

Intégration de données et qualité

(en environnements mono-machine et Big Data)

Master Informatique – 1^{ère} année

Mourad Ouziri
mourad.ouziri@parisdescartes.fr

Maître de conférences
Université de Paris (ex. Paris Descartes)



Traitement de données massives

Plan

Objectif :

Préparer des données de qualité pour les analyses et
les prises de décisions

Partie 1

Intégration de données :
Matérialisée et virtuelle

Partie 2

Qualité de données :
Evaluation et nettoyage

Partie 1 : intégration de données

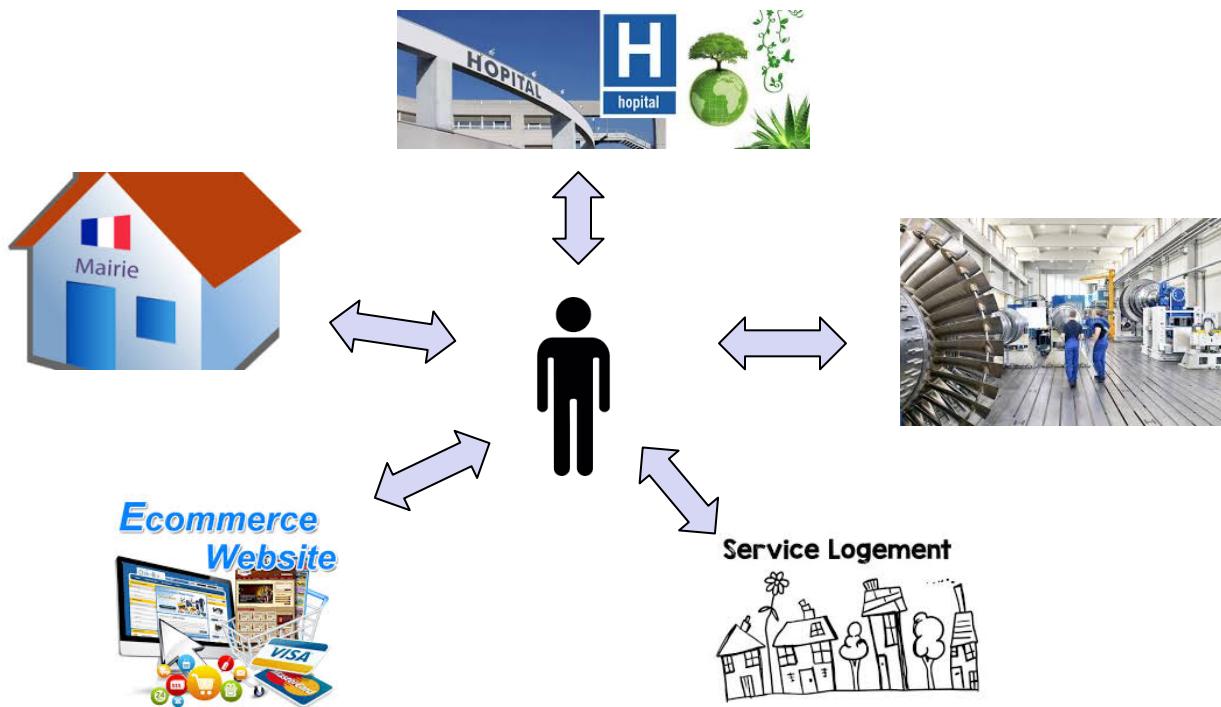
Plan

- ☞ Intégration de données : définition et enjeux
- ☞ Architectures d'intégration de données : matérialisée et virtuelle
- ☞ Intégration de données volumineuses à l'aide de *SGBD*
- ☞ Intégration de données massives en environnement Big Data avec
Apache Spark

Intégration de données

Constat : données en silos

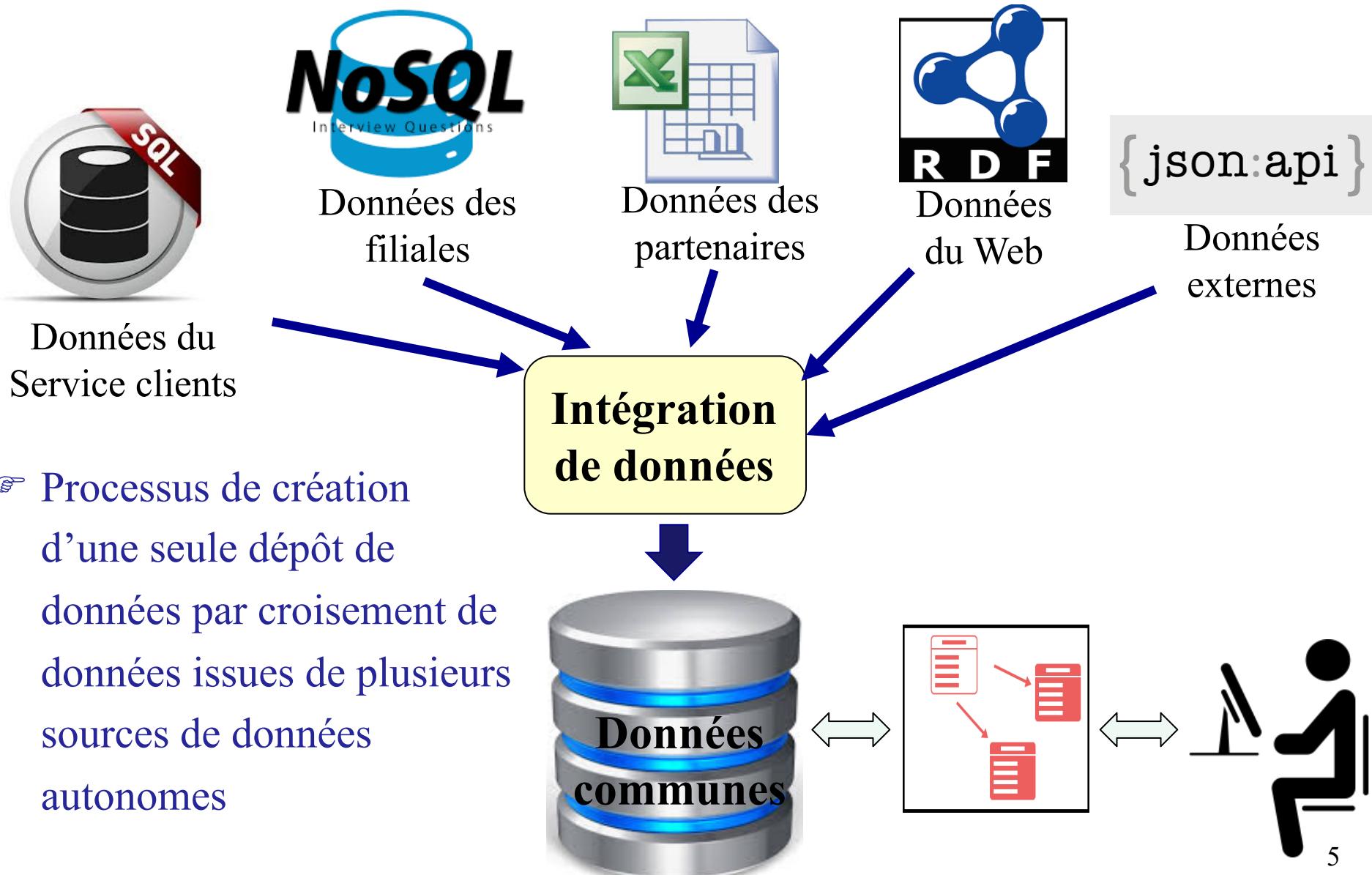
- ☞ Les données sont éparpillées dans plusieurs sources indépendantes



- ☞ Problématique : accéder à l'ensemble des données de manière uniforme tout en préservant l'autonomie des sources

Intégration de données

Définition



Intégration de données

Définition

- ☞ Croiser plusieurs sources de données indépendantes mais complémentaires afin de permettre aux utilisateurs d'accéder de manière uniforme à des données de qualité
- ☞ Sous condition d'autonomie des sources : les sources continuent de fonctionner en toute autonomie, l'intégration est une surcouche et non une rétro-conception des sources en base de données répartie

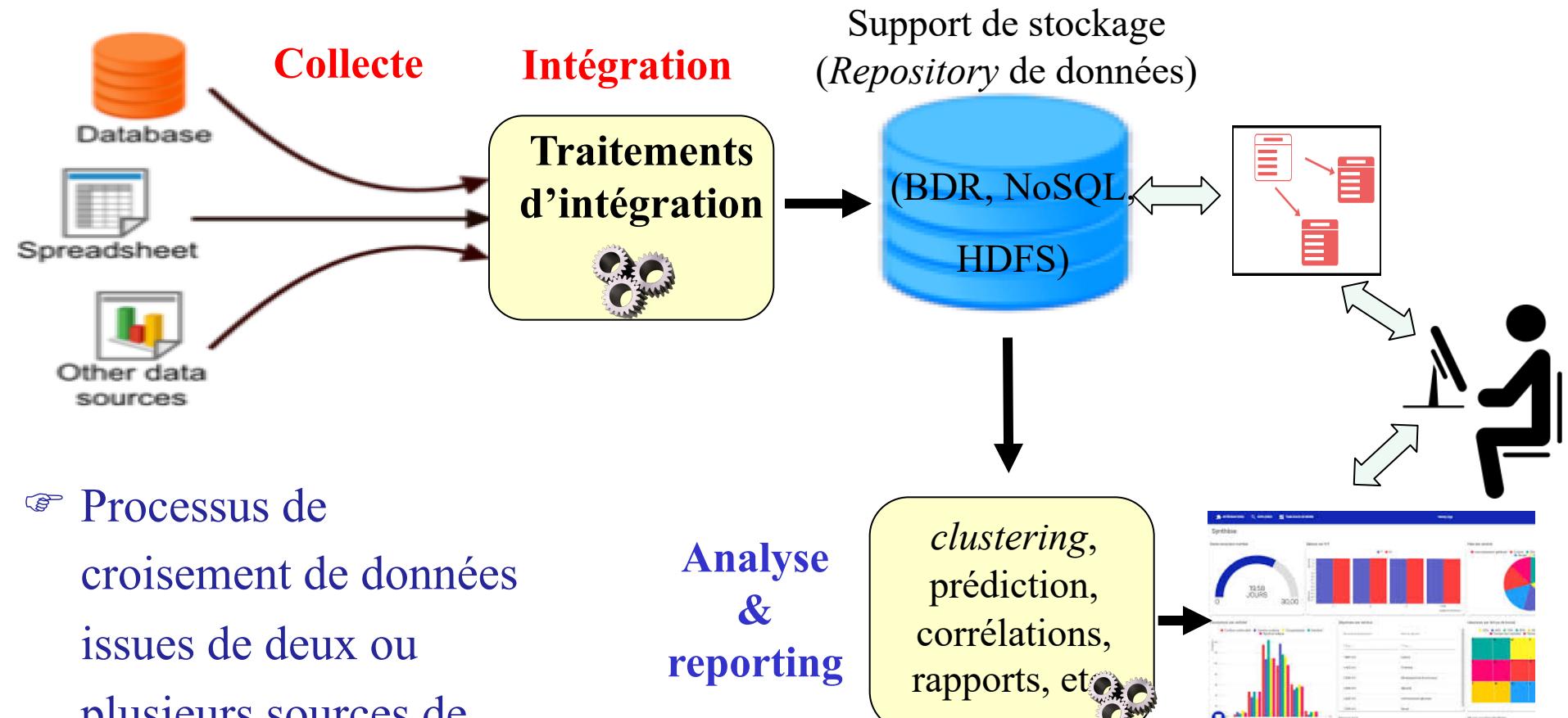
Intégration de données

Enjeux métiers

- ☞ *Croiser* : rapprocher, lier, combiner et consolider
- ☞ *Sources de données* : pas que des bases de données relationnelles mais aussi des BD à objets, NoSQL, fichiers CSV/tabulaires, API, etc.
- ☞ *Indépendantes* : conçues indépendamment les unes des autres avec différents modèles de données, schémas, formats, unités de mesure, identification, vocabulaires, etc.
- ☞ *Complémentaires* : les sources contiennent des données « partielles » pouvant être complétées par d'autres sources
- ☞ *Uniforme* : interface et modèle de données unique permettant un accès unique et transparent aux diverses sources de données

Intégration de données

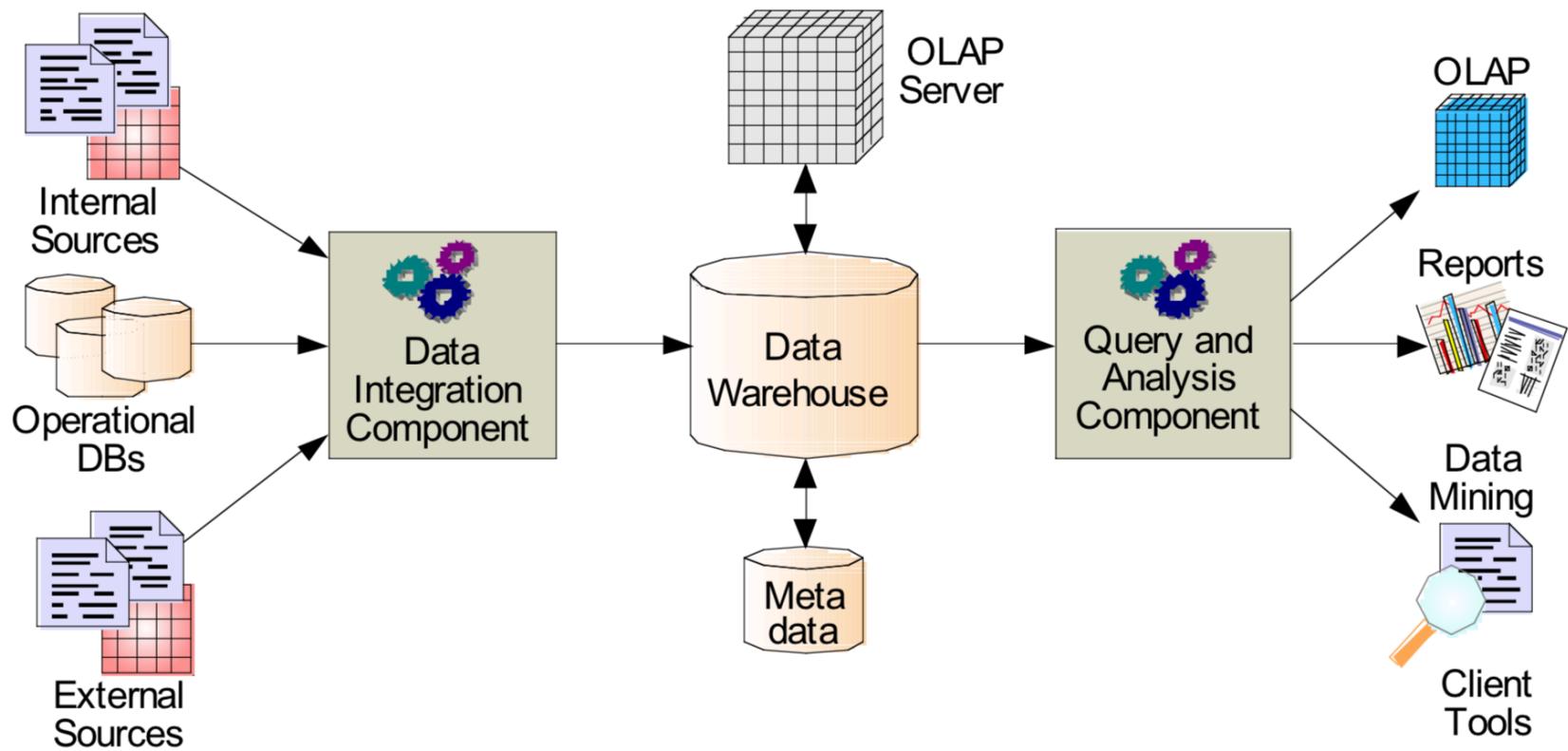
Définition et positionnement dans la chaîne de traitement



☞ Processus de croisement de données issues de deux ou plusieurs sources de données indépendantes

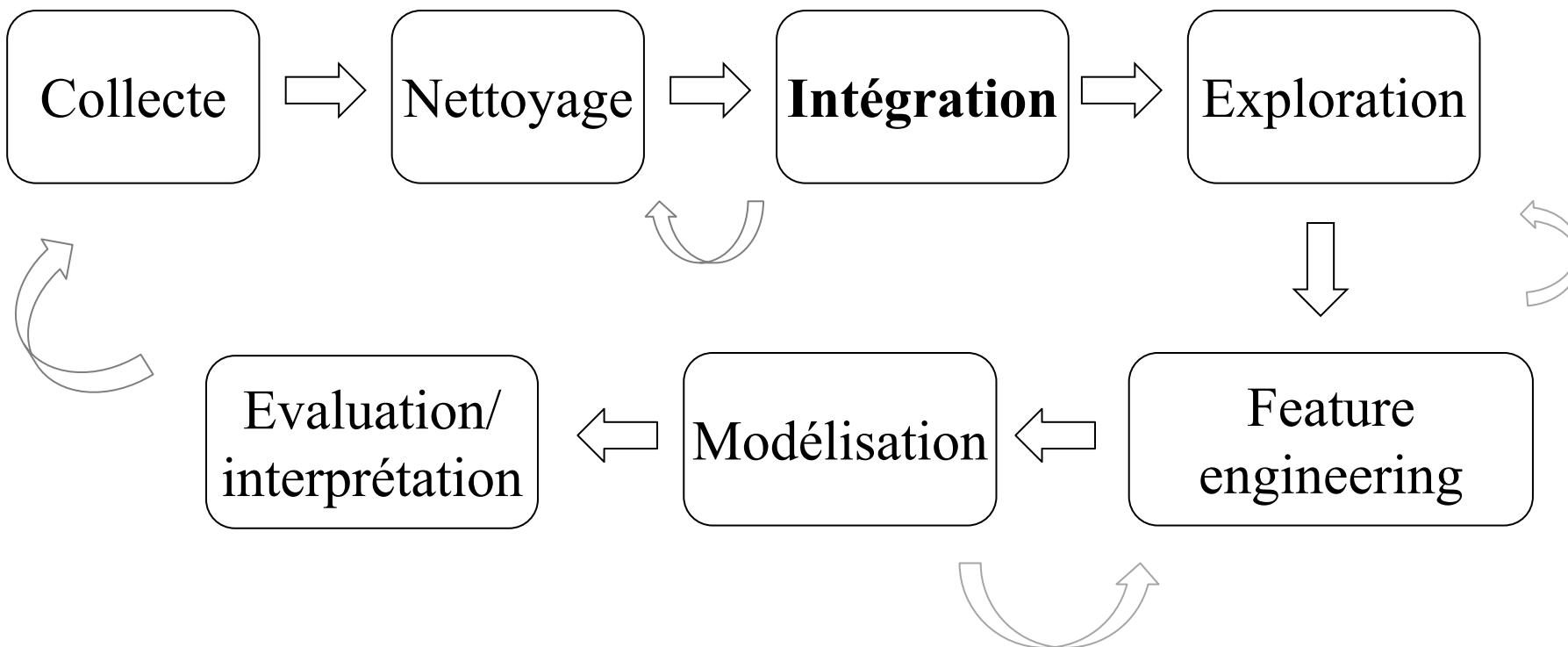
Intégration de données

Architecture d'un SI décisionnel (BI)

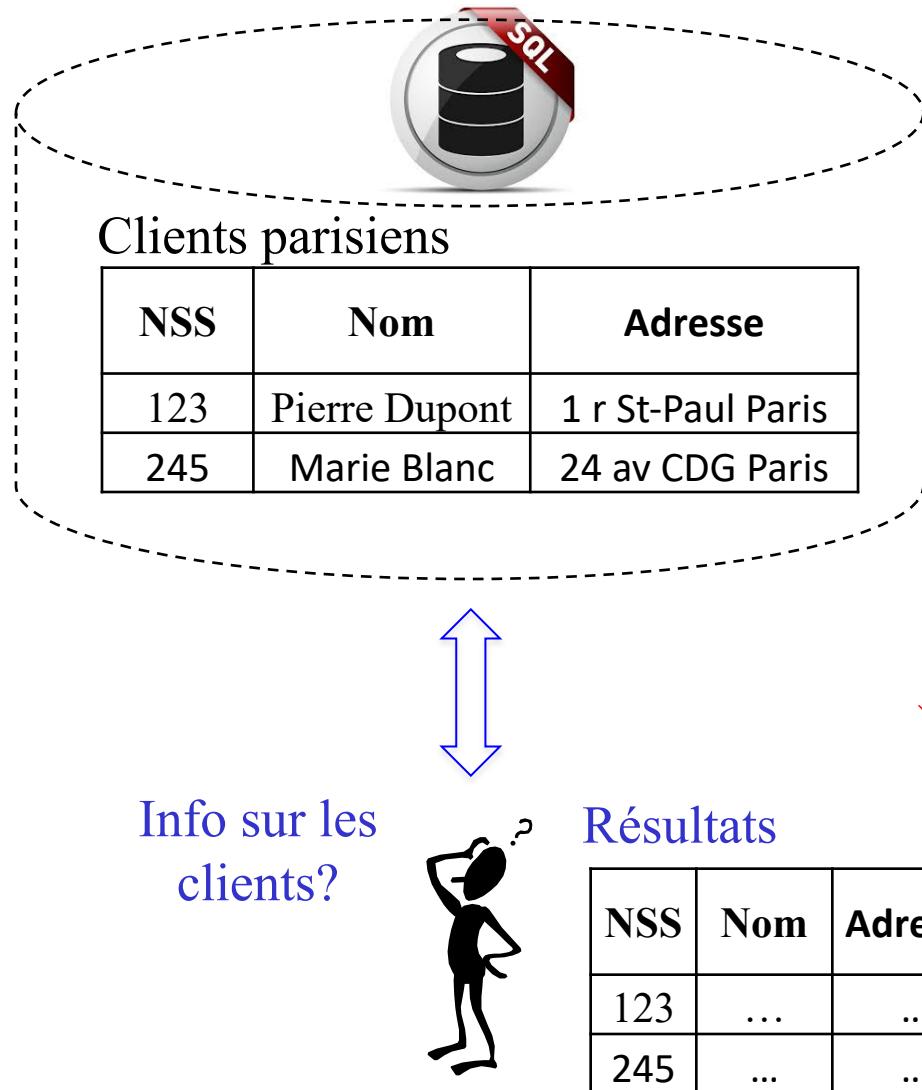


Intégration de données

Etapes de construction d'un modèle d'apprentissage machine

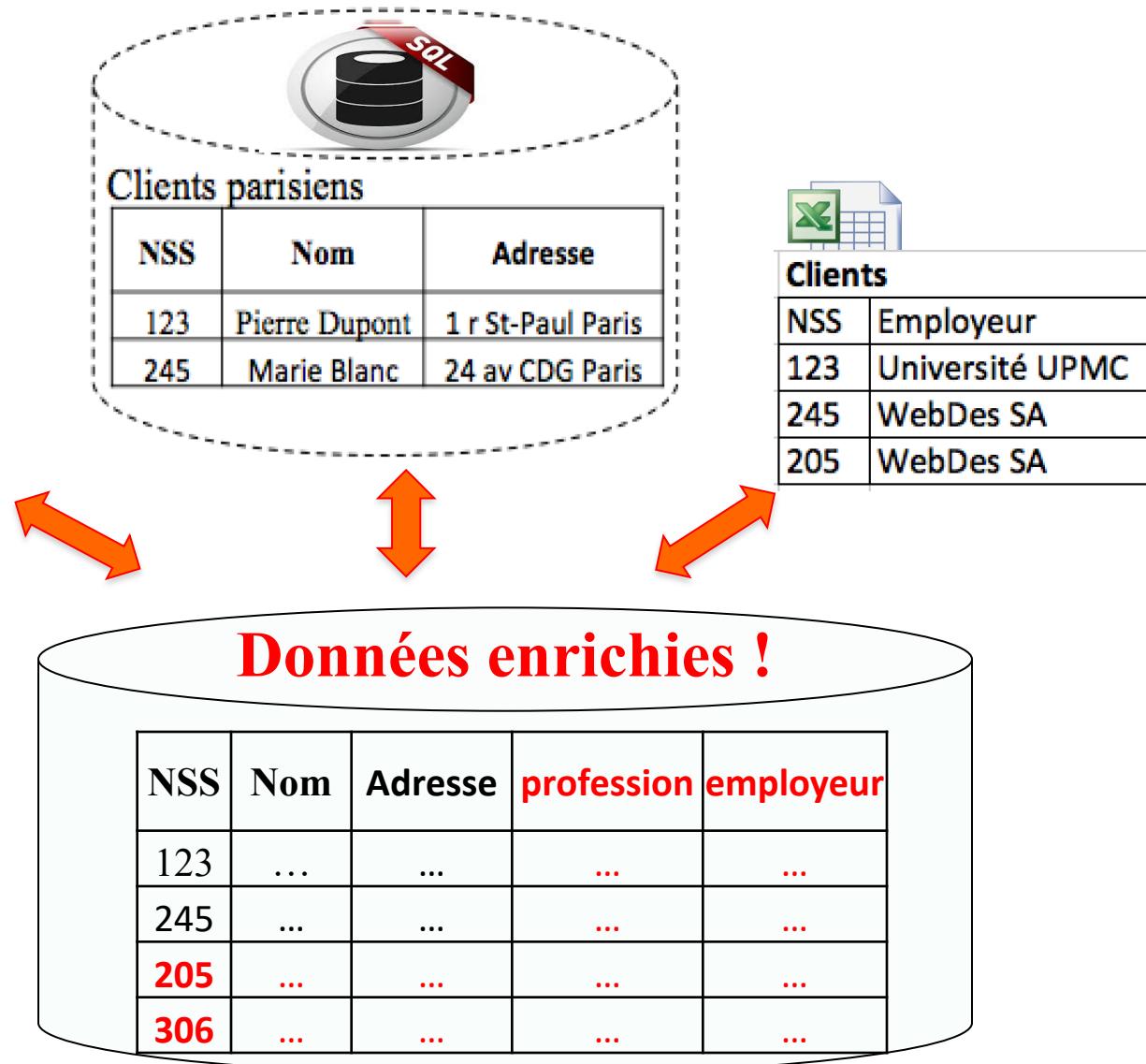


Objectif enrichissement de données



Objectif enrichissement de données

```
{"clients": [ { "NSS":123, "nom":"Pierre Dupont", "profession":"Enseignant", }, { "NSS":205, "nom":"Anne Marchal", "profession":"Informatique", }, { "NSS":306, "nom":"John Smith", "profession":"Informatique", } ]}
```



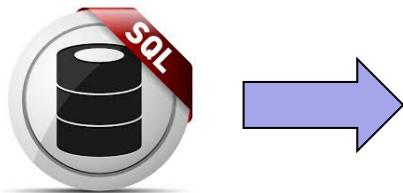
Info sur les clients ?



Enjeux de l'intégration de données

Analyse de données

- ☞ Cas d'usage : gestion immobilière



Clients

id	Nom	Ville	Logement (type)
123	P. Dupont	Paris	locataire
245	M. Blanc	Paris	propriétaire
454	C. Boyer	Paris	?
168	S. Young	Paris	propriétaire
459	K. Sar	Paris	locataire

- ☞ Analyse 1 : profil des clients ?

Pas de profil qui se dégage : 50% des parisiens sont locataires et 50% sont propriétaires

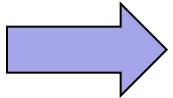
- ☞ Analyse 2 : expliquer le « type de logement » ?

Pas de variable explicative pertinente !

Enjeux de l'intégration de données

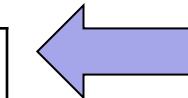
Analyses de données : vente de biens immobiliers

☞ Objectif : profil des clients



Clients

id	Nom	Ville	Sect-activité	Logement
123	P. Dupont	Paris	Santé	locataire
245	M. Blanc	Paris	Informatique	propriétaire
454	C. Boyer	Paris	Santé	locataire
168	S. Young	Paris	Informatique	propriétaire
459	K. Sar	Paris	Santé	locataire



☞ Résultat :

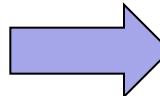
100% des personnes travaillant dans la **santé** sont **locataires** (support 3/5 !)

Sect-activité = "Santé" => Logement = "locataire" (support = 3/5, confiance = 1)

Enjeux de l'intégration de données

Analyses de données : vente de biens immobiliers

☞ Objectif : profil des clients



Clients

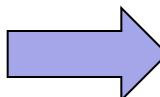
id	Nom	Ville	Logement
123	P. Dupont	Paris	locataire
245	M. Blanc	Paris	propriétaire
454	C. Boyer	Paris	locataire
168	S. Young	Paris	propriétaire
459	K. Sar	Paris	locataire
600	S. McHill	Paris	locataire
601	A. Aubert	Paris	locataire
602	B. Blanc	Paris	locataire
603	B. Bernard	Paris	locataire
604	M. James	Paris	locataire

☞ Résultat :

80% des parisiens sont locataires !

Ville = "Paris" => Logement = "locataire"

(support = 100%, confiance = 0.8)



Intégration de données

Enjeux métiers

👉 Augmenter la valeur de la donnée

- 👉 Valeur marchande : coût de vente ou de mise à disposition
- 👉 Levier de prise de décisions : par une meilleure connaissance de l'entité cible (marché, produits, clients, concurrents, etc.)

👉 Exemples :

- 👉 Marketing : *pricing*, réduction des coûts et compétitivité des entreprises
- 👉 Médical : santé et économie des dépenses de soins

Intégration de données

Valeur de la données pour les entreprises françaises (IDC–2014)

COMPÉTITIVITÉ DES ENTREPRISES ET DONNÉES



Une meilleure exploitation des données pourrait améliorer la compétitivité des entreprises à plusieurs niveaux

Entreprises aux usages avancés



Entreprises aux usages standards



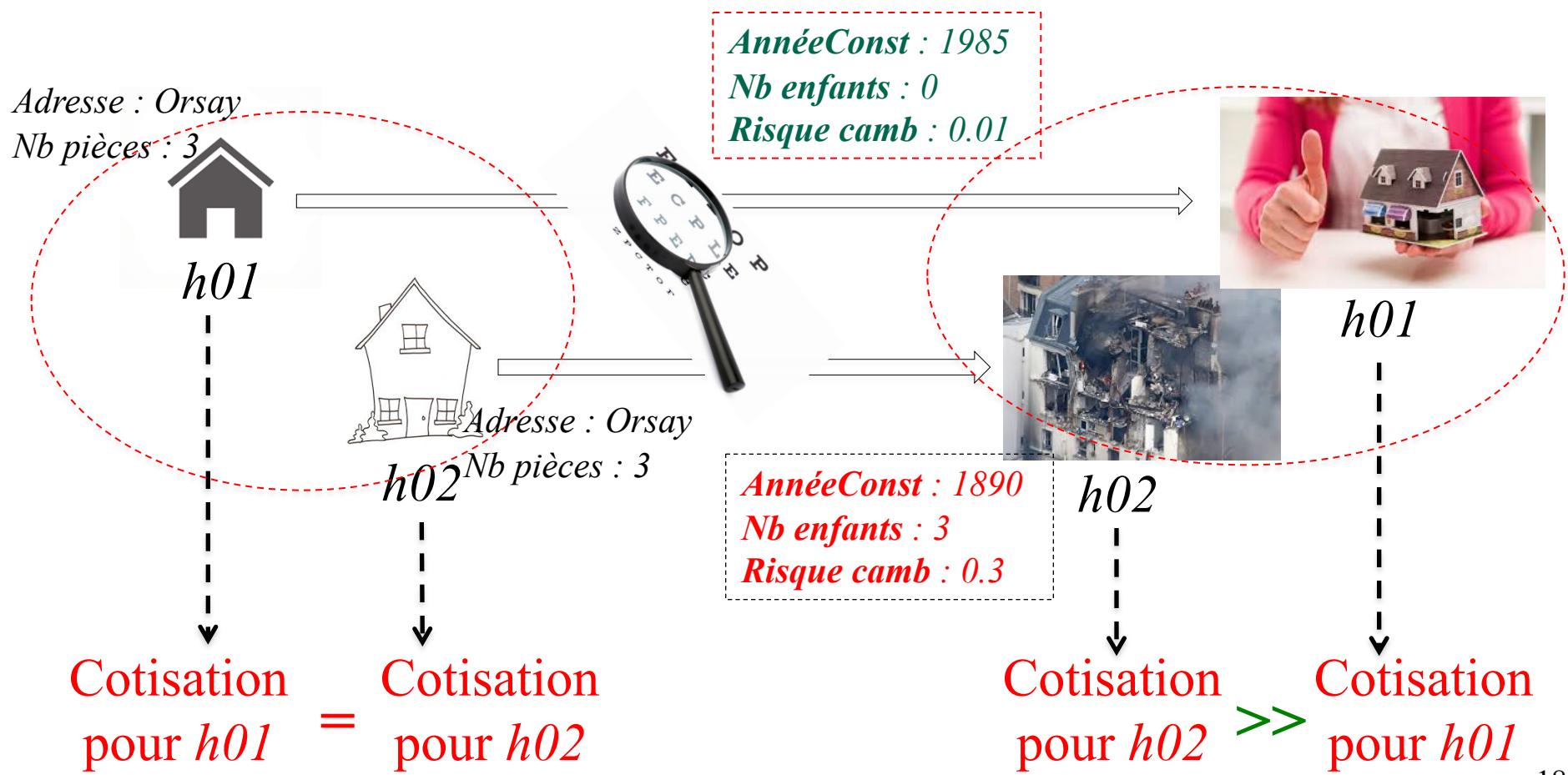
- Réduction des coûts
- Amélioration de la productivité des collaborateurs
- Augmentation du chiffre d'affaires

- Réduction des coûts
- Amélioration de la productivité des collaborateurs
- Augmentation du chiffre d'affaires

Enjeux métiers en marketing

Compétitivité des entreprises

- En assurance habitation, une meilleure connaissance des habitations contribue à rendre les compagnies d'assurance plus compétitives



Enjeux métiers dans le médical

Dossier médical partagé (DMP)

- ☞ Meilleure qualité des soins !
- ☞ Eviter la redondance d'actes médicaux =>

1. Economiser les actes médicaux déjà réalisés
2. Préserver la santé des patients



radios,
IRM,
interprétations



hospitalisations,
analyses, etc.



diagnostics,
prescriptions

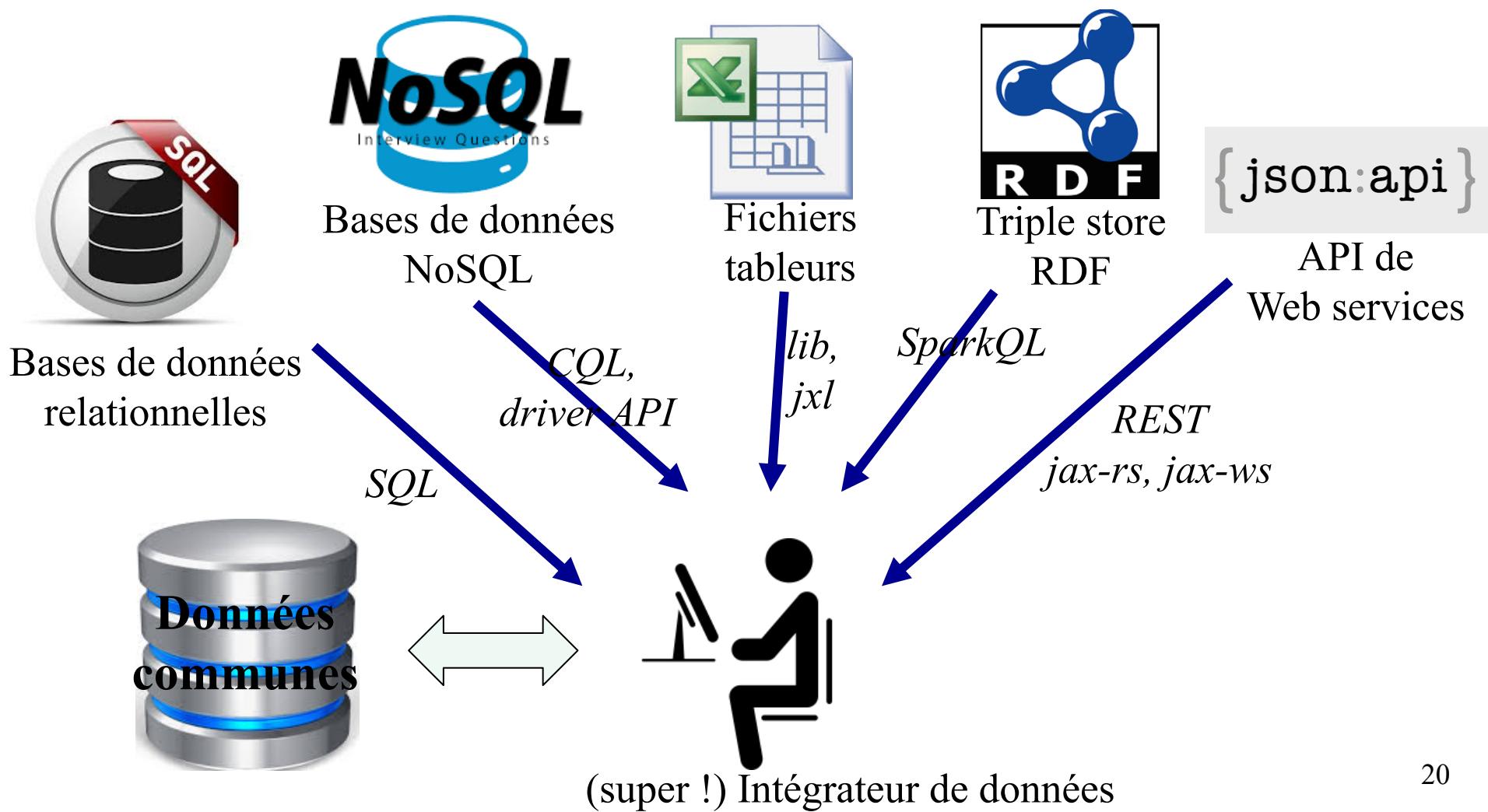


?



Problématique de l'intégration de données

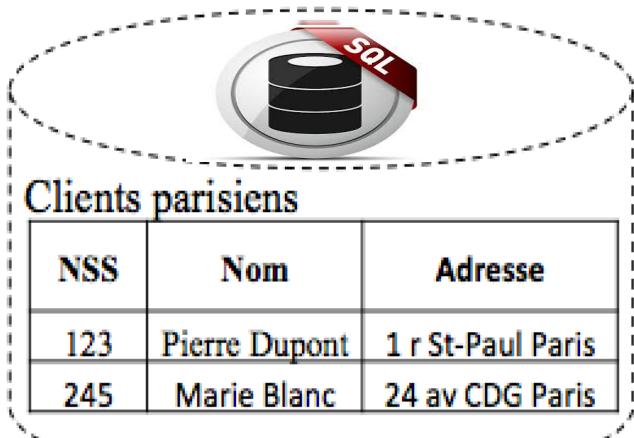
Diversité des systèmes de stockage



Problématiques de l'intégration de données

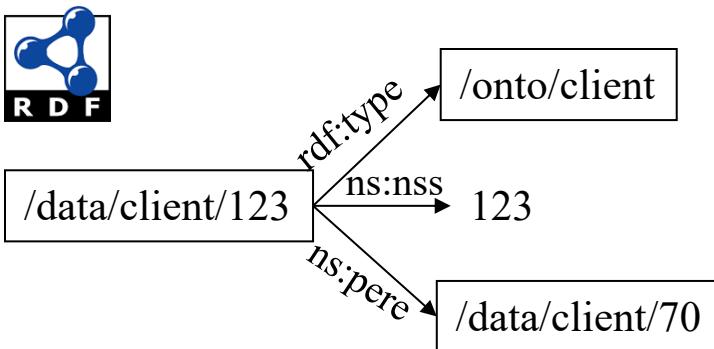
Hétérogénéité des modèles

- Les bases peuvent avoir été conçues avec des modèles de données différents



NoSQL
Introducing NoSQL

```
{"clients": [ { "NSS":123, "nom":"Pierre Dupont", "profession":"Enseignant", }, { "NSS":205, "nom":"Anne Marchal", "profession":"Informatique", }, { "NSS":306, "nom":"John Smith", "profession":"Informatique", } ]}
```



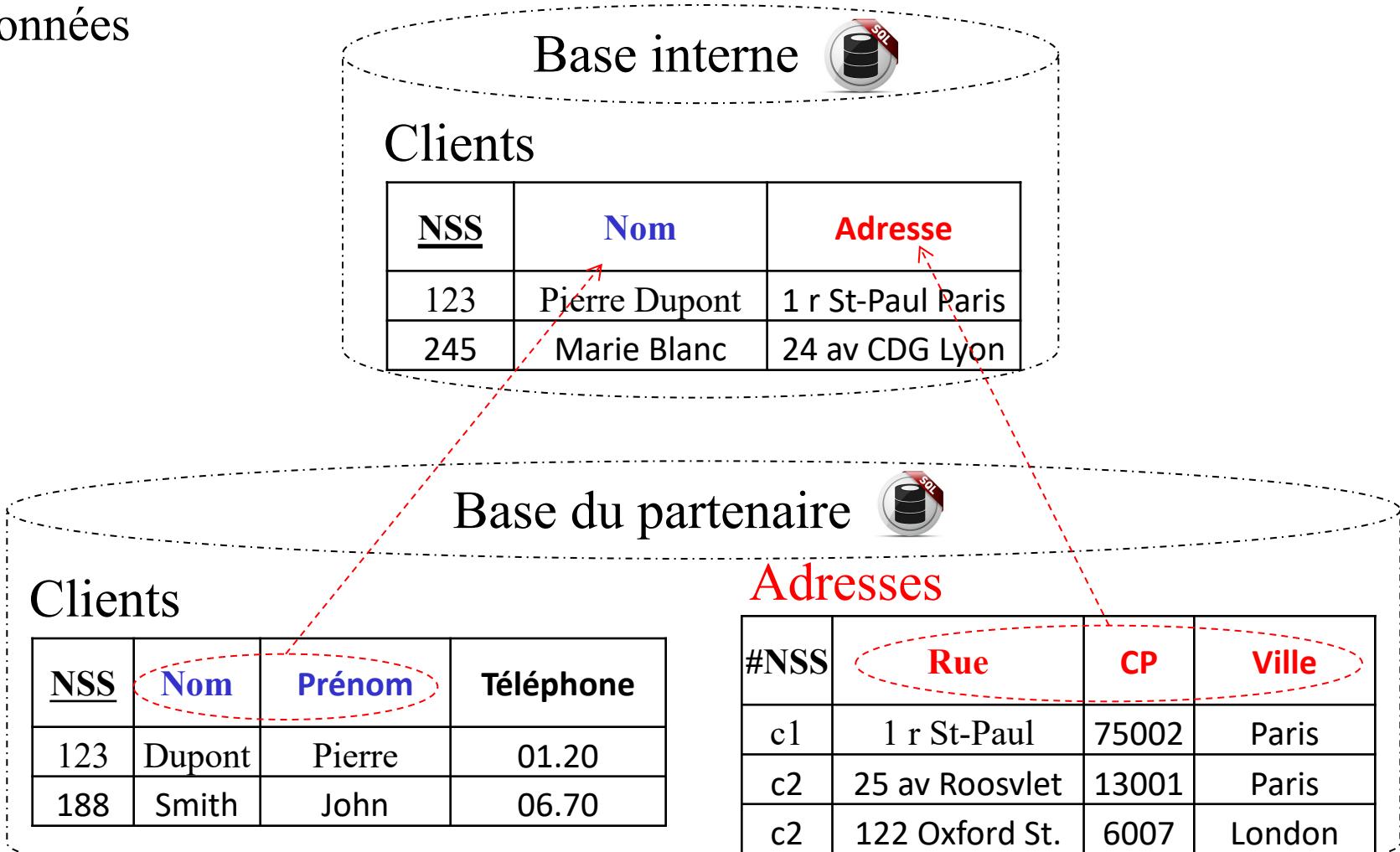
Clients

NSS	Employeur
123	Université UPMC
245	WebDes SA
205	WebDes SA

Problématiques de l'intégration de données

Hétérogénéité structurelle (schéma)

- Les bases peuvent avoir des schémas différents pour stocker les mêmes données



Problématiques de l'intégration de données

Identification des entités du monde réel (*Entity Resolution*)

- La même entité (personne, produit, organisation, etc.) peut apparaître dans plusieurs bases avec des identifiants différents

Base interne



Customer

NSS	Nom	Adresse	Emploi
123	Pierre Dupont	1 r St-Paul Paris	CDD
245	Marie Blanc	24 av CDG Lyon	CDI

Base du partenaire



Clients

Id	Nom	Adresse	Logement
c1	P. Dupont	1 r St-Paul 75001	propriétaire
c2	M. Blanc	24 av CDG 69003	locataire

Q1 : Tout savoir
sur le client 123 ?



Q2 : Clients en
CDI et locataires
de leur logement ?

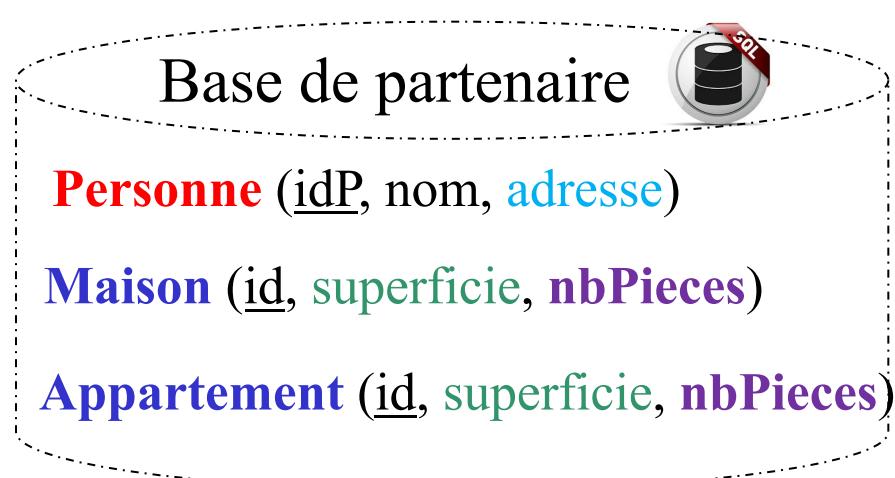
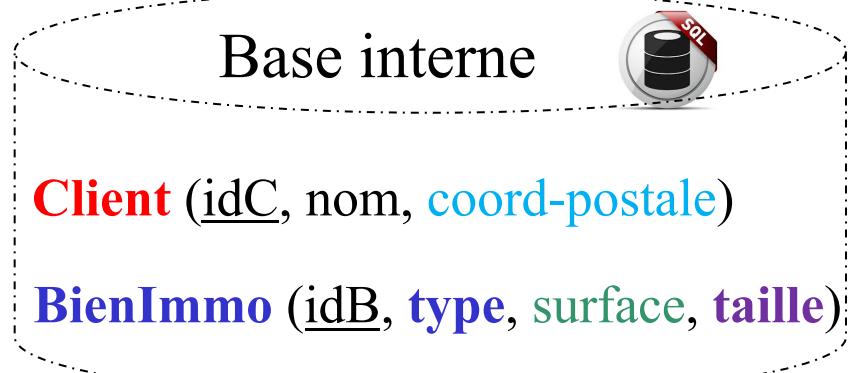
Clients

NSS	Id	Nom	Adresse	Emploi	Logement
123		Pierre Dupont	1 r St-Paul Paris	CDD	
245		Marie Blanc	24 av CDG Lyon	CDI	
	c1	P. Dupont	1 r St-Paul Paris		propriétaire
	c2	M. Blanc	24 av CDG Lyon		locataire

Problématiques de l'intégration de données

Hétérogénéité sémantique

- ☞ Les bases utilisent différentes terminologies pour décrire leurs données



Clients propriétaires de maisons de surface minimum de 100m² ?

Architectures des systèmes d'intégration de données

Formulation logique du problème d'intégration

Formulation logique des requêtes

☞ Une relation (source ou du schéma global) est un prédictat logique de la forme $R(\vec{a})$ où \vec{a} est un vecteur de constantes (donc un fait logique !)

☞ Une requête $Q(\vec{x})$ est une formule logique de la forme :

$\exists \vec{y} \ \phi(\vec{x}, \vec{y})$ pouvant être (en Datalog) : $Q(\vec{x}) \leftarrow \phi(\vec{x}, \vec{y})$)

\vec{x} : vecteur de variables libres

\vec{y} : vecteur de variables existentielles et/ou de constantes

ϕ : formule (souvent conjonctive) de la logique du premier ordre

☞ Exemples :

$Q_1: \exists y \ Place(x) \wedge Capitale(y) \wedge estSitué(x, y)$

$Q_2(x, y) \leftarrow collègue(x, y) \wedge Informaticien(y)$

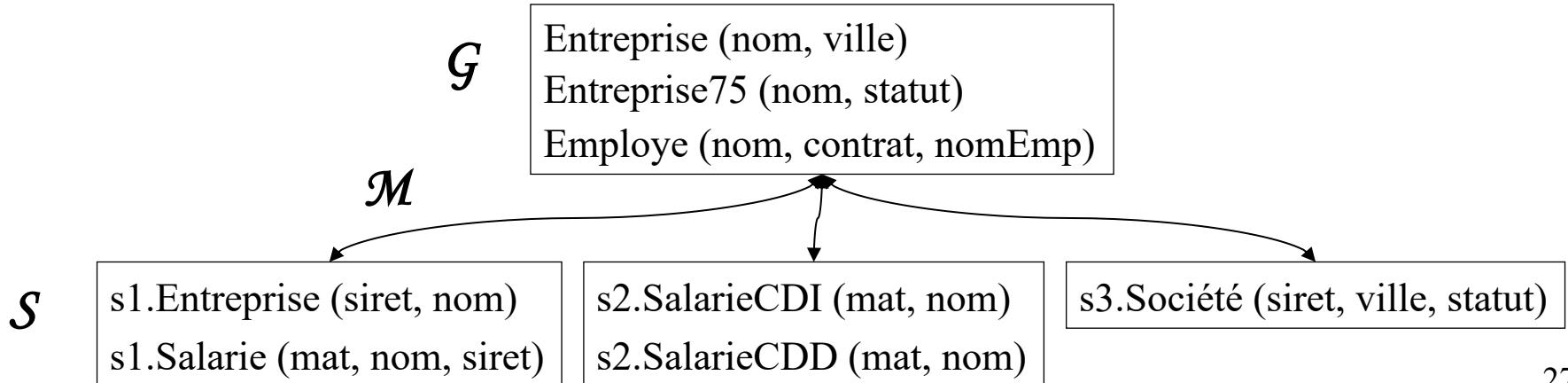
$Q_3(x) \leftarrow Etudiant(y) \wedge inscrit(y, 'Univ. Paris') \wedge connaît(x, y)$

Architecture d'intégration virtuelle

Intégration de schémas

☞ Un système d'intégration de données est un triplet $\mathfrak{I} = \langle G, S, M \rangle$ tels que :

- G : est le schéma global des données de l'ensemble des sources
C'est une théorie logique (ensemble de prédictats) exprimée sur un alphabet \mathcal{A}_G
- S : est le schéma de la source
C'est une théorie logique exprimée sur un alphabet \mathcal{A}_S (disjoint de \mathcal{A}_G)
- M : est un ensemble de correspondances (*mappings*) entre S et G
Deux principales façons d'exprimer M : G en fonction de S ou S en fonction de G



Architecture d'intégration

Description du schéma d'intégration

☞ *Mappings* \mathcal{M} : chaque *relation* du schéma global est exprimée en fonction des *relations* des sources (ou inversement !)

☞ Formellement, \mathcal{M} est un ensemble de règles :

$$\forall \vec{x} \ g(\vec{x}) \leftarrow \phi_S(\vec{x}) \quad \text{pour tout } g \in \mathcal{G}$$

ϕ_S est une requête exprimée sur S de même arité que g

☞ Exemple:

$\mathcal{M}_1: Entrepirse(n, v) \leftarrow s1. Entrepirse(s, n) \wedge s3. Societe(s, v, -)$

$\mathcal{M}_2: Entrepirse75(n, st)$

$\leftarrow s1. Entrepirse(s, n) \wedge s3. Societe(s, v, st) \wedge situeeA(v, 'Paris')$

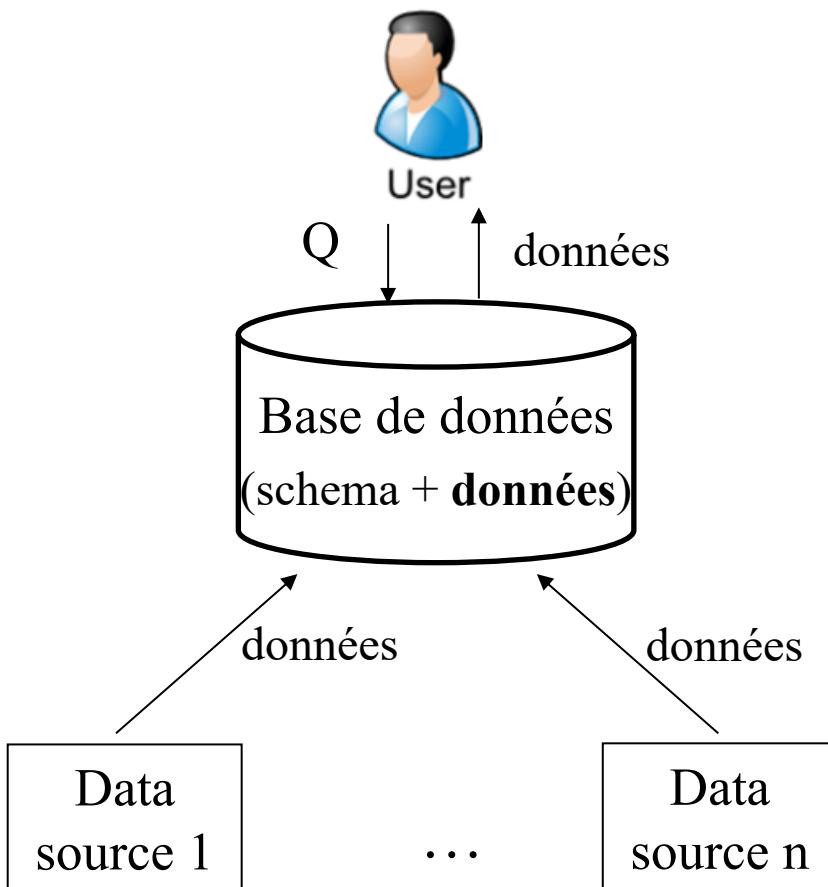
Architectures d'intégration de données

☞ Trois principales architectures

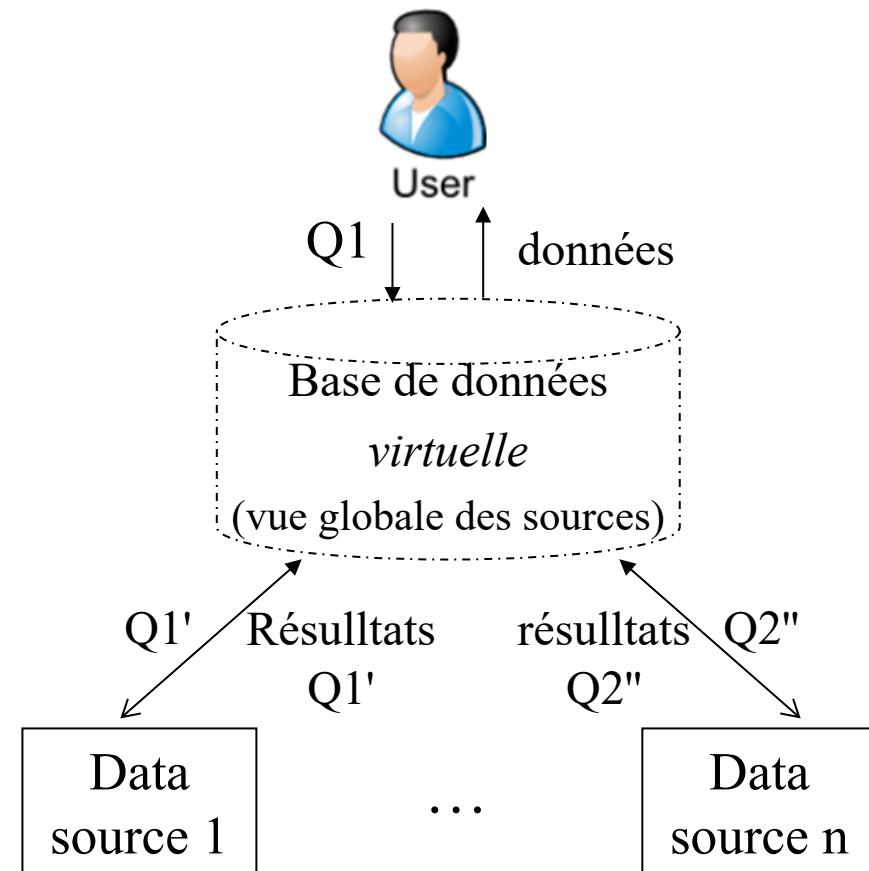
- **Intégration matérialisée (ETL – Extract, Transform, Load)** : les données et les schémas des sources sont intégrés puis stockés dans un dépôt cible unique (*datwarehouse, data lake*).
Les relations de \mathcal{G} sont peuplées avec les données collectées des sources
- **Intégration virtuelle (EII – Enterprise Information Integration)** : seuls les schémas des sources sont intégrés, les données sont maintenues dans leurs sources respectives et leur extraction n'est réalisée qu'au moment de l'évaluation des requêtes utilisateurs.
Les relations de \mathcal{G} sont des vues (ne contiennent aucune donnée)
- **Intégration d'applications (EAI – Enterprise Application Integration, SOA)** : l'intégration est réalisée par les applications/services fournissant les données sources par échange de messages sur un *broker/router* qui réalise le routage et les transformations nécessaires

Architectures d'intégration de données

Matérialisée vs. virtuelle

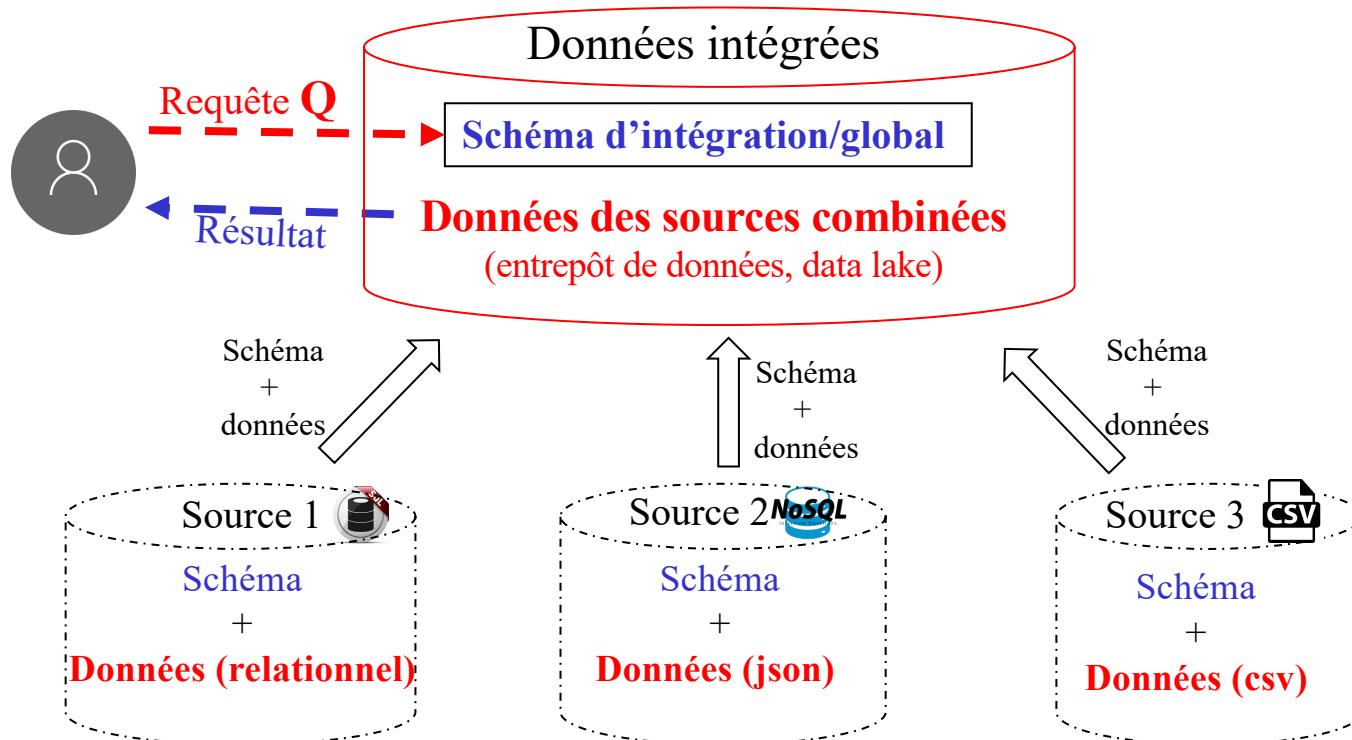


Intégration matérialisée
(données pré-collectées)

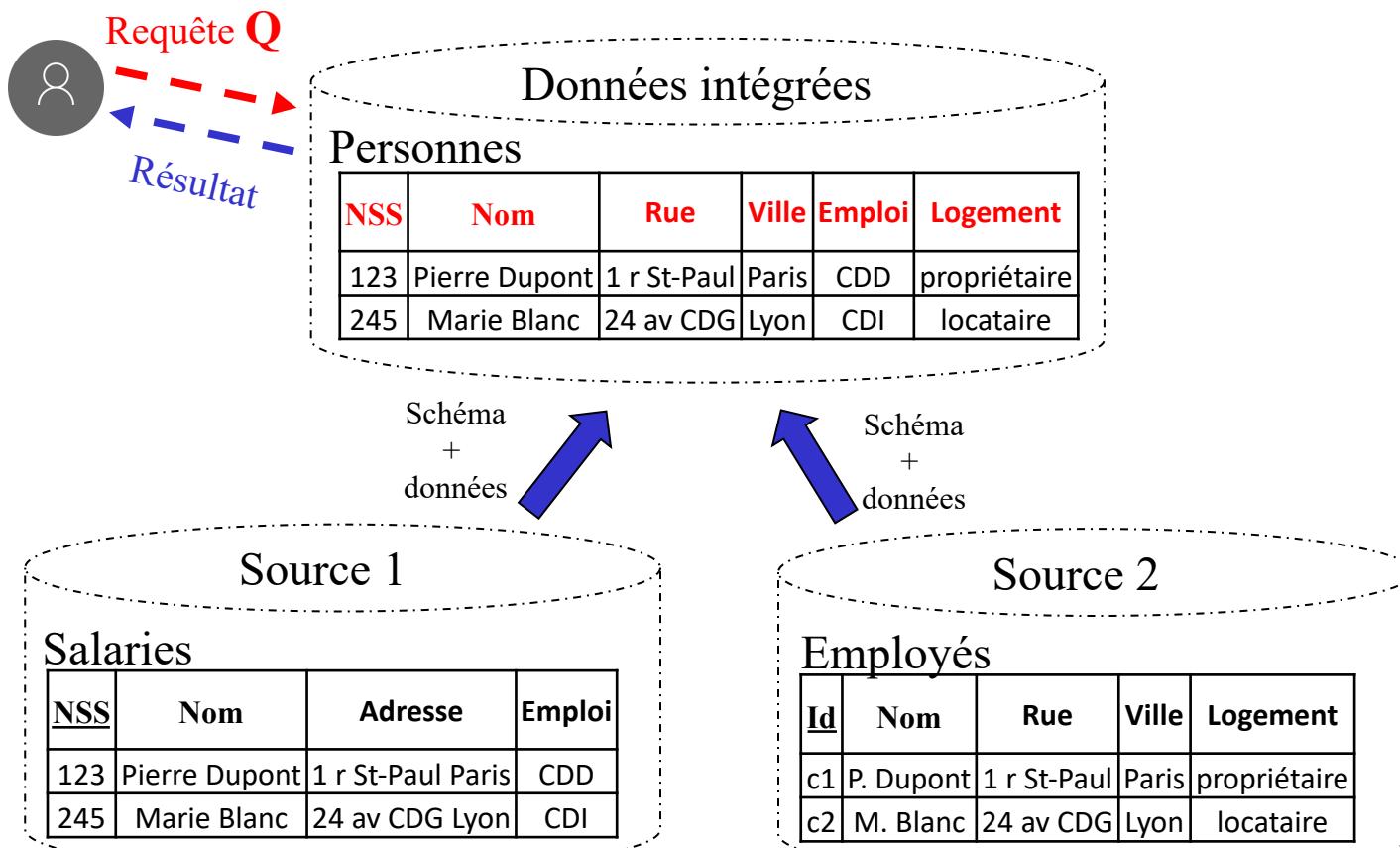


Intégration virtuelle
(données collectées à la demande)

Architectures d'intégration matérialisée (ETL)



Architectures d'intégration matérialisée (ETL)

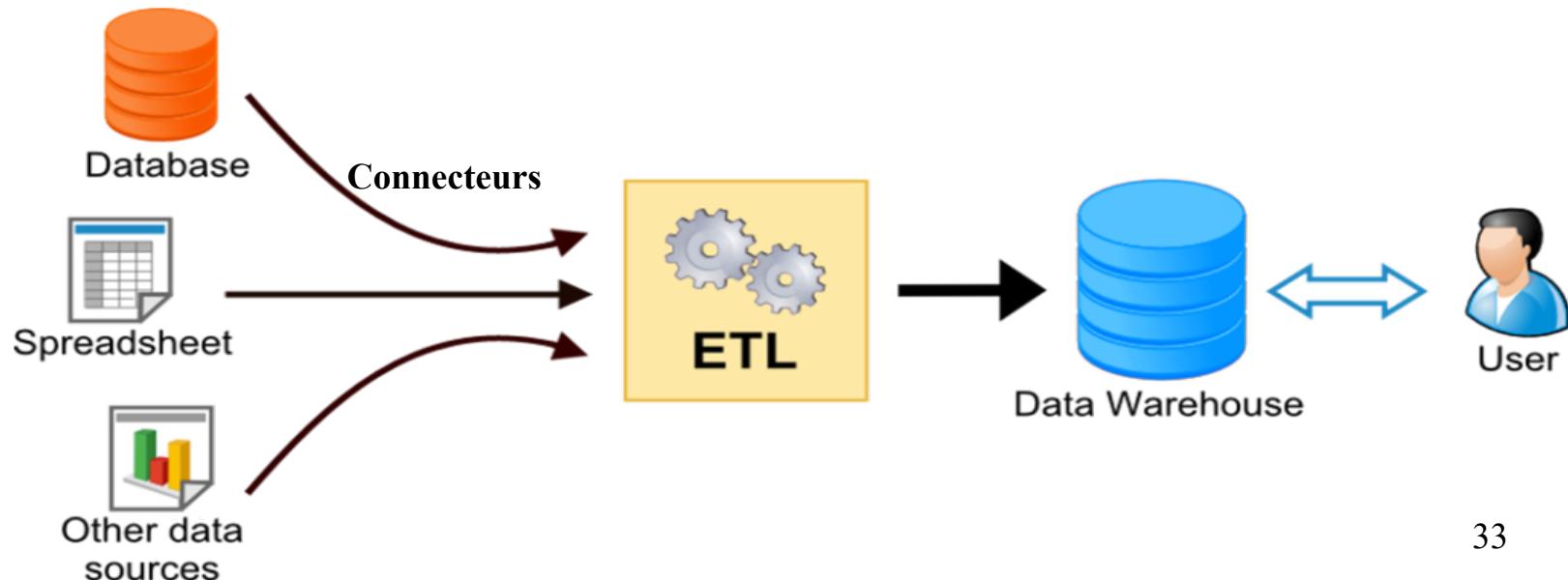


L'intégration matérialisée

Le processus ETL

☞ ETL pour Extract, Transform, Load

- Extract : récupération de données à partir de sources hétérogènes (fichiers textes, bases de données, services web, etc.)
- Transform : traitements de données (conversion de types et formats, de valeurs, nettoyage de données, agrégations, jointures, etc.)
- Load : insérer les données dans les sources cibles (datawarehouse, data lake, etc.)



L'intégration matérialisée

Limites du processus ETL : volume !

- ☞ ETL : le volume de données pouvant être traité est limité !
 - Les moteurs ETL classiques utilisent les ressources (CPU, RAM, HDD) de (une seule !) la machine

Talend : mémoire insuffisante pour traiter 7 millions de lignes

Bonjour,

Je suis sur que je serais aidée ici.

En effet, je travaille avec l'ETL TALEND et mes données sont dans une base MYSQL.

Je fais des jointures(jointures qui marchent et vérifiée sur MYSQL) à l'aide du composant tmysqlINPUT .

Les tables dont je me sert comportent plus de 7 millions de lignes chacune.

Après exécution, j'ai un problème de mémoire, j'ai augmenté dans la configuration (fichier [TalendOpenStudio-win32-x86.ini](#))

**-vmargs
-Xms256m
-Xmx1024m
-XX:MaxPermSize=128m**

mais malgré cela, j'ai toujours ce même message.

Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at com.mysql.jdbc.MysqlIO.nextRow(MysqlIO.java:1340)
at com.mysql.jdbc.MysqlIO.readSingleRowSet(MysqlIO.java:2330)
at com.mysql.jdbc.MysqlIO.getResultSet(MysqlIO.java:427)
at com.mysql.jdbc.MysqlIO.readResultsForQueryOrUpdate(MysqlIO.java:2035)
at com.mysql.jdbc.MysqlIO.readAllResults(MysqlIO.java:1421)
at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:1772)

L'intégration matérialisée

Limites du processus ETL : volume !

☞ ETL : utiliser le disque dur (*swapping*) pour traiter plus de données !

The screenshot shows a web browser displaying the Talend Open Studio for Data Integration Guide. The URL in the address bar is help.talend.com/r/~9WHmPreN~gvkavtiUEdvA/8CFxBxxEc~1_2gNAyVlz1w. The page title is "Talend Open Studio for Data Integration Guide utilisateur". A sidebar on the left lists navigation links: "Conception d'un Business Model", "Conception d'un Job", "Gestion des Jobs", and "Mapping de flux de données". Under "Mapping de flux de données", there are three items: "Interfaces de mapping", "Présentation du fonctionnement du tMap", and "Configuration du flux d'entrée". The main content area is titled "6.1 Français (France)". A yellow box labeled "Decommissioned" is present. The text discusses memory issues when dealing with large datasets, mentioning problems with insufficient memory that prevent the job from executing correctly, particularly when using a tMap component for transformations. A red circle highlights the phrase "des problèmes de mémoire insuffisante empêchant votre Job de s'exécuter correctement". Below this, another paragraph explains a Java-specific option for the tMap component that stores temporary data on disk to reduce memory usage during reference data processing (lookup).

Lorsque vous devez traiter un nombre important de données, par exemple, de nombreuses colonnes, différents types de colonnes ou lignes, votre système peut rencontrer **des problèmes de mémoire insuffisante** empêchant votre Job de s'exécuter correctement, et plus particulièrement lorsque vous utilisez un composant **tMap** pour effectuer des transformations.

Une option (uniquement disponible en Java pour le moment) a été ajoutée au composant **tMap**, pour utiliser moins de mémoire lors du traitement des données de référence (lookup). En effet, au lieu de stocker les données de référence dans la mémoire système et ainsi en atteindre les limites, l'option **Store temp data** vous permet de stocker les données de référence dans un dossier temporaire sur votre disque dur.

Intégration de données dans le Big Data

Limites du processus ETL : temps de traitement

☞ ETL : le *swapping* ralentit sensiblement le temps de traitement !

- Les données de la RAM sont temporairement et fréquemment vidées dans le disque dur...

Pages: 1

2017-01-20 15:47:40

jfaza
Member
26 posts

Hello,
I have created a delta load using Talend.
Everything is working fine except updating the columns seems to take forever.
In tPostresqlOutput I am using 'Update' in Action on data.
The table to be updated has more than 7 million rows but I am updating only 500 rows but still it is taking too much time.
Is there a way to fix this issue?
I might be updating even bigger tables later so this is a big problem for me.
Wrong forum.
How can I delete this post?

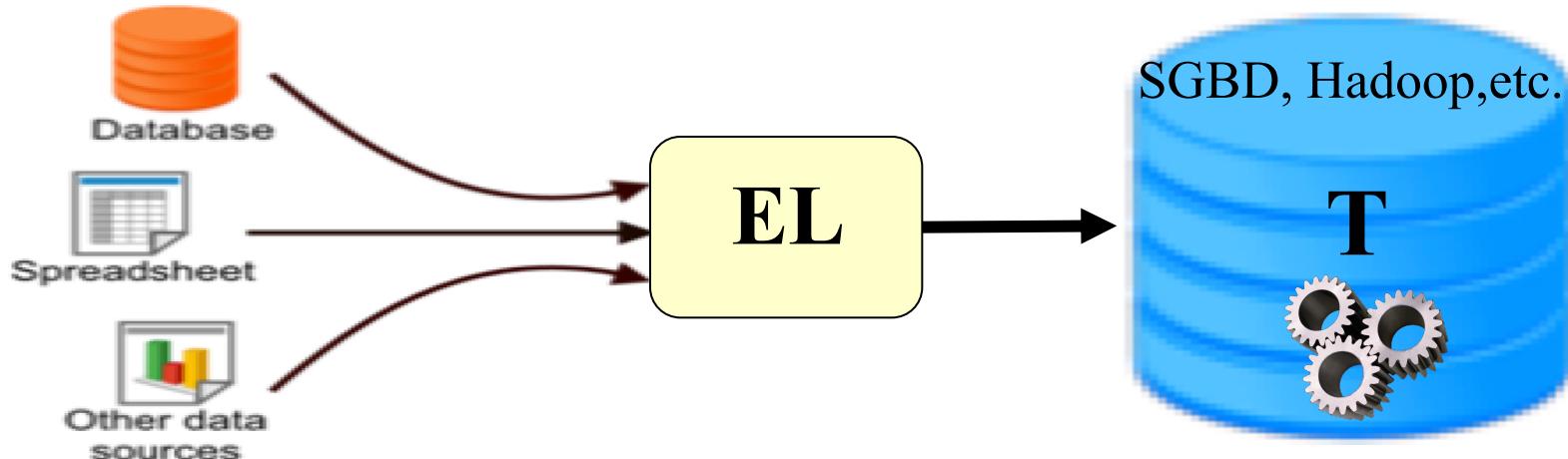
Last edited by jfaza (2017-01-20 16:16:23)

Offline

Intégration de données dans le Big Data : le processus ELT

☞ ELT : Extract, Load, Transform

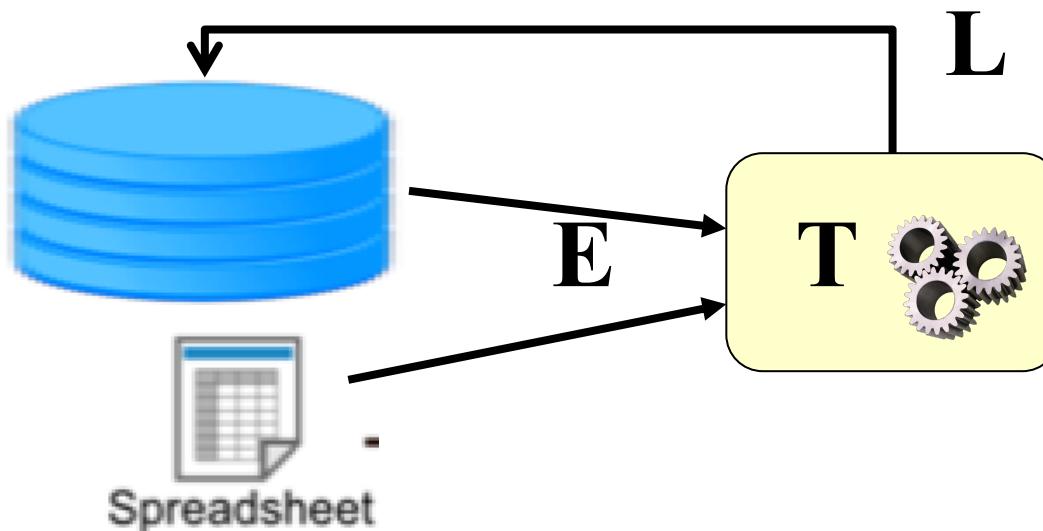
- Mêmes objectifs que l'ETL, mais l'ordre diffère !
- Pousse les traitements (T de ELT) vers les données (philosophie du Big Data !)
- Les traitements sont ainsi réalisés par le système cible hébergeant les données (SGBD, ERP, Hadoop, CRM, etc.)
- Par conséquent, les traitements sont écrits dans le langage cible (SQL, PL/SQL, Hive, Pig, MapReduce, Spark, etc.)



Intégration de données dans le Big Data

Limites du processus ETL

- ☞ Le volume de données pouvant être intégré est limité
 - Les traitements d'intégration sont exécutés par le moteur ETL
 - Les moteurs ETL classiques utilisent les ressources (CPU, RAM, HDD) de (une seule !) la machine
- ☞ Exemple : Mise à jour d'une base de données avec des données issues d'un fichier externe !



Architectures d'intégration matérialisée

Avantages et inconvénients

☞ Avantages

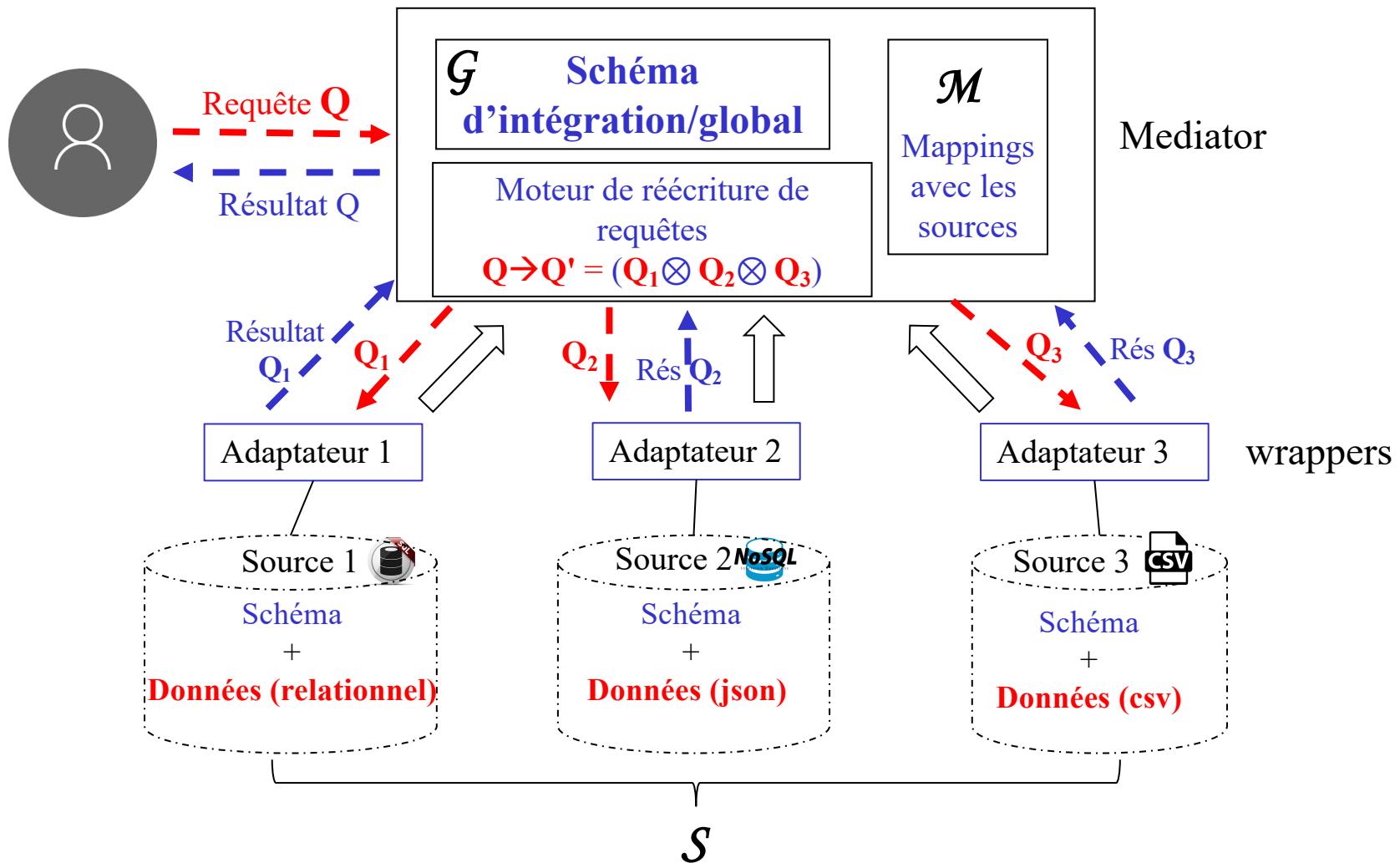
- Adaptée aux processus d'analyse de données (massives)
- Adaptée aux transformations complexes

☞ Inconvénients (maintenabilité de l'entrepôt de données) :

- Peu adaptée aux données changeantes (temps-réel)
- Peu adaptée aux schémas évolutifs
- Peu adaptée en cas d'ajout fréquent de nouvelles sources

Architecture d'intégration virtuelle

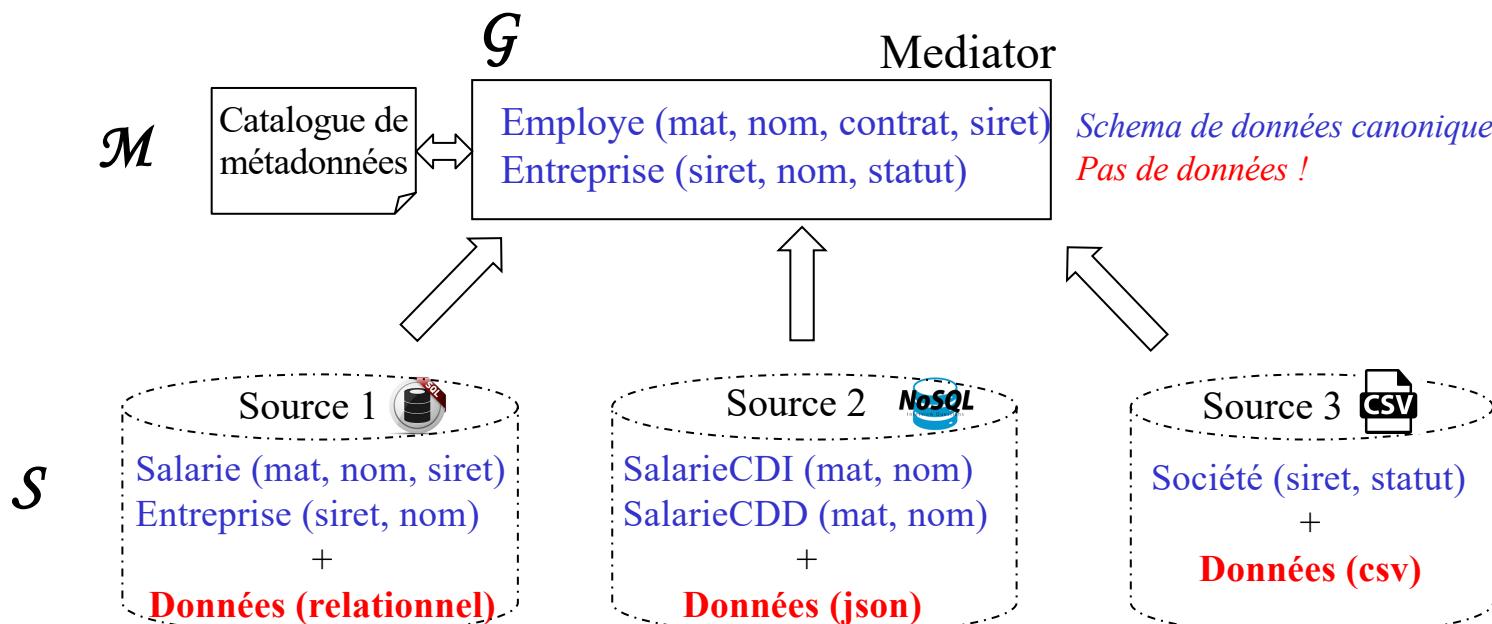
Intégration de schémas



Architecture d'intégration virtuelle

Intégration de schémas

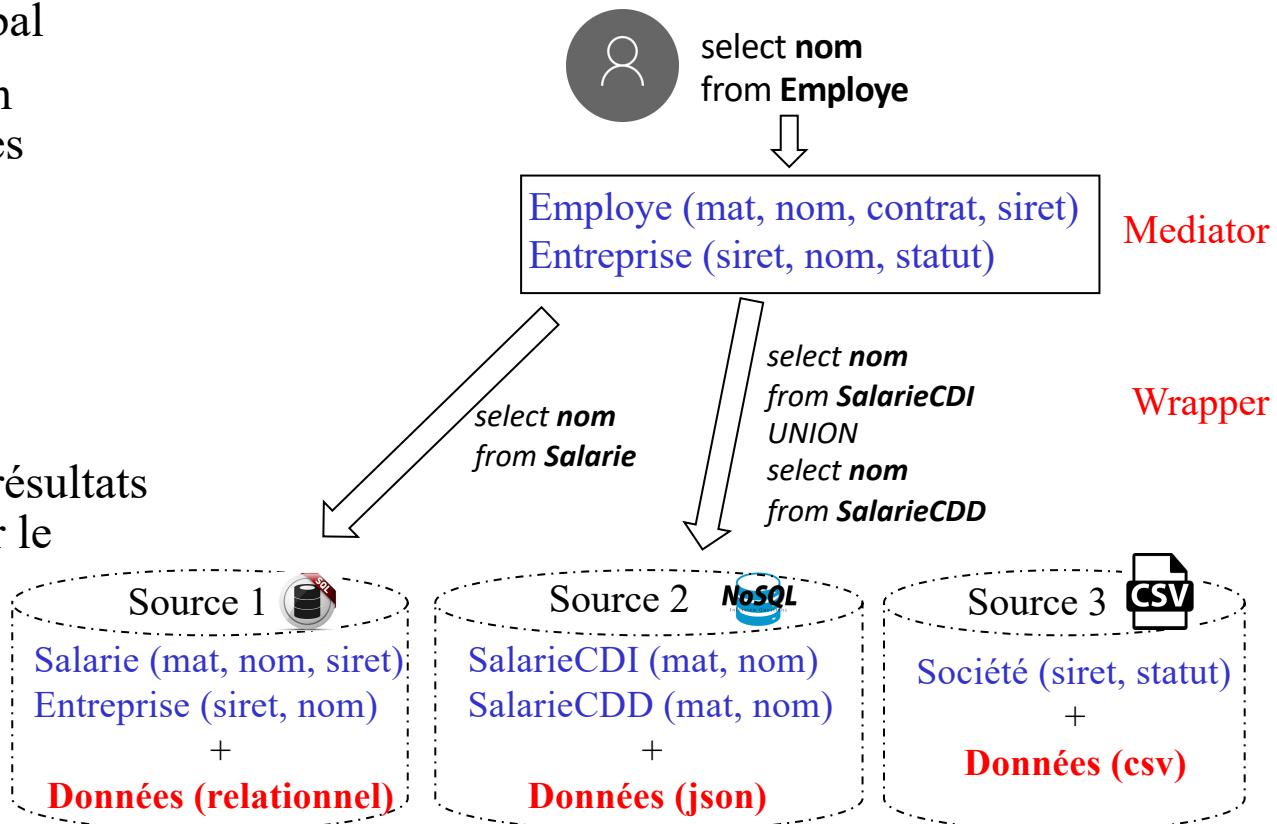
- ☞ Intégration des schémas des sources en un schéma canonique (global)
- ☞ Les données sont maintenues dans les sources et sont combinées (intégrées) pendant l'évaluation de requêtes



Architecture d'intégration virtuelle

Evaluation de requêtes

- Les requêtes utilisateurs sont exprimées sur le schéma global
- Elles sont ensuite traduites en plusieurs sous-requêtes sur les schémas des sources
- Les *Wrappers* évaluent les requêtes sur les sources et renvoient les résultats
- Le médiateur combine les résultats des *Wrappers* pour constituer le résultat final



Architecture d'intégration virtuelle

Intégration de schémas

☞ Modèle de description des schémas G et S :

- Relationnel :

```
Employe (mat, nom, contrat, siret)  
Entreprise (siret, nom, statut)
```

- XML (XMLSchem) :

```
<employe mat=...>  
    <nom>...</nom>  
</employe>
```

- Logique :

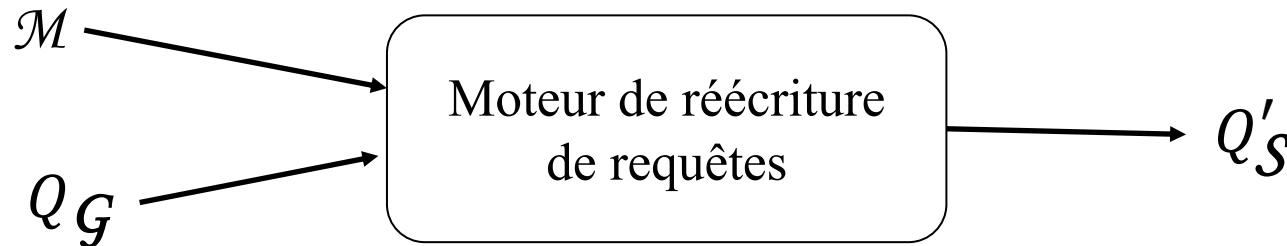
```
∀x Employe(x) ⇒ Personne(x)  
∀x Personne(x) ⇒ ¬Entreprise(x)  
∀x Employe (x) ⇒ ∃y embauche (x, y) ∧ Entreprise (y)  
∀x,y embauche (x, y) ⇒ Employe(x) ∧ Entreprise (y)
```

- Autres : graphe, JSON, etc.

Architecture d'intégration virtuelle

Intégration de données par réécriture de requêtes

- ☞ La réécriture de requêtes dépend des *mappings* \mathcal{M}



- ☞ Types de réécriture

- Réécriture partielle : $Q' \sqsubseteq Q$ (Q' donne une partie de Q)
- Réécriture équivalente : $Q' \sqsubseteq Q$ et $Q \sqsubseteq Q'$ (Q' donne le même résultat que Q)
 - cas d'usage : utilisée par beaucoup de SGBD pour l'optimisation de requêtes
- Réécriture maximale : $\exists Q'' \text{ tel que } Q' \sqsubseteq Q'' \text{ et } Q'' \sqsubseteq Q$
 - (Q' donne le résultat qui se rapproche le plus à celui de Q)
 - cas d'usage : intégration de données

Formulation logique du problème d'intégration

Inclusion de requêtes (*Query containment*)

☞ Inclusion de requêtes (*query containment*), dénotée par $Q_1 \sqsubseteq Q_2$

$Q_1 : \exists \vec{y}_1 \phi_1(\vec{x}_1, \vec{y}_1)$ est incluse dans $Q_2 : \exists \vec{y}_2 \phi_2(\vec{x}_2, \vec{y}_2)$, on écrit $Q_1 \sqsubseteq Q_2$, si :

$$\vec{x}_1 = \vec{x}_2 \text{ et } Q_1 \vDash Q_2$$

Autrement dit, pour toute base de données D et pour tout tuple $\vec{t} \in D$ on a :

$$\vec{t} \in \text{Résultats}(Q_1, D) \Rightarrow \vec{t} \in \text{Résultats}(Q_2, D)$$

Théorème : si Q_1 et Q_2 sont des requêtes conjonctives alors :

$Q_1 \sqsubseteq Q_2$ ssi il existe un homomorphisme σ de Q_2 vers Q_1

☞ Homomorphisme de requêtes

Un homomorphisme σ de $Q_2 (h_2 \leftarrow b_2)$ vers $Q_1 (h_1 \leftarrow b_1)$ est une fonction qui unifie les variables de Q_2 avec les variables ou constantes de Q_1 tout en respectant les deux conditions : $\sigma(h_2) = h_1$ et $\sigma(b_2) \subseteq b_1$

Formulation logique du problème d'intégration

Inclusion de requêtes (*Query containment*)

☞ Inclusion de requêtes (*query containment*) : exemples

vérifier si $Q_1 \sqsubseteq Q_2$ dans les cas suivants :

$$Q_1(x): -R(x, x)$$

$$Q_2(y): -R(y, z)$$



$$\sigma = \{y/x, z/x\}$$

$$Q_1(x): -R(x, y), S(y)$$

$$Q_2(x'): -R(x', y')$$



$$\sigma = \{x'/x, y'/y\}$$

$$Q_1(x): -R(x, y), S(z, t)$$

$$Q_2(x'): -R(x', y'), S(y', z')$$



$$\sigma = \{x'/x, y'/?, z'/t\}$$

$$Q_1(x, y): -R(x, z), S(z, z), R(z, y)$$

$$Q_2(x', y'): -R(x', z'), S(z', t'), R(t', y')$$



$$\sigma = \{x'/x, y'/y, z'/z, t'/t\}$$

Formulation logique du problème d'intégration

Inclusion de requêtes (*Query containment*)

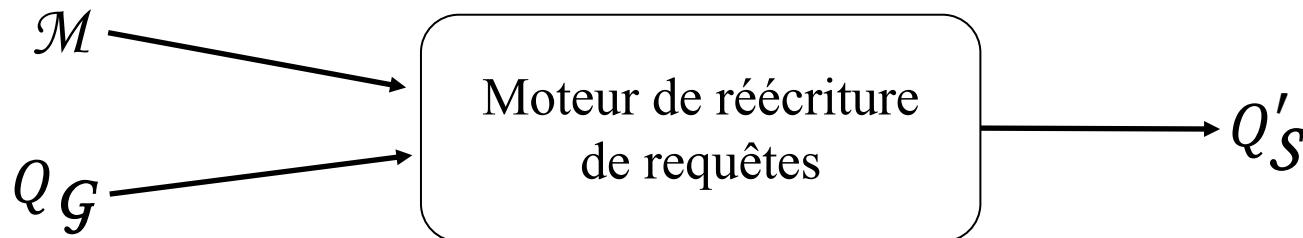
☞ Inclusion de requêtes (*query containment*) : complexité

- Le problème d'inclusion de requêtes est indécidable pour SQL et l'algèbre relationnelle
- Il est décidable pour les requêtes conjonctives
- Dans ce deuxième cas, la décidabilité est NP-complet et sa complexité expérimentale est raisonnable

Architecture d'intégration virtuelle

Intégration de schémas

- ☞ La réécriture de requêtes dépend des *mappings* \mathcal{M}



- ☞ Deux principales approches de représentation de \mathcal{M} :
 - GAV (Global As View) : décrire les relations du schéma global avec celles des schémas des sources
 - LAV (Local As View) : décrire les relations des schémas des sources avec celles du schéma global
 - GLAV (GLAV !)

Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

☞ *Mappings GAV* : chaque *relation* du schéma global est exprimée en fonction des *relations* des sources

☞ Formellement, \mathcal{M} est un ensemble de règles :

$$\forall \vec{x} \ g(\vec{x}) \leftarrow \phi_S(\vec{x}) \quad \text{pour tout } g \in \mathcal{G}$$

ϕ_S est une requête exprimée sur S de même arité que g

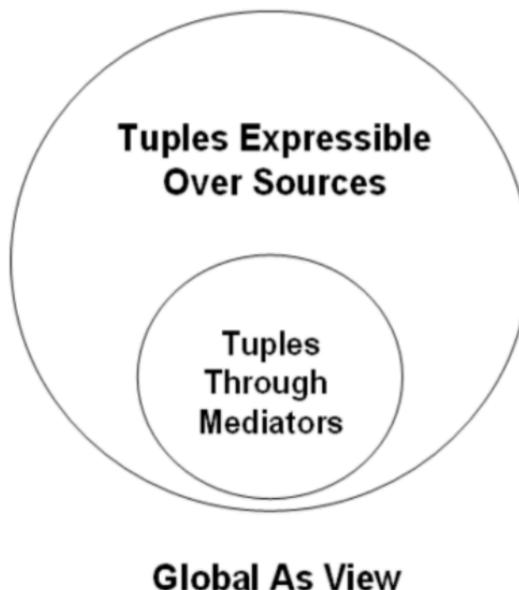
☞ Sémantique d'une règle de *mapping* de \mathcal{M} :

- $db^{\mathcal{G}}$: base de données (virtuelle) du schéma d'intégration
- db^S : base de données (physique) des sources
- s : une relation source ($s \in S$)
- pour tout $g \in \mathcal{G}$ on a : $db^S(s) \subseteq db^{\mathcal{G}}(g)$

Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

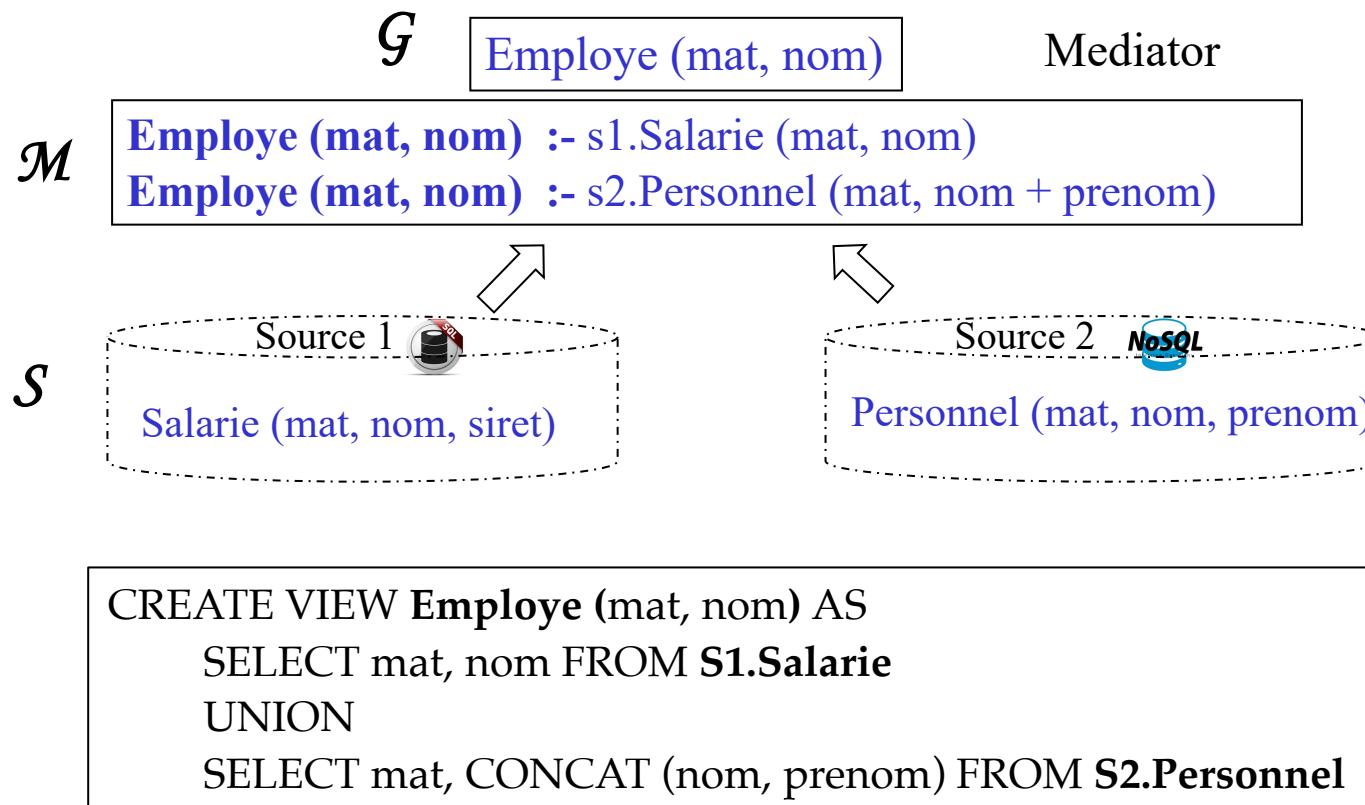
- ☞ On indique au *Médiator* comment peut-il constituer les données de chaque *relation* du schéma global à partir des schémas des sources
- ☞ Le schéma global est une vue du contenu des sources
- ☞ Approche connue en bases de données (G est une vue sur S)



Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

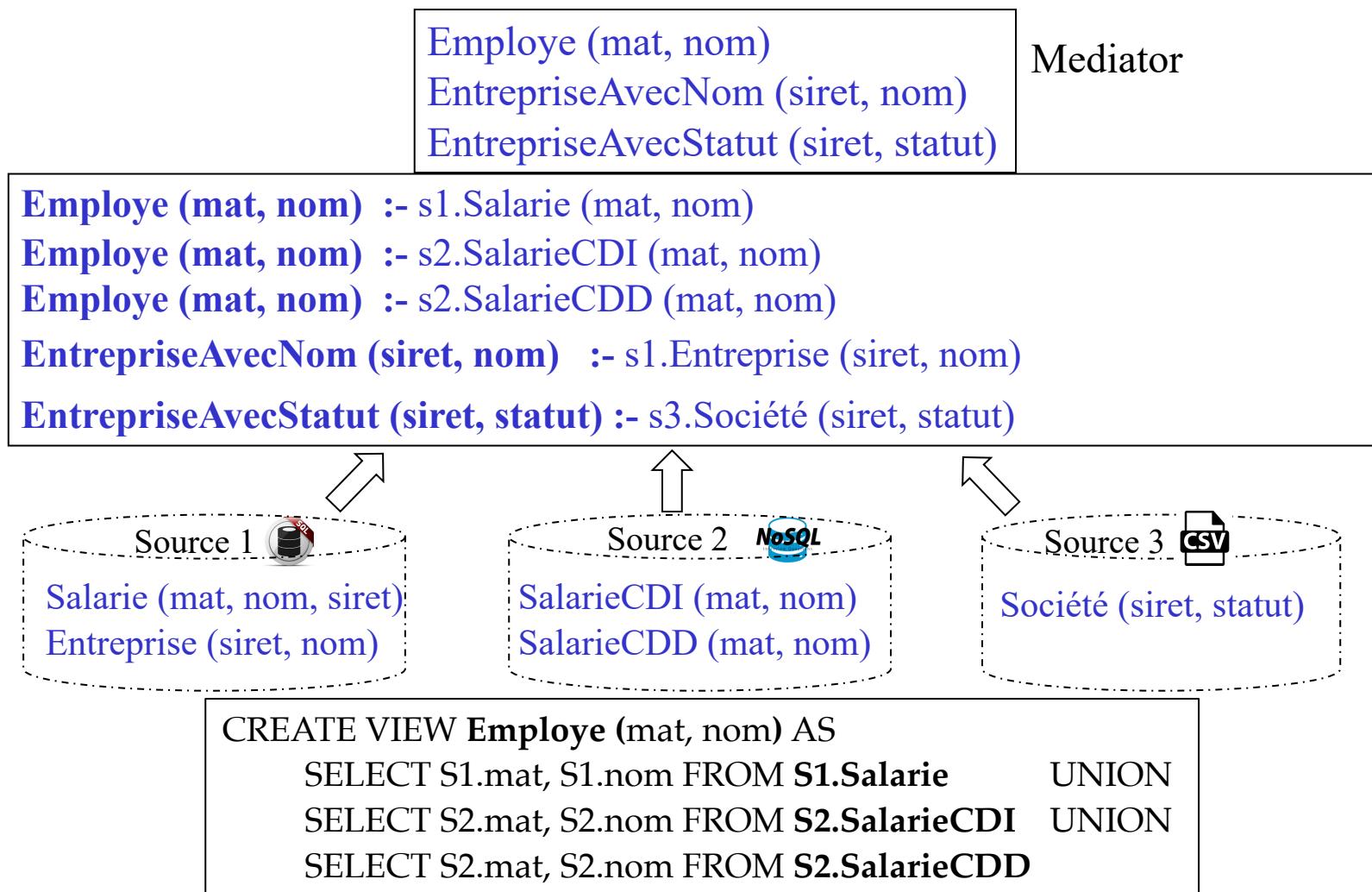
☞ Exemple : vue de tous les employés (exhaustivité des enregistrements)



Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

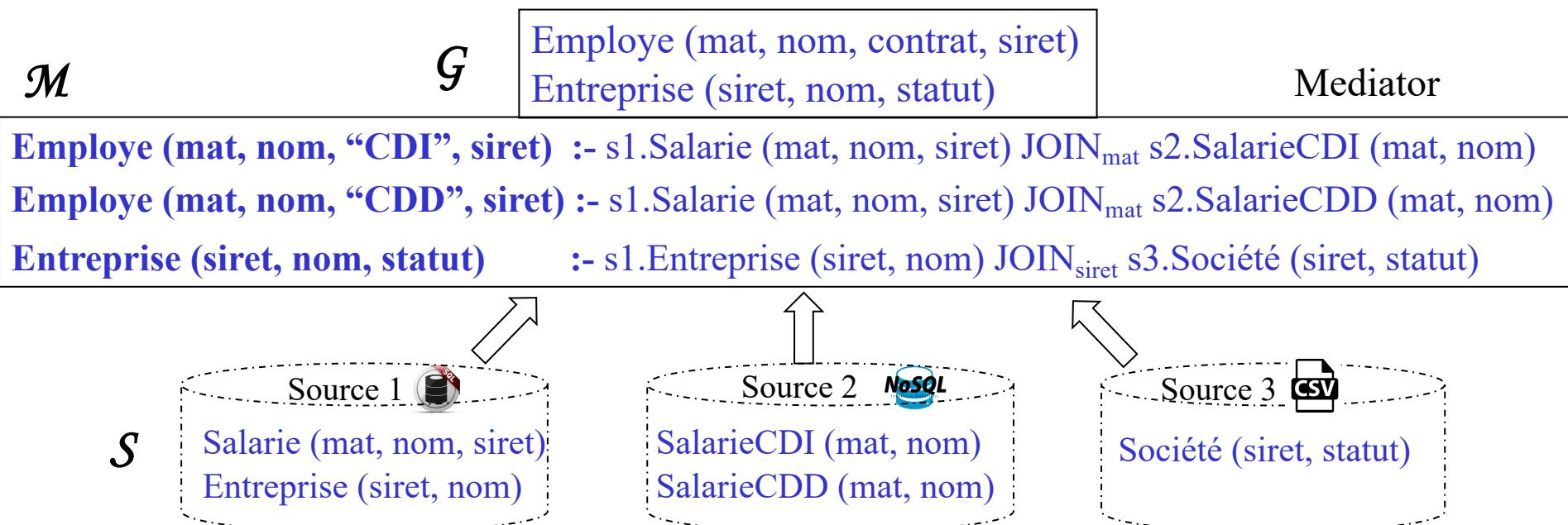
- Exemple : vue de tous les employés (exhaustivité des enregistrements)



Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

- ☞ Exemple : vue des employés ayant tous les attributs (exhaustivité des attributs)

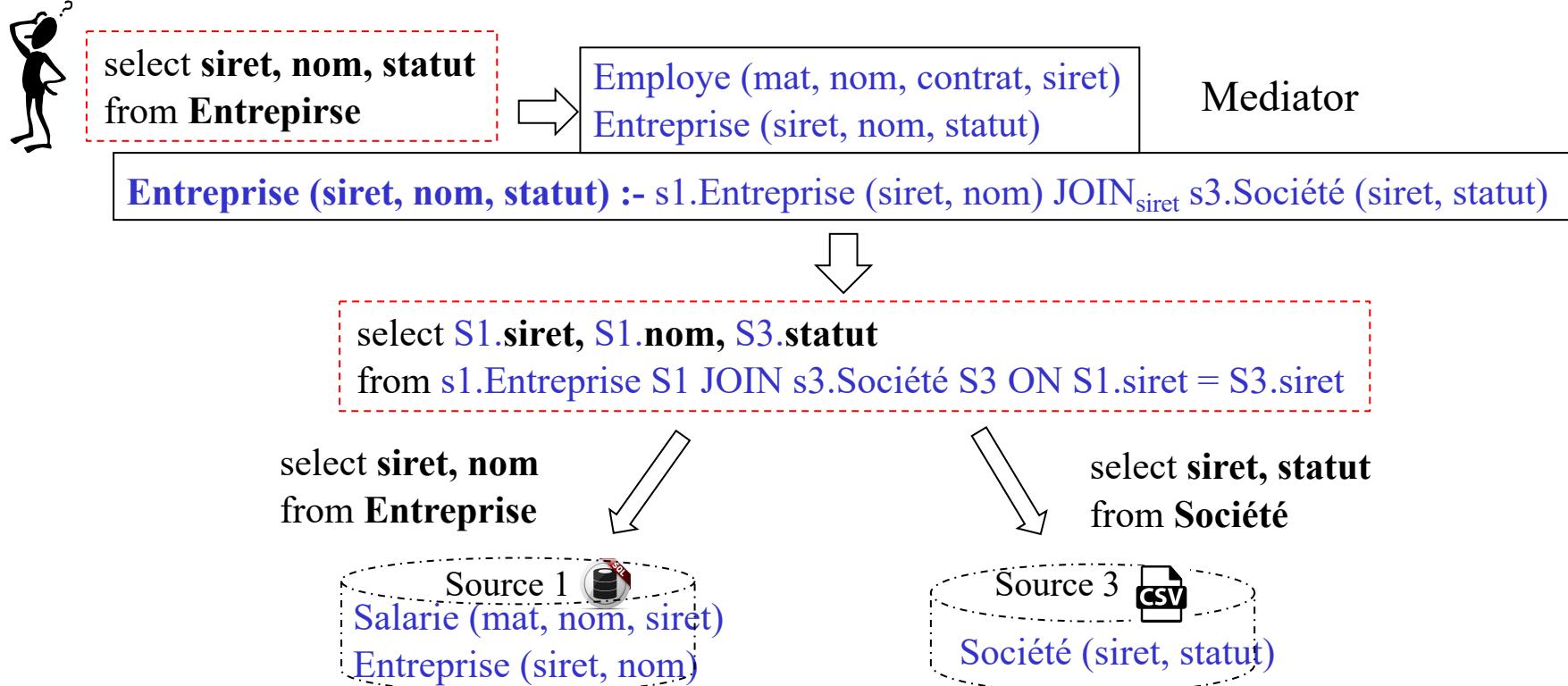


```
CREATE VIEW Employe (mat, nom, contrat, siret) AS
    SELECT S1.mat, S1.nom, "CDI", S1.siret
    FROM S1.Salarie S1 INNER JOIN S2.SalarieCDI S2 ON S1.siret = S2.siret
    UNION
    SELECT S1.mat, S1.nom, "CDD", S1.siret
    FROM S1.Salarie S1 INNER JOIN S2.SalarieCDD S2 ON S1.siret = S2.siret
```

Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

- ☞ Evaluation de requêtes simple : elle se fait par réécriture de requête
- ☞ Réécriture de requêtes par expansion : remplacer les termes/*relations* du schéma global par leurs correspondants dans les schémas des sources



Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Global As View (GAV)

☞ Avantages :

- ☞ Approche classique : le schéma global est une vue sur les sources
- ☞ Simplicité de la reformulation de requêtes

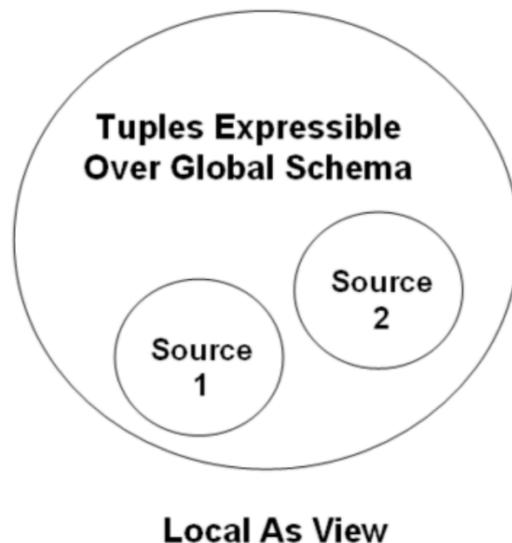
☞ Inconvénients (maintenance des *mappings* \mathcal{M}) :

- ☞ L'ajout et la suppression de sources est complexe car nécessite de redéfinir les règles de *mappings*

Architecture d'intégration virtuelle

Description du schéma d'intégration : Local As View (LAV)

- ☞ *Mappings* : Les *relations* des sources sont décrites en fonction de celles du schéma d'intégration
- ☞ Donc, les sources sont représentées comme des vues (matérialisées) sur le schéma d'intégration
- ☞ On indique au *Médiator* la part du schéma global que chaque source peut fournir



Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Local As View (LAV)

☞ *Mappings LAV* : chaque *relation source* $s \in S$ est exprimée en fonction des *relations* du schéma global G

☞ Formellement, \mathcal{M} est un ensemble de règles :

$$\forall \vec{x} \ s(\vec{x}) \leftarrow \phi_G(\vec{x}) \quad \text{pour tout } s \in S$$

ϕ_G est une requête exprimée sur G de même arité que s

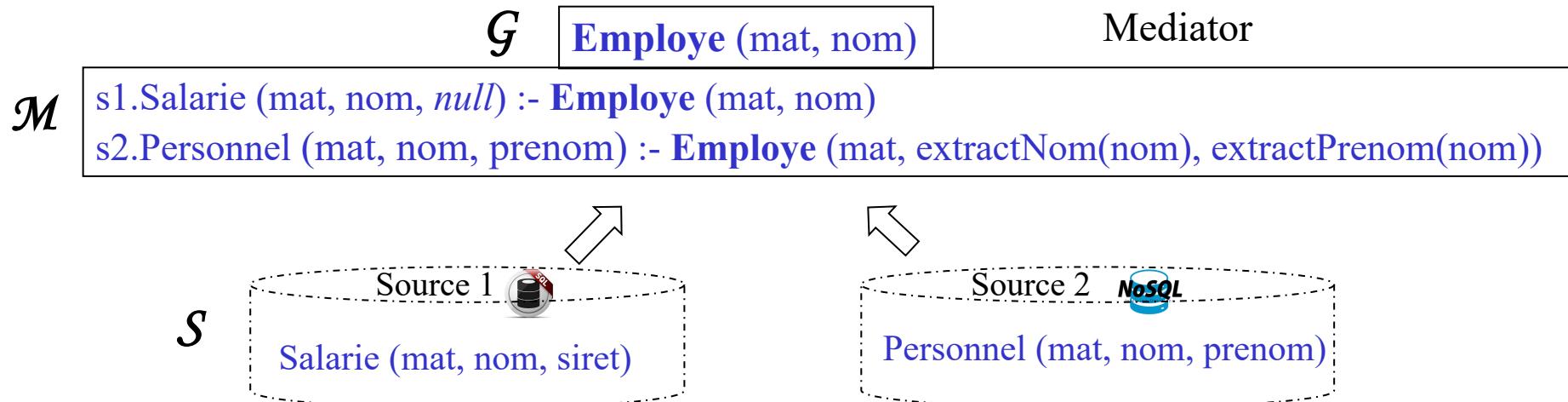
☞ Sémantique d'une règle de *mapping* de \mathcal{M} :

- db^G : base de données (virtuelle) du schéma d'intégration
- db^S : base de données (physique) des sources
- s : une relation source ($s \in S$)
- pour tout $g \in G$ on a : $db^S(s) \subseteq db^G(g)$

Architecture d'intégration virtuelle (EII)

Description du schéma d'intégration : Local As View (LAV)

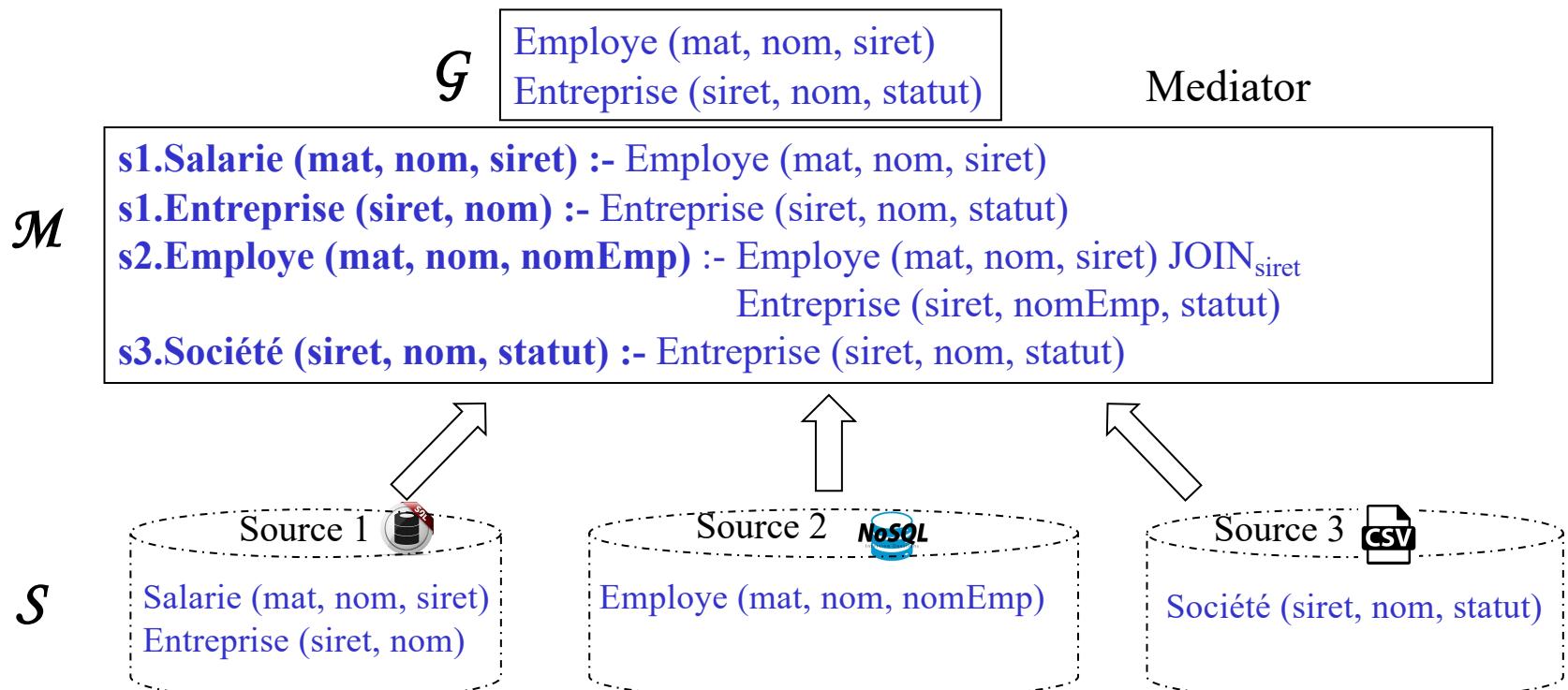
☞ Exemple 1 : décrire les sources en utilisant les relations du schéma global...



Architecture d'intégration virtuelle

Description du schéma d'intégration : Local As View (LAV)

☞ Exemple 2



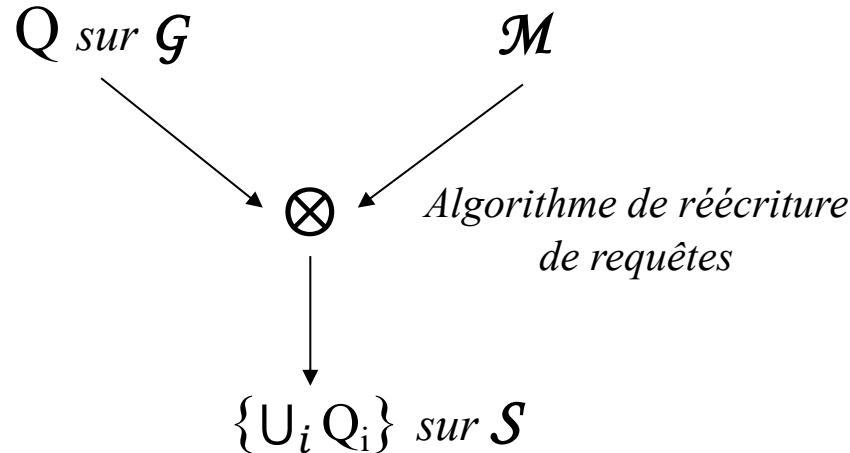
Architecture d'intégration virtuelle

Local As View (LAV) : évaluation de requêtes par réécriture

- ☞ Requête (conjonctive) : nom des employés et celui de leurs employeurs ?

$Q(\text{nom}, \text{nEmp}) = \text{Employe}(\text{mat}, \text{nom}, \text{siret}) \text{ JOIN}_{\text{siret}} \text{Entreprise}(\text{siret}, \text{nEmp}, \text{statut})$

- ☞ Réécrire cette requête en une union de plusieurs requêtes conjonctives exprimées sur les relations des sources



- ☞ Trois principaux algorithmes de réécriture : *Bucket*, *Inverse rules* et *Minicon*

Réf : *Querying Heterogeneous Information Sources Using Source Descriptions*. VLDB : 251-262 (1996)

MiniCon: A scalable algorithm for answering queries using views. VLDB J. 10(2-3): 182-198 (2001)

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme de réécriture de requêtes *Bucket*

☞ L'algorithme de réécriture de requêtes Q dit « *Bucket* »

1. Identifier les sources de Q : créer un *bucket* (ensemble de relations sources $s_i \in S$) pour chaque prédicat/sous-but $g \in Q$ à partir de \mathcal{M} tel qu'il existe une fonction d'unification $\sigma(g \mapsto s_i)$ et que pour tout $x \in \text{head}(Q)$, $\sigma_i(x) \in \text{head}(s_i)$
2. Générer les requêtes candidates (Q_i) par le produit cartésien des *buckets* : une requête candidate est obtenue par combinaison d'une requête de chaque *bucket*
3. Suppression des requêtes candidates donnant un résultat ne faisant pas partie de la requête utilisateur Q : $Q_i \sqsubseteq_{\mathcal{M}} Q$ (*query containment*)
4. Retourner le résultat final à Q : le résultat à Q est l'union des résultats des Q_i

☞ Sa complexité est exponentielle en la taille de la requête

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme de réécriture de requêtes *Bucket*

☞ Etapes de l'algorithme de réécriture de requêtes « *Bucket* » :

$$s_1(a) \leftarrow g_1(a,b)$$

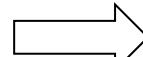
$$s_2(a) \leftarrow g_2(b,a)$$

$$s_3(a) \leftarrow g_1(a,b), g_2(b,a)$$

$$Q(x) :- g_1(x,y), g_2(y,x)$$

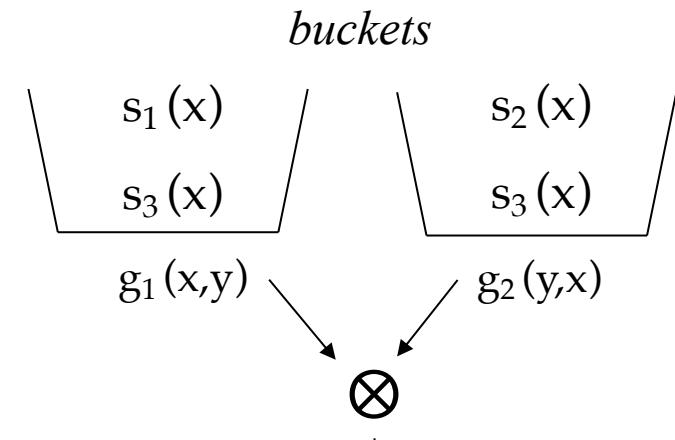
$$\sigma(g_1 \mapsto s_1) = \{a/x, b/y\}$$

$$\sigma(g_1 \mapsto s_3) = \{a/x, b/y\}$$



$$\sigma(g_2 \mapsto s_2) = \{b/y, a/x\}$$

$$\sigma(g_2 \mapsto s_3) = \{b/y, a/x\}$$



Requêtes finales

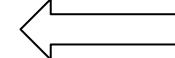
✗ $Q_1(x) :- g_1(x,y), g_2(z,x)$

✓ $Q_2(x) :- g_1(x,y), g_1(x,z), g_2(z,x)$

✓ $Q_3(x) :- g_1(x,y), g_2(y,x), g_2(x,t)$

✓ $Q_4(x) :- g_1(x,y), g_2(y,x)$

$$Q_i \sqsubseteq_{\mathcal{M}} Q ?$$



Requêtes candidates

✗ $Q_1(x) :- s_1(x), s_2(x)$ ✗

✗ $Q_2(x) :- s_1(x), s_3(x)$ ✗

✗ $Q_3(x) :- s_3(x), s_2(x)$ ✗

✓ $Q_4(x) :- s_3(x)$ ✓

$$\text{Rew}_{\sigma}(Q) = Q_2 \cup Q_3 \cup Q_4 = Q_4 \text{ car } Q_2 \sqsubseteq Q_4 \text{ et } Q_2 \sqsubseteq Q_4$$

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme de réécriture de requêtes *Bucket*

- ☞ Requête utilisateur : nom des employés avec celui de leurs employeurs ?

$Q(\text{nom}, \text{nEmp}) = \text{Employe}(\text{mat}, \text{nom}, \text{siret}) \text{ JOIN}_{\text{siret}} \text{Entreprise}(\text{siret}, \text{nEmp}, \text{statut})$

- ☞ Algorithme de réécriture « *Bucket* »

1. Identifier les sources de chaque relation dans la requête (*buckets*) :

s1.Salarie (mat, nom, siret)

s2.Employe (mat, nom, nEmp)

Employe (mat, nom, siret)

Bucket 1

s1.Entreprise (siret, nEmp),

s3.Société (siret, nEmp, statut)

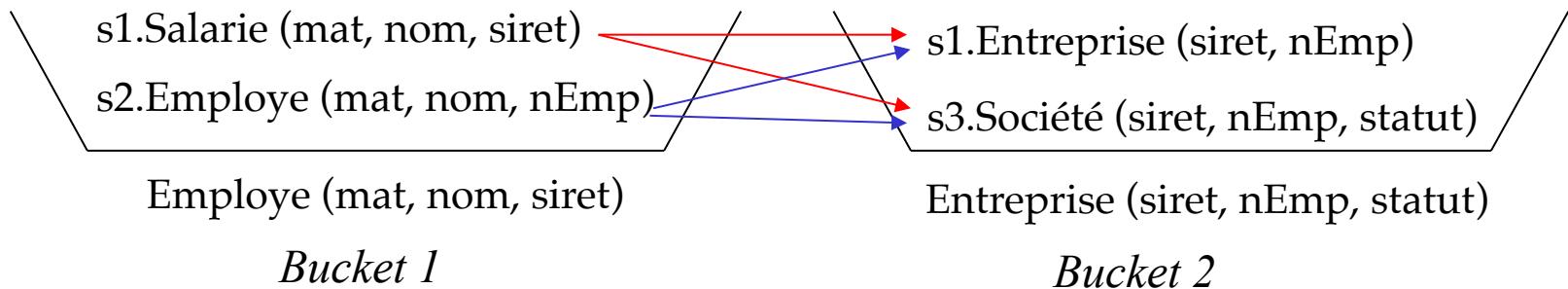
Entreprise (siret, nEmp, statut)

Bucket 2

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme de réécriture de requêtes *Bucket*

2. Générer les requêtes candidates en combinant les relations des *buckets*



$Q_1 (\text{nom}, \text{nEmp}) :- s1.\text{Salarie} (\text{mat}, \text{nom}, \text{siret}), s1.\text{Entreprise} (\text{siret}, \text{nEmp})$

$Q_2 (\text{nom}, \text{nEmp}) :- s1.\text{Salarie} (\text{mat}, \text{nom}, \text{siret}), s3.\text{Société} (\text{siret}, \text{nEmp}, \text{statut})$

$Q_3 (\text{nom}, \text{nEmp}) :- s2.\text{Employe} (\text{mat}, \text{nom}, \text{nEmp}), s1.\text{Entreprise} (\text{siret}, \text{nEmp})$

$Q_4 (\text{nom}, \text{nEmp}) :- s2.\text{Employe} (\text{mat}, \text{nom}, \text{nEmp}), s3.\text{Société} (\text{siret}, \text{nEmp}, \text{statut})$

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme de réécriture de requêtes *Bucket*

3. Validation des requêtes candidates : supprimer les combinaisons non incluses/contenues dans la requête initiale (*query containment*)

Vérifier pour chaque Q_i que : $Q_i \sqsubseteq_{\mathcal{M}} Q$

Cette vérification se fait par expansion de Q_i en utilisant les relations de \mathcal{M} :

$Expansion_{\mathcal{M}}(Q_i) \sqsubseteq Q$

Exemple (suite) :

$Q_2 :- s1.\text{Salarie}(\text{mat}, \text{nom}, \text{siret}), s3.\text{Société}(\text{siret}, \text{nEmp}, \text{statut}) \sqsubseteq Q(\text{nom}, \text{nEmp}) ?$

Employe (mat, nom, **siret**), Entreprise (**siret**, nom, statut) $\sqsubseteq Q(\text{nom}, \text{nEmp})$ 

$Q_3 :- s2.\text{Employe}(\text{mat}, \text{nom}, \text{nEmp}) \sqsubseteq Q(\text{nom}, \text{nEmp}) ?$

Employe (mat, nom, **siret**), Entreprise (**siret**, nEmp, statut) $\sqsubseteq Q(\text{nom}, \text{nEmp})$ 

$Q_5 :- s1.\text{Entreprise}(\text{siret}, \text{nEmp}) \sqsubseteq Q(\text{nom}, \text{nEmp}) ?$

Entreprise (siret, nom, statut) $\sqsubseteq Q(\text{nom}, \text{nEmp})$ 

$Q_6 :- s3.\text{Société}(\text{siret}, \text{nEmp}, \text{statut}) \sqsubseteq Q(\text{nom}, \text{nEmp}) ?$

Entreprise (siret, nom, statut) $\sqsubseteq Q(\text{nom}, \text{nEmp})$ 

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *bucket*

☞ Quelques éléments de complexité algorithmique

n : taille de la requête, m : taille des vues, v : nombre de vues

- ☞ Nombre de buckets : n
- ☞ Taille d'un bucket : $O(m \times v)$
- ☞ Génération de buckets : $O(n \times m \times v)$
- ☞ Nombre de réécritures possibles (sous-requêtes candidates) : $O((m \times v)^n)$
- ☞ Nombre d'inclusions de requêtes : $O((m \times v)^n)$

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

- ☞ L'algorithme *Minicon* est une optimisation de l'algorithme *Bucket*
- ☞ Idée de l'algorithme :
Ne pas ajouter aux *buckets* (*MiniCoin Description – MCD*) les prédictats sources générant des réécritures non valides pour éviter l'étape *query containment*
- ☞ Exemple

$$s_1(a) \leftarrow g_1(a, \mathbf{b})$$

$$s_2(a) \leftarrow g_2(\mathbf{b}, a)$$

$$Q(x) :- g_1(x, \mathbf{y}), g_2(\mathbf{y}, x)$$

s_1 (respectivement s_2) ne peut être utilisée la réécriture pour remplacer g_1 (resp. g_2) car la variable de jointure y/b n'apparaît pas dans l'entête. Par conséquent, la variable de jointure disparaîtra de la réécriture suivante (invalide) :

$$Q'(x) :- s_1(x), s_2(x)$$

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

- ☞ Algorithme en deux étapes (pas de test d'inclusion)
 - ☞ Créer les MCD – MiniCon Description (*buckets*) en évitant les sous-but/prédicats inutiles
 - ☞ Générer les réécritures par combinaison de *MCD*
- ☞ MCD (*MiniCon Description*)
 - ☞ Un MCD lie les prédicats de la requête avec une vue en se basant sur sa règle de mapping
 - ☞ MCD : les prédicats de la requête liés par des variables existentielles doivent être unifiés en même temps avec la définition de s_i dans \mathcal{M} pour ne pas perdre les jointures

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

☞ Etape 1 : création de MCD (*MiniCon Description*)

Un MCD est créée pour chaque règle de *mapping* $\mathcal{M}(s_i)$ pour s_i

$mcd_i = \langle s_i, \sigma_i, cov_i \rangle$: σ_i unification $s_i \mapsto g_i$, $cov_i = \{g_i\}$: prédictats de Q couverts

Ajouter s_i à mcd_i si :

$\exists g_i \in Q$ et $\exists g'_i \in \mathcal{M}(s_i)$ tel que $\exists \sigma_i(g'_i \mapsto g_i)$ σ_i : fonction d'unification

$\forall x \in g_i$ (donc $x \in body(Q)$) tels que $x \in head(Q)$ alors $\sigma_i(x) \in head(s_i)$

Pour toute variable existentielle y' dans $\mathcal{M}(s_i)$ tel que $\{y'/y\} \in \sigma_i$ et tout prédictat $g_k (k \neq i) \in Q$ où $y \in g_k$, σ_i doit couvrir g_k

Ajouter g_i et g_k aux prédictats couverts par mcd_i

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

☞ Exemple d'application de l'algorithme de réécriture « *Minicon* » :

$$\mathcal{M}_1 : s_1(a) \leftarrow g_1(a, b)$$

$$\mathcal{M}_2 : s_2(a) \leftarrow g_2(b, a)$$

$$\mathcal{M}_3 : s_3(a) \leftarrow g_1(a, b), g_2(b, a)$$

$$Q(x) :- g_1(x, y), g_2(y, x)$$

Phase 1: calcul des *mcd*

Calcul du mcd_1

Pour $g_1 \in Q$:

$$\sigma_1 = \{a/x, b/y\}$$

b est existentielle dans \mathcal{M}_1 et $b \in g_2$ mais σ_1 ne peut couvrir g_2
donc, $s_1 \notin mcd_1$

Calcul du mcd_2 : similaire au calcul de mcd_1

Calcul du mcd_3

Pour $g_1 \in Q$:

$$\sigma_3 = \{a/x, b/y\}$$

b est existentielle dans \mathcal{M}_3 : $b \in g_2$ et σ_3 couvre déjà g_2
donc, $s_3 \in mcd_3 = \langle s_3, \sigma_3 = \{a/x, b/y\}, cov_3\{g_1, g_2\} \rangle$

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

☞ Etape 2 : génération des réécritures

Une réécriture est obtenue en combinant les s_i des $mcd_i = \langle s_i, \sigma_i, cov_i \rangle$ qui couvrent l'ensemble de la requête Q sans redondance (en se basant sur l'indice de position des calculé dans les mcd)

Créer la réécriture $Q'_i = \bigwedge_{i=1}^k s_k$ pour toute combinaison de mcd :
 $\{mcd_1, \dots, mcd_k\}$ tel que $\bigcup_{i=1}^k cov_i = Q$

☞ Exemple (suite) : génération des réécritures

$$mcd_1 = \emptyset$$

$$mcd_2 = \emptyset$$

$$mcd_3 = \langle s_3, \sigma_3 = \{a/x, b/y\}, \{g_1, g_2\} \rangle$$

Générer les réécritures en combinant les sources s_i données dans les mcd_i de sorte à couvrir tous les prédictats de Q. Ainsi,

$\text{Rew}_{\sigma_3}(Q(x)) = s_3(x)$ est une réécriture valide car couvre Q

Architecture d'intégration virtuelle

Local As View (LAV) : algorithme *Minicon*

☞ Création de MCD : étude de quelques cas

☞ Cas 1 : $\mathcal{M}_1 : s_1(a, b) \leftarrow g_1(a, b)$

$\mathcal{M}_2 : s_2(a) \leftarrow g_2(a, b)$

$Q(x) :- g_1(x, y), g_2(y, x)$

$mcd_1 = \langle s_1, \sigma_1 = \{a/x, b/y\}, cov_1 = \{g_1\} \rangle$

$mcd_2 = \emptyset$ (*b* est existentielle, g_1 non couvert)

Pas de réécriture possible car aucune combinaison de mcd permettant de couvrir Q

☞ Cas 2 : $\mathcal{M}_1 : s_1(a, b, c) \leftarrow g_1(a, b), g_2(a, c)$

$Q(y) :- g_1(x, y), g_2(x, y)$

$mcd_1 = \langle s_1, \sigma_1 = \{a/x, b/y, c/y\}, cov_1 = \{g_1, g_2\} \rangle$

Réécriture : $Q'(y) :- s_1(x, y, y)$

☞ Cas 3 : $\mathcal{M}_1 : s_1(a, b) \leftarrow g_1(a, b), g_2(a, c)$

$\mathcal{M}_2 : s_2(a, b) \leftarrow g_2(a, b)$

$Q(z, y) :- g_1(x, y), g_2(x, z)$

$mcd_1 = \langle s_1, \sigma_1 = \{a/x, b/y, c/z\}, cov_1 = \{g_1\} \rangle$

$g_2 \notin cov_1$ car $z \in g_2$ et $z \in head(Q)$ or $\sigma_1(z) = c \notin head(s_i)$

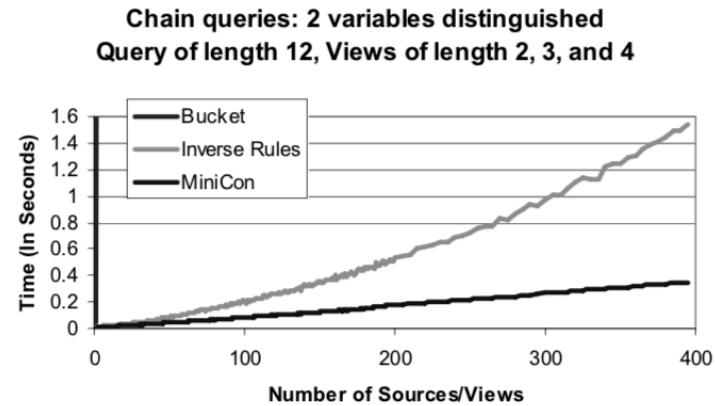
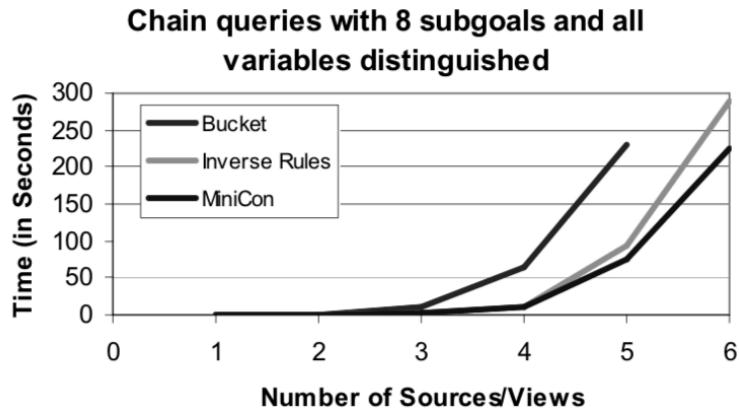
$mcd_2 = \langle s_2, \sigma_2 = \{a/x, b/z\}, cov_2 = \{g_2\} \rangle$

Réécriture : $Q'(z, y) :- s_1(x, y), s_2(x, z)$ obtenue par combinaison de mcd_1 et mcd_2

Architecture d'intégration virtuelle

Local As View (LAV) : algorithmes de réécriture de requêtes

- ☞ La complexité des algorithmes de réécriture de requêtes est exponentielle en la taille des requêtes
- ☞ Du point de vue expérimental, *Minicon* est plus rapide que *Bucket*, notamment si le nombre de relations du schéma global (vues) est grand



- ☞ La réécriture de requêtes (conjonctives) sur des schémas relationnels (global et sources) produit un nombre fini de réécritures
- ☞ La réécriture de requêtes (conjonctives) en présence de contraintes est indécidable

Architecture d'intégration virtuelle

Local As View (LAV)

- ☞ Avantage :
 - ☞ Simplicité des ajouts et suppressions de sources (pas de liens entre les sources !)
- ☞ Inconvénients :
 - ☞ La réécriture de requêtes est complexe (*NP-complet* en cas de requêtes conjonctives sans contraintes, réduction à la 3-coloration de graphes)
 - ☞ Les données recherchées par les utilisateurs (sur le schéma global) peuvent ne pas être fournies par les sources (données manquantes)

Architecture d'intégration virtuelle

Choix de l'architecture GAV vs. LAV

→ Avantages :

- ☞ Accès à des données fraîches
- ☞ Ajout et suppression de sources dynamiquement
- ☞ Passage à l'échelle (*scalabilité horizontale*)

☞ Inconvénients :

- ☞ Forte dépendance de la disponibilité des sources
- ☞ Temps d'accès aux données (reformulation/agrégation) peut être important
- ☞ Adaptée plutôt à la BI (connaissance préalable des variables d'analyse) plutôt qu'à l'apprentissage machine (pas de connaissances préalables des variables explicatives)

Outils d'intégration de données matérialisée *(les outils ETL)*

Quelques outils ETL

☞ Propriétaires

- Talend (Talend SA)
- Informatica (Informatica Corporation)
- Oracle Data Integrator
- SSIS (MS SQLServer)
- IBM InfoSphere DataStage
- SAP SI, SAS SI, etc,

☞ Open source

- Talend Open Studio
- Pentaho DI
- CloverETL (Javlin)

Talend

Talend Open Studio (3.2.0.M1_r26328)

File Edit View Window Help

Repository Selezione

Business Models Job Designs

- t01_Campo
- CustomCode
- Databases
 - Bulk
 - InOut
 - SCD
 - SP
 - ComponentRow 0.1
 - Connection 0.1
- DataQuality
- ELT
- File
- Internet
- LogError
- Misc
- Orchestration
- Processing
- System_

Create table

Insert the data

Drop the primary key

Set the primary key

tCreateTable_1 "demutable"

tRowGenerator_1 (Main)

tMySQLOutput_1

tMySQLRow_1

tMySQLRow_2

OnSubjobOk

tCreateTable_1 (tCreateTable_1 "demutable")
tMySQLOutput_1
tMySQLRow_1 (tMySQLRow_1)
tMySQLRow_2 (tMySQLRow_2)
tRowGenerator_1 (tRowGenerator_1)

Designer Code

Job(Compo) Contexts Component Run (Job C) Problems Modules Talend Exc Scheduler Job Hierarc

0 errors, 0 warnings, 1 info

Description Resource

Errors (0 items)
Warnings (0 items)
Infos (1 item)

Palette Find component...

Business Business Intelligence Custom Code Data Quality Databases

- AS400
- Access
- DB Generic
- DB JDBC
- DB2
- Firebird
- Greenplum
- HSQLDb
- Informix
- Ingres
- Interbase
- JavaDB

ELT

- File
- Internet
- Logs & Errors
- Misc
- MultiSchema
- Orchestration
- Processing
- System
- XML

```
graph TD; subgraph Job [Job demo03_tMap 0.1]; Create[tCreateTable_1 "demutable"]; Insert[tRowGenerator_1 (Main)]; Drop[tMySQLRow_1]; Set[tMySQLRow_2]; end; Create -- OnSubjobOk --> Drop; Drop -- OnSubjobOk --> Set; Set -- OnSubjobOk --> Insert;
```

Talend

Plusieurs éditions adaptées

BIG DATA

Simplifiez l'ETL pour les ensembles de données variés et volumineux

[TÉLÉCHARGEMENTS →](#)

DATA QUALITY

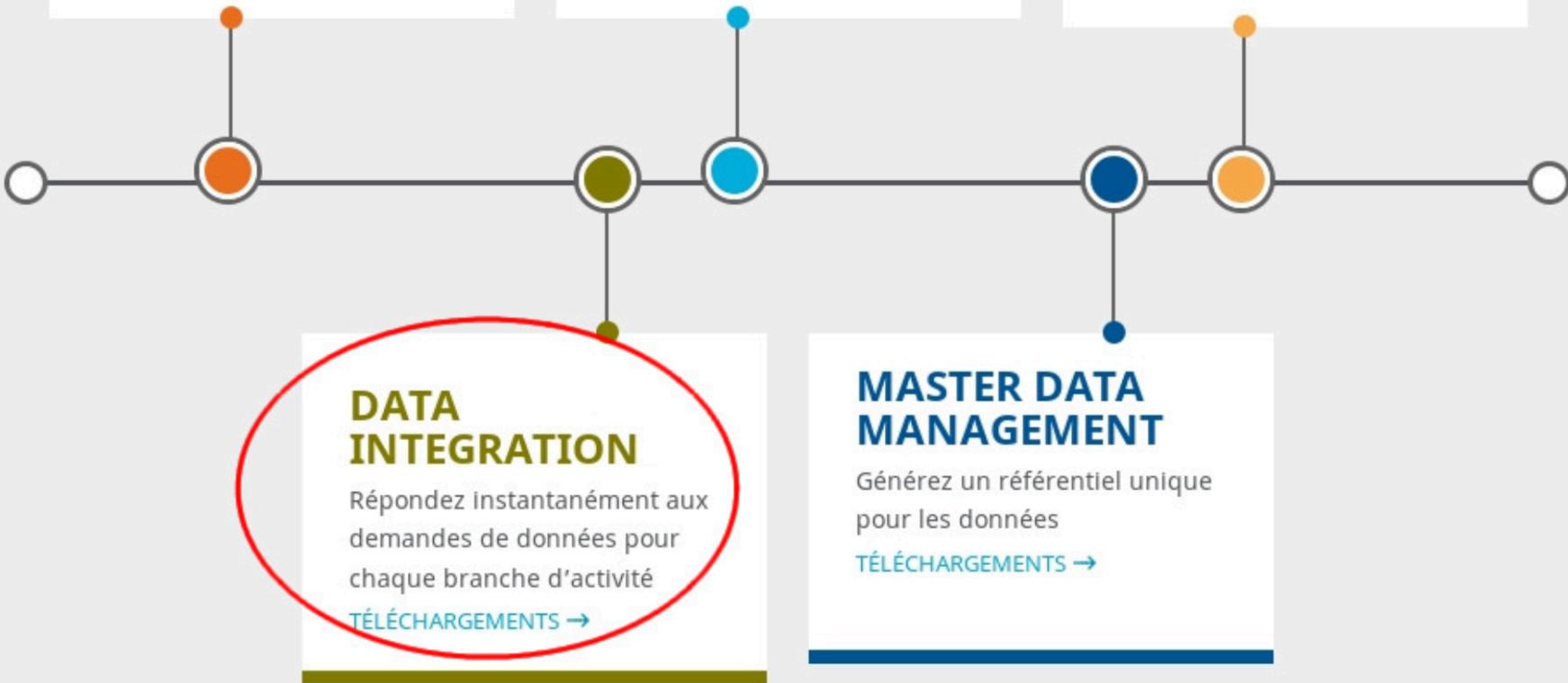
Améliorez la précision et l'intégrité des données

[TÉLÉCHARGEMENTS →](#)

ENTERPRISE SERVICE BUS (ESB)

Simplifiez la connexion des services et des applications

[TÉLÉCHARGEMENTS →](#)



Talend

Quelques composants graphiques

☞ Lecture de données (E)

- Fichiers : tFileInputDelimited, tFileInputExcel, tFileInputXML,...
- Bases de données : tMySQLInput, tOracleInput, tMongoDBInput,...
- Services Web : tREST, tSOAP, etc.

☞ Traitement et transformation de données (T)

- tMap : calcul/transformation, jointure, filtre.
- tSortRow, tFilterRow, tUniqRow : tri, filtre, déduplication
- tNormalize et tDenormalize : normaliser les données tabulaires

☞ Ecriture de données (L)

- Fichiers : tFileOutputDelimited, tFileOututExcel...
- Bases de données : tMySQLOutput, tMongoDBOutput,...

Talend

Composants graphiques

☞ Visualiser des résultats temporaires pour déboguer

- tLogRow : afficher les résultats sur la sortie standard de Talend

☞ Assemblage des composants : trois types de liens

- Liens de transmission de données (row)
- Lien de boucle (Iterate) : pour répéter les actions qui n'acceptent pas de flux entrant (comme les *tFileInput*)
- Lien de précédente entre composants (trigger)

☞ Les matrices de données sont traitées sous forme de flux

- Chaque ligne est traitée dans une itération
- tAggregateRow permet de faire l'agrégation de plusieurs lignes

Talend

Composants graphiques

☞ *globalMap* : une zone mémoire partagée par les composants du projet

- Permet la transmission de données entre composants
- Utilisée par les composants de Talend comme *tMsgBox*, *tFileDialog* et *tIterateToFlow*
- *tMsgBox* y dépose la valeur saisie par l'utilisateur (nom de la clé : NomComposantMsgBox_RESULT)
- *tFileDialog* parcourt les fichiers d'un répertoire et transmet leur nom aux composants suivants via *globalMap* (exemple : CURRENT_FILEPATH)
- *tFlowToIterate* génère une itération sur les composants suivants pour chaque ligne du flux de données entrant en leur transmettant les valeurs de la ligne dans la *globalMap*

Outils d'intégration de données virtuelle *(les outils de médiation)*

Quelques outils d'intégration virtuelle

☞ Propriétaires

- Denodo
- IBM WebSphere Federation Server
- SAP Business Objects Data Federator
- Sybase Avaki
- Xcalia Intermediation Core

☞ Open source

- Denodo