Datawarehouse and OLAP

Datawarehousing

Syllabus, materials, notes, etc.

See http://www.info.univ-tours.fr/~marcel/dw.html

today

architecture

ETL

refreshing

warehousing projects

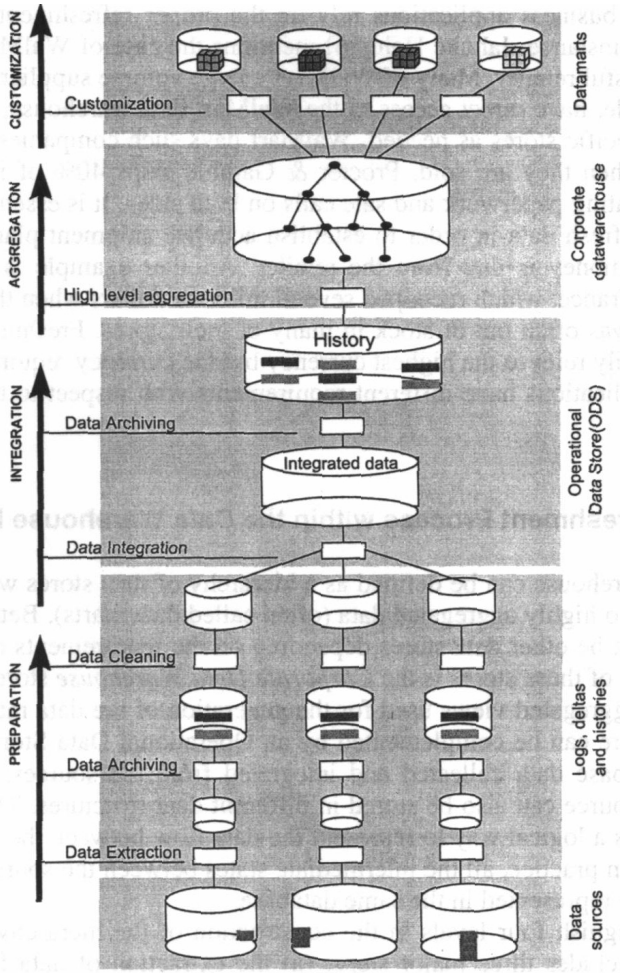
architecture

architecture

que faut-il faire ?

1. regrouper les données sources
2. concevoir le schéma de l'entrepôt
3. remplir l'entrepôt
4. maintenir l'entrepôt

architecture



architecture

vue logique de l'entrepôt : une hiérarchie de dépôts

ODS *O*perational *D*ata *S*tore
regroupe les données intégrées
récupérées des sources

CDW *C*orporate *D*ata *W*arehouse
regroupe les vues agrégées

4 niveaux de construction

1. préparation des données
2. intégration des données
3. agrégation des données
4. personnalisation des données

préparation et intégration des données

jusqu'à 80 % du temps de développement d'un entrepôt

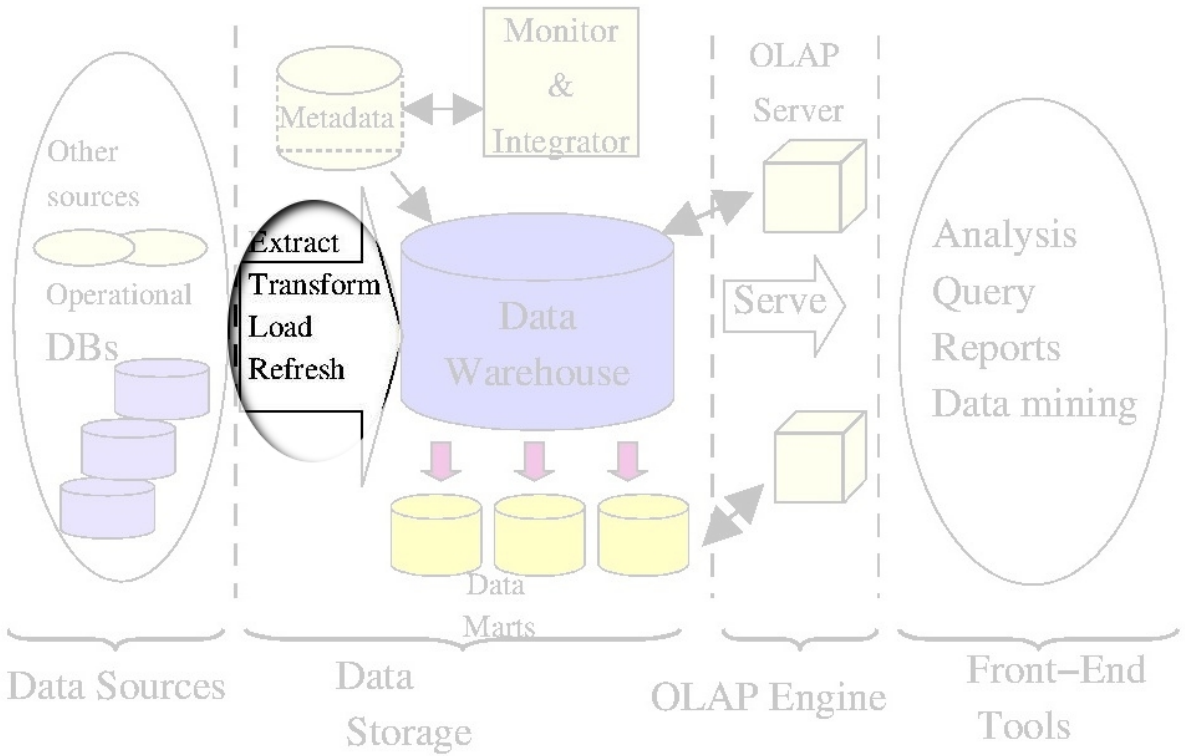
1. préparation
   - 1.1 identification des données à extraire des sources
   - 1.2 extraction de ces données
   - 1.3 nettoyage des données extraites
   - 1.4 archivage éventuel
2. intégration
   - 2.1 définition d'un format commun
   - 2.2 transformation des données vers ce format
   - 2.3 stockage dans l'ODS

agrégation et personnalisation

préparation à la restitution

- agrégation
  - calcul de vues agrégées
  - définition des indexes
  - stockage dans le CDW
- personnalisation
  - construction de magasin de données (data marts)
  - construction de cubes de données
  - construction des présentations requises par les utilisateurs

ETL


Monitor
&
Integrator
OLAP
Server
Metadata
Other
sources
Operational
DBs
Extract
Transform
Load
Refresh
Data
Warehouse
Serve
Analysis
Query
Reports
Data mining
Data
Marts
Data Sources
Data
Storage
OLAP Engine
Front–End
Tools

ETL tools (Extract Transform Load)

support et/ou automatisation des tâches suivantes :

extraction	méthode d'accès aux différentes sources
nettoyage	recherche et résolution des inconsistances dans les sources
analyse	e.g., détection de valeurs non valides ou inattendues
transformation	entre différents formats, langages, etc.
chargement	alimentation de l'ODS

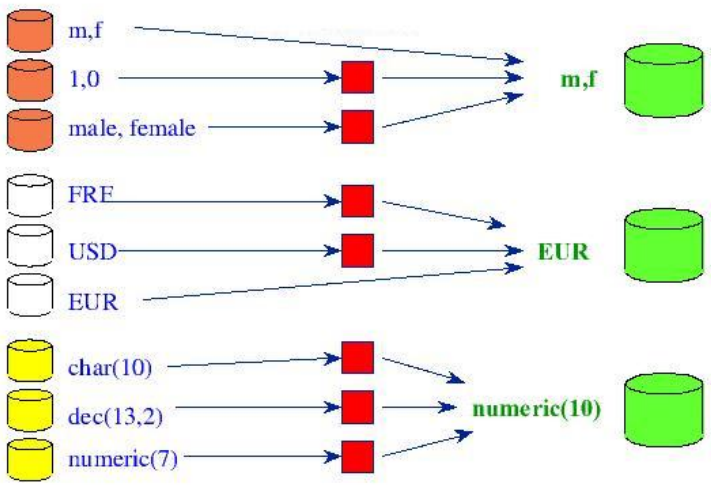
extraction

un extracteur (wrapper) est associé à chaque source

exemple :

- ODBC
- Oracle OCI
- JDBC

nettoyage et transformation



nettoyage

5 à 30 % des données des BD commerciales sont erronées

une centaine de type d'inconsistances ont été répertoriées

but : résoudre le problème de *consistance des données au sein de chaque source*

types d'inconsistances

- présence de données fausses dès leur saisie
  - faute de frappe
  - différent format dans une même colonne
  - texte masquant de l'information (e.g., "N/A")
  - valeur nulle
  - incompatibilité entre la valeur et la description de la colonne
  - duplication d'information
- persistance de données obsolètes
- confrontation de données sémantiquement équivalentes mais syntaxiquement différentes

nettoyage

un outil de nettoyage comprend

- des fonctions d'analyse
- des fonctions de normalisation
- des fonctions de conversion
- des dictionnaires de synonymes ou d'abréviations

normalisation, conversion, dictionnaires, ...

définition de table de règles

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

utilisation d'expression régulière, suppression de doublons, de valeur nulle, ...

transformation

but : suppression des *incohérences sémantiques entre les sources*

pouvant survenir lors de l'intégration

- des schémas
- des données

intégration des schémas

- problème de modélisation
  - différents modèles de données sont utilisés
- problèmes de terminologie
  - un objet est désigné par 2 noms différents
  - un même nom désigne 2 objets différents
- incompatibilités de contraintes
  - 2 concepts équivalents ont des contraintes incompatibles

intégration des schémas

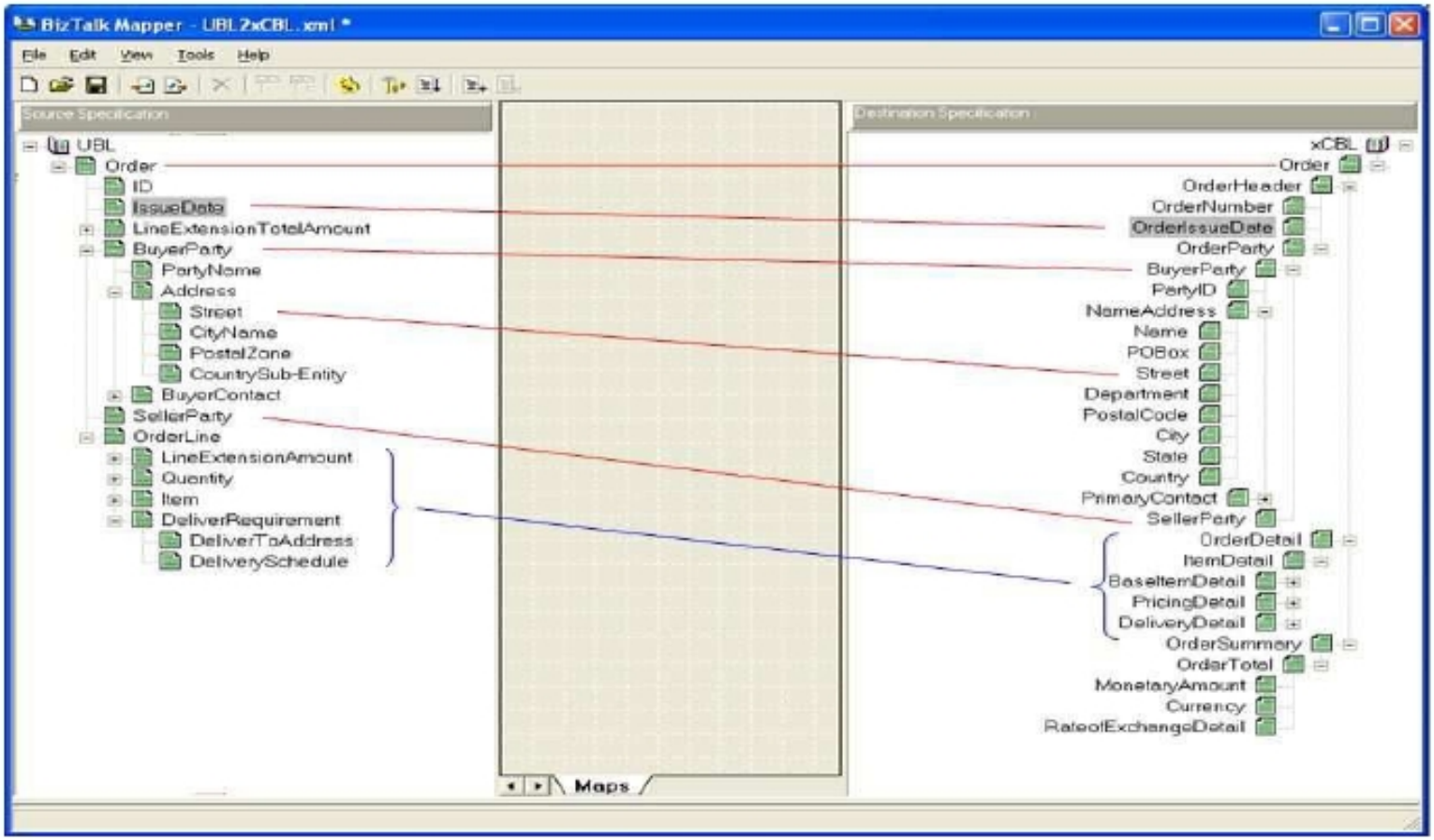
- conflit sémantique
  - choix de différents niveaux d'abstraction pour un même concept
- conflits de structures
  - choix de différentes propriétés pour un même concept
- conflits de représentation
  - 2 représentations différentes choisies pour les mêmes propriétés d'un même objet

intégration des schémas

- demande une solide connaissance de la sémantique des schémas
- peu traité par les produits du marché
- nombreux travaux de recherche

généralement faite à la main...

intégration des schémas



intégration des schémas

les heuristiques de réconciliation de schémas se basent sur l'existence de similarités entre :

- noms de tables et d'attributs
- types de données
- instances
- structure des schémas
- contraintes d'intégrité

équivalence de champs

deux chaînes sont équivalentes si l'une est le préfixe de l'autre

pour 2 champs c_1 et c_2

- $n(c_i) :=$ nombre de chaînes de c_i
- $n_e :=$ le nombre de chaînes équivalentes

compatibilité $:= n_e/((n(c_1) + n(c_2))/2)$

équivalence d'enregistrements

fusion d'enregistrements

pour tous les tuples concernés
si noss1 = noss2 *et* nom1 = nom2
fusionner personne1 et personne2
si (noss1 = noss2 *ou* nom1 = nom2) *et* adresse1 = adresse2
fusionner personne1 et personne2
...

chargement

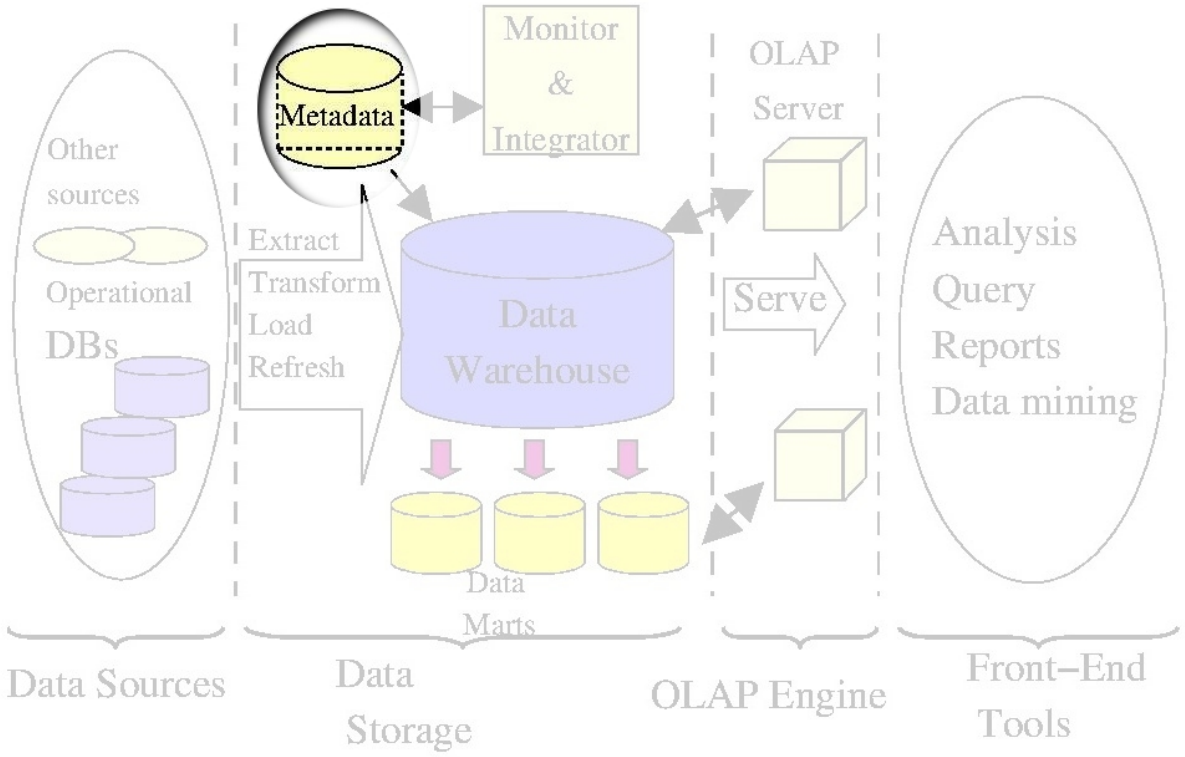
définitions de vues relationnelles sur les données sources

matérialisation des vues dans l'entrepôt

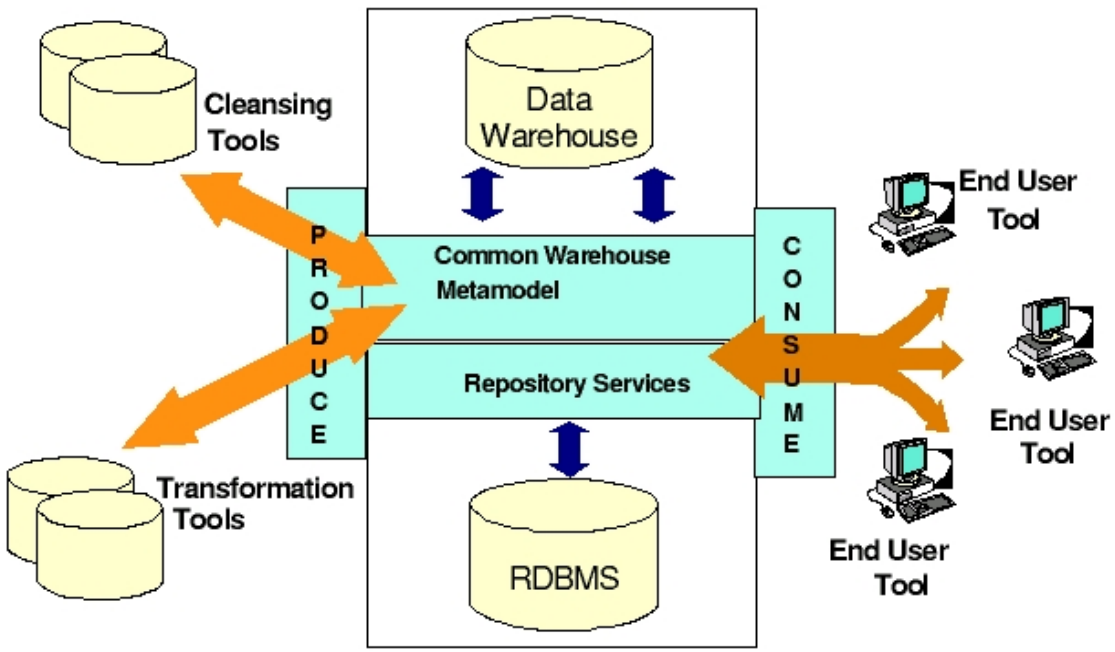
plus

- tris
- consolidations (pré-agrégation)
- indexation
- partitionnement des données
- enregistrement de méta-données
- ...

méta-données



méta-données



définition : toutes les informations nécessaires pour la construction et l'administration de l'entrepôt

méta-données

- informations présentes dans l'entrepôt
  - données source
  - données dérivées, dimensions, hiérarchies
  - contraintes d'intégrités
  - schéma de l'entrepôt
  - indexes, partitions
  - requêtes prédéfinies
  - ...

méta-données

- informations d'administration
  - règles de nettoyage, transformation, extraction
  - politique de rafraîchissement
  - sécurité
  - monitoring, statistiques
  - traçage des données
  - ...

méta-données

chaque composant de l'entrepôt

- fournit des méta-données
- doit connaitre celles des autres composants
- doit savoir où ces méta-données sont situées

une BD est dédiée aux méta-données

méta-données

exemple de standard : *C*ommon *W*arehouse *M*etamodel

- proposé par l'OMG
- basé notamment sur UML, XML
- conçu par IBM, Unisys, NCR, Oracle, Hyperion, ...

architecture typique (1)

peut être distribué pour des raisons de

- taille
- équilibrage de charge
- disponibilité

ce qui suppose une

- réplications des méta-données
- administration centrale

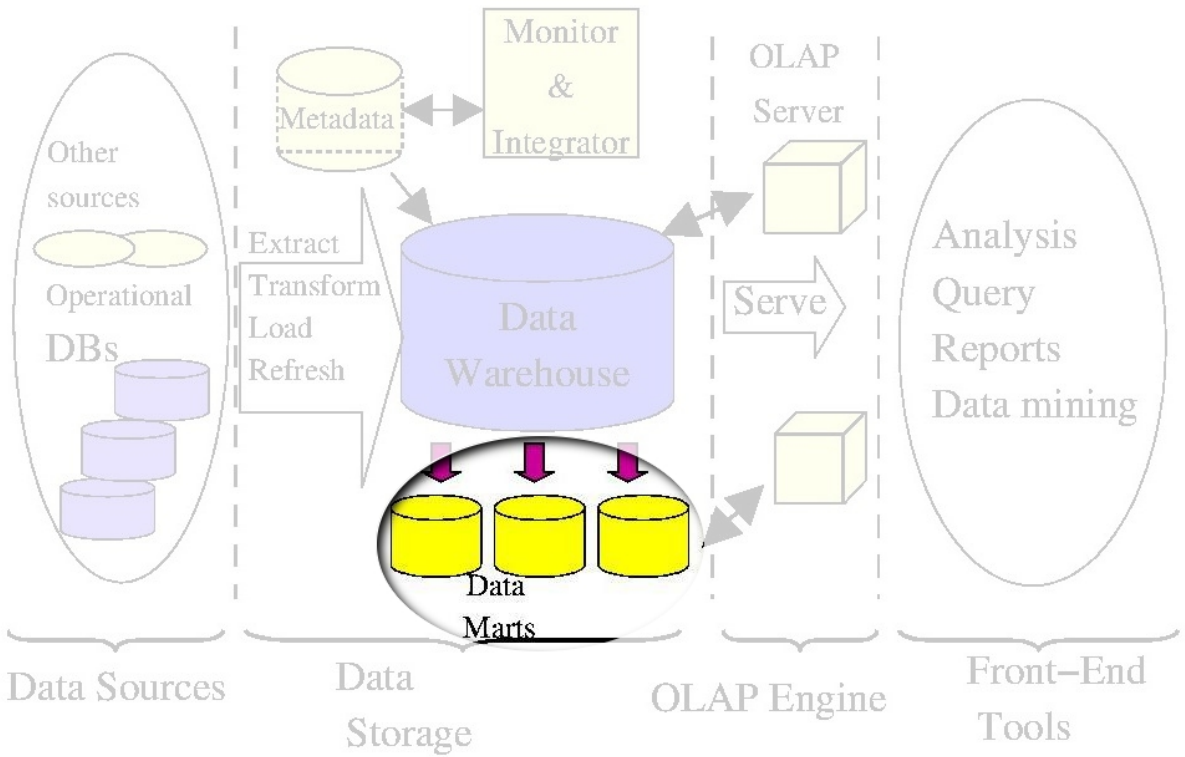
architecture typique (2)

fédération de petits entrepôts (data marts)

lorsqu'un seul entrepôt est trop couteux

- un data mart particulier à un département/secteur
- stockage décentralisé
- administration décentralisée

data mart

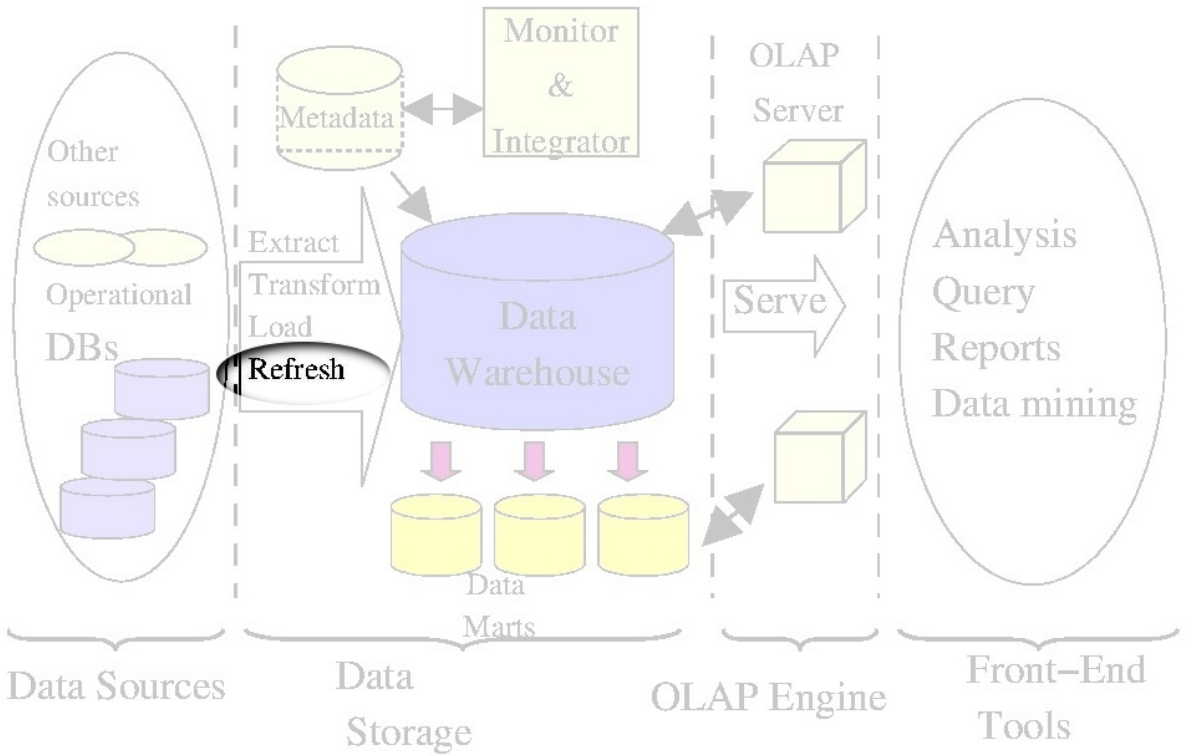

Monitor
&
Integrator
Metadata
OLAP
Server
Other
sources
Operational
DBs
Extract
Transform
Load
Refresh
Data
Warehouse
Serve
Analysis
Query
Reports
Data mining
Data
Marts
Data Sources
Data
Storage
OLAP Engine
Front–End
Tools

data warehouse / data mart

caractéristiques	data warehouse	data mart
utilisateur	toute l'entreprise	un département
BD	SQL type serveur	BD MD, OLAP
échelle du modèle de données	entreprise	département
champs applicatifs	multi-sujet	quelques sujets
sources de données	multiples	quelques unes
stockage	plusieurs BD distribuées	une BD
taille	$>$ 100 Go	10 à 20 Go
temps de mise en place	9 à 18 mois	6 à 12 mois
coût	plusieurs MF	500 KF à 3 MF
matériel	unix	NT petit serveur

refreshing

refreshing


Monitor
&
Integrator
Metadata
OLAP
Server
Other
sources
Operational
DBs
Extract
Transform
Load
Refresh
Data
Warehouse
Serve
Analysis
Query
Reports
Data mining
Data
Marts
Data Sources
Data
Storage
OLAP Engine
Front–End
Tools

refreshing

3 approches

1. reconstruction périodique
   - la plus simple
   - la plus longue
   - suppose une longue période d'indisponibilité

refreshing

2. mise à jour périodique
   - volume de données concerné plus petit
   - algorithmes plus complexes que pour une reconstruction
3. mise à jour instantanée
   - nécessite de nombreuses communications

ne pas reconstruire

rafraîchissement périodique et de manière incrémentale

- prise en compte des changements des sources
- suppression des données anciennes

détection des changements

dépend des sources

- triggers utilisés pour déclencher la mise à jour
- exploitation des logs des changements
- extraction des changements pertinents par requêtes
- comparaison de différentes images de la sources

comparaison d'image de la source : principe

F1 et F2 deux images de la source
ensemble d'enregistrements de la forme $\langle \underline{K}, B \rangle$

- calculer F1 - F2 et F2 - F1
- déduire $\langle insert, K, B \rangle$ et $\langle delete, K, B \rangle$
- calculer la jointure de F1 et F2
- sélectionner les enregistrements où la partie B contient des différences
- déduire $\langle update, K, B \rangle$

maintenance de vue

équipe de Stanford, conférence SIGMOD'95

contexte

- la source signale les mises à jour
- l'entrepôt questionne la source
- la source envoie les données concernées
- l'entrepôt met la vue à jour

maintenance de vue

cas simple

- une seule vue
- une seule source
- vue définie par $V = \pi_{proj}(\sigma_{cond}(r_1 \times r_2 \times \ldots \times r_n))$

extraction de changements par requête

sources

r_1	W	X
	1	2

r_2	X	Y
	2	4

vue de l'entrepôt $V = \pi_Y(r_1 \bowtie r_2)$

l'entrepôt contient $MV = \{(4)\}$

la source prévient l'entrepôt : insertion de (2,3) dans r_2

l'entrepôt envoie la requête : $Q_1 = \pi_Y(r_1 \bowtie (2,3))$

extraction de changements par requête

sources

r_1	W	X
	1	2

r_2	X	Y
	2	4
	2	3

vue de l'entrepôt $V = \pi_Y(r_1 \bowtie r_2)$

l'entrepôt contient $MV = \{(4),(3)\}$

extraction de changements par requête

sources

r_1	W	X
	1	2

r_2	X	Y

entrepôt $V = \pi_{W,Y}(r_1 \bowtie r_2)$ contenant $MV = \emptyset$

la source prévient l'entrepôt : insertion de (2,3) dans r_2

l'entrepôt envoie la requête : $Q_1 = \pi_{W,Y}(r_1 \bowtie (2,3))$

la source prévient l'entrepôt : insertion de (4,2) dans r_1

extraction de changements par requête

sources

r_1	W	X
	1	2
	4	2

r_2	X	Y
	2	3

évaluation de Q_1 : $MV = \{(1,3),(4,3)\}$

l'entrepôt envoie la requête : $Q_2 = \pi_{W,Y}((4,2) \bowtie r_2)$

évaluation de Q_2 : $MV = \{(1,3),(4,3),(4,3)\}$

solutions possibles

- verrouillage des sources pour la mise à jour de l'entrepôt
  - contraignant pour les sources
- recalcul de la vue
  - coûteux en temps et en ressources
- garder des copies de chaque relation impliquée dans une vue
  - coûteux en espace et en propagations de mises à jour

en fait

problème : requêtes évaluées

- à la source
- *après* changement de l'état de la source

$\rightsquigarrow$ compensation à la demande (eager compensating) :
compenser l'effet de la mise à jour par requêtes

Eager Compensating Algorithm (ECA)

notations

- U : mise à jour
- Q : requête
- MV : vue matérialisée
- A : résultat d'une requête
- t : tuple
- r : relation
- ss : état courant de la source
- $V < U >$: expression V où une relation est remplacée par U
- UQS : requêtes envoyées non traités

tuples signés

+t : tuple inséré ou existant

-t : tuple supprimé

$pos(r) = \{+t \in r\}$

$neg(r) = \{-t \in r\}$

$r_1 + r_2 = (pos(r_1) \cup pos(r_2)) - (neg(r_1) \cup neg(r_2))$

$r_1 - r_2 = r_1 + (-r_2)$

événements

- S_up : mise à jour sur la source
- S_qu : requête sur la source
- W_up : mise à jour pour l'entrepôt
- W_ans : réponse pour l'entrepôt

ECA : à la source

S_up_i exécuter la mise à jour U_i
envoyer U_i à l'entrepôt
signaler l'événement W_up_i à l'entrepôt

S_qu_i recevoir la requête Q_i
$A_i = Q_i[ss_i]$
envoyer A_i à l'entrepôt
signaler l'événement W_ans_i à l'entrepôt

ECA : à l'entrepôt

$COLLECT = \emptyset$
$UQS = \emptyset$

W_up_i recevoir la mise à jour U_i
$Q_i = V < U_i > - \sum_{Q_j \in UQS} Q_j < U_i >$
envoyer Q_i à la source
$UQS = UQS + Q_i$
signaler l'événement S_qu_i à la source

W_ans_i recevoir la réponse A_i
$COLLECT = COLLECT + A_i$
si $UQS = \emptyset$
alors $MV = MV + COLLECT$
$COLLECT = \emptyset$
sinon rien

extraction de changements par requête

sources

r_1	W	X
	1	2

r_2	X	Y

entrepôt $V = \pi_{W,Y}(r_1 \bowtie r_2)$ contenant $MV = \emptyset$

la source prévient l'entrepôt : insertion de (2,3) dans r_2

l'entrepôt envoie la requête : $Q_1 = \pi_{W,Y}(r_1 \bowtie (2,3))$

la source prévient l'entrepôt : insertion de (4,2) dans r_1

extraction de changements par requête

sources

r_1	W	X
	1	2
	4	2

r_2	X	Y
	2	3

l'entrepôt envoie la requête :
$Q_2 = \pi_{W,Y}((4,2) \bowtie r_2) - \pi_{W,Y}((4,2) \bowtie (2,3))$

évaluation de Q_1 : $MV = \{(1,3),(4,3)\}$

évaluation de Q_2 : $MV = \{(1,3),(4,3)\}$

ECA^k

si la définition de la vue contient les clés des sources

exemple : $r_1[W,X]$, $r_2[X,Y]$ et $V = \pi_{W,Y}(r_1 \bowtie r_2)$
cas 1 : W non clé, cas 2 : W clé

- les suppressions sont faites à l'entrepôt
  - la clé permet de retrouver les tuples à supprimer
    $I(r_1) = \{(3,1),(3,2)\}$, $I(r_2) = \{(1,4)\}$
    suppression (3,2) de r_1

ECA^k

si la définition de la vue contient les clés des sources

exemple : $r_1[W,X]$, $r_2[X,Y]$ et $V = \pi_{W,Y}(r_1 \bowtie r_2)$
cas 1 : W non clé, cas 2 : W clé

- les suppressions sont faites à l'entrepôt
  - la clé permet de retrouver les tuples à supprimer
    $I(r_1) = \{(3,1),(3,2)\}$, $I(r_2) = \{(1,4)\}$
    suppression (3,2) de r_1
- les insertions sont faites sans compensation
  - les duplicats sont simplement ignorés
    $I(r_1) = \{(3,1)\}$, $I(r_2) = \{\}$
    ajout (1,4) puis (1,5) dans r_2

Warehousing projects

conception

- chaque entrepôt est unique
- construction coûteuse, longue et complexe
- construction souvent itérative

il n'existe pas de méthodologie concensuelle

conception

- de quelles données a-t-on besoin ?
- d'où proviennent-elles ?
- comment les nettoyer ?
- quel est le schéma de l'entrepôt ?
- quelles données agréger ?
- quelles données matérialiser ?
- quels indexes ?

principe de construction

selon Kimball, 1996

1. identification des utilisateurs, interviews
   - sujet : ventes, commandes, envois, inventaire, ...
2. identification des sources, interviews des DBA
3. choix de la granularité des faits
   - granularité : transactions individuelles, images quotidiennes, ...
   - mesures : quantités, prix, coûts, marges, ...
4. identification des dimensions
   - dimensions : temps, produits, clients, vendeurs, ...
5. définition des processus ETL
6. choix des matériels/logiciels BD, ETL, OLAP, frontends

efficacité

une bonne analyse suppose des données

- propres
- valides
- consistantes
- pertinentes
- informative

efficacité

de l'analyse

- plus la granularité des faits est fine
- plus l'analyse sera précise
- plus l'entrepôt sera volumineux

de l'exécution des requêtes

- précalculer et préstocker des agrégats
- dénormaliser les tables de dimensions

estimation de la taille : exemple

dimensions

- 4 ans $\times$ 365 jours $=$ 1460 jours
- 300 magasins
- 200 000 références, 10% vendues par jours
- promotion : 1 valeur booléenne

estimation de la taille : exemple

faits

- $1460 \times 300 \times 20000 = 8{,}76 \times 10^9$
- 4 clés
- 4 mesures
- $8{,}76 \times 10^9 \times 8$ champs $\times$ 4 octets $= 280$ Go

sans agrégats...

estimation de la taille : exemple

- 3 ans = 1095 jours
- 100 million d'appels téléphoniques par jour
- 5 clés
- 3 mesures
- $109 \times 10^9 \times 8$ champs $\times$ 4 octets = 3,49 To

sans agrégats...

pratiques courantes dans l'industrie

- extraction et intégration faite off-line
- tout est copié dans l'entrepôt

success story

cas de la grande distribution

apports constatés

- augmentation des ventes grâce à un meilleur marketing
- amélioration des taux de rotation
- élimination des produits obsolètes
- réduction des rabais, remises, ristournes
- meilleure négociation des achats

exemple 1 : le groupe Casino

- solution Teradata
- un des premiers entrepôts en France
- plusieurs millions de dollars économisés en s'apercevant que les stocks de coca-cola faisaient souvent défaut...

1994	80 Go 50 utilisateurs	2002	+ de 10 To, 1500 utilisateurs 25000 requêtes/jour

exemple 2 : France Télécom région centre

12 BD sources
récupération des données : 1,5 année

- données régionales et nationales
- parfois chez des prestataires de services
- parfois au prix d'un intense lobbying

en 2003 : environ 5 années de travail

exemple 2 (suite)

- entreposage : SQL server
- DW de 3 bimestres, vidé périodiquement
- 1,2 million d'individus
- 1 fait = 1 client
- 250 colonnes
- intégration faite à la main périodiquement
- exploitation : progiciel de DM développé spécifiquement

exemple (3) : Walmart

- solution Teradata
- en 2006 : 0.5 Po de données
- le plus gros entrepôt de données du monde
  - distributeurs, magasins, clients ($> 10^8$), produits ($> 10^9$)...
- un des plus secret également...

Wal-Mart, for example, discovered that people who buy Pampers often buy beer, so they moved Pampers and beer close together. The result was that sales of both increased (Computer Business Review, October 1996).

échec des projets de construction d'entrepôt

- manque de littérature sur la méthodologie de conception

échec des projets de construction d'entrepôt

- manque de littérature sur la méthodologie de conception
- mauvaises prévisions techniques dans l'évaluation et le choix du matériel

échec des projets de construction d'entrepôt

- manque de littérature sur la méthodologie de conception
- mauvaises prévisions techniques dans l'évaluation et le choix du matériel
- problème de déploiement, e.g., sensibilisation des utilisateurs finaux

échec des projets de construction d'entrepôt

- manque de littérature sur la méthodologie de conception
- mauvaises prévisions techniques dans l'évaluation et le choix du matériel
- problème de déploiement, e.g., sensibilisation des utilisateurs finaux
- considérations sociotechniques ou éthiques, e.g., propriétés des données sources

échec des projets de construction d'entrepôt

- manque de littérature sur la méthodologie de conception
- mauvaises prévisions techniques dans l'évaluation et le choix du matériel
- problème de déploiement, e.g., sensibilisation des utilisateurs finaux
- considérations sociotechniques ou éthiques, e.g., propriétés des données sources

conclusion

So far: we have the warehouse...

Next: what do to with it?