

Systemes d'information décisionnels et entrepôts de données

Giuseppe Berio

giuseppe.berio@univ-ubs.fr

2024

Organisation/Contenu

- Intervenants : Giuseppe Berio (GB- UBS), Michel Dubois (MD-UBS), Intervenant Extérieur (AL - **Jems Group, à confirmer**)
- **Partie I (*Architectures des solutions décisionnelles* - GB)** : Introduction, Modélisation (3 niveaux), Introduction à Intégration de données
- **Partie II (*mise en œuvre ROLAP/MOLAP + ETL* - MD)** :
 - Schéma en Etoile avec Oracle, optimisations ROLAP, Solution HOLAP
 - Solutions ROLAP et HOLAP avec un serveur MySQL+Mondrian
 - Solutions Desktop OLAP avec SAP BusinessObjects
 - ETL : TALEND, Pentaho Data Integration
- **Partie III (*Reporting* - MD et AL)** :
 - MDX avec un serveur Mondrian
 - Power BI
- Évaluation : projet d'envergure en trinôme/binôme (soutenance) + 2 épreuves individuelles
- Plateforme pédagogique : 2 pages distinctes pour GB et MD

Organisation Partie I

- 10 séances CM (à réajuster) dont la moitié en salle info
- 2/3 séances : Introduction
- 3,5 séances : Modélisation de l'entrepôt → SQL Developer, ORACLE
 - 0,5 séance de présentation de la modélisation à 3 niveaux dans SQL Developer
- Reste des séances : Intégration de données
 - Modélisation zone de staging → SQL Developer, SQL
 - Traitements ETL → spécification, prototypes en SQL

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

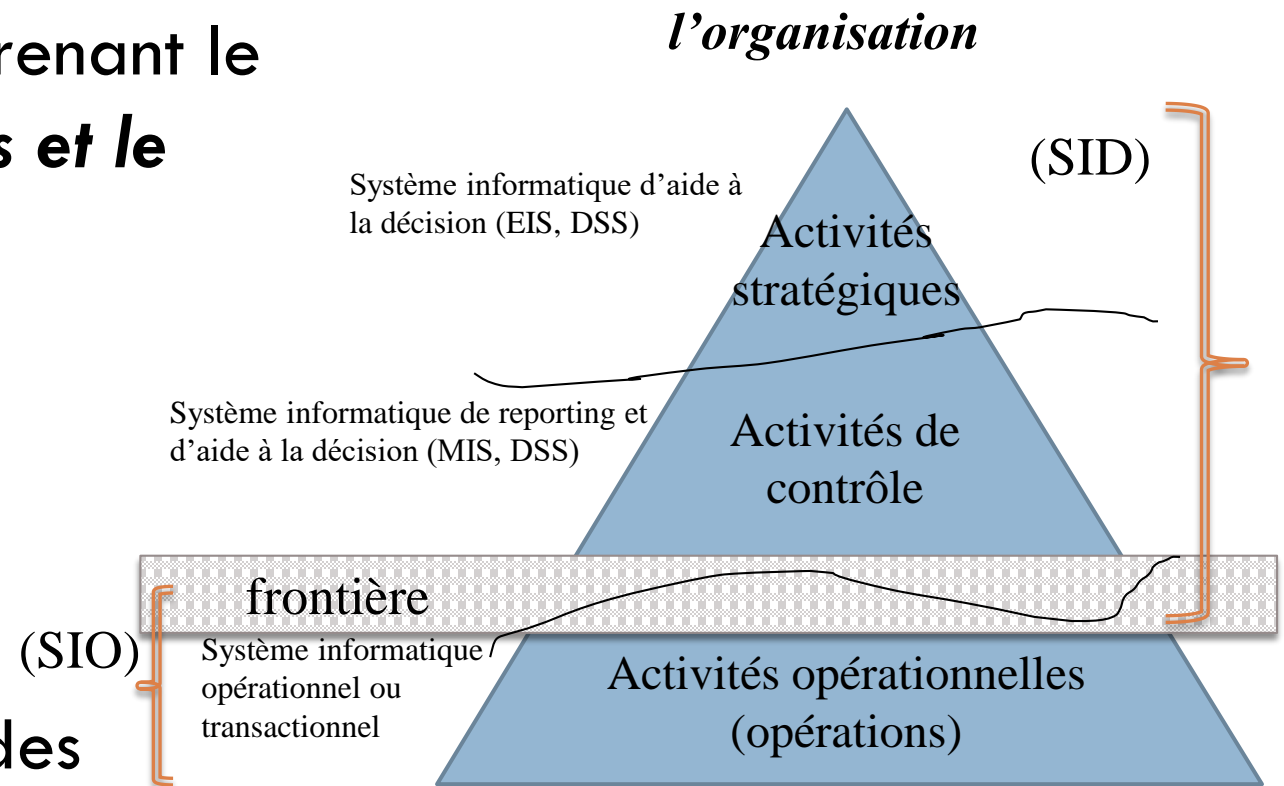
- **Collecter**

- **Archiver**

- **Élaborer**

- **Échanger**

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

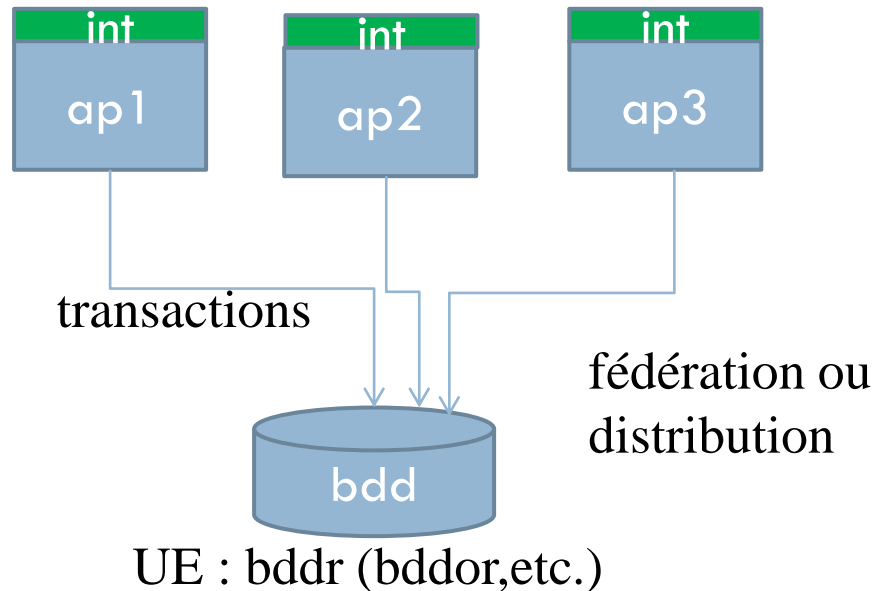


Système informatique \subseteq matériel + logiciel

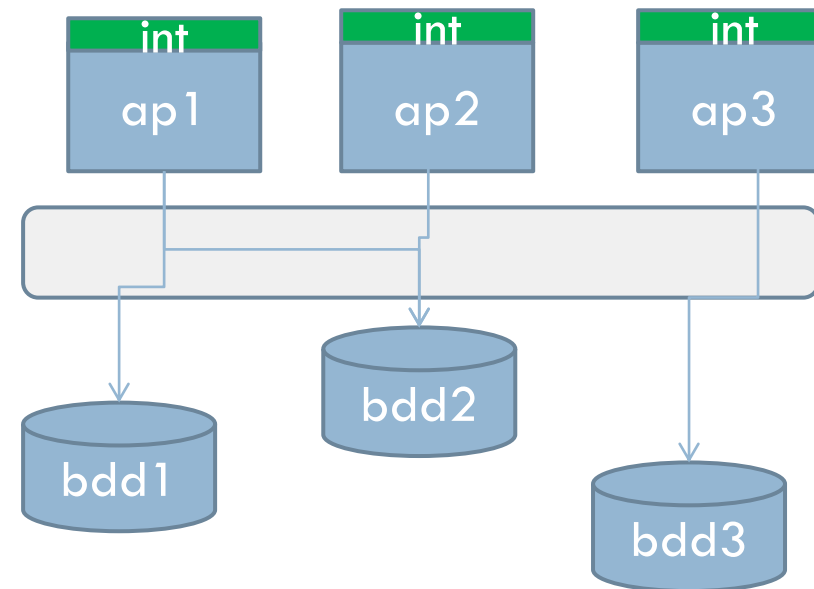
Le système informatique opérationnel (SIO) théorique (architecture)

6

système centralisé



système distribué



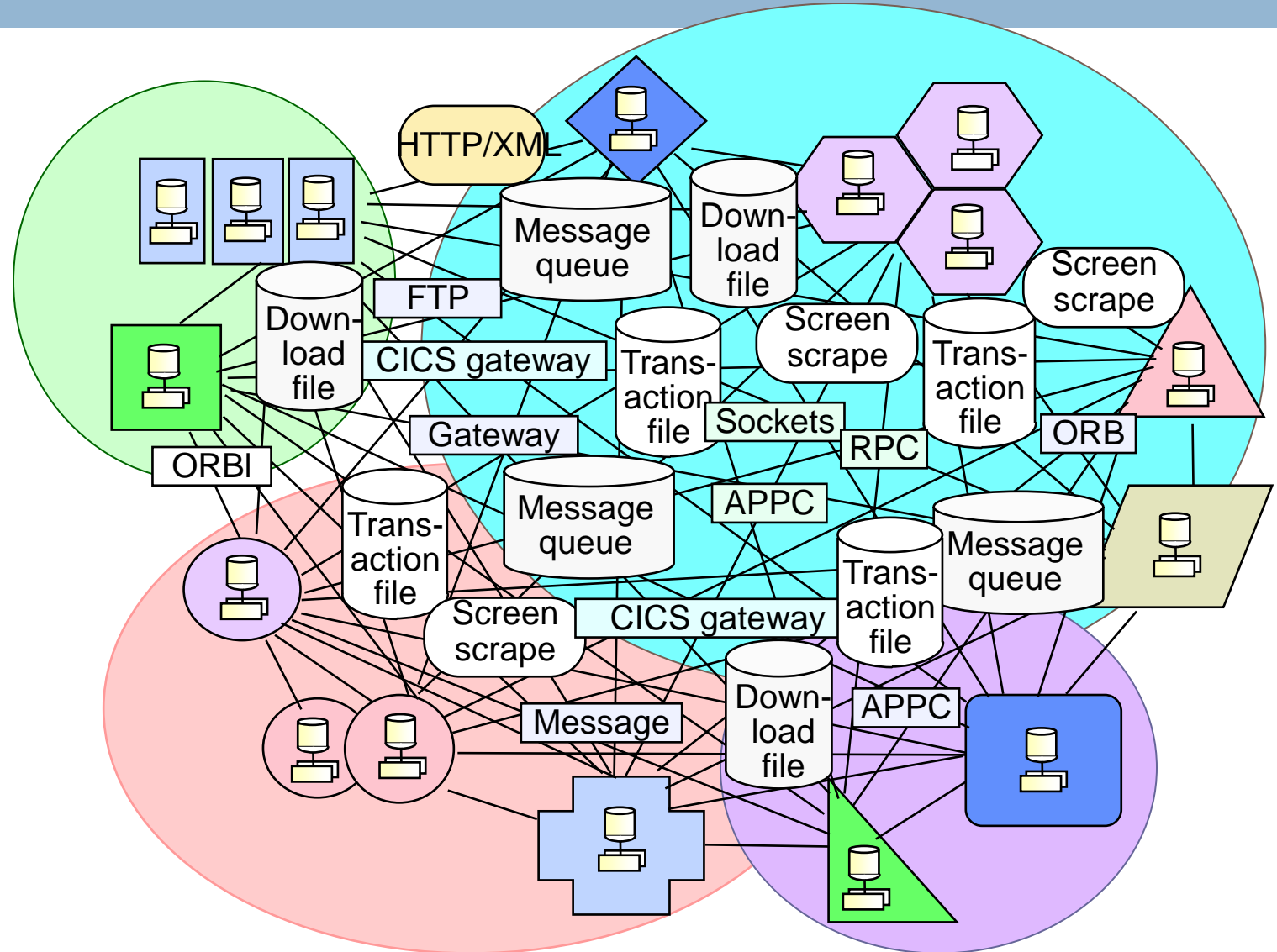
- ***Nombre élevé de transactions par unité de temps*** : maximiser le nombre d'utilisateurs travaillant en parallèle sur les mêmes bases
- ***Intégrité de données (sous concurrence)*** : maximiser la qualité de données stockées, ne pas avoir un comportement/résultat erroné

Architecture réelle du SIO

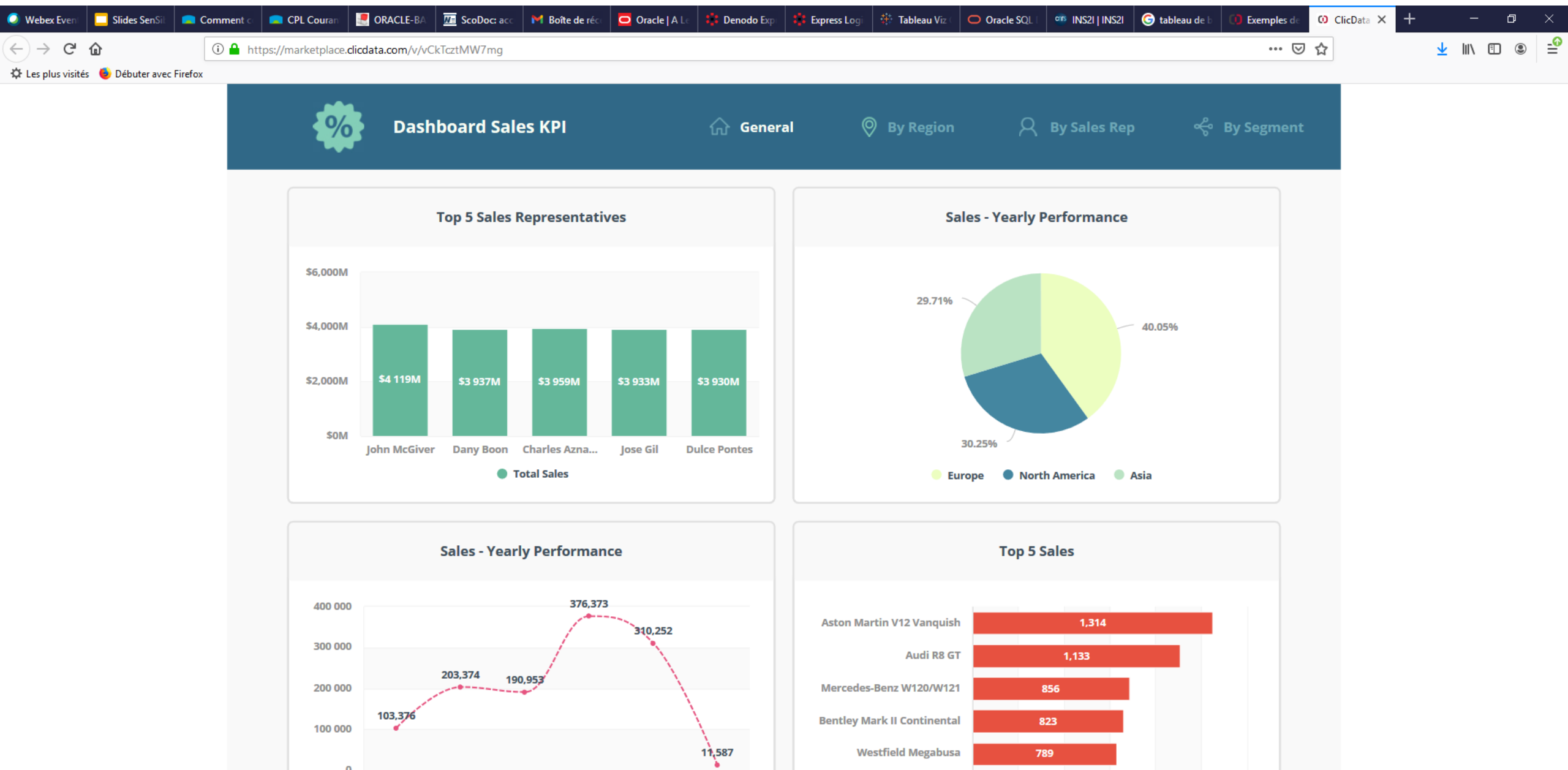
Architecture réelle du SIO = ensemble d'applications autonomes et communicantes



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



Front-end : Tableau(x) de bord statiques



Front-end : Tableau(x) de bord dynamiques

9



OTHER LEVEL'S FINANCE STATUS DASHBOARD

Last Update 10 Sep 2020, Thursday

Main Dashboard

Human Resource

Status Report

Year

2019

2020

2021

Department

CCP

EZE

ICCD

IMC

PAN

SBN

SSBD

SSMC

Project Status

Cancelled

Closed

Current

Total Project's

99 PO

C APEX

774,111,895

Lockers

427,057,284

Popular

503,392,307

Saving

657,920,172

OPEX

61%

100%

Current 57 Closed 40 Canceled 2



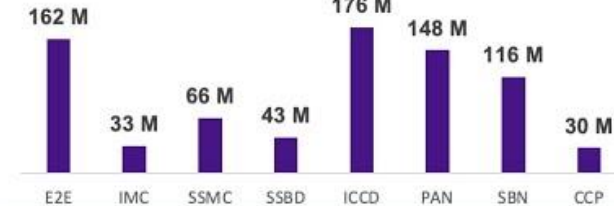
InIntrodsuie Project's

20

Standard Project's

79

LOI Issued by Departments



Total Targets For September

9,817

Total Actuals For September

9,113

GAP

-704

-7%



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



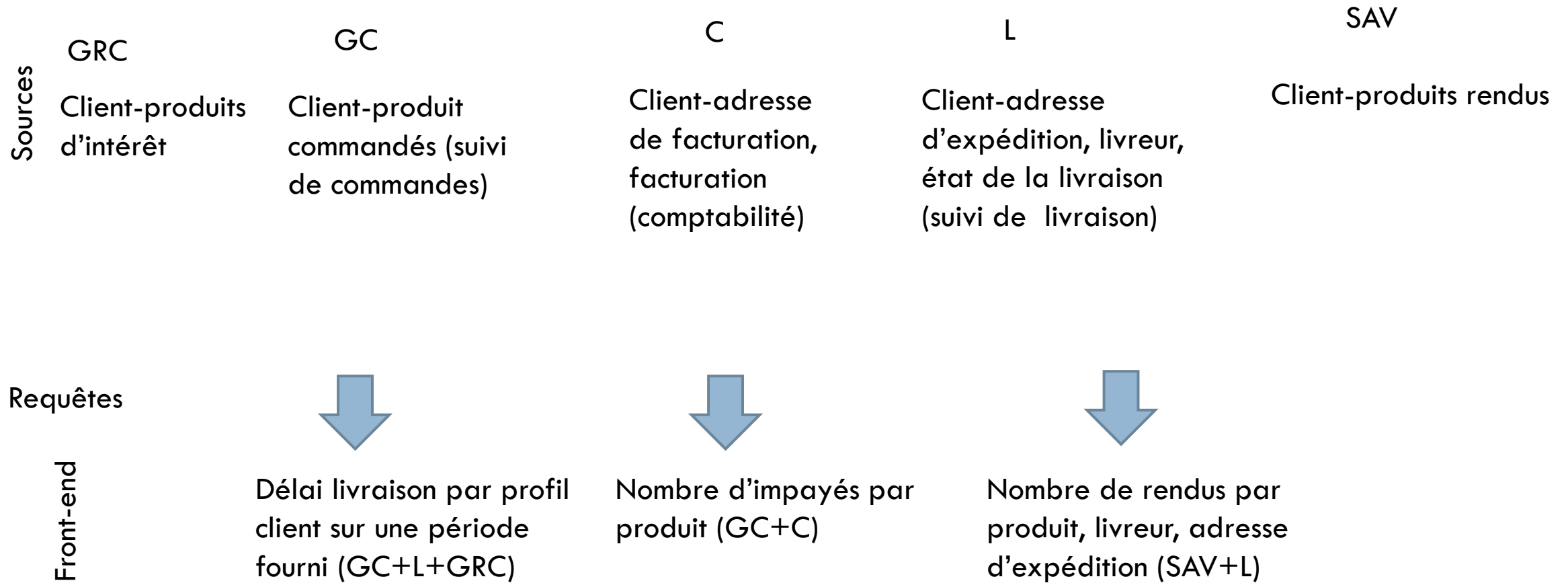
GAP 4
As Built vs Invoiced
-27,049,456

	GAP 1	GAP 2	GAP 3	GAP 4
EZE	-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

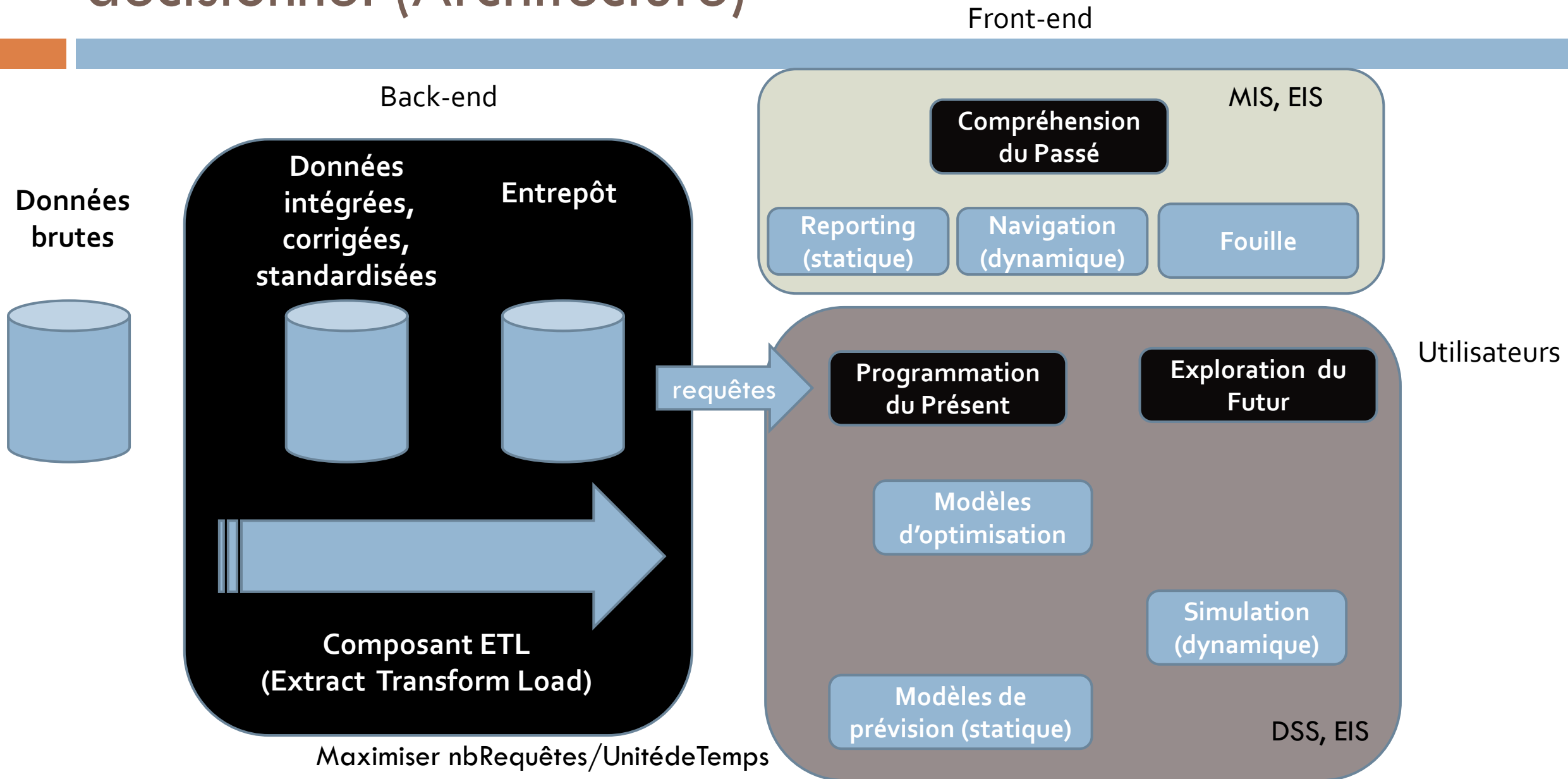
Commercial Negiation

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

Idée



Composants typiques d'un système informatique décisionnel (Architecture)



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 (Raffinement Architecture)

SIO (extrait)

Applications

Gestion
Relation
Client

Gestion
commandes

Gestion
production

Logistique

Sources
Internes

Databases

Sources
Externes

Data staging

ETL
processus

Back End

Metadonnées

Entrepôt
De données

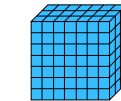
Magasins de
données

Serveur
OLAP

Front End

SID

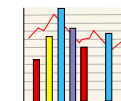
Outils OLAP



Outils de
Reporting



Statistiques



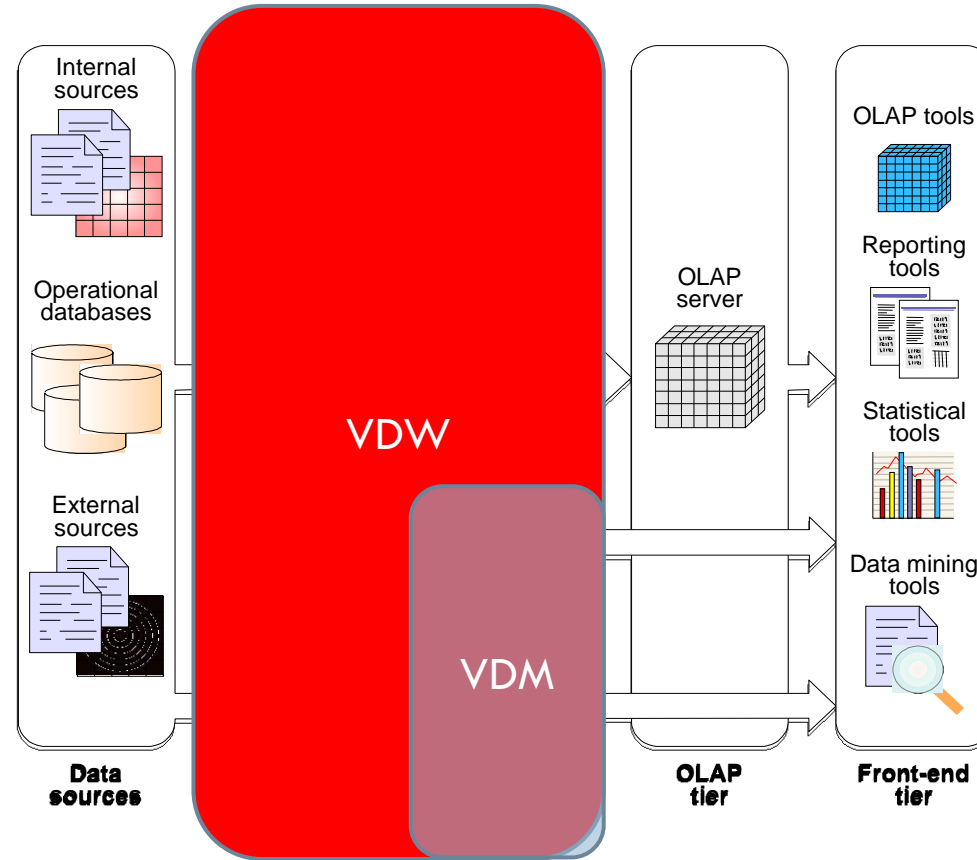
Outils
Data mining



MIS
DSS
EIS

simulation

Architectures simplifiées



VDW : Virtual DataWarehouse

VDM : Virtual Data Marts

Structure d'une application SID

15

- 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 (même si le concept d'entrepôt virtuel contredit cet propriété)

principalement utilisées pour l'aide à la décision W.H.

Inmon (1996)

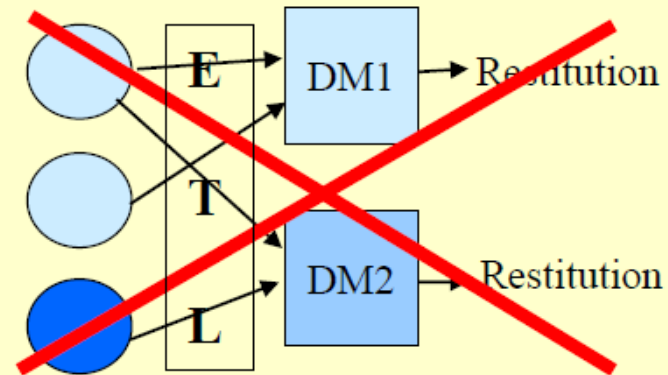
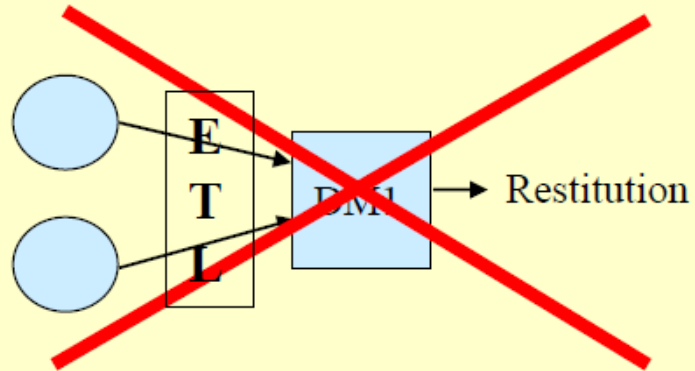
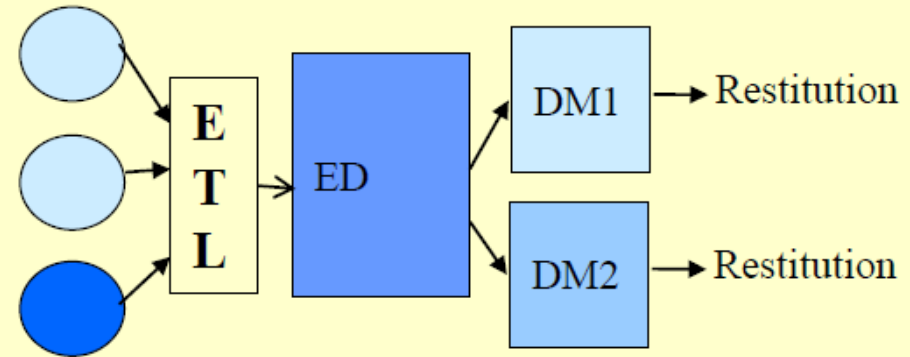
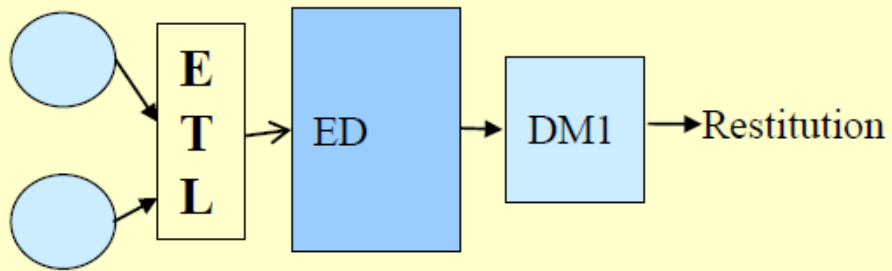
Magasin de données

17

- 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

Data-marts et entrepôt

18



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

19



Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms



Source: Gartner (February 2020)

© Gartner, Inc

Données (back-end) : marché 2019

Figure 1. Magic Quadrant for Operational Database Management Systems



Source: Gartner (November 2019)

Processus ETL : décomposition en traitements

21

Extraction

Données
extraites
(partiellement
corrigées et
standardisées)

Stockage dans l'espace de
staging : typiquement BDD
Relationnelle si nécessaire

- Correction
- Préparation
- Intégration

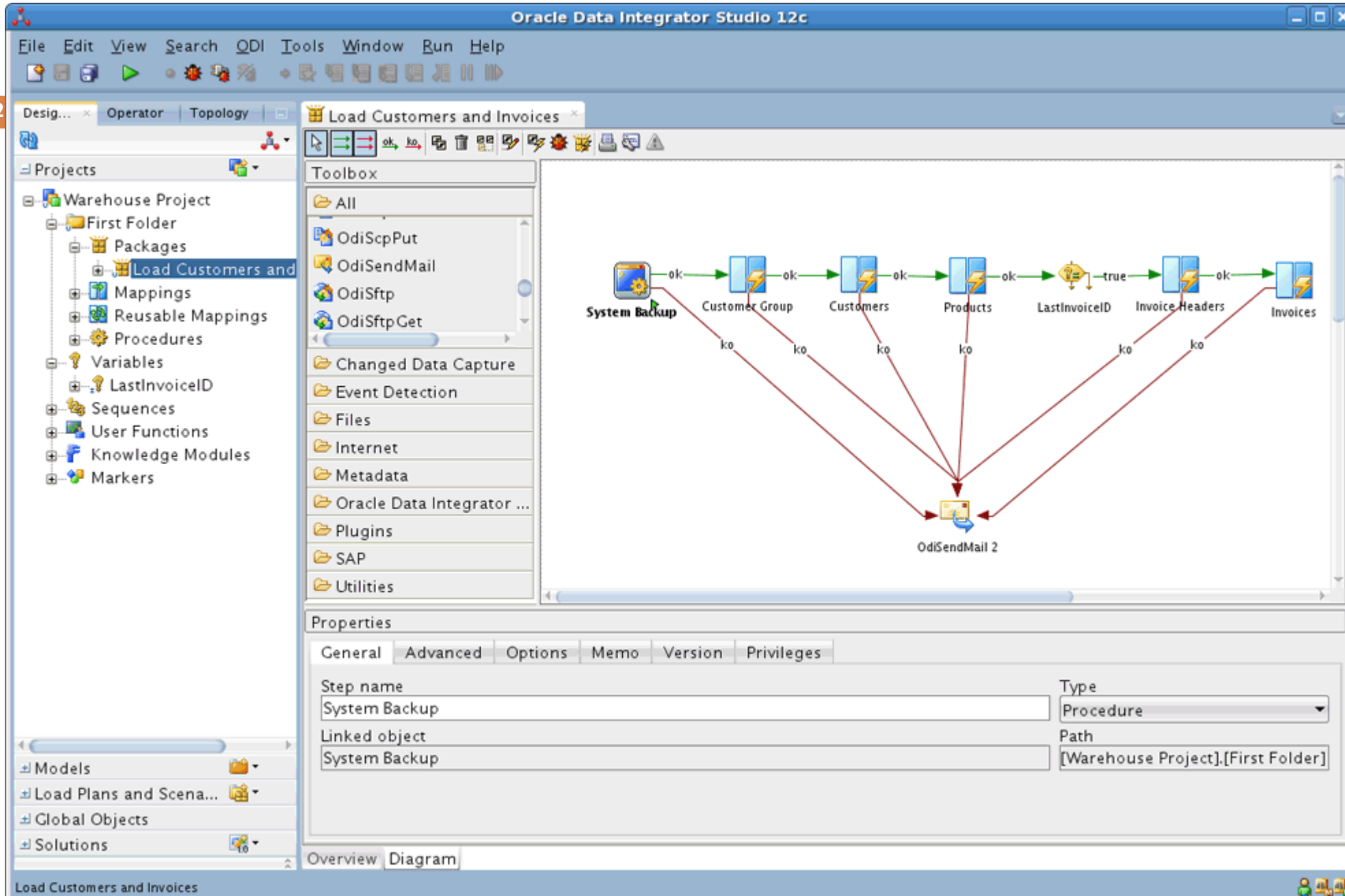
Données
corrigées et
intégrées

Stockage dans l'espace de
staging : typiquement BDD
Relationnelle

- Préparation
- Chargement

Stockage : Entrepôt

Données
chargées dans
l'entrepôt



Prepare Customer Target TRAIN data *

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

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

View Data: Transpose_OUTPUT2 (290 rows) (Browse)

#	customer_id	_NAME_	COL1	COL2	COL3	
19	cx115	requested_file	/Cookie_C...	/Departme...	/Email.jsp ...	/Home
20	cx116	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Cooki
21	cx117	requested_file	/Product.js...	
22	cx118	requested_file	/CDMA/Err...	
23	cx119	requested_file	/Home.jsp ...	/Product.js...	/Search.js...	/Site_
24	cx12	requested_file	/Billing.jsp ...	/Cart.jsp ...	/Confirm.js...	/Depai
25	cx120	requested_file	/Catalog_...	/Catalog_...	/Cookie_C...	/Depai
26	cx121	requested_file	/Home.jsp	
27	cx122	requested_file	/MiniDisc/E...	/Product.js...	/Retail_Sto...	
28	cx123	requested_file	/Cookie_C...	/Departme...	/Product.js...	/Subc

Completed successfully

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)


27

Approfondissement des concepts


Points à développer

28

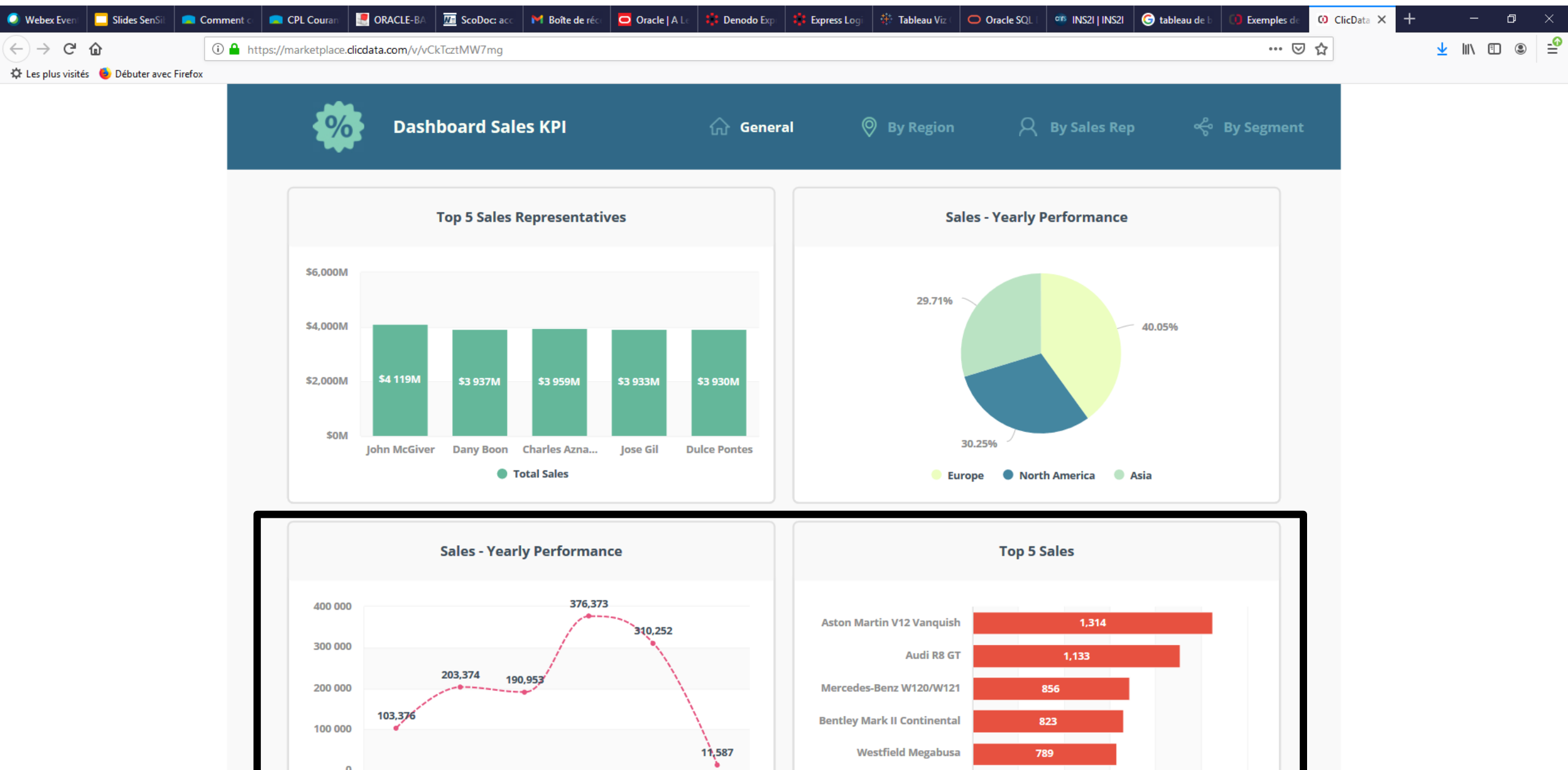
- Compréhension des éléments centraux pour la modélisation de l'entrepôt et de l'espace de staging (données standardisées, corrigées et intégrées)
- Modélisation de l'entrepôt
- Modélisation de l'espace de staging
 - ▣ Zone d'extraction
 - ▣ Zone d'intégration
- Principes de modélisation du processus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

- Éléments 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

- Éléments 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

Démo fonctionnement 1 (ROLAP)

Données brutes

Non typé

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

correction/typage

Typé

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

Composant ETL (Données
extraites) – Staging

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

Démo fonctionnement 2 (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

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

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

Mesure précalculée
(fait inféré)

Σ Quantité	ClientID
300	1
100	3
200	2

Mesure

ClientID	Σ Quantité	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	Σ Quantité
54	200
76	200
56	200

Entrepôt

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

Σ Quantité	Mois
300	01/2013
100	02/2013

Σ Quantité	Année
600	2013

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

Dimension et hiérarchie

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

Entrepôt

ClientID	
1	
3	
2	

ClientID	Σ Quantité	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

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

Exemple III (modèle logique)

ClientID		
1		
3		
2		

ClientID	Σ Quantité	Produit	Mois	$\Sigma(\text{quantité} \times \text{prix})$ 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

Σ Quantité		
200		
200		
200		

Entrepôt : table de faits
avec plusieurs faits

	Année	CA
	2013	null

Exemple IV (modèle logique)

Table 1 - ventes

ClientID			ClientID	Σ Quantité	Produit	Mois	$\Sigma(\text{quantité} \times \text{prix})$ as CA	Promo		Produit		
1			1	100	54	01/2013	Null	Null		54		
3			3	100	54	02/2013	1000	3		56		
2			2	200	56	01/2013	1100	7		76		
			1	200	76	01/2013	null	null				

Entrepôt : modèle
logique en
constellation

Σ Quantité	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)

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 fraîches)
- Données agrégées
- Données précalculées

Fiabilité de la décision
: garantie par le
processus ETL

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



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*

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

Magasin
|
Region
|
Pays

Magasin
|
Ville
|
Region
|
Pays

Magasin
|
Ville
|
Region
|
Pays

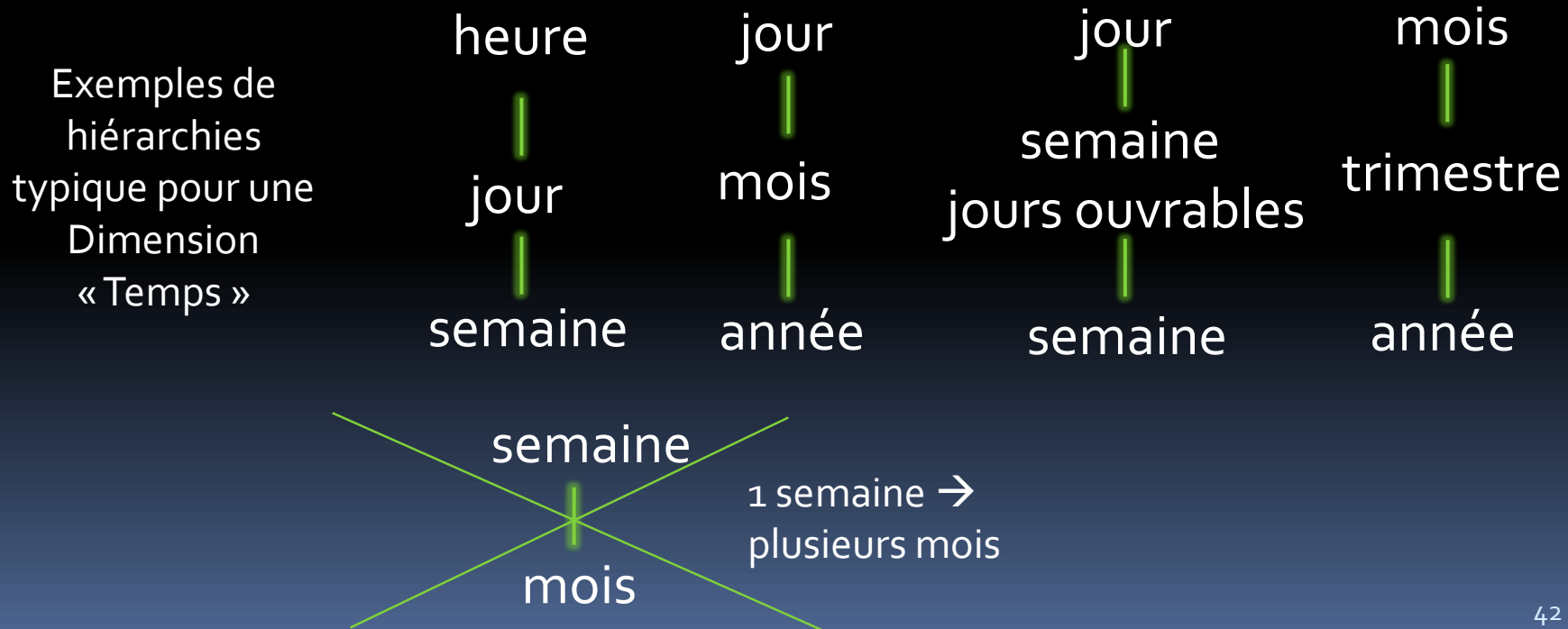
Nom,
Habitants

Hiérarchies



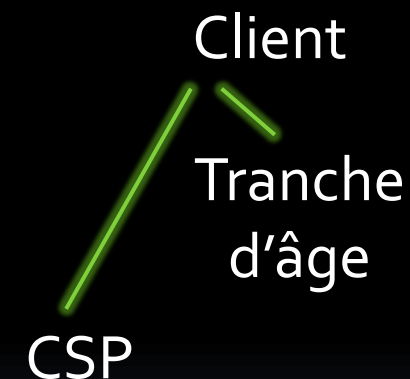
Hiérarchies temporelles

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



Dimensions et hiérarchies

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





Dimensions conformes

- Dimensions pouvant être utilisées pour plusieurs tables de faits (ou **cubes**) distinctes
- Une dimension conforme typique est la *dimension temporelle*, souvent pré-réalisée dans les systèmes disponibles
- Les dimensions conformes représentent toute 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*



Contrainte de modélisation pour les dimensions

- La modélisation d'une dimension ne peut se faire qu'en liaison avec les données disponibles (traitées par l'ETL)
- Par exemple, serait-il possible de représenter une dimension « client » avec un niveau « tranche d'âge » sans pour autant disposer de données sur l'âge de client ?
- Conceptuellement oui (sachant que même si inconnu) un client a un âge et donc une tranche d'âge ; mais le modèle logique de données resterait incomplet ne pouvant pas associer la tranche d'âge aux clients (SQLDeveloper évite ce genre de désagrément)

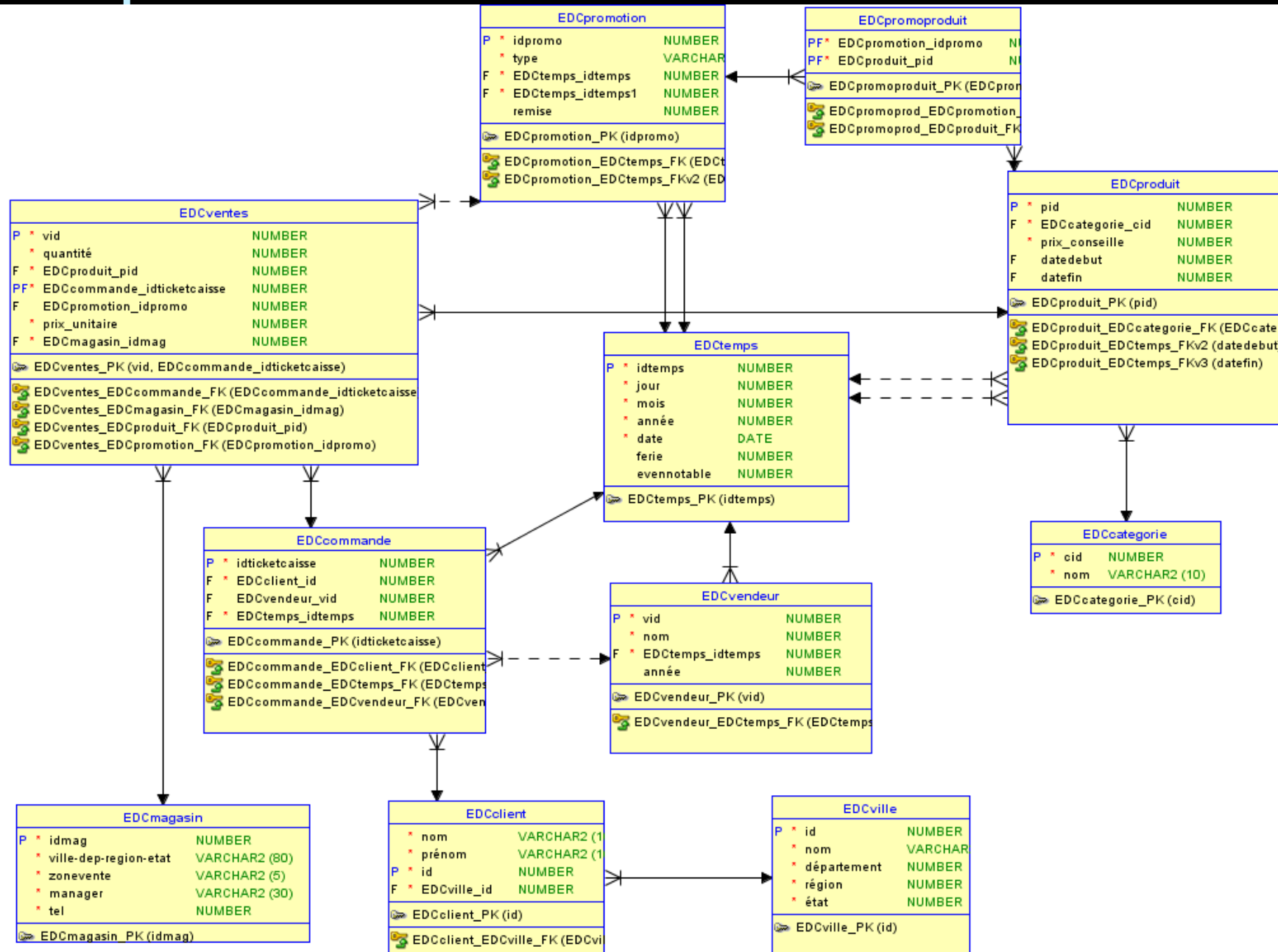
Objectif et définition d'une mesure

- L'objectif d'une mesure est de
 - **Spécifier comment alimenter la table de faits** à partir de données traitées par l'ETL donc de **constituer les faits observés, peu importe les dimensions définies**
 - **D'établir (d'inférer) les faits** à un (ou plusieurs) niveau(x) à partir des faits observés et inférés, si possible
- Plusieurs moyens d'inférence peuvent définir une mesure, selon les cas
 - Operateur d'agrégation SQL (+where éventuel)
 - Formule libre éventuellement 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 peut nécessiter à la fois d'une formule et un operateur d'agrégation utilisé pour inférer les faits

Objectif et définition d'une mesure

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

Compréhension des mesures : exemple

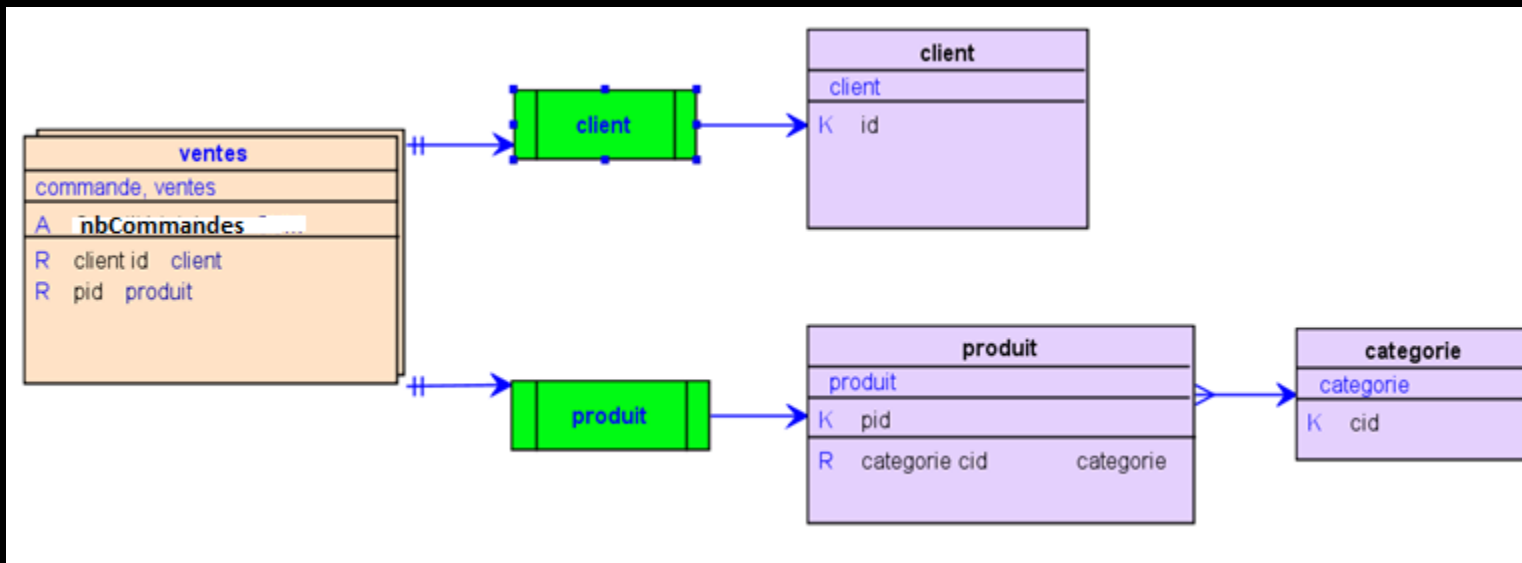


Source ou données transformées (schéma) au sein de l'ETL

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 égale** à celle que la mesure aurait utilisant les données disponibles (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)



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

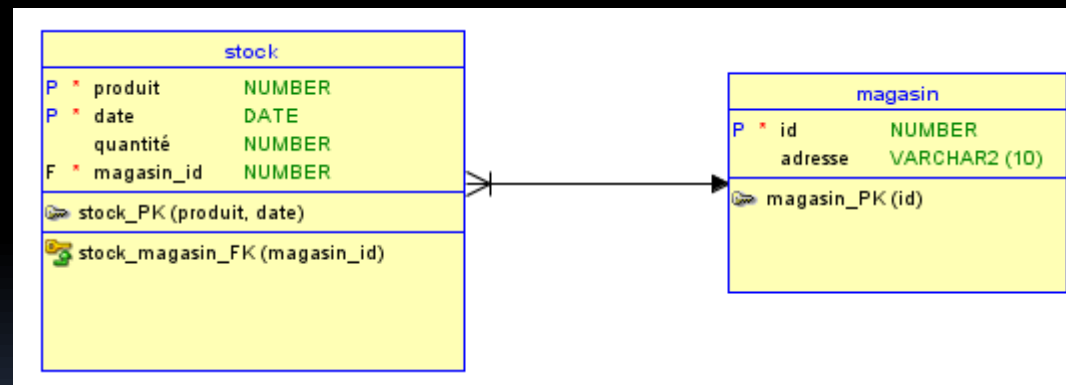
- Une mesure est « agrégable » par rapport à une dimension, s'il existe un opérateur d'agrégation (SQL) fournissant la valeur escomptée lorsqu'il est utilisé avec la dimension
- Une mesure est agrégable si elle est agrégable pour toute dimension
- Autrement la mesure n'est pas agrégable
- Si la mesure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

Operateurs vs propriétés

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

Une mesure définie par une somme n'est pas toujours agrégable (ni additive) !

- Parfois, effectuer une somme pour certaines dimensions n'est pas significatif car les données représentent **un niveau** (à une certaine date), contrairement à **un flux**



Formalisation de la notion de mesure

- Hypothèses :
 - **Fait** est l'observation comme valeur numérique (ou * est l'évènement sans fait) repérable dans les données disponibles sans élaboration
 - **FI** est un fait inféré
 - N_{ji} est un niveau hiérarchique i (par exemple le nom d'une colonne d'une table) d'une dimension j
 - **M** est la mesure associée à un Fait, calculable sur **les données disponibles** permettant d'obtenir un FI à savoir $FI(N_{ht}, \dots, N_{ij}) = M(\text{Fait}, N_{ht}, \dots, N_{ij})$; **M** réalise l'inférence (il s'agit d'une convention – donc il n'est pas l'observable/observé mais FI est le mesurable/mesuré)
 - N_{ji} et N_{ji+1} sont 2 niveaux hiérarchiques directement liés au sein de la même dimension j , étant N_{ji} le détail et N_{ji+1} l'agrégé
- **M est additive** ssi pour chaque dimension (j) et tout niveau (i)
 - $M(\text{Fait}, N_{ji+1}) = \text{Sum}(M(\text{Fait}, N_{ji}), N_{ji+1})$
 - S'il existe au moins 1 dimension ne satisfaisant pas la condition ci-dessus, alors **M est semi-additive**
- **M est agrégable** ssi il existe un *opérateur d'agrégation* OP tel que
 - $M(\text{Fait}, N_{ji+1}) = OP(M(\text{Fait}, N_{ji}), N_{ji+1})$
- **M est calculé** (ou calculable) ssi il existe une formule F telle que
 - $M(\text{Fait}, N_{ht}, \dots, N_{ij}) = F(M_1(\text{Fait}_1, N_{ht}, \dots, N_{ij}), \dots, M_n(\text{Fait}_n, N_{ht}, \dots, N_{ij}))$
- **M est dérivée** (ou dérivable) ssi il existe une formule F telle que
 - $\text{Fait}_{(n+1)} = F(\text{Fait}_1, \dots, \text{Fait}_n, N_{ht}, \dots, N_{ij})$ et
 - $M(\text{Fait}_{(n+1)}, N_{ji+1}) = OP(M(\text{Fait}_{(n+1)}, N_{ji}), N_{ji+1})$

Mesures et table de faits

- Dans un modèle logique ROLAP, la table de faits contient des faits observés ou des faits inférés
- Pour les faits inférés, elle contient les faits inférés aux premiers niveaux des dimensions ; à savoir, s'il y a n dimensions alors la table de faits contient

$$FI(N_{11}, \dots, N_{n1})$$

- Typiquement, elle ne contient pas des colonnes correspondantes aux mesures calculées

Mesures et 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

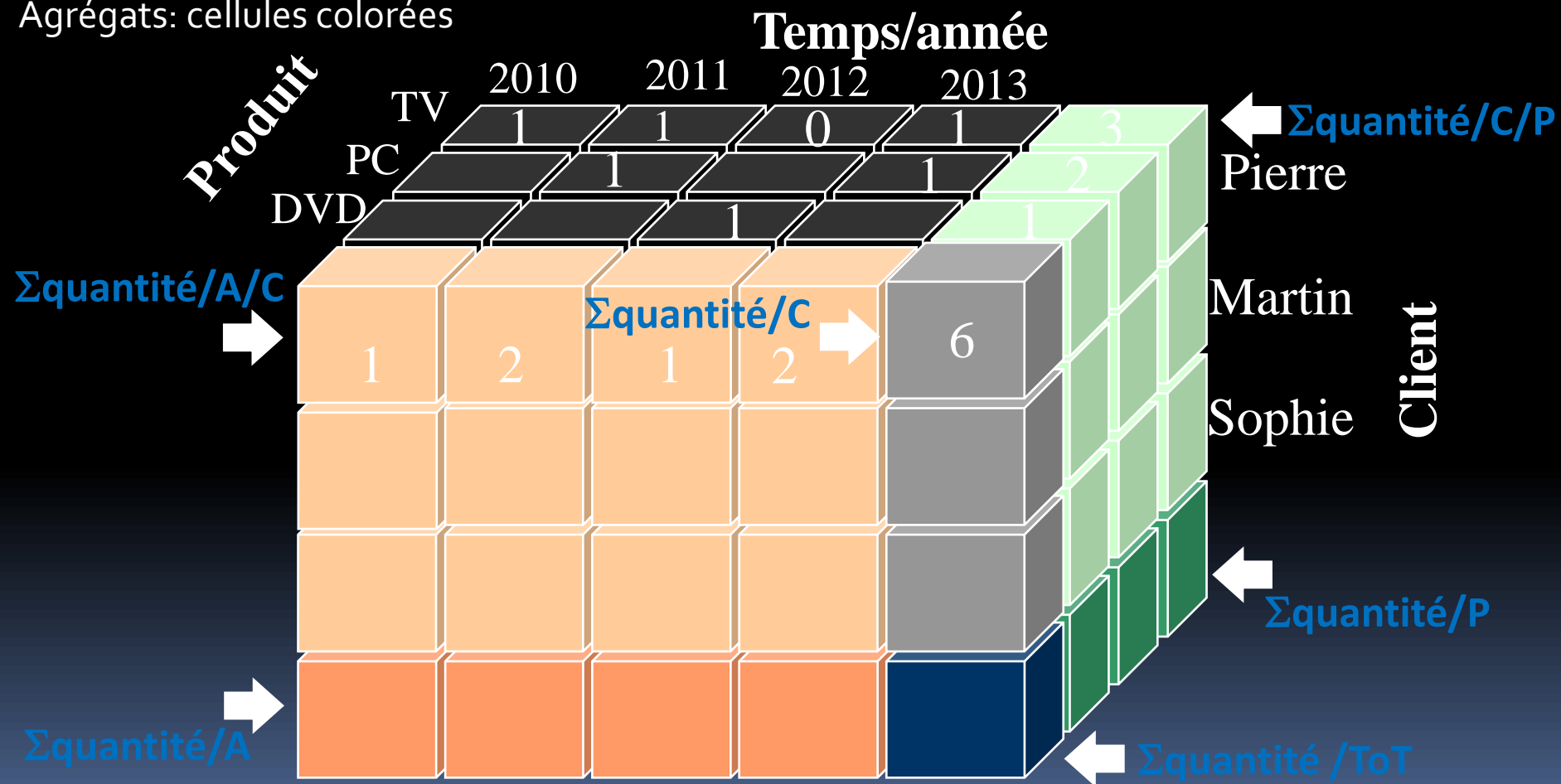
Modèles logiques (choix de base)

- 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 aux 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)

Cuboïde

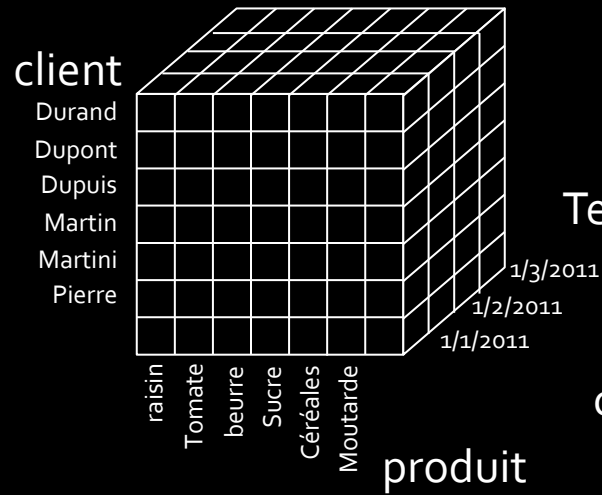
Cuboïde : cellules en blanc (produit,année,client)

Agrégats: cellules colorées

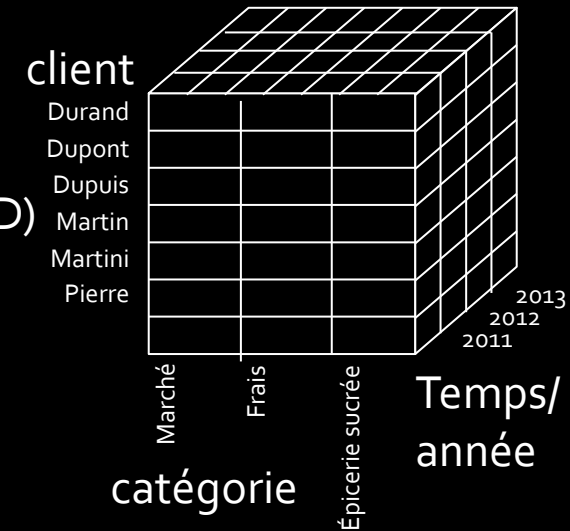


MOLAP (cuboïdes possibles)

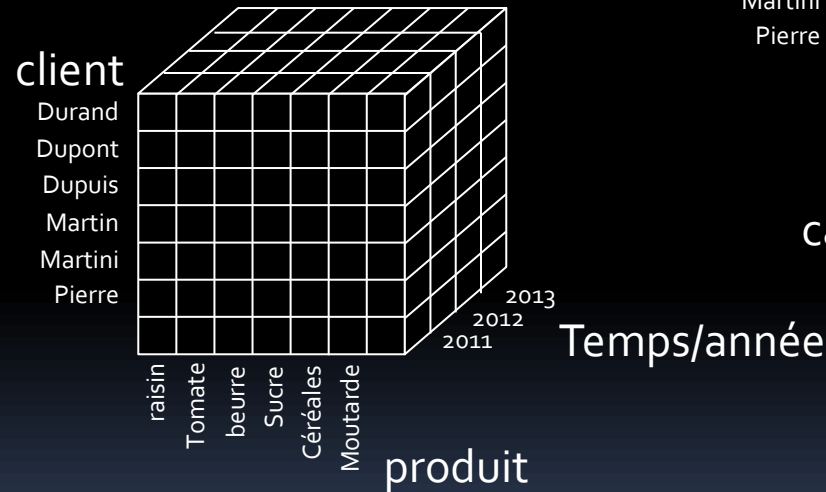
Cuboïde - Faits : client,produit,date (3D)



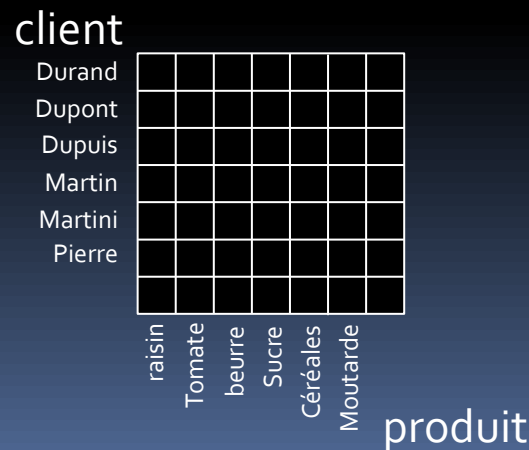
Cuboïde : client,catégorie,année (3D)



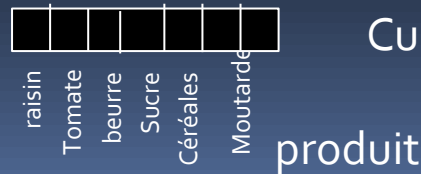
Cuboïde : client,produit,année (3D)



Cuboïde : client,produit (2D)



Cuboïde : produit (1D)





Modélisation logique (ROLAP)

- Schéma flocon vs schéma étoile
- Dimensions changeantes
- Clés de la table de faits

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

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	Mois
1100	01/2013
450	02/2013

ΣQuantité	Année
1550	2013

Modèle logique
dimensionnel ROLAP
(agrégats en vert pour info)



ΣQuantité
1250
100
200

ClientID	
1	
3	
2	

ClientID	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 structure de la table de faits contient les clés étrangères pouvant faire référence au 1^{er} niveau de toute dimension
- La structure de la table de faits contient des colonnes additionnelles correspondantes aux mesures définies, sauf mesures calculées (autant de colonnes que de mesures)
- Mais dans un **modèle étoile** certaines mesures calculées pourraient être réintroduites
- Cette structure permet de stocker dans la table de faits, en fonction de la clé primaire choisie, 2 distinctes **granularité de faits**
 - Granularité transactionnelle (les faits correspondent à ce que l'on trouve dans les données traitées par l'ETL)
 - Granularité temporelle (les faits correspondent à une agrégation de ce que l'on trouve dans les données traitées par l'ETL)

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 les sources (clé d'origine) 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 d'Origine/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

Nouvelle colonne d'historique

Type 3 (update)

ID	Customer	Country	Previous Country
1	Bob	United States	United Kingdom

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

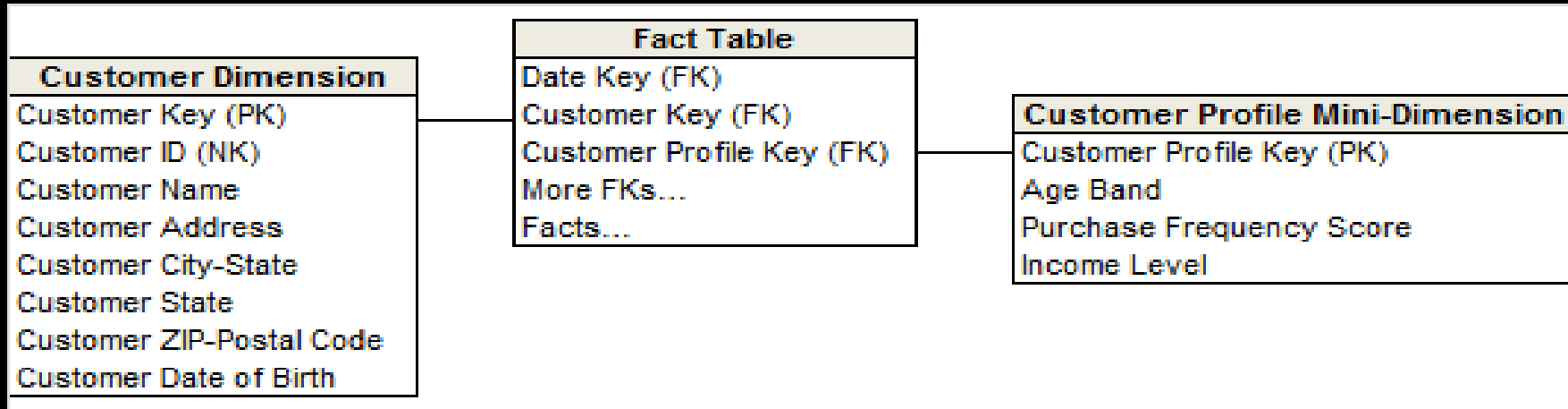
Clé d'Origine/clé ETL

Type 6 = Types 1+2+3

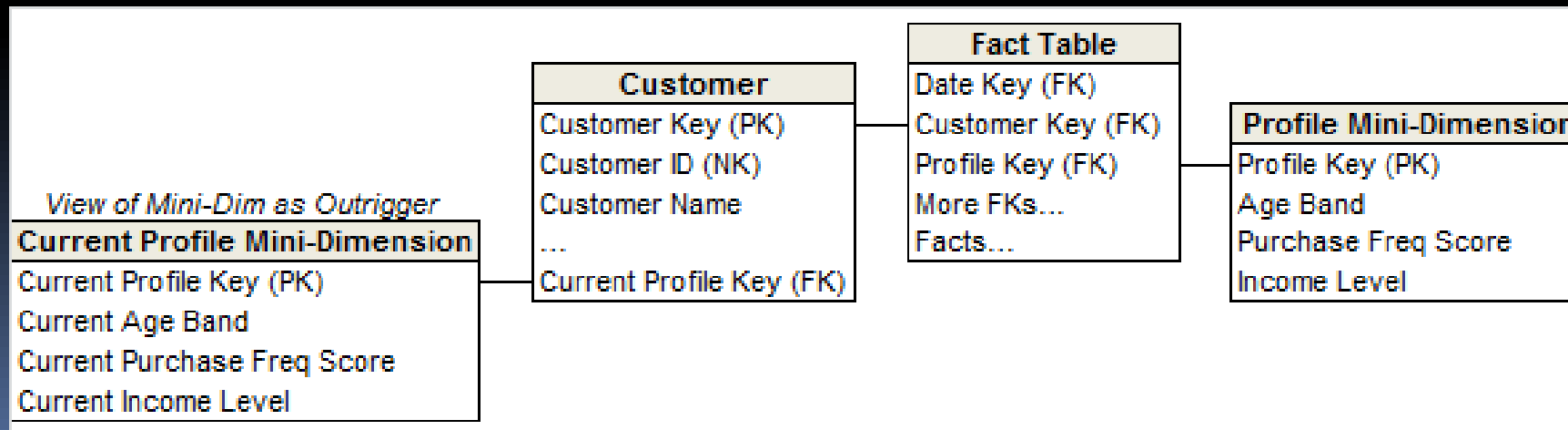
Surrogate Key	ID	Customer	Current Country	Historical Country	Current Flag	Record Start Date	Record End Date
1	1	Bob	United States	United Kingdom	0	01/01/2000	12/03/2014
2	1	Bob	United States	United States	1	12/03/2014	NULL

Modélisation RCD

Type 4 (minidimension)



Type 5 (minidimension +
current values)



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 pouvoir réaliser le **SID et le rendre performant**, il est nécessaire 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, modularisent la réalisation de dimensions changeantes et permettent le stockage de données intégrées)
 - 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) permettant d'établir la « provenance » de données ; mais pour des scénarios complexes, cela peut avoir des conséquences négatives si non correctement géré

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 certaines analyses pourraient ne plus être disponibles (mais d'autres les deviendraient) ; les inférences de faits pourraient être limitées, obligeant l'utilisation d'agrégats explicites



Clés de la table de faits

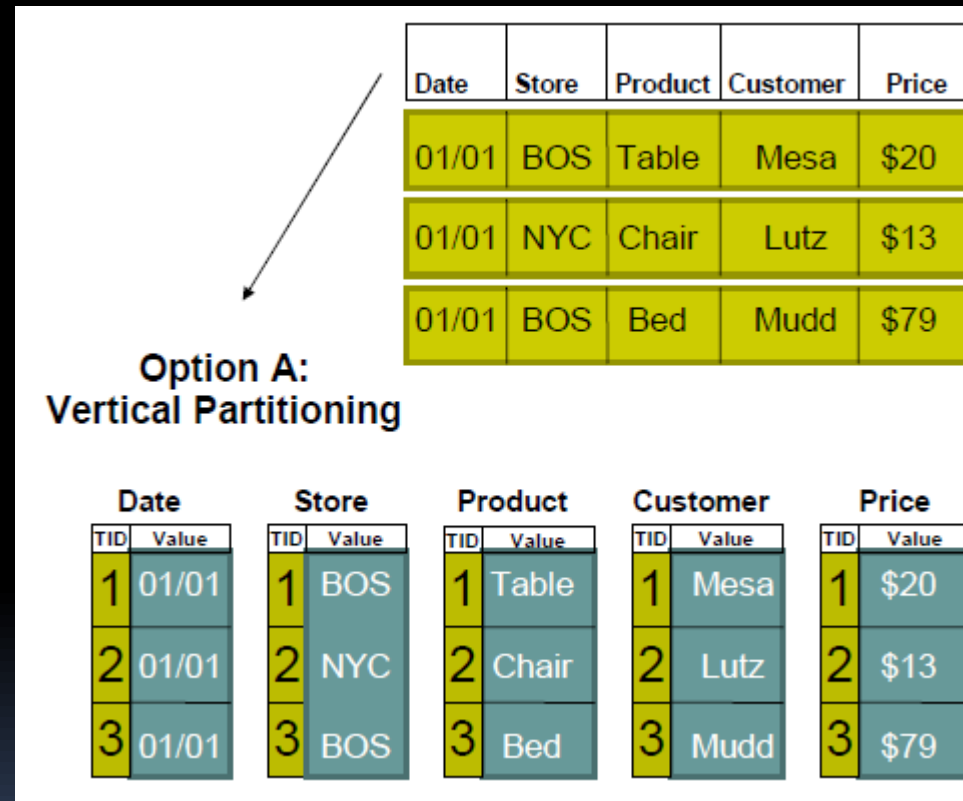
- La clé de la table de faits dépend donc de 3 paramètres
 - De la granularité de faits à stocker
 - Du type de schéma (flocon vs étoile)
 - De la prise en compte des dimensions changeantes
- Il est donc généralement nécessaire de choisir une clé « surrogate » pour pouvoir réaliser la table de faits
 - Cette clé pourrait éventuellement être la composition des clés étrangères vers les dimensions
- Généralement il n'est pas pertinent (même si éventuellement possible) d'utiliser des clés provenant du SIO pour la table de faits



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



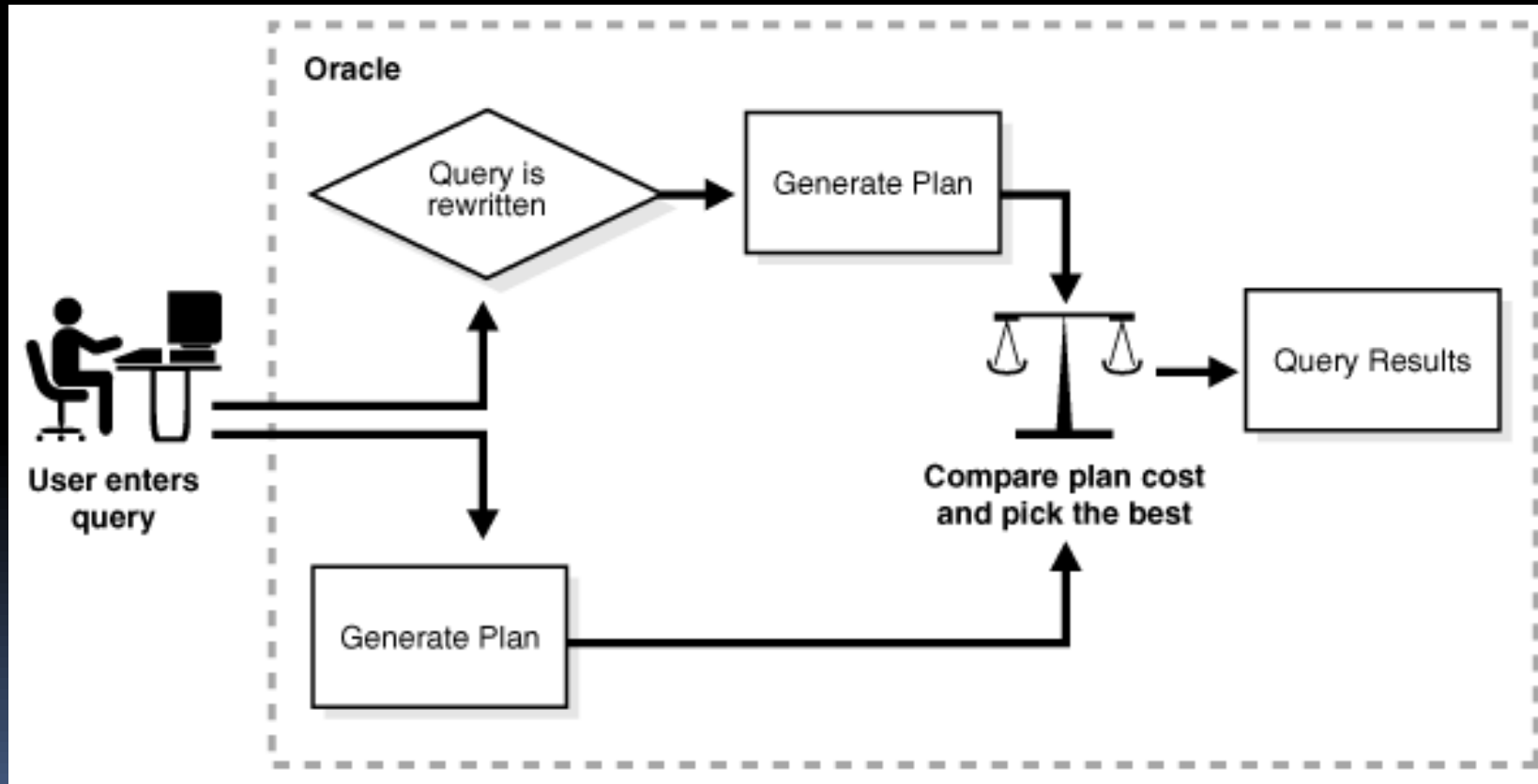
Exemple : ORACLE InMemory (utilisation du même RowID pour les projections)
CREATE TABLE purchases (date, store, product, customer, price) INMEMORY

Réécriture

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

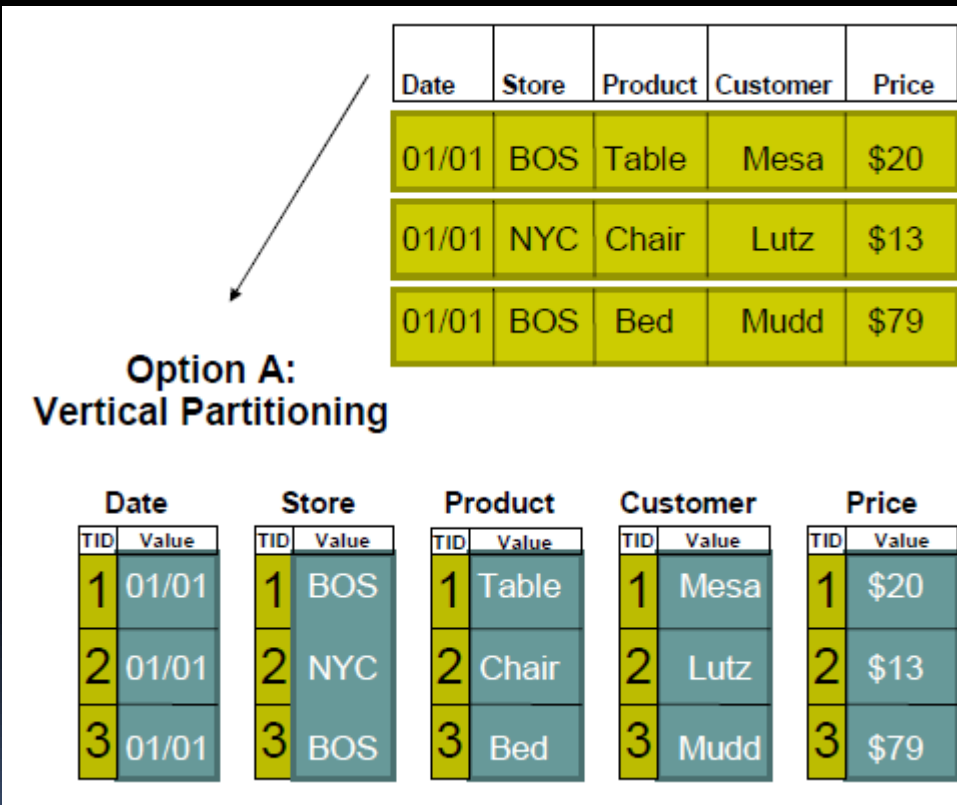
La requête saisie par l'utilisateur est réécrite automatiquement suivant la requête elle-même et les informations dont ORACLE dispose

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)

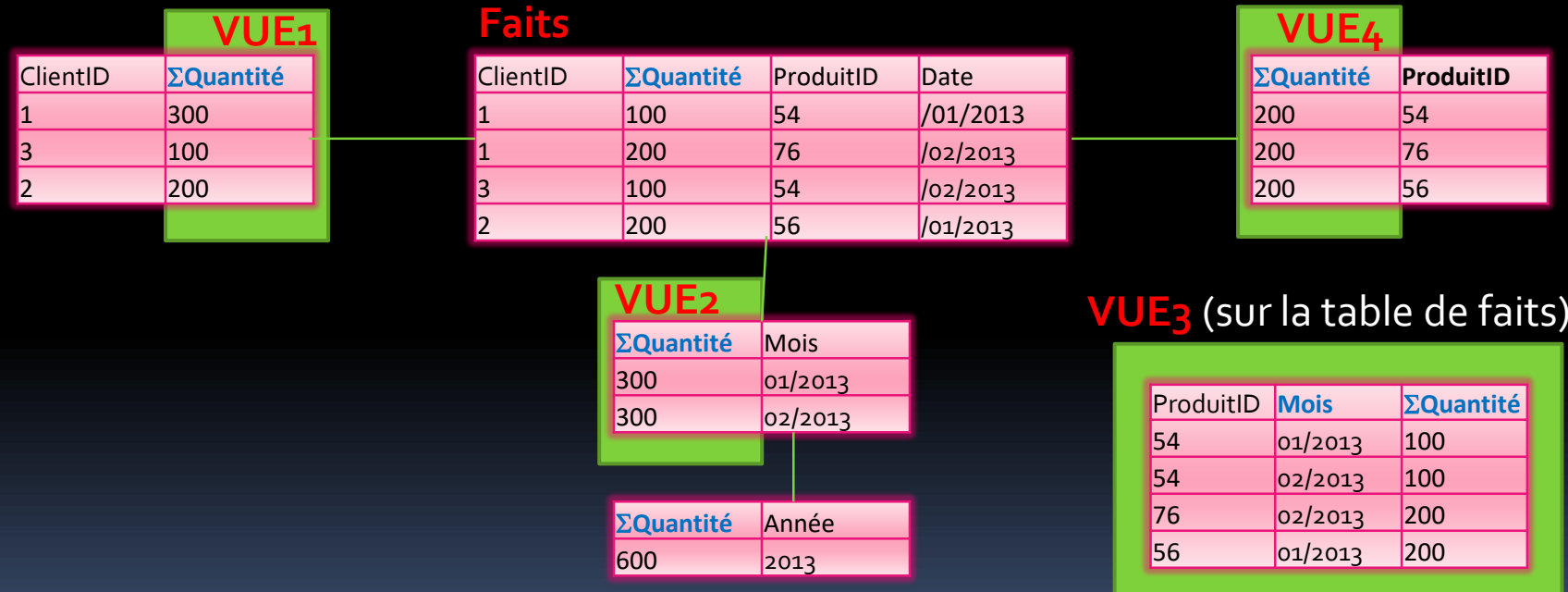
Select A.a, B.b from (Select B.b, B.id
from B where B.b=x), A where
A.id=B.id

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>

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*



Select produitID, Year(Date), sum(Σ quantité) **from Faits** group by produitID, Year(Date)

→ Select produitID, Year(mois), sum(Σ 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....;
```

Journalisation (LOG) pour le
« refresh fast »

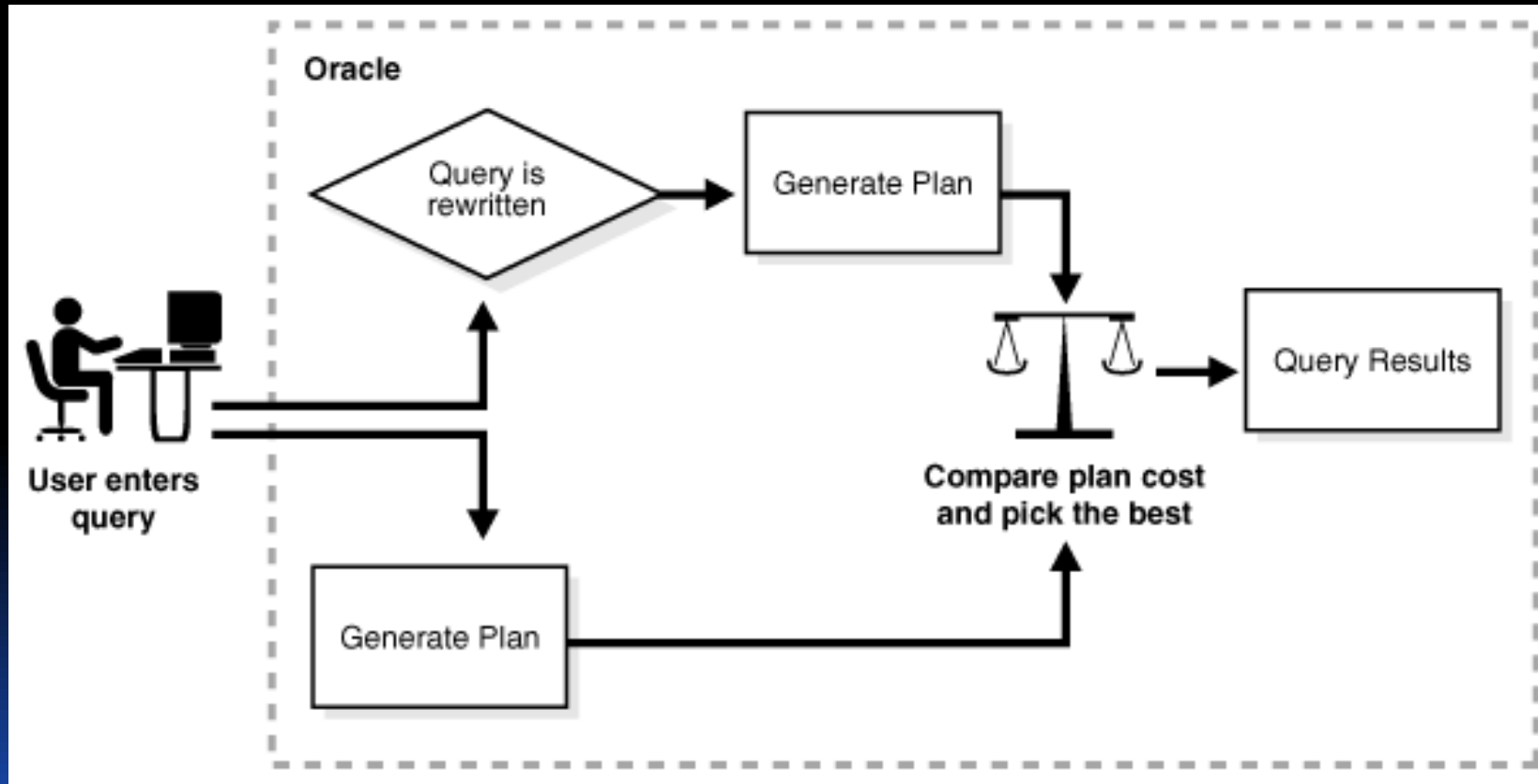
```
CREATE MATERIALIZED VIEW MatView1 REFRESH FAST ON COMMIT ENABLE QUERY REWRITE
AS
SELECT clientid,nom,
sum(quantité)
FROM commande,
Ventes, client
WHERE commande.cid=ventes.commande_cid and commande.clientid=client.id
GROUP BY client.id, nom;
```

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

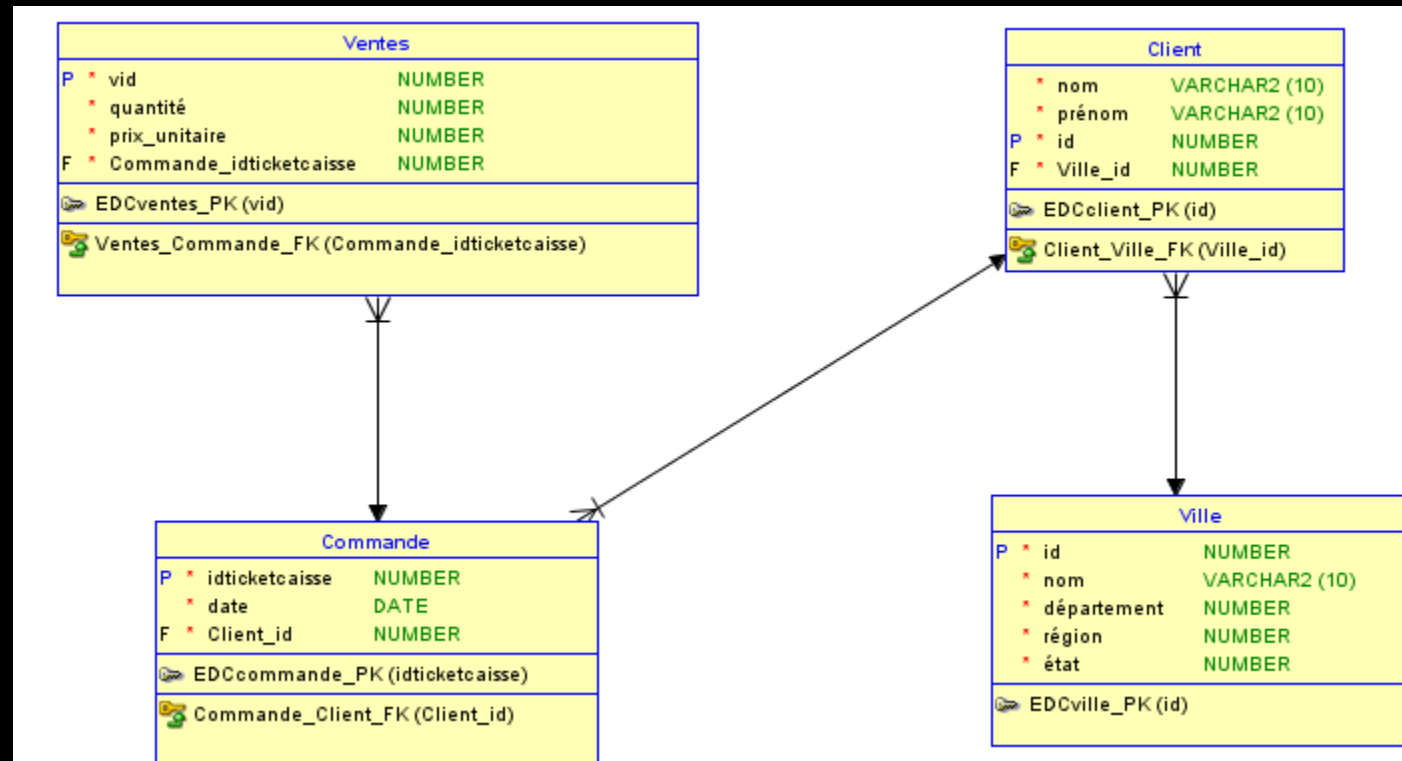


Le plan d'exécution de la requête est celui de base sans utiliser aucune information

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'opérateur et de données)

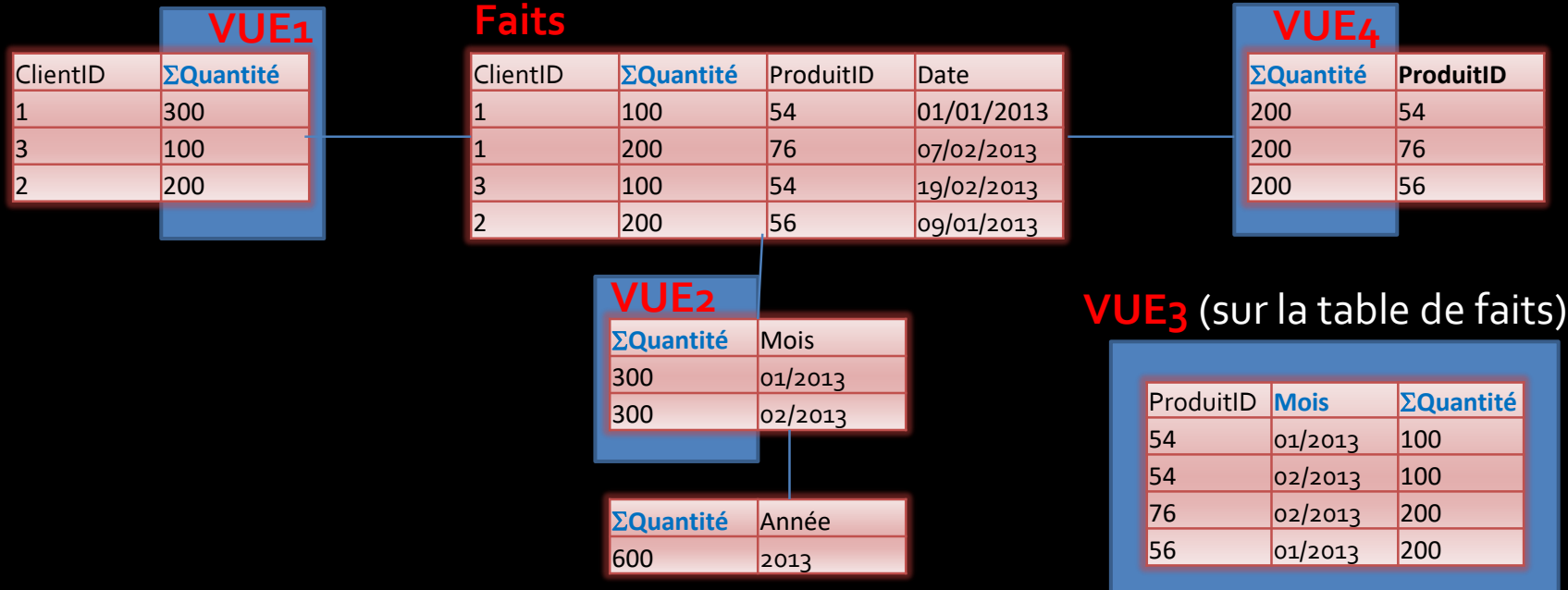
```
SELECT Y.villeid,  
       SUM(X.sommeqt)  
FROM (SELECT client.id as id,  
            SUM(quantité) as sommeqt  
      FROM commande,  
            ventes  
      WHERE commande.cid=ventes.commande_cid  
      GROUP BY client.id) X,
```

```
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



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

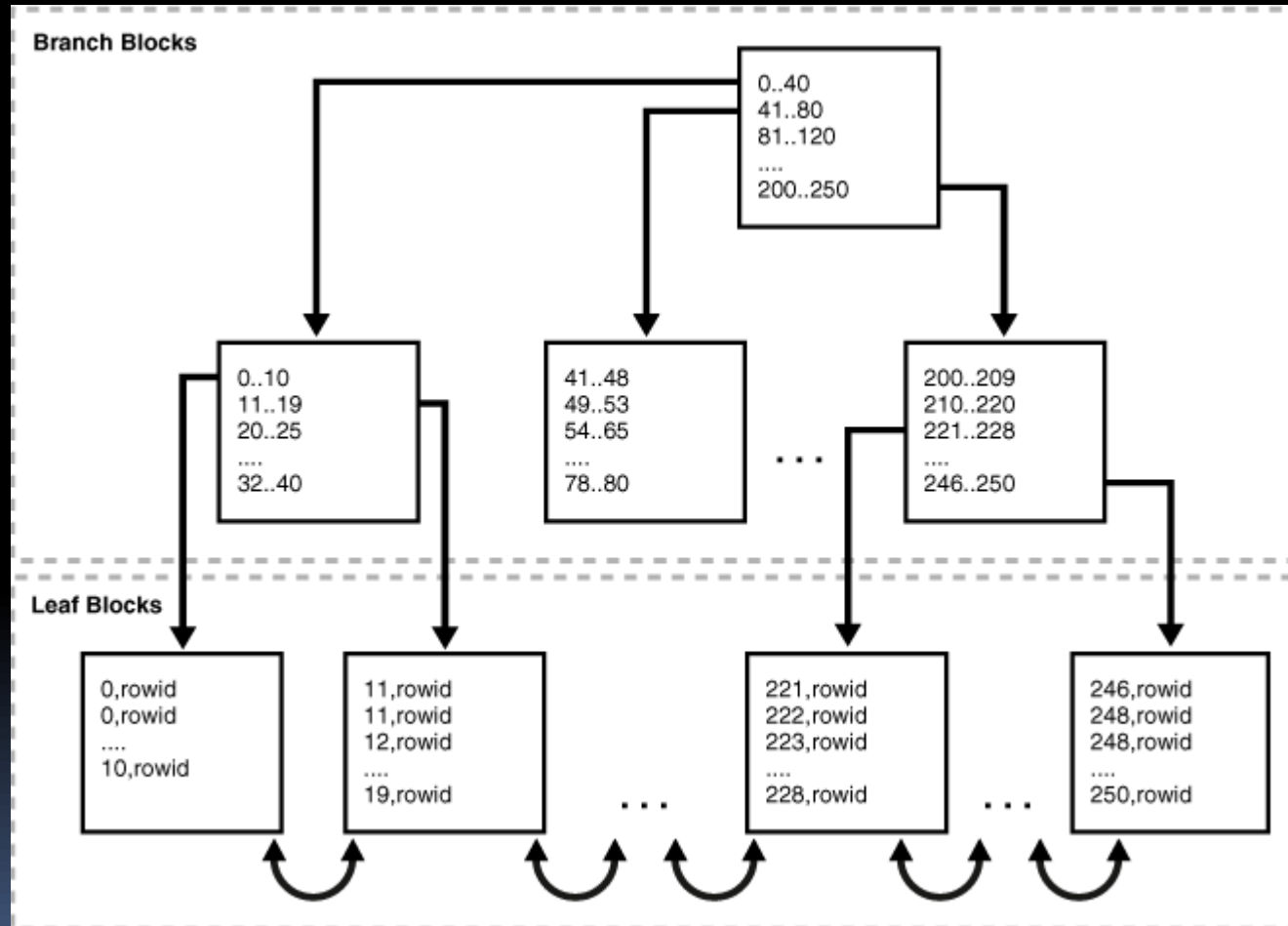
- Dans certains cas (simples) la table de faits peut être alimentée en données par la définition d'une vue matérialisées (au lieu d'un ETL)
- Cela a certaines implications sur la table de faits et sur la vue à cause des conditions strictes de fonctionnement des vues :
 - Indisponibilité éventuelle de certains opérateur 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

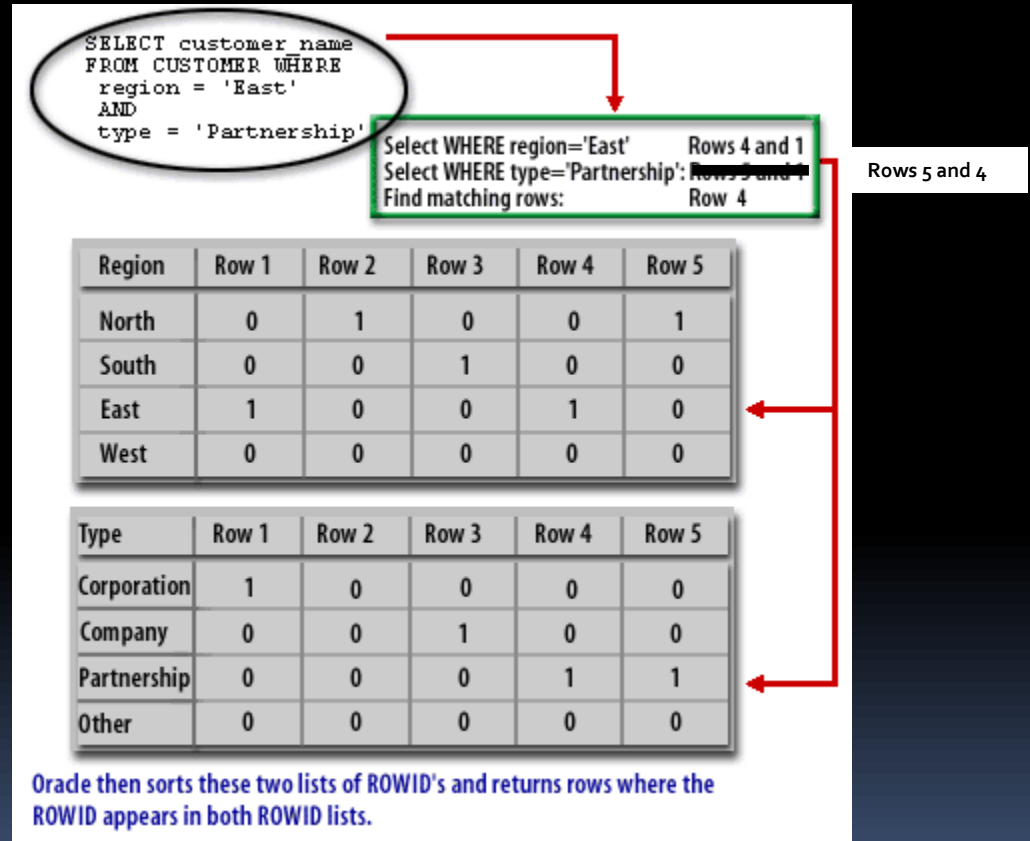


Index

Rowid
blocks

Index Bitmap (Ex. ORACLE)

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

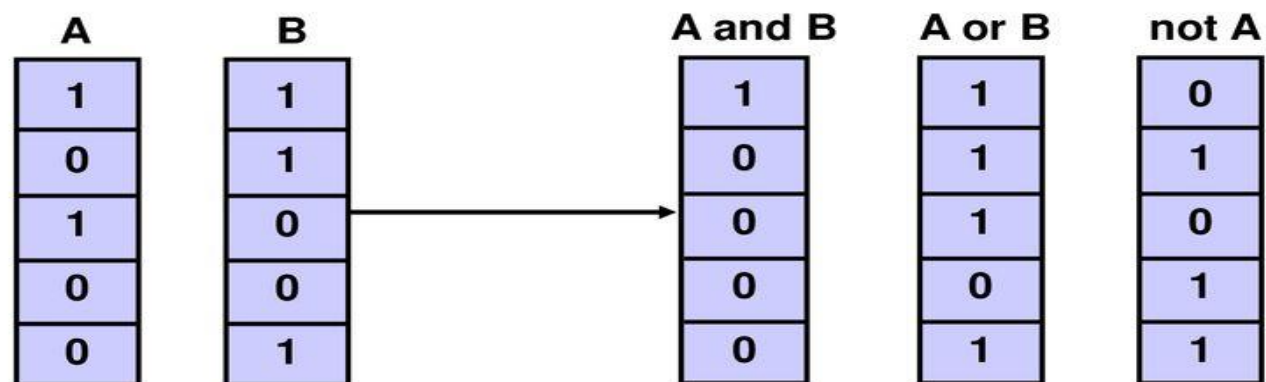


Opérations sur les index

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`



Index de jointure (Join index, Ex. ORACLE)

- 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
AAAQNKAFAAAAABSAAL	Vannes
AABQNHAFAAAABSAAL	Brest
BCAQNKAAFAAAAABTAAL	Nantes

	AAAQ NKAA FAAA ABSA AL	AABQ NHAF AAAA BSAAL	BCAQ NKAA FAAA ABTA AL	...
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'));
```

Table de faits : Calcul du stockage (par ligne, ORACLE)

Calculate the number of rows inside each of the dimensions:

Date dimension: 365 rows for 1 year

Product dimension: 100 rows (100 products)

Store dimensions: 2 rows (2 stores)

Customer dimension: 100000 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:

$365 * 100 * 2 * 100000 * 50 * 10 = 3650000000000$ 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 or degenerate dimension : 7

Number of measures: 8 (as an example)

Assume that the fact table takes 4 bytes (required for referring to about 100000 distinct customers) for a NUMBER column (all columns are typed NUMBER), and calculate the size of a single row:

$(7 + 8) * 4 \text{ bytes} = 60 \text{ bytes}$

Calculate the maximum data growth for a single year for the fact table: $3650000000000 \text{ rows} * 60 \text{ bytes} = 199 \text{ TiB (219TB)}$

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. What will be approximate size of this data warehouse in 5 years?

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) =

$(233600000 * 5) / (1024 * 1024) = 1113.89$ MiB

Size of fact table (gigabytes) = $1113.89 / 1024 = 1.089$ GiB



Partitionnement

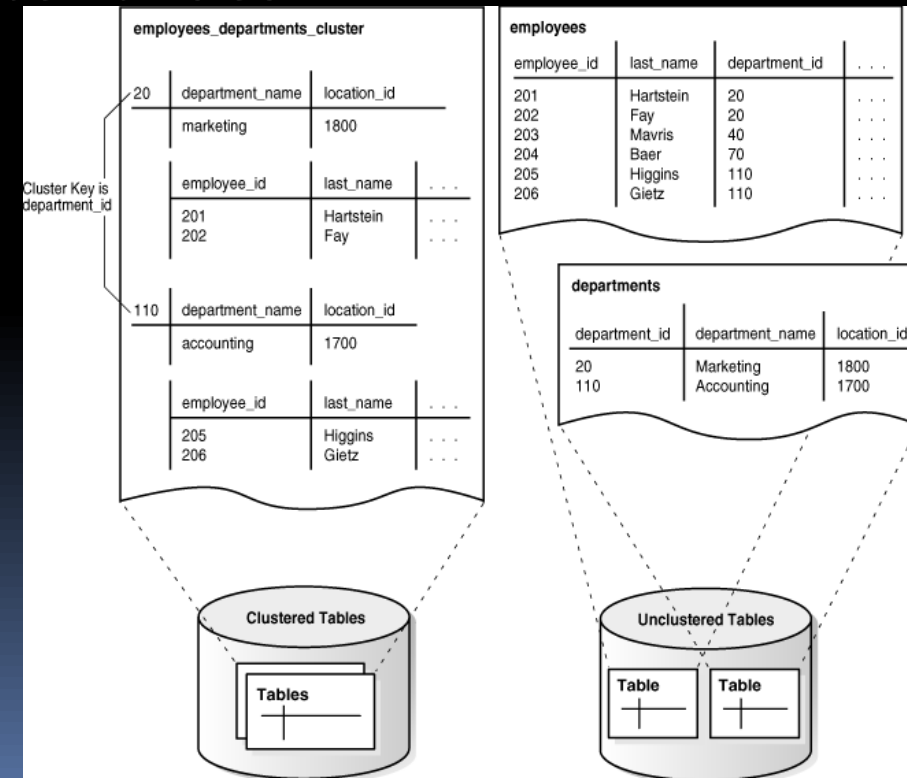
- Le partitionnement permet de créer des *sous-ensembles disjoints* de données (ayant ou pas une signification)
 - Le partitionnement est utile, entre autre, pour l'optimisation des requêtes, introduisant la possibilité d'accéder un des sous-ensembles de données (au lieu de toutes les données)
 - Le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

Partitionnement vertical (Ex.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 ou vues matérialisés

```
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 (Ex.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, aux vues matérialisées, et aux index :

<https://docs.oracle.com/en/database/oracle/oracle-database/12.2/vldbg/partition-create-tables-indexes.html#GUID-oCAB4231-E7DB-4245-9C43-C9CA352EC298>

Partitionnement par valeur (Ex. ORACLE)

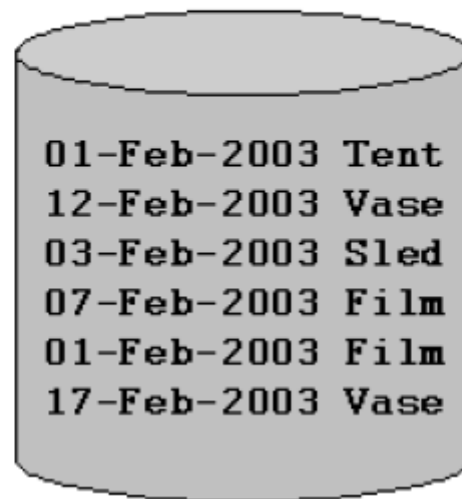
Range Partitioning by MONTH

Jan 2003



01-Jan-2003	Tent
02-Jan-2003	Book
03-Jan-2003	Sled
05-Jan-2003	Tent
01-Jan-2003	Film
11-Jan-2003	Vase

Feb2003



01-Feb-2003	Tent
12-Feb-2003	Vase
03-Feb-2003	Sled
07-Feb-2003	Film
01-Feb-2003	Film
17-Feb-2003	Vase

Mar2003



11-Mar-2003	Tent
03-Mar-2003	Book
03-Mar-2003	Sled
15-Mar-2003	Tent
10-Mar-2003	Film
12-Mar-2003	Film

Partitionnement par valeur (Ex. 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

Mot-clé

Partitionnement par hachage (Ex. ORACLE)

Hash Partitioning by Product

Partition 1	Partition 2	Partition 3	Partition 4
01-Jan-2003 Book 02-Feb-2003 Book 03-Feb-2003 Vase 05-Jan-2003 Book 01-Apr-2003 Book 11-Mar-2003 Vase	01-Jan-2003 Tent 12-Feb-2003 Tent 03-Jan-2003 Sled 07-Jan-2003 Sled 01-Mar-2003 Tent 17-Feb-2003 Sled	11-Apr-2003 Film 03-Mar-2003 Film 03-Jan-2003 Film 15-Mar-2003 Film 10-Jul-2003 Film 12-Mar-2003 Film	11-Apr-2003 Lamp 07-Jul-2003 Ball 01-Jan-2003 Lamp 12-May-2003 Hat 09-Aug-2003 Lamp 12-Mar-2003 Hat

Fonction d'hachage

- Une fonction de transformation $F: \mathbb{N} \rightarrow \{i, \dots, j\}$
- De préférence une F « à distribution uniforme » à savoir
$$P(F(x)=k)=P(F(x)=h), h \neq k$$
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 $\{i, \dots, j\}$ (par exemple ORACLE utilise une certaine fonction et conseille d'utiliser une cardinalité de l'ensemble $\{i, \dots, j\} = 2^m$)

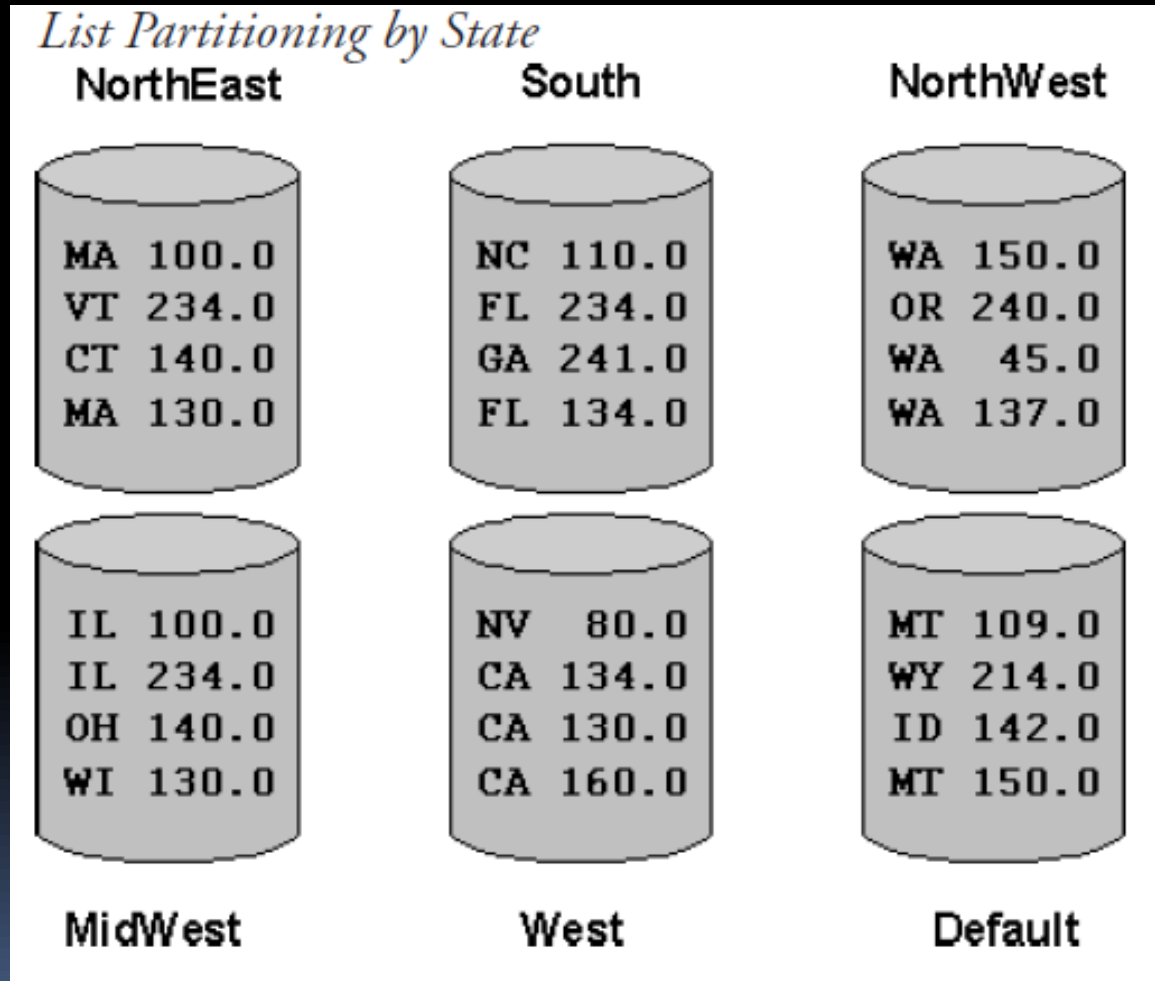
Partitionnement par hachage (Ex. ORACLE)

```
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 (Ex. ORACLE)



Partitionnement par liste (Ex. 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)

SIO (extrait)

Applications

Gestion
Relation
Client

Gestion
commandes

Gestion
production

Logistique

Sources
Internes

Databases

Sources
Externes

Data staging

ETL
processus

Back End

Metadonnées

Entrepôt
De données

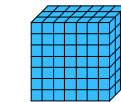
Magasins de
données

Serveur
OLAP

Front End

SID

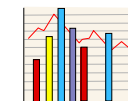
Outils OLAP



Outils de
Reporting



Statistiques



Outils
Data mining



simulation

MIS
DSS
EIS



Magasin de données

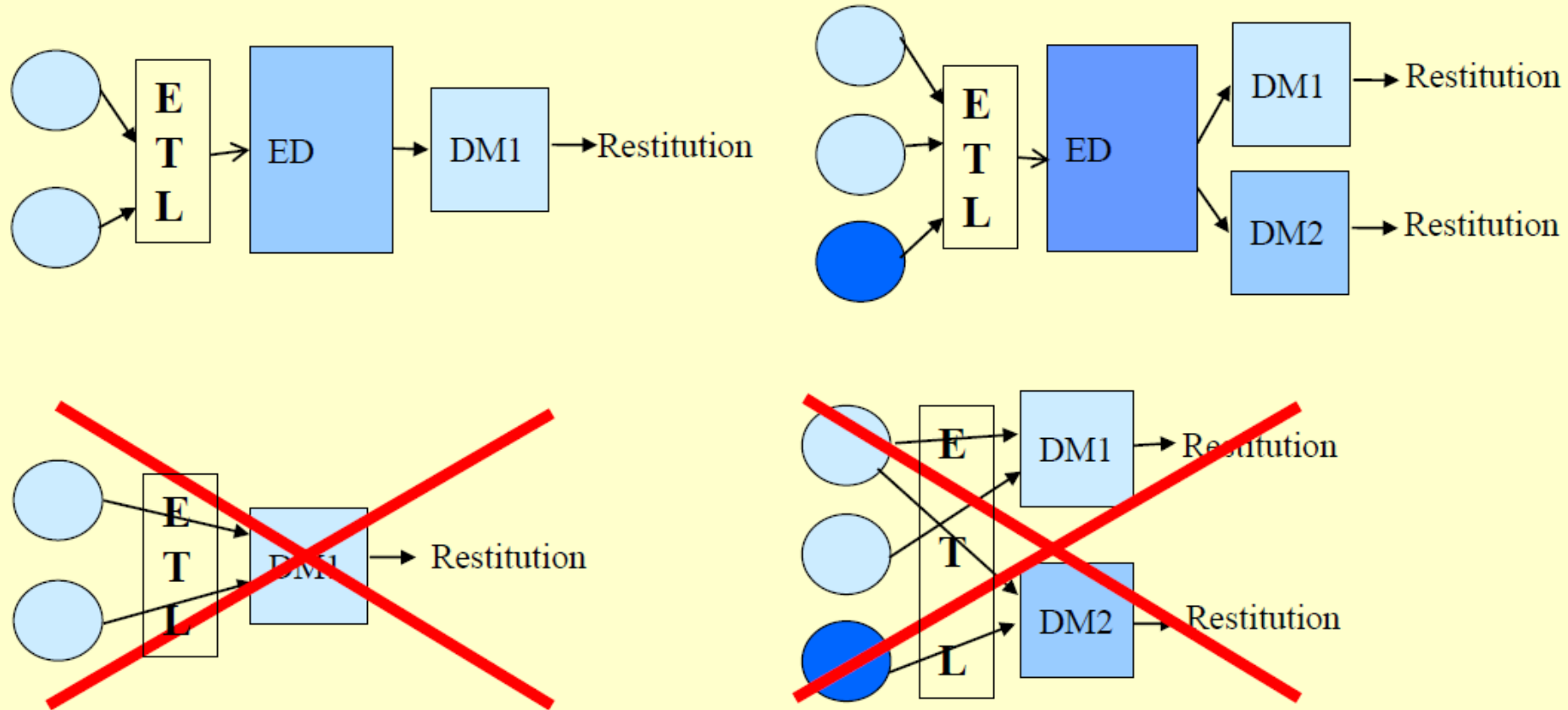
- Orienté vers une étude particulière, pour un traitement spécifique ; exemples :
 - Comportement de la clientèle → quels produits sont achetés le plus fréquemment par certains clients (types), pourquoi certains clients n'achètent plus certains produits, etc.
 - Impact des promotions sur les ventes d'un 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



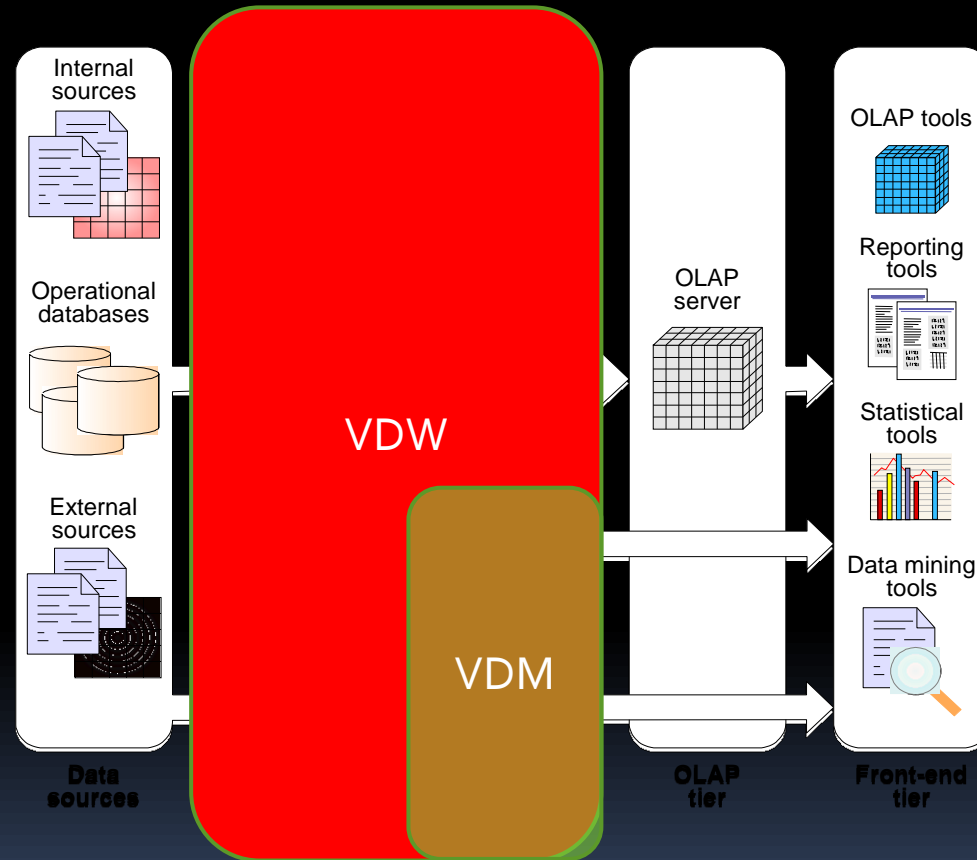
Focus sur les Architectures

- 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	prodlid	maglid	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	prold	magld	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

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

MOLAP (cuboide de faits)

FAITS+Agrégats

jour	prold	magld	solde
jour 2	p1	s1	44
		s2	4
		s3	NULL
	p2	s1	8
		s2	4
		s3	NULL
jour 1	p1	s1	12
		s2	NULL
		s3	50
	p2	s1	11
		s2	8
		s3	NULL

dimensions = 3

Faits memorisées dans des tableaux : accès direct aux données

Dimensions utilisées pour indexer les tableaux

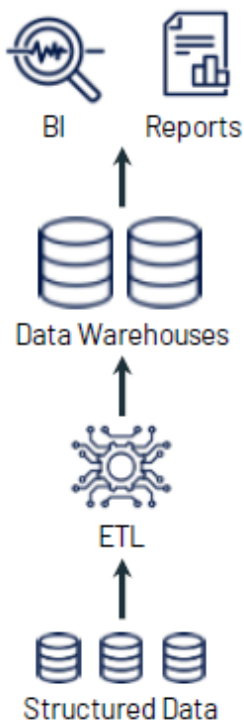
Langage de requêtes pour schémas ROLAP/MOLAP/HOLAP

- 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)

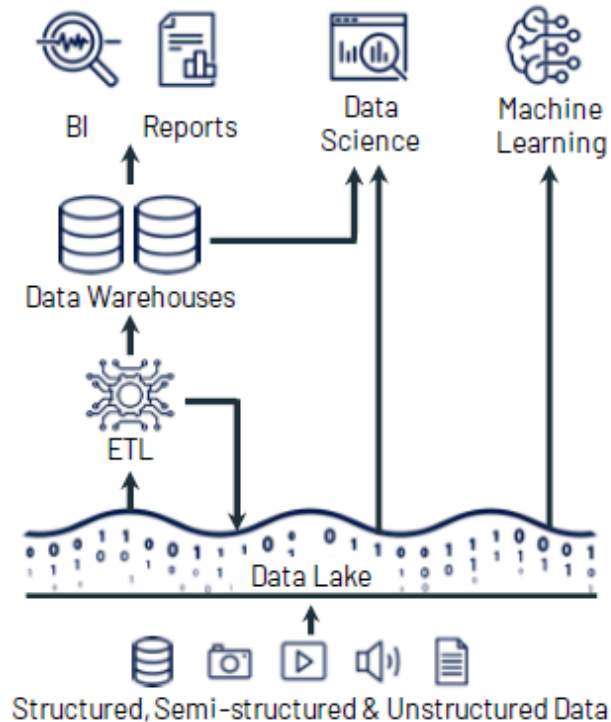
Tendances

- Entrepôt vs « datalake » et similaires :
 - Stockage de données hétérogènes en format « file » (CSV, JSON, Parquet....), actuellement dans le cloud, sans besoin d'accès via API JDBC/ODBC typiques
 - Évite l'obligation de la structuration et la latence propres aux entrepôts (car les données ne sont pas maîtrisées et demandent de l'exploration avant toute utilisation)
 - Permet éventuellement la compression de données, suivant le format « file » choisi
 - Les données sont difficilement interrogeables (langages procéduraux) et l'optimisation est manquante (sauf celle induite par la parallélisation)
- ETL et « data fabric »
- Introduction d'un « metric layer »

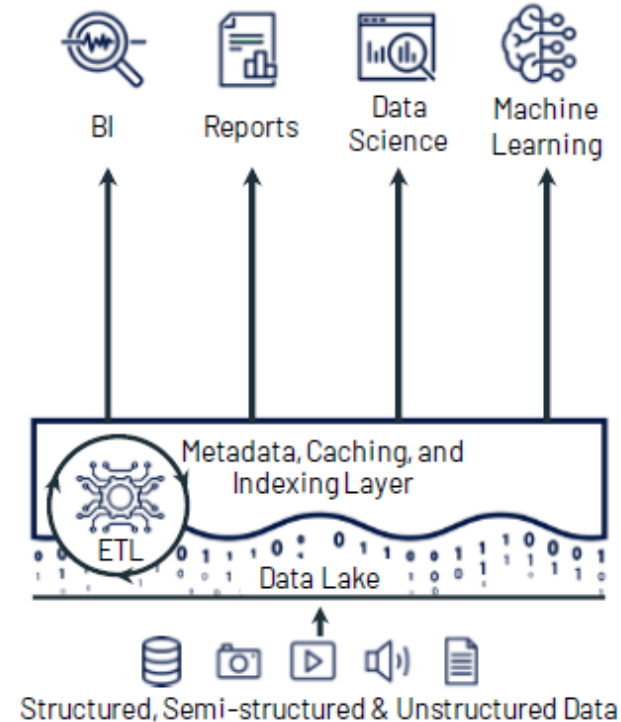
Tendances



(a) First-generation platforms.



(b) Current two-tier architectures.



(c) Lakehouse platforms.

https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf



Tendances

- Le datalake évolue reprenant les concepts classiques disponible en tout SGBD
 - Table (structure virtualisée) disponible sur le stockage physique en « file »
 - Operations classiques SQL, transactions
 - Partitionnement comme base de la parallélisation
 - Clustering pour rendre rapide l'accès aux données

https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

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>)
- <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9442782/>
- <https://www.quantmetry.com/blog/pourquoi-faut-il-modeliser-les-donnees/>



Synthèse

- Système informatique décisionnel : motivation, modèles de données (conceptuel, logique, physique ; ROLAP (en particulier), molap, holap)
 - Mesures et leurs propriétés (en particulier, additivité)
 - Dimensions, hiérarchies
 - Mécanismes d'optimisation
- 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