI/NON	NON
COUNT(DISTINCT)	NON	NON/OUI	NON

Modèles logiques (choix de base)

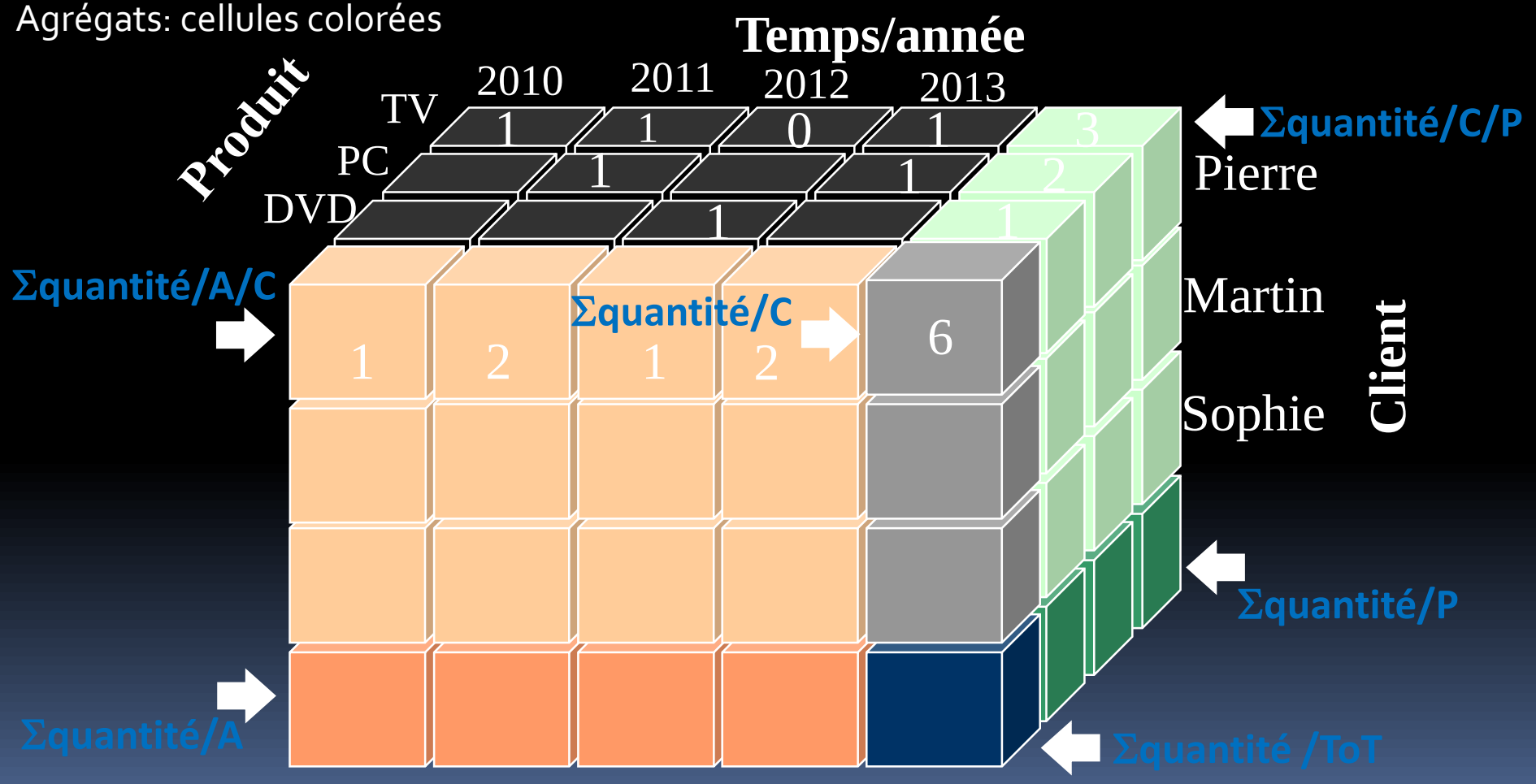
- ROLAP : Relational OLAP (souvent utilisé pour l'entrepôt pour sa capacité à gérer une grande quantité de données garantissant de performances acceptables)
- MOLAP : Multidimensional OLAP (souvent utilisé pour optimiser l'accès au données stockées dans l'entrepôt pour la capacité à exécuter des requêtes très complexes mais limité en quantité de données ; peut être utilisé par les data-marts/serveur olap)
  - → il n'y a pas de véritable modélisation logique !
- HOLAP : Hybrid OLAP (un mix entre ROLAP et MOLAP)

OLAP : OnLine Analytical Processing

Cuboïde

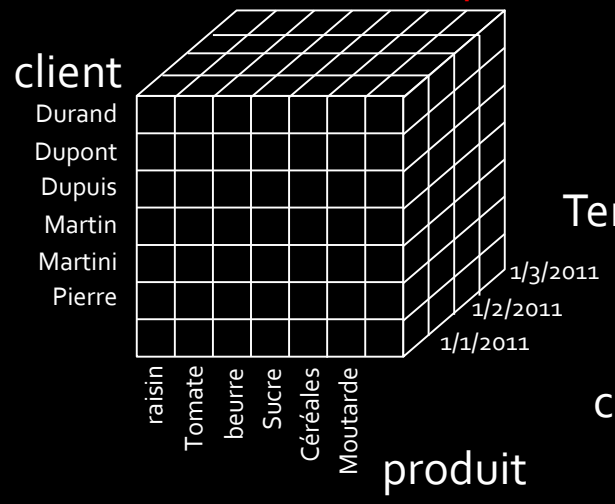
Cuboïde : cellules en blanc (produit,année,client)

Agrégats: cellules colorées

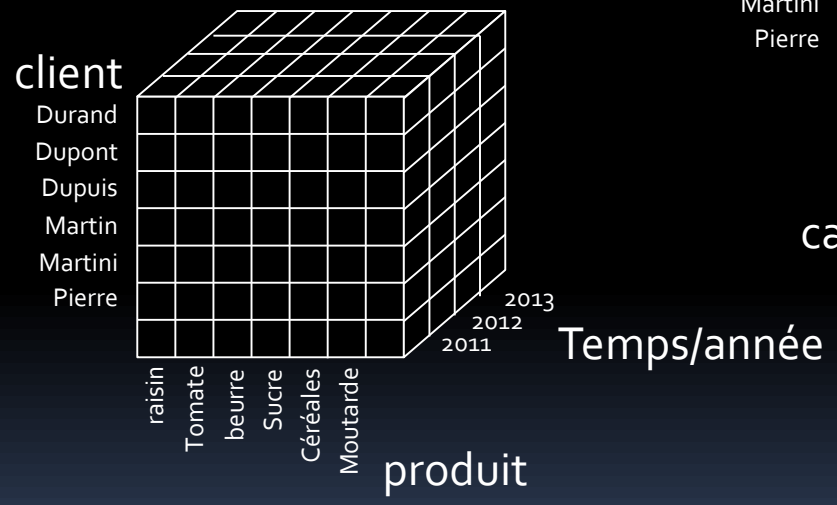


MOLAP (cuboïdes possibles)

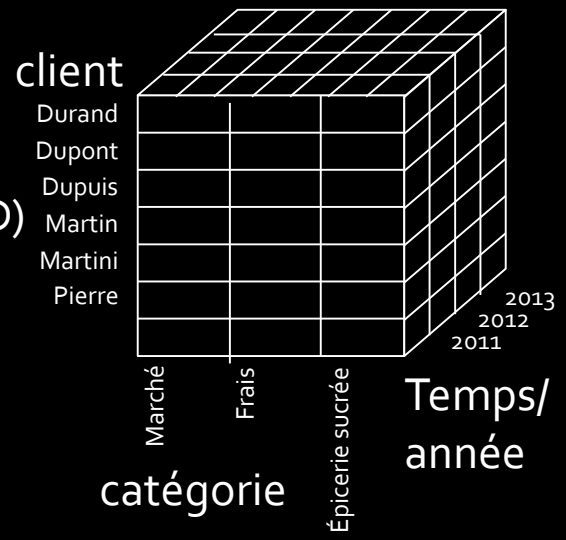
Cuboïde - Faits : client,produit,date (3D)



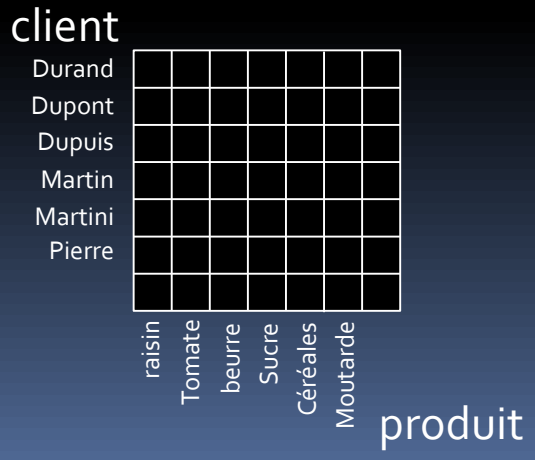
Cuboïde : client,produit,année (3D)

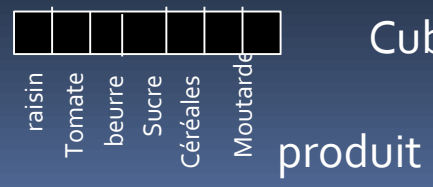


Cuboïde : client,catégorie,année (3D)



Cuboïde : client,produit (2D)





Cuboïde : produit (1D)

Modélisation logique (ROLAP)

- Table de faits
- Schéma flocon vs schéma étoile
- Dimensions changeantes
- Clés

Table de faits (contenu)

ΣQuantité
1250
100
200

ClientID	
1	
3	
2	

ClientID	ΣQuantité	Produit	Date
1	900	54	/01/2013
1	200	76	/02/2013
3	100	54	/02/2013
2	200	56	/01/2013
1	150	54	/02/2013

	Produit
	54
	76
	56

ΣQuantité
1150
200
200

Flocon de neige (snowflake) : 1 niveau = 1 table (normalisation)

ΣQuantité
1100
450

	Mois
	01/2013
	02/2013

ΣQuantité
1550

	Année
	2013

Modèle logique dimensionnel ROLAP (agrégats en vert pour info)

ΣQuantité
1250
100
200

ClientID	
1	
3	
2	

ClientIID	ΣQuantité	Produit	Date
1	900	54	1
1	200	76	2
3	100	54	2
2	200	56	1
1	1050	54	3
1	200	76	3
3	100	54	3
2	200	56	3

	Produit
	54
	76
	56

ΣQuantité
1150
200
200

Etoile (star) : une seule table par dimension générant des tables dénormalisées (rappel : peu ou aucune modification de données)

key	Mois	année	
1	01/2013	2013	
2	02/2013	2013	
3	NULL	2013	

ΣQuantité
1100
450
1550

Modèle logique-physique dimensionnel ROLAP (agrégats en vert pour info)

Table de faits (structure)

- La clé de la table de faits est établie en fonction de la granularité de faits à atteindre
  - Granularité transactionnelle (les faits correspondent à ce que l'on trouve dans les sources)
  - Granularité temporelle (les faits correspondent à une agrégation de ce que l'on trouve dans les sources)
- La table de faits contient les clés étrangères pouvant faire référence au 1er niveau de toute dimension
- La table de fait contient des colonnes additionnelles correspondantes aux mesures définies (autant de colonnes que de mesures)

Types de schéma logique en ROLAP

- Schéma Etoile
  (star schema)
- Schéma Flocon de neige
  (snowflake schema)

Comme schéma normalisé ou denormalisé au sein du modèle relationnel

Dimensions changeantes (ROLAP)

- Les informations accessibles via les dimensions (informations dimensionnelles) peuvent évoluer dans le temps ; le problème est comment prendre en compte ces évolutions
- Trois solutions classiques pour le « slowly changing dimensions » :
  - Ecraser l'information ancienne (update) ; Type 1
  - Insérer la nouvelle information utilisant - par exemple la même clé définie dans le SIO pour l'ancienne information (insert) ; Type 2
  - Rajouter des colonnes pour que l'ancienne information et la nouvelle puisse coexister (update) ; Type 3
- Solutions spécifiques pour le « rapidly changing dimensions »

Modélisation SCD

Clé SIO/clé ETL

Table d'origine

ID	Customer	Country
1	Bob	United Kingdom

Type 1 (update)

ID	Customer	Country
1	Bob	United States

Dates de validité/données courantes

Type 2 (insert)

Surrogate Key	ID	Customer	Country	Current Flag	Record Start Date	Record End Date
1	1	Bob	United Kingdom	0	01/01/2000	12/03/2014
2	1	Bob	United States	1	12/03/2014	NULL

Surrogate key (utilisé dans la table de faits en remplacement de la clé SIO/ETL)

Clé SIO/clé ETL

Nouvelle colonne d'historique

Type 3 (update)

ID	Customer	Country	Previous Country
1	Bob	United States	United Kingdom

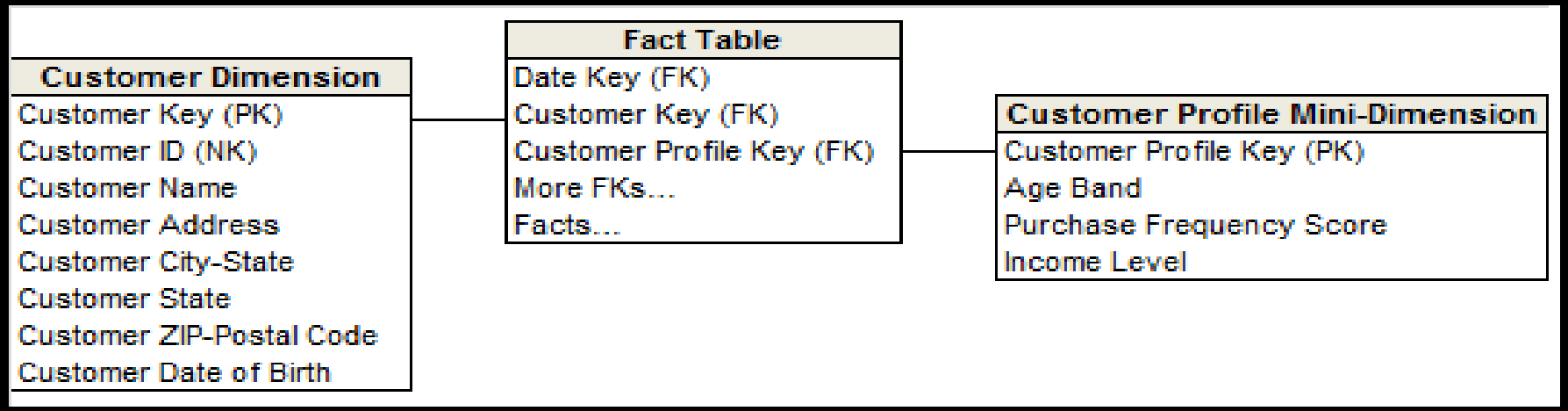
Type 6 = Types 1+2+3

Surrogate Key	ID	Customer	Current Country	Historical Country	Current Flag	Record Start Date	Record End Date
1	1	Bob	United States	United Kingdom	0	01/01/2000	12/03/2014
2	1	Bob	United States	United States	1	12/03/2014	NULL

Modélisation RCD

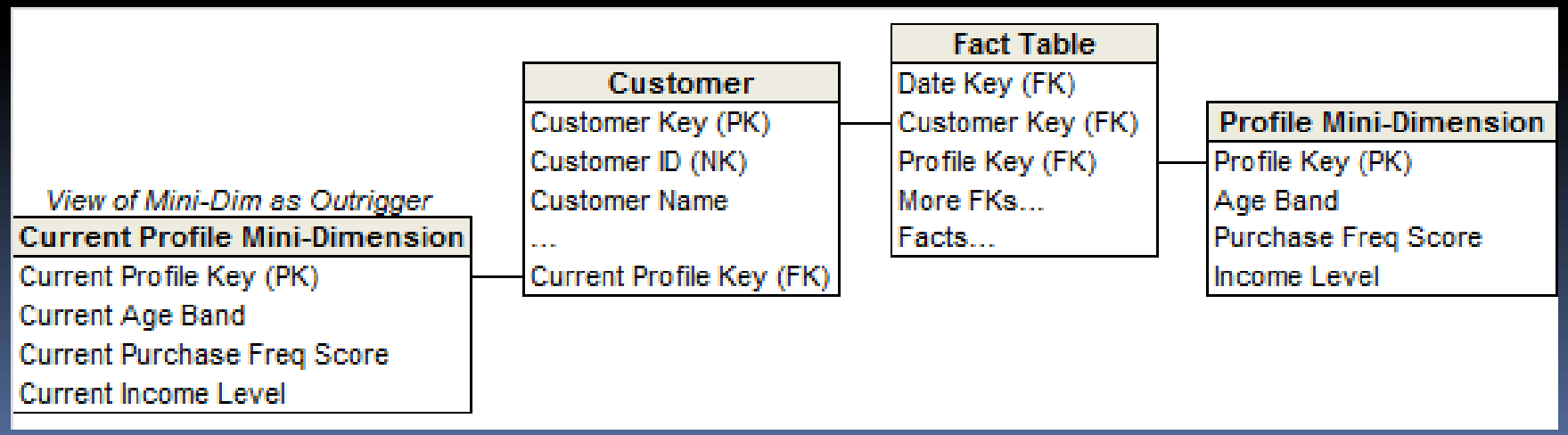
Type 4 (minidimension)

Segmentation



Type 5 (minidimension + current values)

Segmentation



Dimensions changeantes : synthèse

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change to attribute value	Facts associated with attribute's original value
Type 1	Overwrite attribute value	Facts associated with attribute's current value
Type 2	Add new dimension row for profile with new attribute value	Facts associated with attribute value in effect when fact occurred
Type 3	Add new column to preserve attribute's current and prior values	Facts associated with both current and prior attribute alternative values
Type 4	Add mini-dimension table containing rapidly changing attributes	Facts associated with rapidly changing attributes in effect when fact occurred
Type 5	Add type 4 mini-dimension, along with overwritten type 1 mini-dimension key in base dimension	Facts associated with rapidly changing attributes in effect when fact occurred, plus current rapidly changing attribute values
Type 6	Add type 1 overwritten attributes to type 2 dimension row, and overwrite all prior dimension rows	Facts associated with attribute value in effect when fact occurred, plus current values
Type 7	Add type 2 dimension row with new attribute value, plus view limited to current rows and/or attribute values	Facts associated with attribute value in effect when fact occurred, plus current values

Clés (ROLAP)

- Comme les informations stockées dans un entrepôt/data-mart proviennent principalement du SIO, ces informations possèdent naturellement des clés (définies au sein du SIO)
- Pour rendre le ***SID indépendant du SIO, modulaire, et performant***, il convient de redéfinir des nouvelles clés dans toutes les tables réalisant l'entrepôt/data mart, sans signification particulière, au sein du SID
  - Ces clés sans signification, nommées « surrogate key », purement numériques (garantissant plus de performance, par exemple en cas de jointure ou pour construire les index, de liberté dans l'établir la granularité de faits, modularisant la réalisation de dimensions changeantes)
  - Toute clé définie au sein du SIO peut être également stockée dans l'entrepôt/data-mart comme simple colonne (si nécessaire car l'ETL/Vues matérialisées contiennent de mécanismes automatiques pour par exemple faire des mises à jour – voir modèle physique)

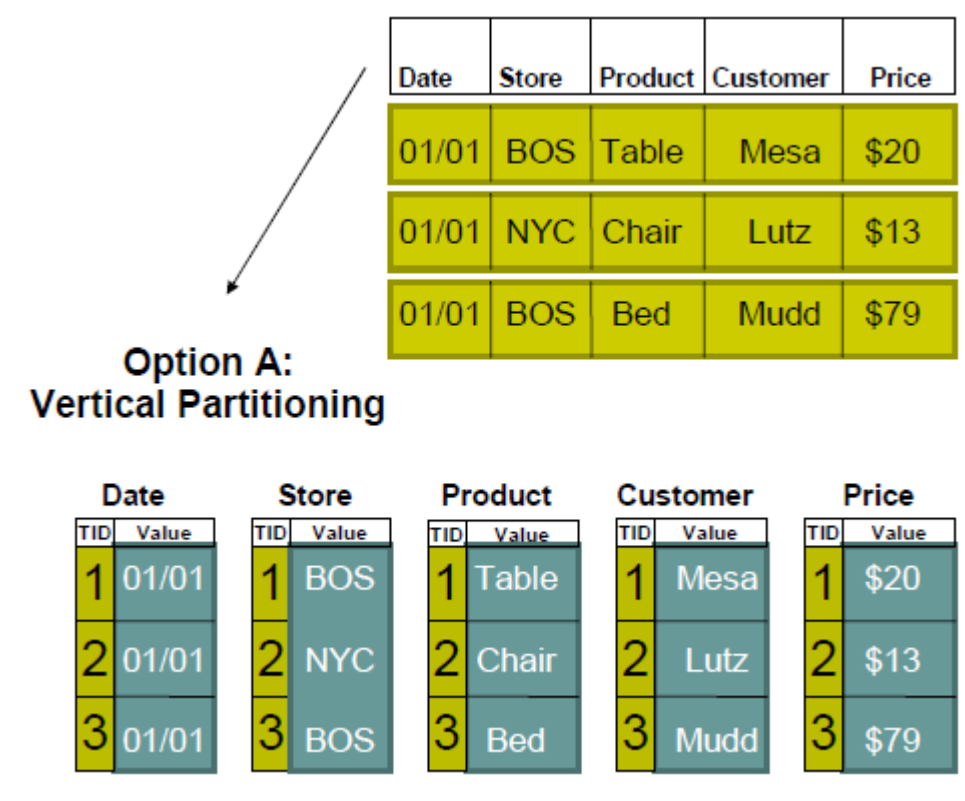
Clés de la table de faits

- La clé de la table de faits détermine la granularité de faits
- Granularité transactionnelle
  - la clé remplace les clés du SIO permettant de stocker directement les faits observés
  - Avantage : l'entrepôt représente un maximum d'information ; l'obtention de faits inférés peut être simplifiée
  - Désavantage : l'espace de stockage devient très important pouvant affecter les performances de requêtes en absence de mécanismes d'optimisation
- Granularité temporelle
  - La clé ne permet que de stocker de faits inférés compte tenu des dimensions
  - Avantage : espace de stockage réduit permettant plus de requêtes complexes et donc d'analyses ;
  - Désavantage : il y a une perte d'information et donc certains analyses pourraient ne plus être disponibles (mais d'autres les deviendraient) ; les inférences de faits pourraient être limitées, obligeant l'utilisation d'agrégats explicites

Modélisation physique (ROLAP)

- Stockage
- Vues matérialisées (en fonction de la plateforme)
  - Agrégats
  - Table de faits (alternative à l'ETL)
- Index
- Partitionnement

Modèle physique (ROLAP) : Stockage par ligne vs stockage par colonne



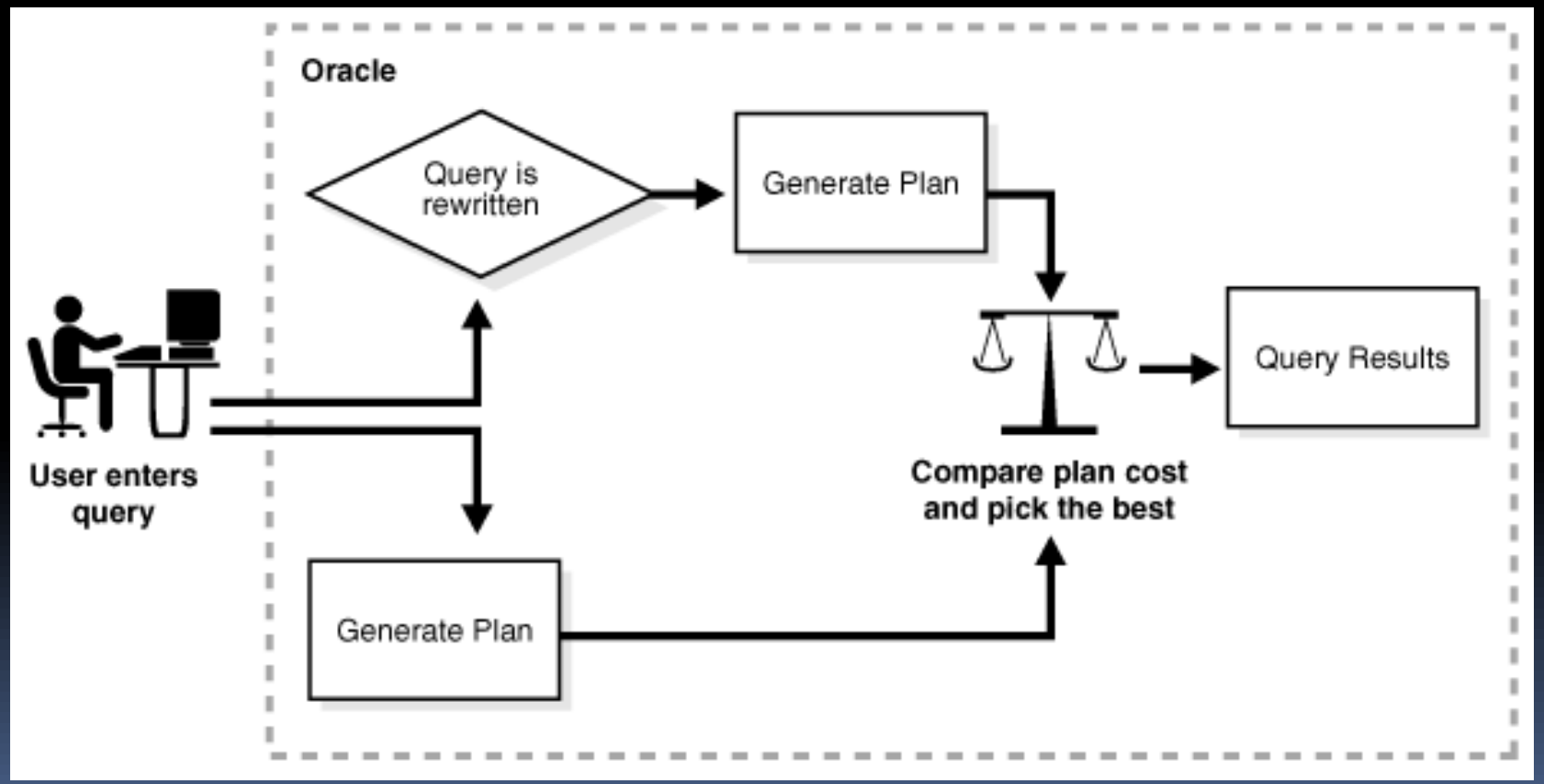
Exemple : ORACLE InMemory (utilisation du même RowID pour les projections)
CREATE TABLE purchases (date, store, product, customer, price) INMEMORY

Réécriture

- Rappel du processus d'exécution d'une requête (ORACLE)

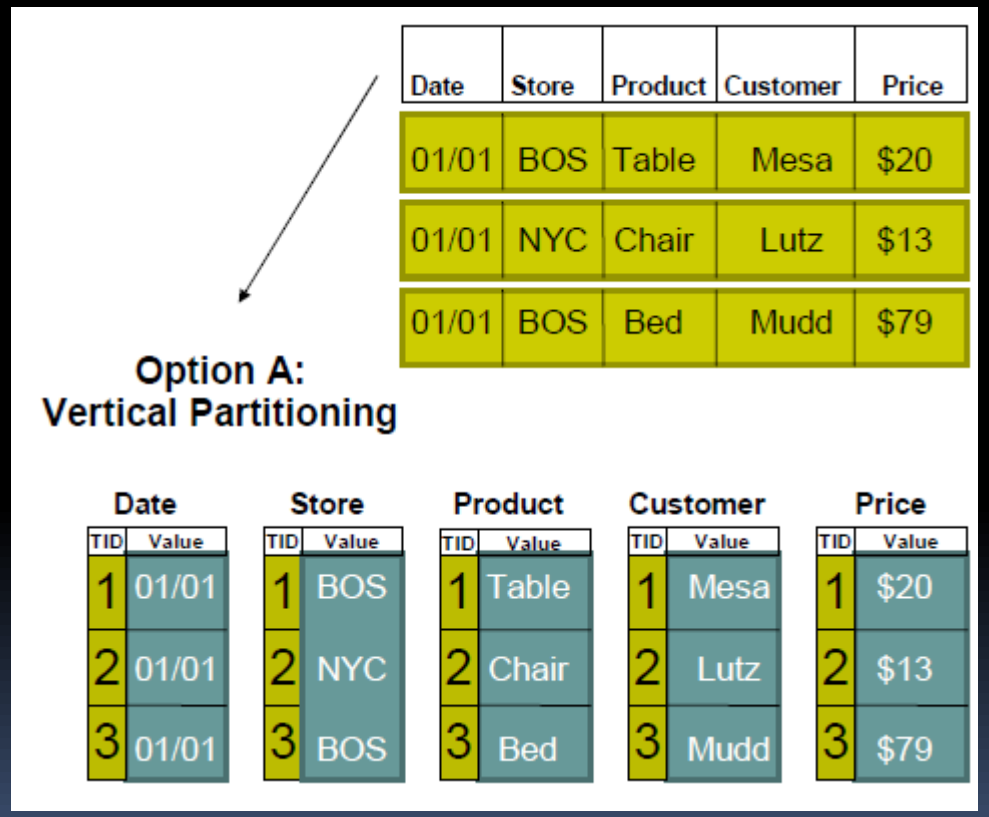
La requête saisie par l'utilisateur est réécrite automatiquement suivant la requête elle-même et les informations dont ORACLE dispose

Les plans d'exécutions utilisent les informations dont ORACLE dispose



Le plan d'exécution de la requête est celui de base sans utiliser aucune information

Modèle physique (ROLAP) : Stockage par ligne vs stockage par colonne – réécriture



Select A.a, B.b where B.b=x

Équivalent à (réécriture)

Select A.a, B.b from (Select B.b, B.id from B where B.b=x), A where A.id=B.id

Avantages de l'organisation par colonnes

- **Columnar databases have higher read efficiency.** If you're running a query like "give me the average price of all transactions over the past 5 years", a relational database would have to load all the rows from the previous 5 years even though it merely wants to aggregate the price field; a columnar database would only have to examine one column — the price column. This means that a columnar database only has to sift through a fraction of the total dataset size.
- **Columnar databases also compress better than row-based relational databases.** It turns out that when you're storing similar pieces of data together, you can compress it far better than if you're storing very different pieces of information. (In information theory, this is what is known as 'low entropy'). As a reminder, columnar databases store *columns* of data — meaning values with identical types and similar values. This is far easier to compress compared to row data, even if it comes at the cost of some compute (for decompression during certain operations) when you're reading values. But overall, this compression means more data may be loaded into memory when you're running an aggregation query, which in turn results in faster overall queries.
- The final benefit is that compression and dense-packing in columnar databases free up space — space that may be used to sort and index data within the columns. In other words, **columnar databases have higher sorting and indexing efficiency**, which comes more as a side benefit of having some leftover space from strong compression. It is also, in fact, mutually beneficial: researchers who study columnar databases point out that sorted data compress better than unsorted data, because sorting lowers entropy.
- When using column store in ORACLE : https://www.oracle.com/technetwork/database/in-memory/overview/twp-dbim-usage-2441076.html

Table de faits : Calcul du stockage (par ligne)

Calculate the number of rows inside each of the dimensions:

Time dimension: 4 rows
Date dimension: 365 rows for 1 year
Product dimension: 100 rows (100 products)
Store dimensions: 2 rows (2 stores)
Customer dimension: 1000000 customers
Supplier dimension: 50 suppliers
Employee dimension: 10 employees

Calculate the base level of fact records by multiplying together the number of rows for each dimension. Use the numbers that you gathered in the previous step:

4 * 365 * 100 * 2 * 1000000 * 50 * 10 = 730000000 rows (This number might be larger than you expect. The number only applies if every product is sold in every store by every employee to every customer)

Calculate the maximum fact table size growth:

Number of foreign keys: 8
Number of degenerate dimensions (not a foreign key): 1
Number of measures: 8

Assume that the fact table takes 4 bytes for an INTEGER column, and calculate the size of a single row:

(8 + 1 + 8) * 4 bytes = 68 bytes

Calculate the maximum data growth for a single year for the fact table: 730000000 rows * 68 bytes = 45 GB

Table de faits : Calcul du stockage (par ligne)

E.g. A data warehouse will store facts about the help provided by a company's product support representatives. The fact table is made of up of a composite key of 7 indexes (int data type) including the primary key. The fact table also contains 1 measure of time (datetime data type) and another measure of duration (int data type). 2000 product incidents are recorded each hour in a relational database. A typical work day is 8 hours and support is provided for every day in the year. <u>What will be approximate size of this data warehouse in 5 years?</u>

First calculate the approximate size of a row in bytes (int data type = 4 bytes, datetime data type = 8 bytes):

Size of a row = size of all composite indexes (add the size of all indexes) + size of all measures (add the size of all measures).

Size of a row (bytes) = (4 * 7) + (8 + 4).
Size of a row (bytes) = 40 bytes.

Number of rows in fact table = (2000 product incidents per hour) * (8 Hours) * (365 days in a year). → granularity per incidents
----Number of rows in fact table = (8 Hours) * (365 days in a year). → granularity per hour

Number of rows in fact table = 2000 * 8 * 365
Number of rows in fact table = 5840000

Size of fact table (1 year) = (Number of rows in fact table) * (Size of a row) i.e.:

Size of fact table (bytes per year) = 5840000 * 40
Size of fact table (bytes per year) = 233600000 bytes.
Size of fact table (megabytes per year) = 233600000 / (1024*1024)
Size of fact table (in megabytes for 5 years) =
(23360000 * 5) / (1024 *1024)= 1113.89 MB
Size of fact table (gigabytes) = 1113.89 / 1024=1.089 GB

Modèle physique (ROLAP) : Vues Matérialisées (Agrégats)

- Mécanisme incontournable des systèmes ROLAP
  - Nécessaire pour enregistrer les calculs intermédiaires et les réutiliser (« query rewrite »)
  - A prendre en compte en ***modélisation physique***

VUE1

ClientID	∑Quantité
1	300
3	100
2	200

Faits

ClientID	∑Quantité	ProduitID	Date
1	100	54	/01/2013
1	200	76	/02/2013
3	100	54	/02/2013
2	200	56	/01/2013

VUE4

∑Quantité	ProduitID
200	54
200	76
200	56

VUE2

∑Quantité	Mois
300	01/2013
300	02/2013

∑Quantité	Année
600	2013

VUE3 (sur la table de faits)

ProduitID	Mois	∑Quantité
54	01/2013	100
54	02/2013	100
76	02/2013	200
56	01/2013	200

Select produitID, Year(Date), sum(∑quantité) **from Faits** group by produitID, Year(Date)
→ Select produitID, Year(mois), sum(∑quantité) **from VUE2** group by Year(mois)
(réécriture grâce à additivité + hiérarchie)

Exemples de vues matérialisées (ORACLE)

```
CREATE MATERIALIZED VIEW MatView1 REFRESH COMPLETE ON COMMIT ENABLE QUERY REWRITE
AS
 SELECT clientid, nom,
  sum(quantité)
 FROM commande,
  ventes, client
 WHERE commande.cid=ventes.commande_cid and commande.clientid=client.id
 GROUP BY client.id, nom ;
```

```
CREATE MATERIALIZED VIEW MatView1 REFRESH COMPLETE
START WITH ROUND(sysdate)+11/24
NEXT ROUND(sysdate)+7+11/24 ENABLE QUERY REWRITE
AS
 SELECT clientid, nom,
  SUM(quantité)
 FROM commande,
  ventes, client
 WHERE commande.cid=ventes.commande_cid and commande.clientid=client.id
 GROUP BY client.id, nom ;
```

Exemple : à partir du prochain lundi →

NEXT NEXT_DAY(TRUNC(SYSDATE), 'MONDAY'))

LOG d'une vue matérialisée

- Journalisation (fichier LOG) des modifications d'une table, essentielle pour le « refresh fast »
- Exemple d'ORACLE :
  - permet de décider quelle information mettre dans la journalisation
  - Par défaut, on peut tout mettre, indiquant explicitement les données utilisées par les requêtes dans les vues ; exemples :
    - create materialized view log on ventes with rowid, primary key, sequence (quantité) including new values;
    - create materialized view log on commande with rowid, primary key, sequence (clientid) including new values;
    - create materialized view log on client with rowid, primary key, sequence (nom) including new values;

Exemples avec LOG (ORACLE)

```
CREATE MATERIALIZED VIEW MatView1 REFRESH COMPLETE ON COMMIT ENABLE QUERY REWRITE
AS
SELECT clientid,nom,
sum(quantité),
FROM commande,
Ventes, client
WHERE commande.cid=ventes.commande_cid and commande.clientid=client.id
GROUP BY client.id , nom;

CREATE MATERIALIZED VIEW LOG ON ventes....;
CREATE MATERIALIZED VIEW LOG ON commande....;
CREATE MATERIALIZED VIEW LOG ON client....;
CREATE MATERIALIZED VIEW MatView1 REFRESH FAST ON COMMIT ENABLE QUERY REWRITE
AS
SELECT clientid,nom,
sum(quantité)
FROM commande,
Ventes, client
WHERE commande.cid=ventes.commande_cid and commande.clientid=client.id
GROUP BY client.id, nom;
```

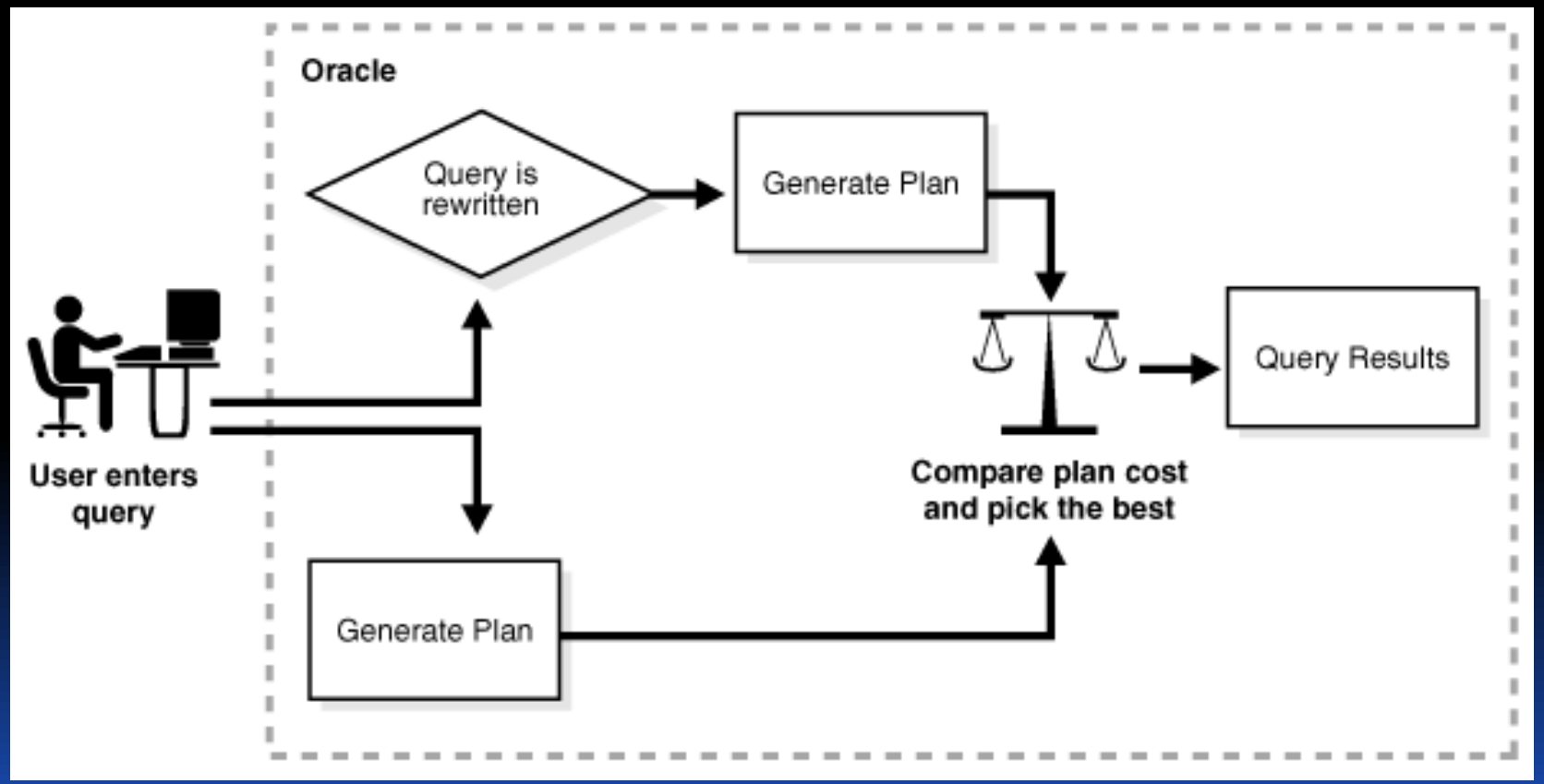
Journalisation (LOG) pour le « refresh fast »

Réécriture

- Rappel du processus d'exécution d'une requête (ORACLE)

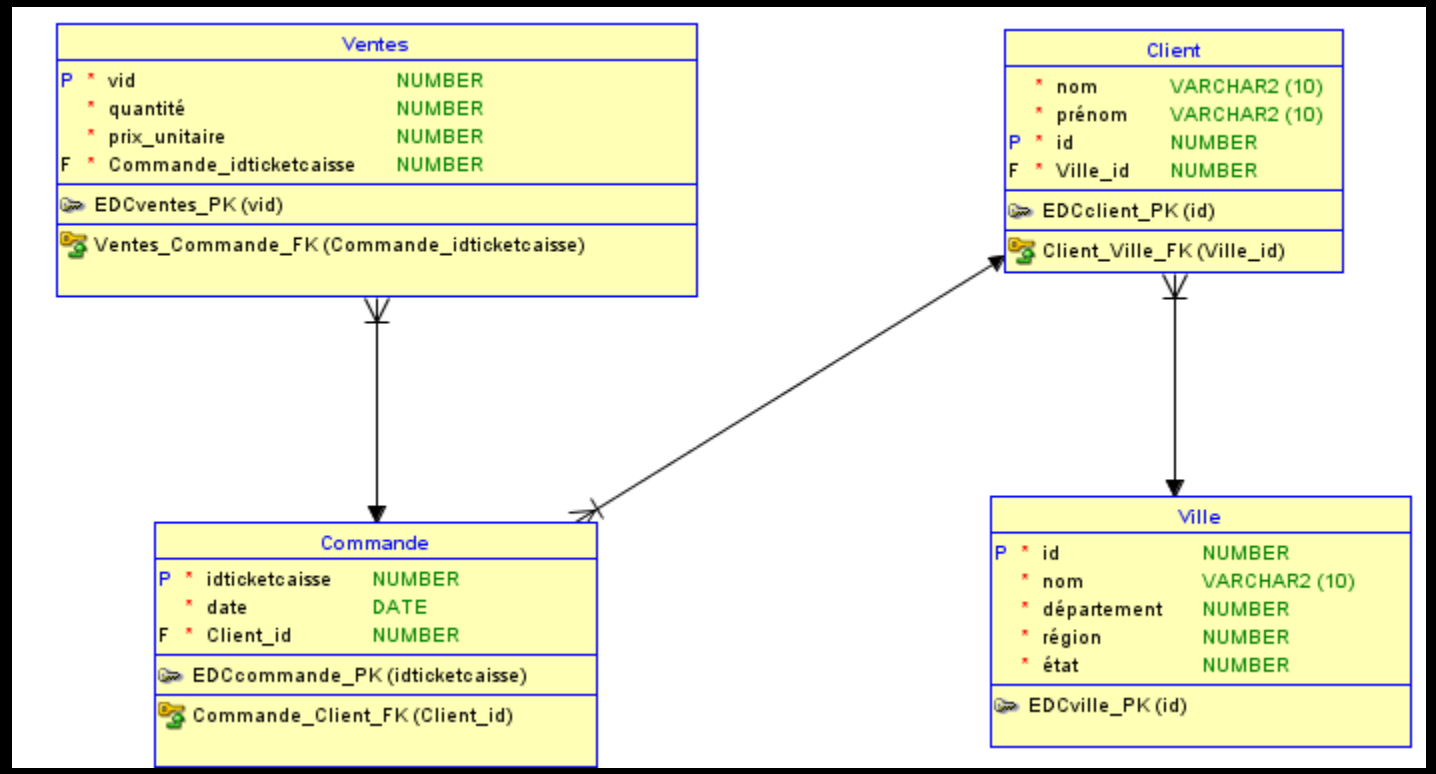
La requête saisie par l'utilisateur est réécrite automatiquement suivant la requête elle-même et les informations dont ORACLE dispose

Les plans d'exécutions utilisent les informations dont ORACLE dispose



Le plan d'exécution de la requête est celui de base sans utiliser aucune information

Exemple: réécriture par vue matérialisée



Information importante :
1 client → 1 seule ville

```
create materialized view MatView enable
query rewrite refresh complete on commit as
SELECT client.id as id
   SUM(quantité) as sommeqt
  FROM commande,
   ventes
  WHERE
commande.cid=ventes.commande_cid
  GROUP BY clientid ;
```

Exemple réécriture

Requête écrite par l'utilisateur

```
SELECT villeid,
  SUM(quantité)
 FROM commande,
  ventes, client
 WHERE
commande.cid=ventes.commande_cid
and commande.clientid=client.id
 GROUP BY villeid ;
```

Requête équivalente (en fonction de l'operateur et de données)

```
SELECT Y.villeid,
  SUM(X.sommeqt)
 FROM (SELECT client.id as id,
            SUM(quantité) as sommeqt
           FROM commande,
           ventes
           WHERE commande.cid=ventes.commande_cid
           GROUP BY client.id) X,
Client Y
 WHERE X.id=Y.id
 GROUP BY Y.villeid ;
```

Requête réécrite (visible avec le plan d'exécution)

```
SELECT Y.villeid,
  SUM(X.sommeqt)
 FROM MatView X,
Client Y
 WHERE X.id=Y.id
 GROUP BY Y.villeid ;
```

Vues matérialisées dans l'entrepôt

VUE1

ClientID	ΣQuantité
1	300
3	100
2	200

Faits

ClientID	ΣQuantité	ProduitID	Date
1	100	54	01/01/2013
1	200	76	07/02/2013
3	100	54	19/02/2013
2	200	56	09/01/2013

VUE4

ΣQuantité	ProduitID
200	54
200	76
200	56

VUE2

ΣQuantité	Mois
300	01/2013
300	02/2013

ΣQuantité	Année
600	2013

VUE3 (sur la table de faits)

ProduitID	Mois	ΣQuantité
54	01/2013	100
54	02/2013	100
76	02/2013	200
56	01/2013	200

```
create materialized view VUE1 enable
query rewrite refresh complete on
commit as SELECT client.id as id
   SUM(quantité) as sommeqt
 FROM  FAITS
GROUP BY clientid ;
```

```
create materialized view VUE2 enable
query rewrite refresh complete on
commit as SELECT month(Date),
Year(Date),  SUM(quantité) as sommeqt
 FROM  FAITS
GROUP BY month(Date), year(Date) ;
```

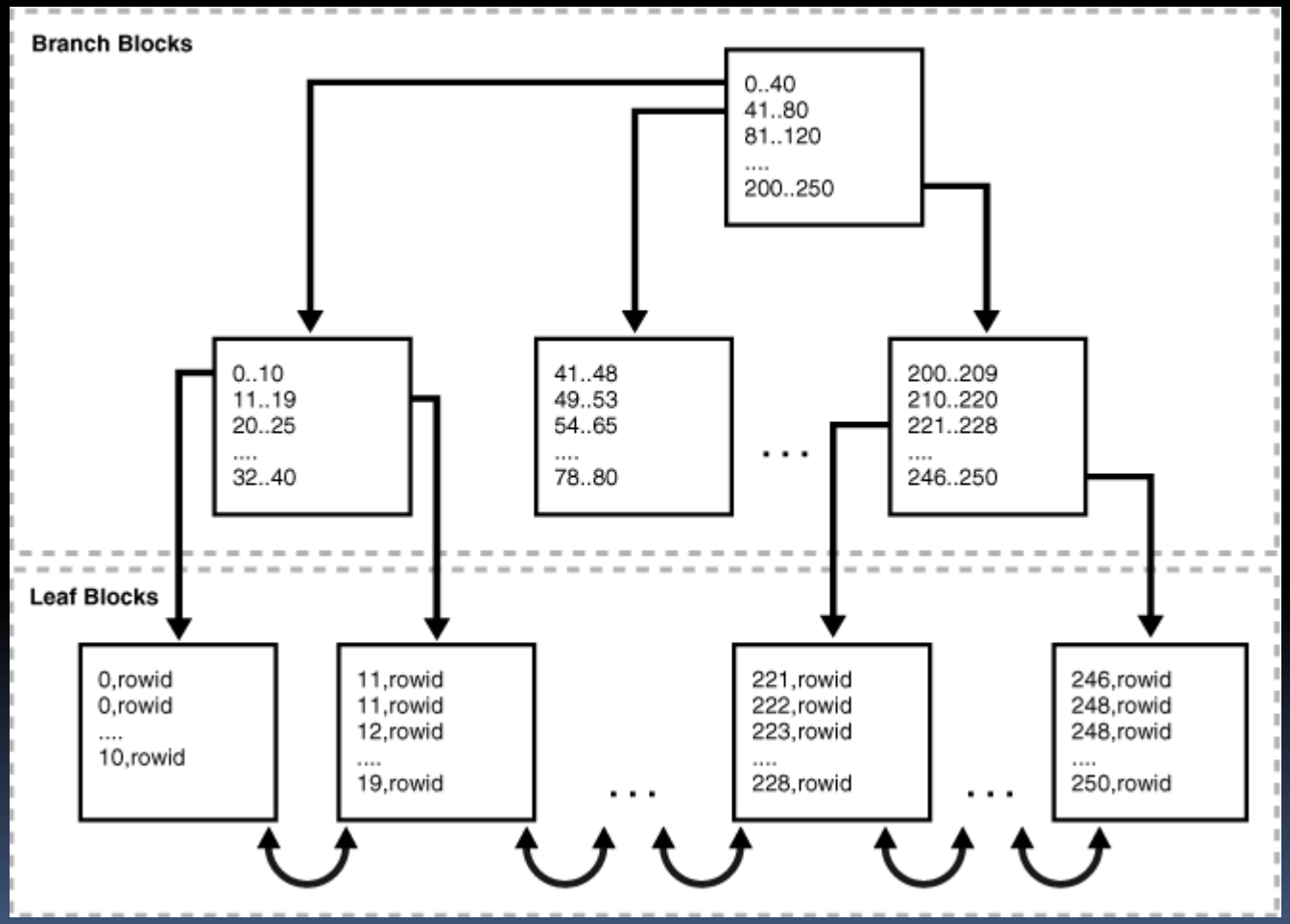
Vues matérialisées et table de faits

- Dans certains cas (simple) la table de faits peut être alimentée en données par la définition d'une vue matérialisées (au lieu d'un ETL)
- Cela a certaines implications sur la table de faits et sur la vue à cause des conditions strictes de fonctionnement des vues :
  - Indisponibilité éventuelle de certains operateur SQL dans les vues (par exemple OP(Distinct) pour ORACLE)
  - Table de faits, données nécessaires, et vues dans le même espace de stockage

Modèle physique (ROLAP) : indexes et partitionnement

- Optimisation des opérations group-by et jointure par des indexes spécifiques
  - Index bitmap
  - Index de jointure (join index)
- Parallélisation des opérations de manipulation de données
  - Partitionnement des tables

Principe des index



Index

RowId blocks

Index Bitmap

RowID	…	Region
1		East
2		North
3		South
4		East
5		North

```
SELECT customer_name
FROM CUSTOMER WHERE
 region = 'East'
 AND
 type = 'Partnership'
```

Select WHERE region='East' Rows 4 and 1
Select WHERE type='Partnership': [redacted]
Find matching rows: Row 4

Rows 5 and 4

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

Type	Row 1	Row 2	Row 3	Row 4	Row 5
Corporation	1	0	0	0	0
Company	0	0	1	0	0
Partnership	0	0	0	1	1
Other	0	0	0	0	0

Oracle then sorts these two lists of ROWID's and returns rows where the ROWID appears in both ROWID lists.

Opérations sur les index

Combining Bitmap Indexes

Due to fast bit-and, bit-minus, and bit-or operations, bitmap indexes are efficient:

- When using `IN (value_list)`
- When predicates are combined with `AND/OR`

A	B	A and B	A or B	not A
1	1	1	1	0
0	1	0	1	1
1	0	0	1	0
0	0	0	0	1
0	1	0	1	1

ORACLE

12-8

Copyright © 2005, Oracle. All rights reserved.

Index de jointure (Join index)

- Il s'agit d'un index fournissant directement les enregistrement d'une table satisfaisants une condition de jointure
- Il s'agit d'un index bitmap
- Il est typiquement utilisé pour calculer les jointures entre table de faits et dimensions dans un schéma étoile

```
CREATE BITMAP INDEX sales_idx
ON sales (employees.town) FROM
employees, sales WHERE
employees.id = sales.employeeid;
```

Sales.rowid	Employees.town
AAAQNKAAFAAAABSAAL	Vannes
AABQNHAFAAAABSAAL	Brest
BCAQNKAAFAAAABTAAL	Nantes

	AAAQNKAAFAAAABSAAL	AABQNHAFAAAABSAAL	BCAQNKAAFAAAABTAAL	…
Vannes	1	0	0	
Brest	0	1	0	
Nantes	0	0	1	

Star transformation

```
SELECT ch.channel_class, c.cust_city,
t.calendar_quarter_desc, SUM(s.amount_sold)
sales_amount FROM sales s, times t,
customers c, channels ch
WHERE s.time_id = t.time_id AND s.cust_id =
c.cust_id AND s.channel_id = ch.channel_id
AND c.cust_state_province = 'CA' AND
ch.channel_desc in ('Internet','Catalog') AND
t.calendar_quarter_desc IN ('1999-Q1','1999-
Q2')
GROUP BY ch.channel_class, c.cust_city,
t.calendar_quarter_desc;
```

réécrit comme

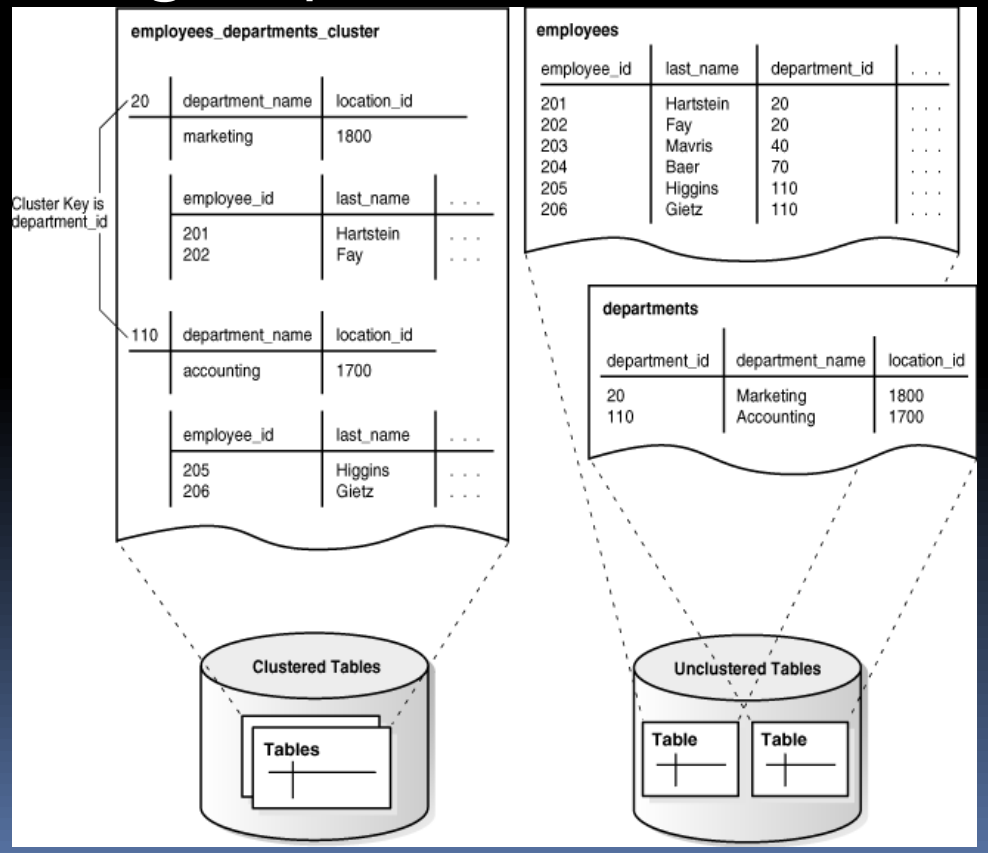
```
SELECT ch.channel_class, c.cust_city,
t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount FROM
sales
WHERE
time_id IN (SELECT time_id FROM times
WHERE calendar_quarter_desc IN('1999-
Q1','1999-Q2')) AND
cust_id IN (SELECT cust_id FROM
customers WHERE
cust_state_province='CA')
AND channel_id IN (SELECT channel_id
FROM channels WHERE channel_desc
IN('Internet','Catalog'));
```

Partitionnement

- Le partitionnement permet de créer des ***sous-ensembles disjoints*** de données (ayant ou pas une signification)
- Le partitionnement est utile, entre autre, pour l'optimisation des requêtes, introduisant la possibilité d'accéder un des sous-ensembles de données (au lieu de toutes les données)
- Le partitionnement peut être
  - Vertical (en fonction des colonnes)
  - Horizontal (en fonction de données)

Partitionnement vertical (ORACLE)

- Il n'existe pas un concept de partitionnement vertical en ORACLE à proprement parler
- Cependant, il est possible d'utiliser les **clusters** pour le mettre en place, un cluster étant un moyen pour regrouper les données provenant de plusieurs tables



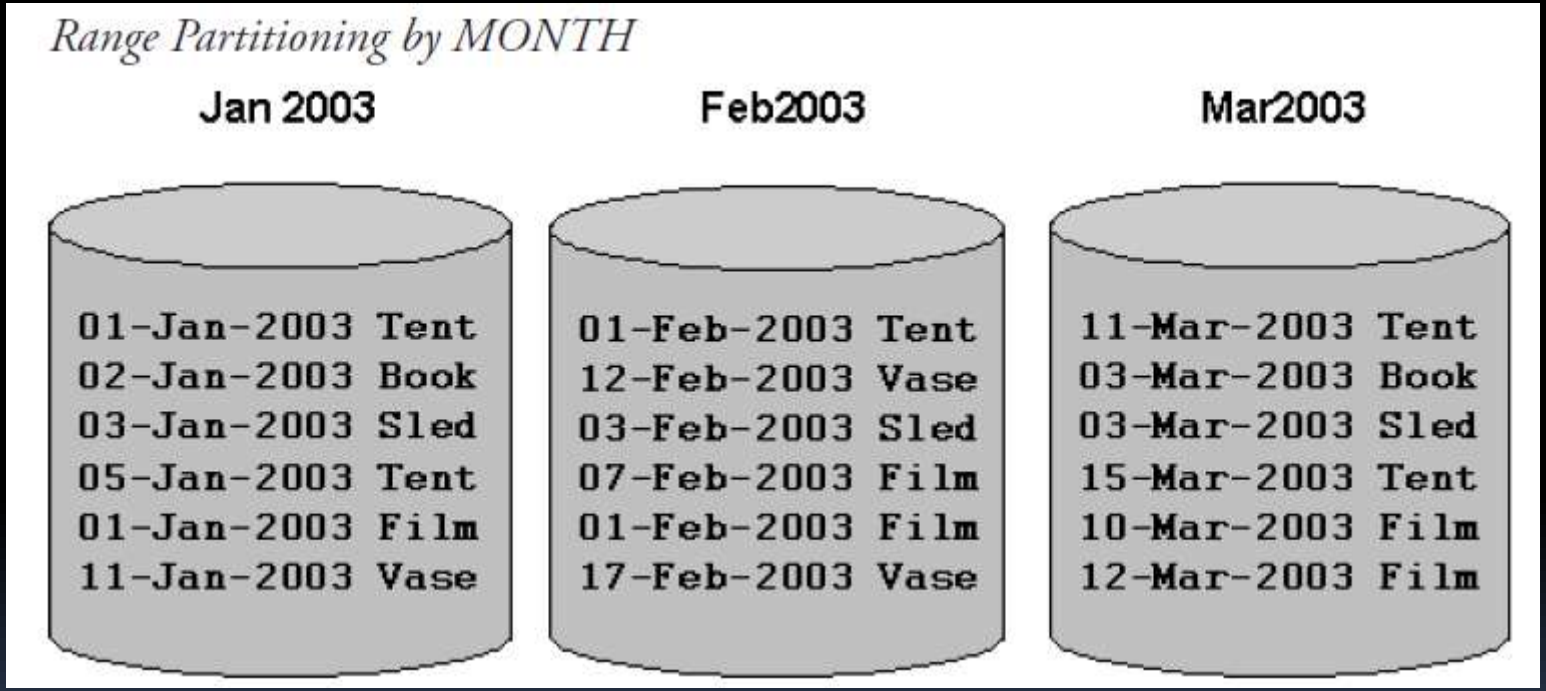

```
CREATE TABLE employees ( ... ) CLUSTER
employees_departments_cluster
(department_id);
 CREATE TABLE departments ( ... ) CLUSTER
employees_departments_cluster
(department_id);
```

Cluster : imposer le stockage de lignes avec la même cluster key dans le même bloc de données sur disque et sans répéter la valeur de la cluster key

Partitionnement Horizontal (ORACLE)

- ORACLE permet de définir 3 types de partitionnement horizontal de base (autres partitionnements sont possibles)
  - Par valeur
  - Par hachage
  - Par liste
- ORACLE permet d'appliquer ce partitionnement aux tables et aux vues matérialisées

Partitionnement par valeur : exemple

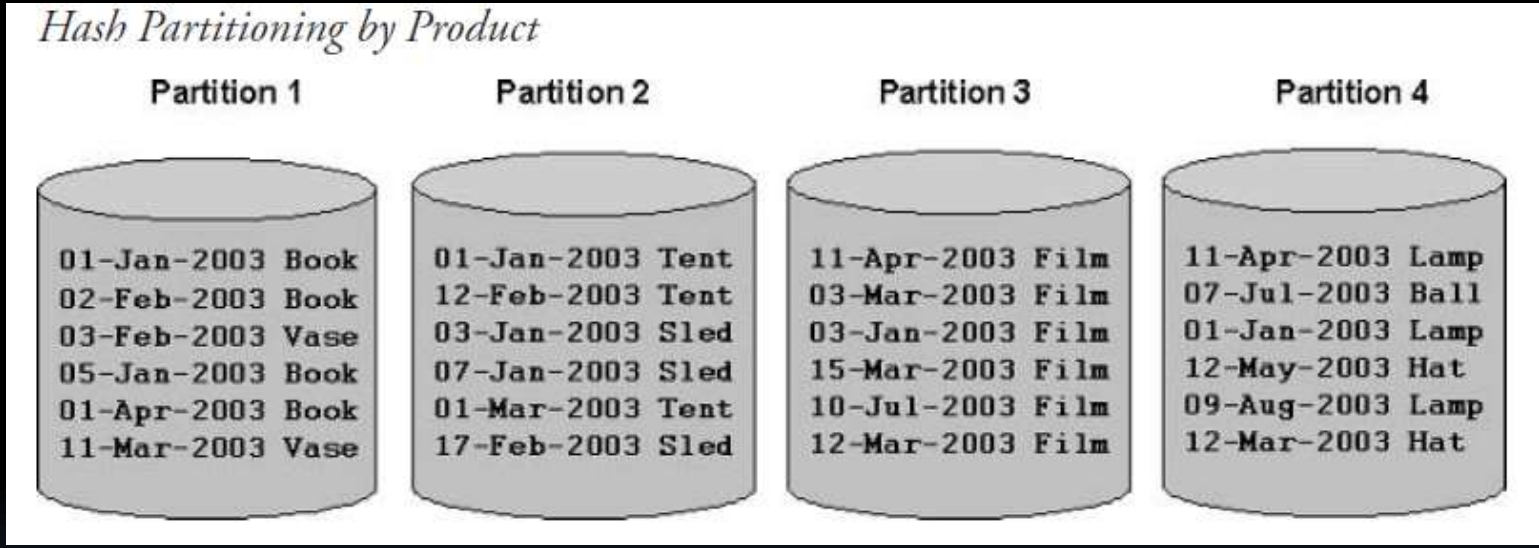


Partitionnement par valeur : syntaxe (ORACLE)

```
CREATE TABLE easydw.purchases
(product_id              varchar2(8),
 time_key                date,
 customer_id             varchar2(10),
 purchase_date           date,
 purchase_time           number(4,0),
 purchase_price          number(6,2),
 shipping_charge         number(5,2),
 today_special_offer     varchar2(1))
PARTITION by RANGE (time_key)
 (partition purchases_jan2003
     values less than (TO_DATE('01-FEB-2003', 'DD-MON-YYYY'))
     tablespace purchases_jan2003,
  partition purchases_feb2003
     values less than (TO_DATE('01-MAR-2003', 'DD-MON-YYYY'))
     tablespace purchases_feb2003,
  partition purchases_mar2003
     values less than (TO_DATE('01-APR-2003', 'DD-MON-YYYY'))
     tablespace purchases_mar2003,
  partition purchase_catchall
     values less than (MAXVALUE)
     tablespace purchases_maxvalue);
```

Utilisation de tablespaces distincts

Partitionnement par hachage : exemple



Fonction d'hachage

- Une fonction de transformation F: $\mathbb{N} \rightarrow \{i,\ldots,j\}$
- De préférence une F « à distribution uniforme » à savoir

  $P(F(x)=k)=P(F(x)=h),\ h<>k$

  sinon, risque de « surcharger » k ou h !
- Dans le partitionnement, la surcharge correspond à utiliser une partition plus qu'une autre
- La propriété ci-dessus est génériquement établie pour une fonction donnée, pour une cardinalité de $\{i,\ldots,j\}$ (par exemple ORACLE utilise une certaine fonction et conseille d'utiliser une cardinalité de $\{i,\ldots,j\} = 2^m$)

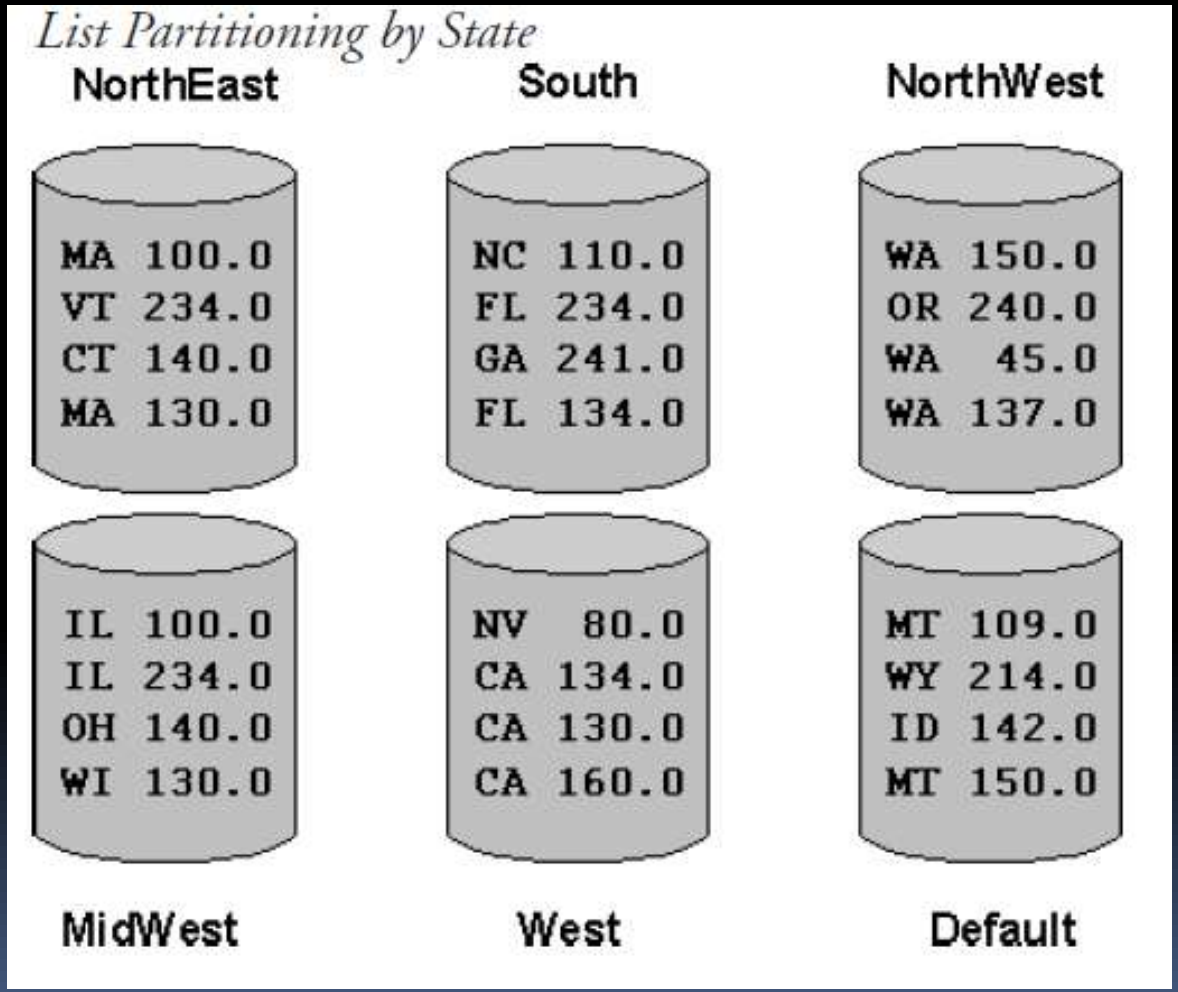
Partitionnement par hachage : syntaxe (ORACLE)

```
CREATE TABLE easydw.purchases
 (product_id              varchar2(8),
  time_key                date,
  customer_id             varchar2(10),
  purchase_date           date,
  purchase_time           number(4,0),
  purchase_price          number(6,2),
  shipping_charge         number(5,2),
  today_special_offer     varchar2(1))
PARTITION BY HASH(product_id)
PARTITIONS 4;
```

Colonnes « unique » typiquement (pour éviter la surcharge)

Puissance de 2 conseillée (la fonction de hachage n'est pas visible)

Partitionnement par liste : exemple



Partitionnement par liste : syntaxe (ORACLE)

```
CREATE TABLE easydw.regional_sales
(state                    varchar2(2),
 store_number             number,
 dept_number              number,
 dept_name                varchar2(10),
 sales_amount             number (6,2)
)
PARTITION BY LIST(state)
(
  PARTITION northeast   VALUES ('NH', 'VT', 'MA', 'RI', 'CT'),
  PARTITION southeast   VALUES ('NC', 'GA', 'FL'),
  PARTITION northwest   VALUES ('WA', 'OR'),
  PARTITION midwest     VALUES ('IL', 'WI', 'OH'),
  PARTITION west        VALUES ('CA', 'NV', 'AZ'),
  PARTITION otherstates VALUES (DEFAULT));
```

Parallélisation des opérations de manipulation de données (réécriture)

- Étant donné que les partitions distinguent de données « indépendantes », elles peuvent être utilisées comme base pour introduire du parallélisme pour INSERT, UPDATE, DELETE et SELECT
- Par exemple (ORACLE)

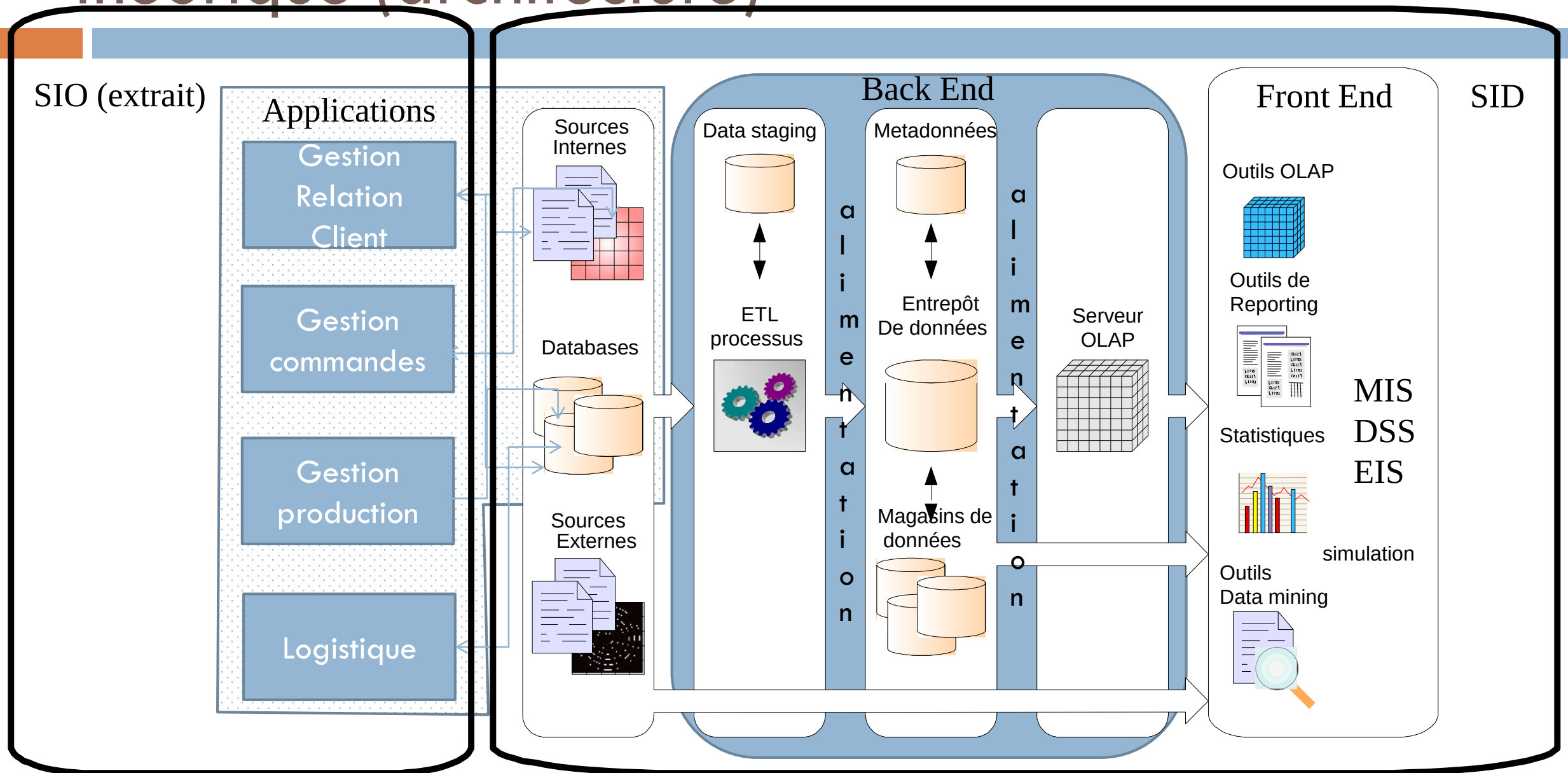
```
DELETE /*+ PARALLEL(PRODUCTS) */ FROM
PRODUCTS  WHERE category_id = 39;

UPDATE /*+ PARALLEL(employees) */ employees SET
salary=salary * 1.1 WHERE job_id='CLERK' AND
department_id IN (SELECT department_id FROM
DEPARTMENTS WHERE location_id = 'DALLAS');
INSERT /*+ PARALLEL(employees) */ INTO employees ;

SELECT /*+ PARALLEL(ACME_EMP) */ * FROM ACME_EMP;
```

Réécrit avec n partitions

```
DELETE FROM PRODUCTS  WHERE
category_id = 39 and Prodid in P1
DELETE FROM PRODUCTS  WHERE
category_id = 39 and Prodid in P2
…
DELETE FROM PRODUCTS  WHERE
category_id = 39 and Prodid in Pn
```

ASPECTS COMPLÉMENTAIRES/SYNTHÈSE

Le système informatique décisionnel (SID) théorique (architecture)



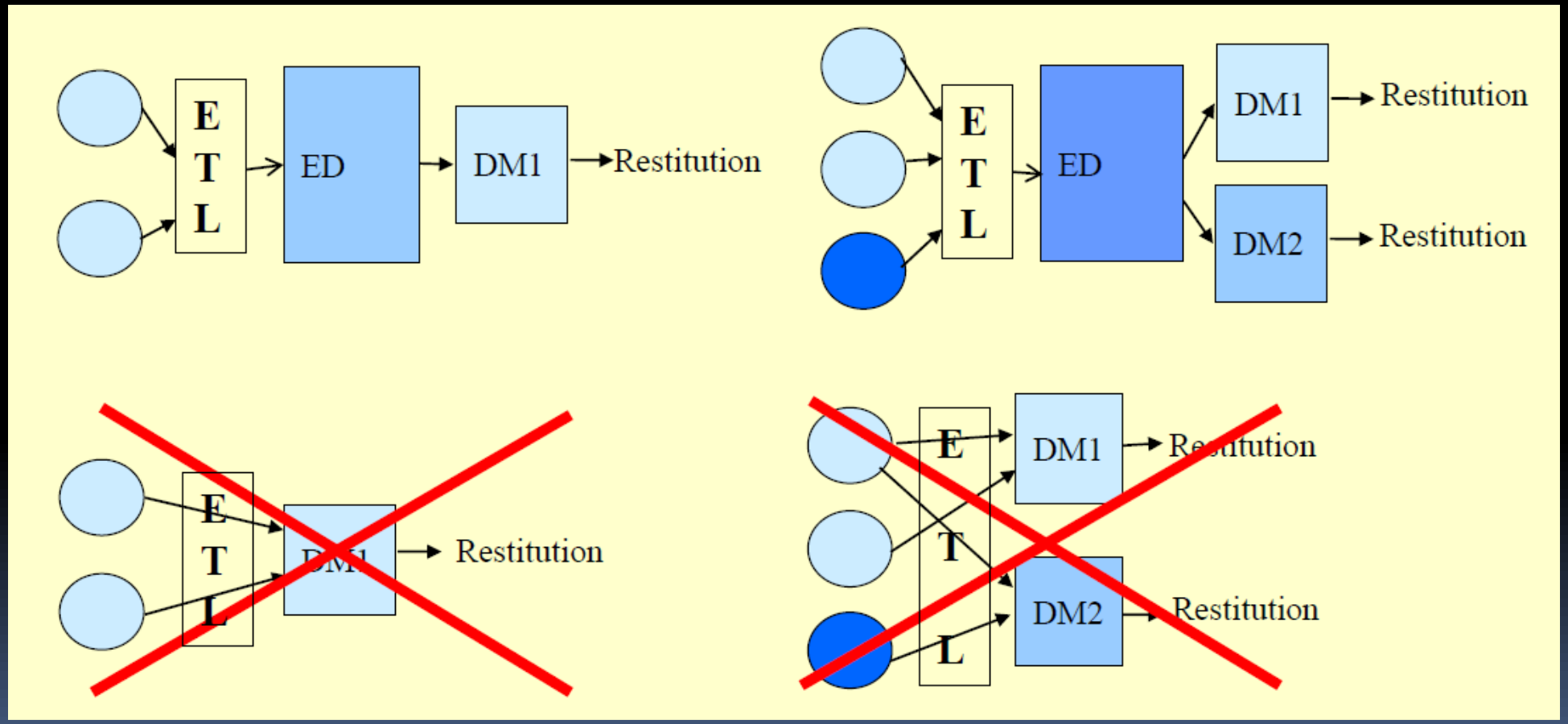
Magasin de données

- Orienté vers une étude particulière, pour un traitement spécifique ; exemples :
  - Comportement de la clientèle → quels produits sont achetés le plus fréquemment par certains clients (types), pourquoi certains clients n'achètent plus certains produits, etc.
  - Impact des promotions sur les ventes d'un produit → est ce que la promotion fait vendre plus d'un produit, est ce qu'elle attire de nouveaux clients, etc.
- Sous-ensemble de données, normalement dérivées de l'entrepôt
- Organisation dimensionnelle de données

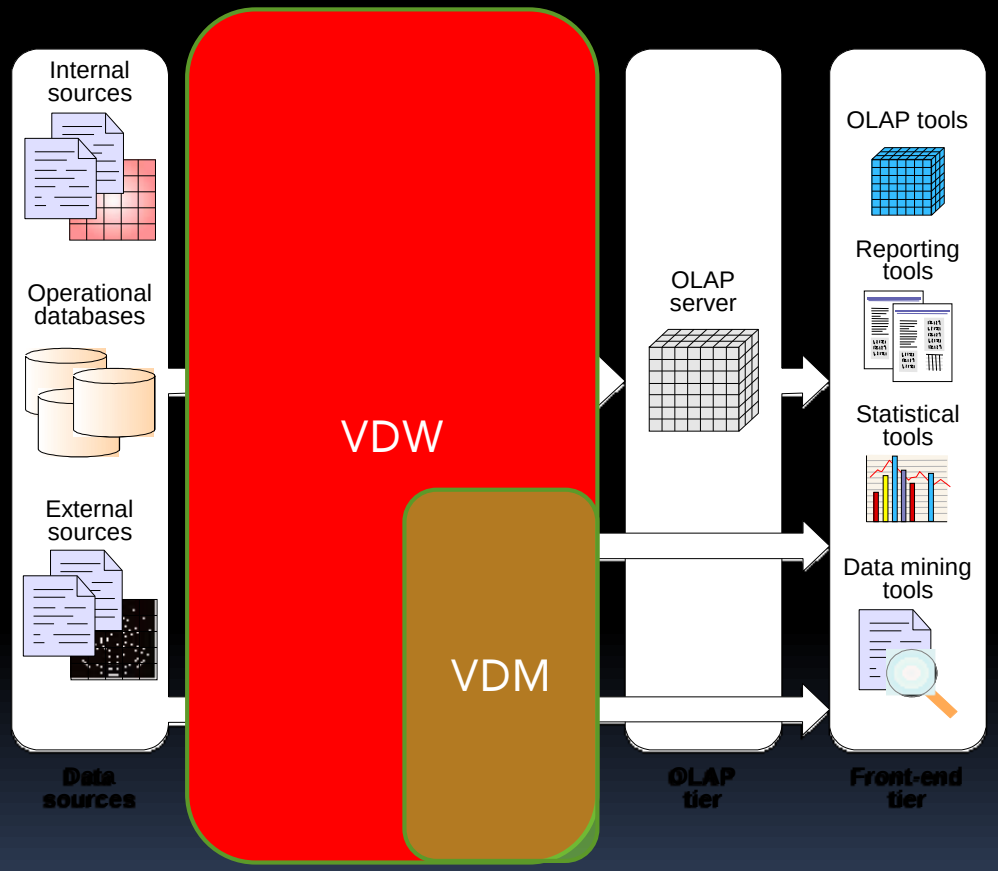
Focus sur les Architectures

- Choix architecturaux
- Choix modèle de données

Architectures alternatives



Architectures simplifiées



VDW : Virtual DataWarehouse
VDM : Virtual Data Marts

Architecture/Modèles de données

- Entrepôt → ROLAP
- Magasin → ROLAP/MOLAP
- Serveur OLAP → MOLAP
- Serveur OLAP + Entrepôt/Magasin → HOLAP

MOLAP (modèle logique)

- Il n'y a pas de tables (liées) comme en ROLAP
- Le modèle suit exactement le modèle conceptuel dimensionnel à savoir des faits organisés en dimensions
- Le modèle s'apparente donc à des structures à matrice où les cellules sont indexées par les dimensions
- Par conséquent, il permet un accès direct aux données, contrairement à l'accès indirect aux données stockées dans une table

ROLAP vs MOLAP I

ROLAP (table de faits)

FAITS

vente

prodId	magId	solde
p1	s1	12
p2	s1	11
p1	s3	50
p2	s2	8

MOLAP (cuboide de faits)

FAITS

	s1	s2	s3
p1	12		50
p2	11	8	

Dimensions du cube = 2

A noter que ces cellules sont vides générant des matrices creuses (« sparse matrix ») : cela ne se produit pas en ROLAP

ROLAP vs MOLAP II

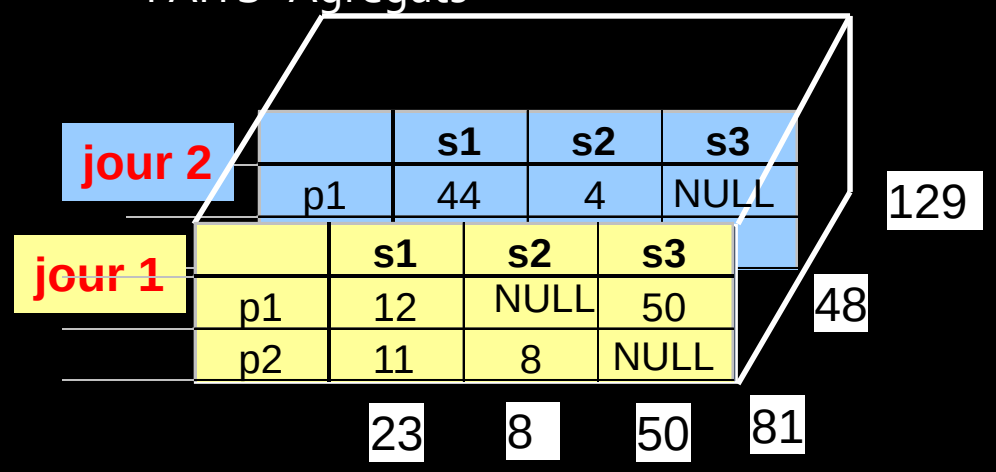
ROLAP (table de faits)

FAITS

vente	prodId	magId	jour	solde
	p1	s1	1	12
	p2	s1	1	11
	p1	s3	1	50
	p2	s2	1	8
	p1	s1	2	44
	p1	s2	2	4

MOLAP (cuboide de faits)

FAITS+Agrégats



dimensions = 3

Select jour, sum(solde) from Faits where jour = 1 →
MOLAP: Somme(Faits(jour=1)(i,j))
ROLAP: Pour chaque Page memorisant des Faits
Pour chaque ligne in Page
Si jour=1 alors
Sum=Sum+ligne.solde

Faits memorisées dans des tableaux : accès direct aux données
Dimensions utilisées pour indexer les tableaux

Langage de requêtes pour schémas ROLAP/MOLAP/HOLAP

- Langage permettant des opérations particulières définies par rapport aux faits/dimensions, telles que :
  - Rollup (Drill-Down)
  - Pivoting
  - Slice
  - Dice
- Pour schéma ROLAP → **Extensions OLAP de l'SQL** (**group by rollup**, **group by cube** + fonctions analytiques réalisant de calculs de regroupement dans le select **« over » « partition by »**)
- Pour schéma MOLAP → Pas de vrai standard, comme l'SQL, langage spécifique (comme MDX, proposé par Microsoft mais utilisé ailleurs, par exemple ESSBASE d'ORACLE)

Tendances

- Entrepôt et « datalake »
- ETL et « data fabric »

Datawarehouse vs Datalake (Gartner)

- **Data warehouse —** A data warehouse is a collection of data in which two or more disparate data sources can be brought together in an integrated, time-variant information management strategy.
  - Data warehouses generally house well-known and structured data. They support well-known, predefined and repeatable analytics needs that can be scaled across many users in the enterprise.
  - As such, data warehouses are best suited to the requirements of moderate to highly consistent semantics.
  - Data warehouses are suited to complex queries, high levels of concurrent access and stringent performance requirements.
- **Data lake —** A data lake collects unrefined data (that is, data in its native form, with limited transformation and quality assurance) and events captured from a diverse array of source systems.
  - Data lakes usually support data preparation, exploratory analysis and data science activities — potentially across a wide range of subjects and constituents.
  - As a result, data lakes support highly variable semantics, a generic set of analytics use cases, and a range of different processing styles and approaches (including data discovery, machine learning and heavy batch computation).
- Datalake et entrepôt peuvent donc être combinés (voir document complet https://www.gartner.com/doc/reprints?id=1-24IZJZ2F&ct=201103&st=sb)

Synthèse

- Système informatique décisionnel : motivation, modèles de données (conceptuel, logique, physique ; ROLAP (en particulier), molap, holap)
- Approche « supply driven » pour la réalisation des entrepôts (applicable aussi aux magasins de données) : transformations de schémas, échanges de données, intégration de schémas, intégration de données
- Aspects complémentaires : architectures/modèles de données, langages de requêtes, tendances