Intégration de données

Dan VODISLAV

CY Cergy Paris Université

Master M2 Recherche SIC

Plan

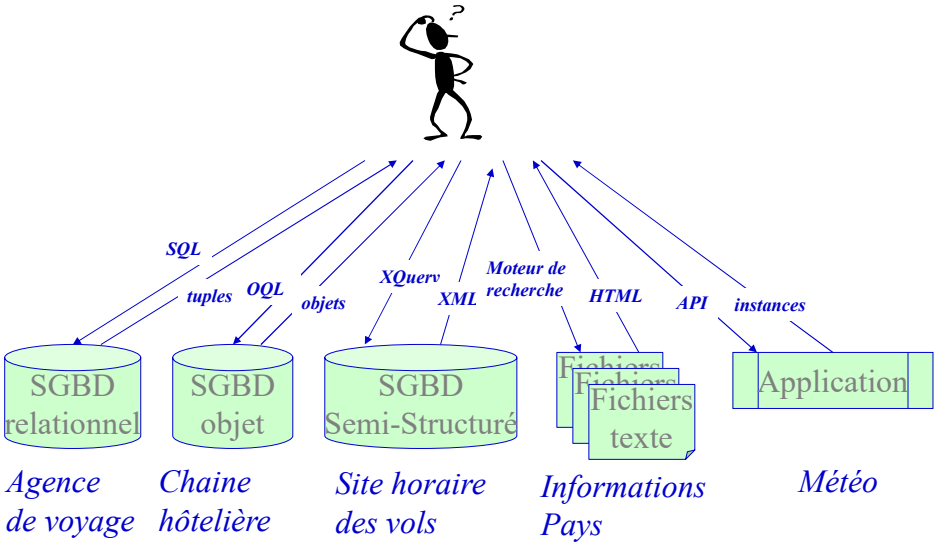
- Objectifs, principes, enjeux, applications
- Architectures d'intégration de données
  - Médiateurs et entrepôts
  - Traitement des requêtes
- Schémas d'intégration
  - Global-as-view
  - Local-as-view

Intégration de données

- Contexte
  - Sources d'information nombreuses et variées
    - SGBD relationnels/XML, pages HTML, LDAP, tableurs, fichiers, applications, …
  - Interfaces d'accès variées
    - Langages d'interrogation: SQL, XPath, XQuery, URL, …
    - Modèle de données: relationnel, XML, HTML, tableurs
    - Protocoles de communication: JDBC, ODBC, SOAP, HTTP
    - Interfaces d'appel: ligne de commande, API, formulaire, interface graphique
- *Objectif général* : utiliser plusieurs sources comme si elles constituaient une seule base de données homogène → ***l'intégration de données*** doit fournir
  - *un accès* (requêtes, éventuellement mises-à-jour)
  - *uniforme* (comme si c'était une seule BD homogène)
  - *à des sources* (pas seulement des BD)
  - *multiples* (déjà deux est un problème)
  - *autonomes* (sans affecter leur comportement, indépendant des autres sources ou du système d'intégration)
  - *hétérogènes* (différents modèles de données, schémas)
  - *structurées* (ou semi-structurées)

Page 3

Exemple

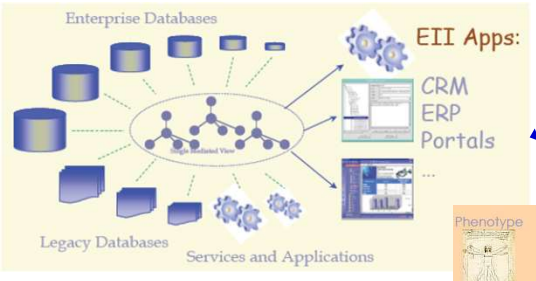


Page 4

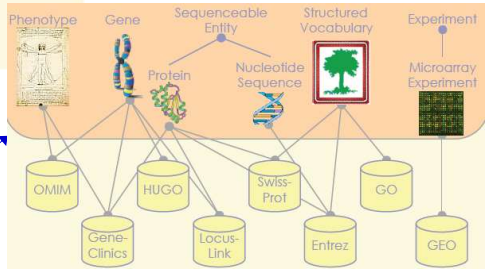
Enjeux

- Dans l'entreprise
  - Données dispersées dans une grande variété de sources hétérogènes:
    - *internes* à l'entreprise (protégées)
    - *externes*, chez des fournisseurs, des partenaires ou des clients
  - Objectif « *business intégration* »: accès *efficace*, *facile* et *sûr* à ces données
  - Études: une partie très importante des budgets IT sont dépensés en intégration
- Grand public
  - Accès simple, rapide et efficace aux informations disponibles sur le web
    - Texte/HTML, images, vidéo, XML, fils RSS, cartes
    - Le web caché, services web
  - Commerce électronique: comparateurs de prix, intégration de magasins en ligne

Applications



Données d'entreprise



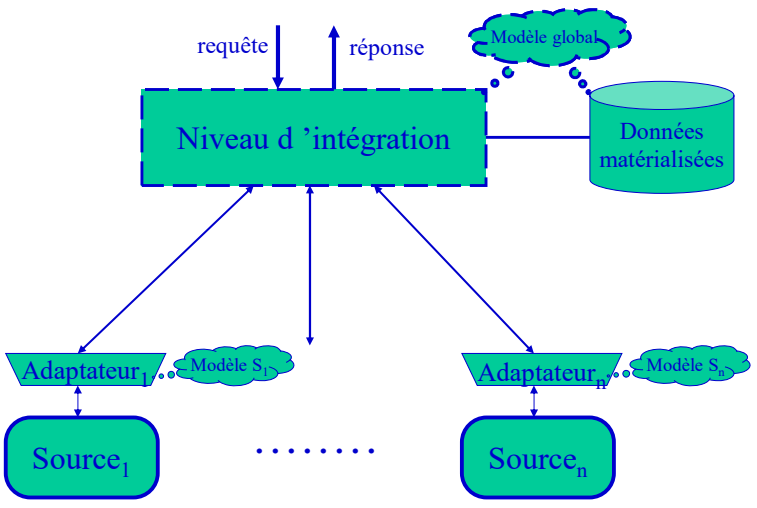
Données d'une communauté

+ **le Web !** → moteurs de recherche, agrégateurs

Caractéristiques des sources de données

- … qui rendent l'intégration de données difficile
  - → *Distribution*
    - Répartition géographique des sources sur le réseau
    - Échelle
  - → *Autonomie*
    - Les sources décident de ce qu'elles partagent, comment et quand
  - → *Hétérogénéité*
    - De format, de structure, de mode d'accès, de capacité de traitement

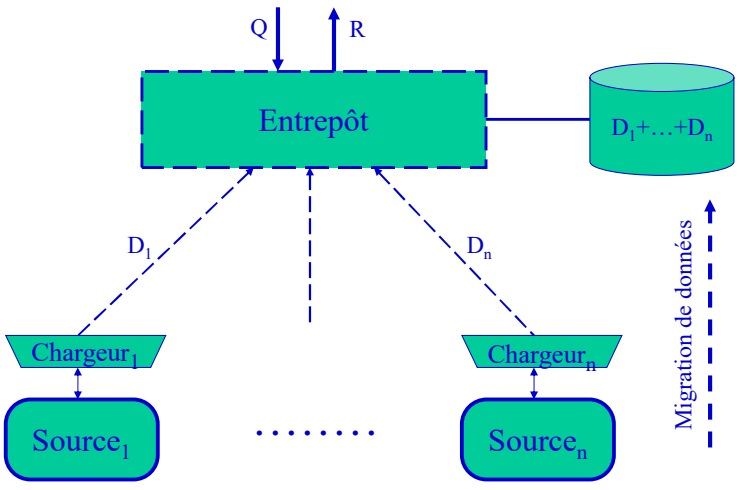
Architecture générale d'intégration



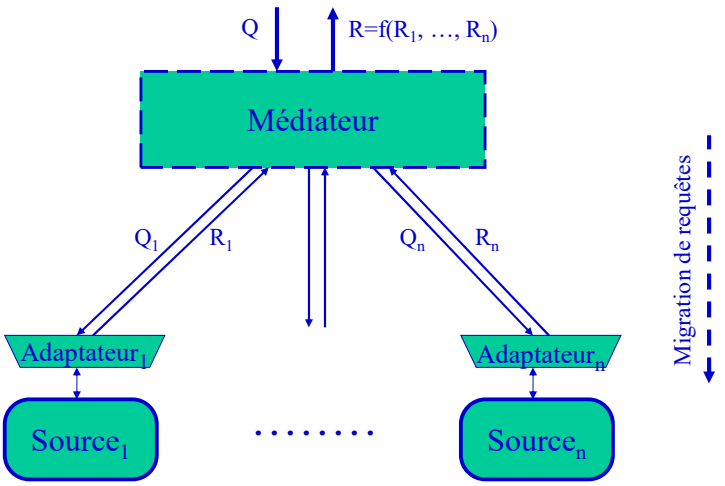
Intégration matérialisée et virtuelle

- Intégration matérialisée → ***entrepôt de données***
  - Les données provenant des sources sont transformées et stockées sur un support spécifique (entrepôt de données).
  - L'interrogation s'effectue comme sur une BD classique
- Intégration virtuelle → ***médiateur***
  - Les données restent dans les sources
  - Les requêtes sont exprimées sur le schéma global, puis décomposées en sous-requêtes sur les sources
  - Les résultats des sources sont combinés pour former le résultat final
- En pratique on peut avoir des architectures intermédiaires, entre ces deux extrêmes

Architecture d'entrepôt



Architecture de médiation



Entrepôt ou médiateur?

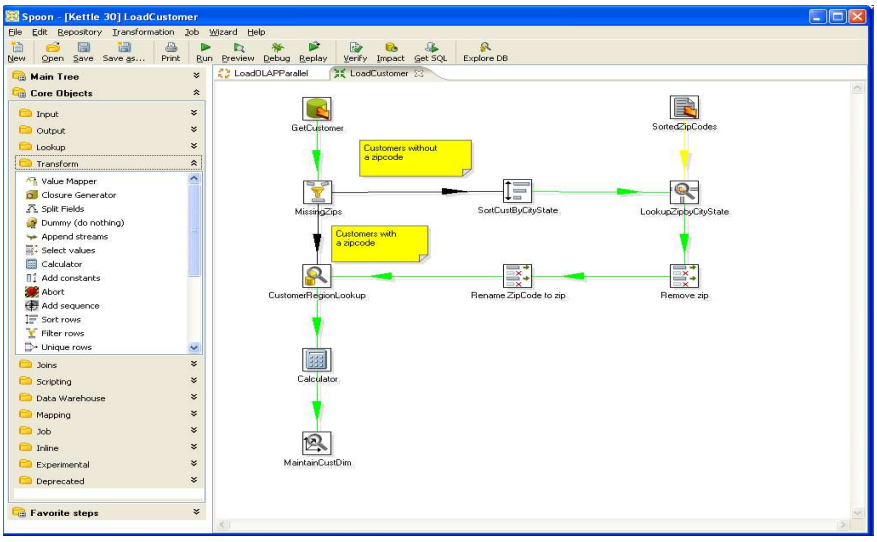
- Médiateur : accès direct aux sources
  - approche « paresseuse », pas de matérialisation
  - migration de requêtes vers les sources
  - *avantages* : données toujours fraîches, plus facile d'ajouter de nouvelles sources, plus grande échelle, distribution de l'effort
  - *inconvénients* : performances, traduction de requêtes, capacités différentes des sources

- Entrepôt de données : accès efficace à une copie des données
  - matérialisation des sources au niveau du modèle global
  - migration de données vers l 'entrepôt
  - *avantages* : performances, personnalisation des données (nettoyage, filtrage), versions
  - *inconvénients* : données pas toujours fraîches, cohérence, gestion des mises-à-jour, gestion de gros volumes de données

Entrepôts de données

- L'approche la plus populaire d'intégration de données
  - Gros avantages: performances, contrôle plus facile à réaliser sur l'hétérogénéité des données
- Utilisation pour les systèmes décisionnels OLAP
- Transformation de données pour alimenter l'entrepôt
  - Chargeurs = *systèmes ETL* (« Extract, Transform, Load »)
  - Outils graphiques pour définir *des flots de traitements/transformations*
  - Une fois le flot de traitement défini → appliqué au contenu des sources

Exemple d'interface graphique d'ETL

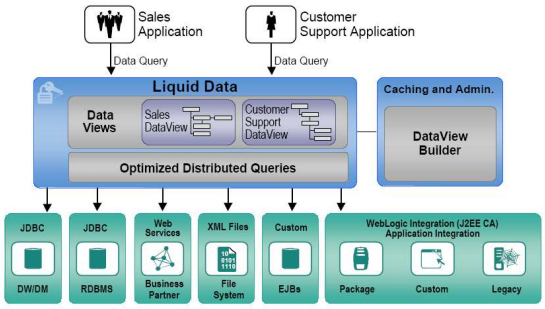


Médiateurs

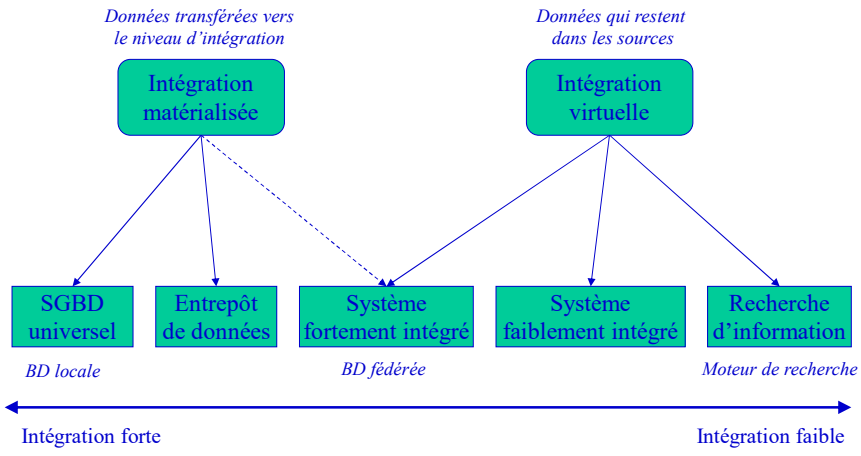
- Bien que moins utilisés en pratique, ils ont plus de potentiel
  - Meilleur passage à l'échelle
  - Acceptent mieux les changements dynamiques (nouvelles sources)

  → mieux adaptés à l'intégration de sources web

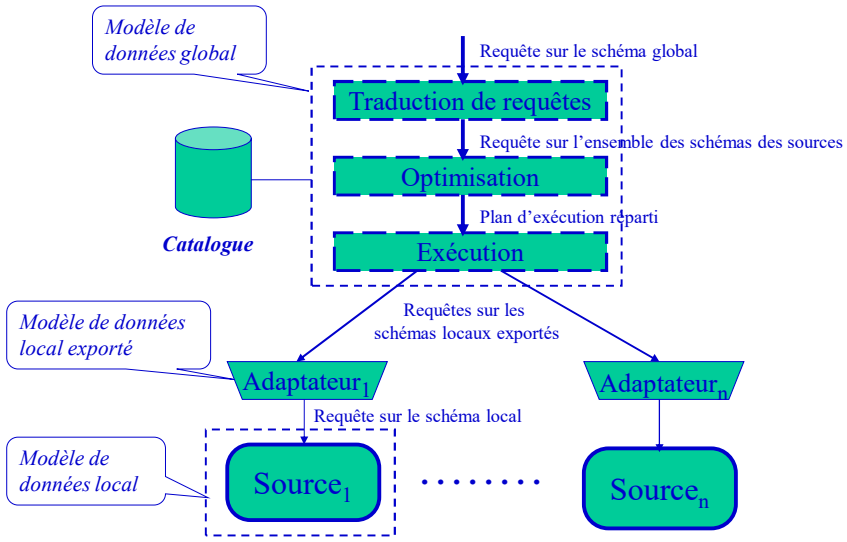
- En entreprise: EII
  « Enterprise Information Integration »
  - Ex. BEA Liquid Data (Oracle WebLogic), IBM Websphere Information Integrator



Degré d'intégration des données

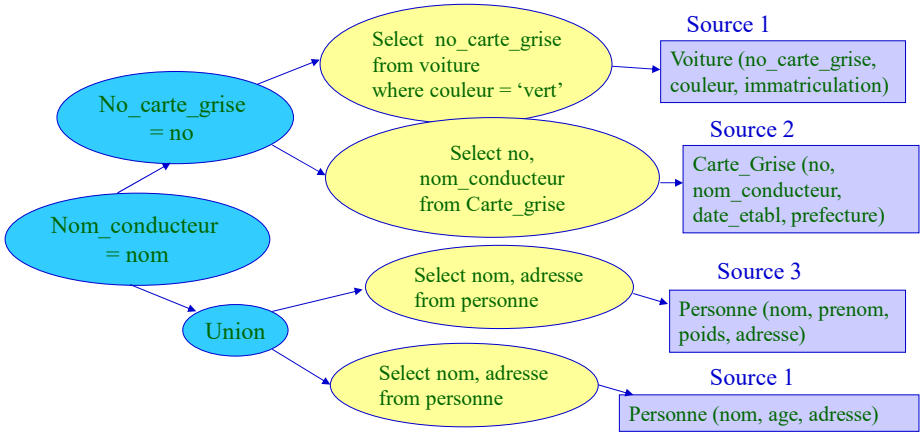


Architecture plus détaillée



Décomposition des requêtes

- Exemple : chercher l'adresse de tous les propriétaires de voitures vertes



Schémas d'intégration

- Problèmes
  - *Intégration de schéma*: comment définir un schéma global d'intégration à partir des schémas des sources?
  - *Fusion de données*: comment rendre compatibles, transformer les données en provenance des sources?
  - *Mappings/vue d'intégration*: comment décrire le lien entre le modèle global et les modèles locaux des sources?

Définition de la vue d'intégration

- Le lien entre schéma global et schémas locaux est défini à travers des vues
  → Mapping entre ces schémas
- Deux façons principales de définir ce lien
  - Le schéma global en fonction des schémas locaux → « ***global as view*** »
    - Approche *ascendante*: on part des sources pour produire le schéma global
  - Les schémas locaux en fonction du schéma global → « ***local as view*** »
    - Approche *descendante*: on fixe le schéma global et on décrit les sources par rapport à ce schéma fixé

« Global-as-View » (GAV)

- Le modèle global = vue sur les sources
  - élément global = f(éléments des sources)
    $\mathbf{M = V(S_1, \ldots, S_n)}$ (ou $\mathbf{M \supseteq V(S_1, \ldots, S_n)}$)
- Avantages
  - approche naturelle
  - la traduction de requêtes se fait facilement
- Inconvénients
  - nouvelle source → modification du modèle global
    - il faut considérer l'interaction de la nouvelle source avec les autres

Page 21

« Local-as-View » (LAV)

- Les sources = vues matérialisées du modèle global
  - une source décrit les données du modèle global qu'elle peut fournir
  - élément source = f(éléments modèle global)
    $\mathbf{S_i \subseteq V_i (M)}$
- Avantages
  - les sources sont décrites indépendamment les unes des autres
    - très simple de rajouter une nouvelle source
- Inconvénients
  - traduction de requêtes plus complexe

Page 22

Exemple GAV

- Modèles global et locaux

  $M = V(S_1, \ldots, S_n)$

  - Modèle global **M**
    - En général modèle spécifique, indépendant de celui des sources
    - Parfois, basé juste sur les modèles des sources, composés par la vue
    - Type de modèle: relationnel (le plus souvent), XML, JSON, RDF, etc.
  - Modèles des sources $\mathbf{S_i}$
    - Sources hétérogènes
    - Uniformisation du type de modèle à travers les adaptateurs
    - Type de modèle: en général le même pour les sources et le modèle global
  - Mapping / vue d'intégration **V**
    - Requête(s) déclarative(s) sous forme de clauses logiques du premier ordre
    - Correspondance à travers des variables

Exemple GAV: TSIMMIS

- TSIMMIS
  - Sources : informations sur les personnes d'une université
    - **Inf** : BDR avec des employés et des étudiants du département Informatique
      Employé(Nom, Prénom, Titre, Chef)
      Étudiant(Nom, Prénom, Année)
    - **Ann** : Annuaire pour l'université (nom, département, catégorie, e-mail, …)
  - Médiateur : les personnes du département Informatique
    - nom, catégorie, titre, chef, e-mail, année, ...
  - langage de spécification de médiateur MSL
    - règles (mapping) : PM :- $P_1, \ldots, P_k$, avec PM, P_i « patterns »

TSIMMIS : modèle

Adaptateur **Inf**

```
<employe>
   <nom>Dupont</nom>
   <prenom>Michel</prenom>
   <titre>professeur</titre>
   <chef>Jean Martin</chef>
</employe>
<etudiant>
   <nom>Hugo</nom>
   <prenom>Victor</prenom>
   <annee>2</annee>
</etudiant> …
```

Adaptateur **Ann**

```
<personne>
   <nom>Michel Dupont</nom>
   <dept>Informatique</dept>
   <categ>employé</categ>
   <email>md@univ.fr</email>
</personne>
<personne>
   <nom>Zoé Durand</nom>
   <dept>Informatique</dept>
   <categ>étudiant</categ>
   <annee>3</annee>
</personne> ...
```

Médiateur

```
<pers_inf>
   <nom>Michel Dupont</nom>
   <categorie>employé</categorie>
   <titre>professeur</titre>
   <chef>Jean Martin</chef>
   <email>md@univ.fr</email>
</pers_inf> ...
```

Spécification MSL du médiateur

```
<pers_inf>
   <nom>N</nom>
   <categorie>C</categorie>
   Reste1 Reste2
</pers_inf>  :-
   <personne>
      <nom>N</nom> <dept>Informatique</dept>
      <categ>C</categ> Reste1
   </personne>@Ann   AND
   <C>
      <nom>NF</nom><prenom>P</prenom> Reste2
   </C>@Inf  AND
   decomp(N, NF, P)
```


TSIMMIS : requêtes

- Exemple de requête
  - trouver toutes les informations sur Michel Dupont

    <pers_inf> <nom>Michel Dupont</nom></pers_inf>@Med

  - « dépliage » (unfolding) de la requête: substitution des éléments de la requête par leur mapping du médiateur

<pers_inf> <nom>Michel Dupont</nom> <categorie>C</categorie> Reste1 Reste2 </pers_inf> **:-**

<personne>
<nom>Michel Dupont</nom> <dept>Informatique</dept> <categ>C</categ> Reste1
</personne>**@Ann** AND

<C>
<nom>NF</nom><prenom>P</prenom> Reste2
</C>**@Inf** AND

decomp("Michel Dupont", NF, P)

  - chaque source répondra à la sous-requête qui la concerne

Exemple LAV

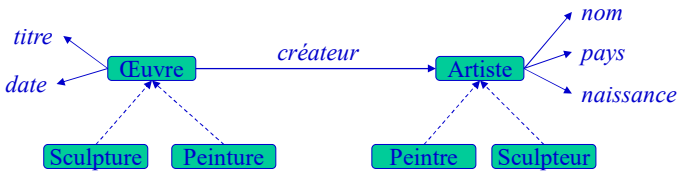
- Modèles global et locaux

  $S_i \subseteq V_i(M), i \in \{1, \ldots, n\}$

  - Modèle global **M**
    - Modèle spécifique, indépendant de celui des sources
  - Modèles des sources S_i
    - Type de modèle: le même que pour le modèle global
    - Chaque source est vue comme un seul élément structurel
  - Mapping / vue d'intégration **V**
    - Requête(s) déclarative(s) sous forme de clauses logiques du premier ordre
    - Correspondance à travers des variables

Exemple LAV: Information Manifold

- Information Manifold
  - modèle global : de type entité – association, exprimé par des relations
- Exemple de modèle global
  - Œuvre(titre, date, créateur), Artiste(nom, pays, naissance)
  - Sculpture, Peinture < Œuvre (sous-classes de Œuvre)
  - Peintre, Sculpteur < Artiste (sous-classes de Artiste)
    - Sculpture(titre, date, créateur), Peinture(titre, date, créateur),
    - Peintre(nom, pays, naissance), Sculpteur(nom, pays, naissance)



Information Manifold : sources

- Sources : vues sur le modèle global
  - définition = requête conjonctive + inégalités
- Exemple de description de sources / mapping LAV

  S_1 : noms/dates naissance des peintres nés après 1800 et les titres/dates de leurs peintures

  $S_1(t, d, n, dn) \subseteq$ Peintre(n, p, dn), Peinture(t, d, n), $dn \geq 1800$

  S_2 : titres/dates des œuvres réalisées avant 1940 et le nom/pays de leurs auteurs

  $S_2(t, d, n, p) \subseteq$ Œuvre(t, d, n), Artiste(n, p, dn), $d \leq 1940$

  S_3 : noms et dates de naissance des sculpteurs français

  $S_3(n, dn) \subseteq$ Sculpteur(n, 'France', dn)

Page 29

Information Manifold : requêtes

- Exemple de requête
  - Le titre et la date des œuvres après 1900, ainsi que le nom et la date de naissance de leurs créateurs

  Q(t, d, n, dn) : Œuvre(t, d, n), Artiste(n, p, dn), $d > 1900$
- Algorithmes de traitement des requêtes LAV
  - Difficultés
    - « Dépliage » des requêtes impossible à cause du sens inversé du mapping
    - Pas de jointure entres sources dans le mapping, il faut déduire les jointures
  - Trois algorithmes principaux
    - **Bucket**: le plus simple
      - Identification des sources pouvant répondre à chaque clause de la requête
      - Génération des combinaisons (jointures) de sources pour la requête
      - Elimination des combinaisons redondantes
    - **Minicon**: amélioration de Bucket pour éliminer les jointures invalides
    - **Inverse Rules**: basé sur l'inversion du sens des mappings → GAV

Page 30

Information Manifold : requêtes

Q(t, d, n, dn) : Œuvre(t, d, n), Artiste(n, p, dn), d > 1900

- Algorithme Bucket
  - Pour chaque clause de la requête → « bucket » (ensemble) de sources pouvant répondre à la clause (avec vérification des contraintes)
    - Deux types de variables dans les requêtes/vues
      - Distinguées: présentes dans les en-têtes des requêtes/vues
      - Existentielles: dans le corps, sans apporter d'information dans la requête/vue
    - S_i inclus dans le bucket de la clause C de Q si
      - Le mapping de S_i contient C
      - La clause C de Q est compatible avec la clause C du mapping (variables)
      - Les variables distinguées de la requête dans C doivent correspondre à des variables distinguées dans la vue
      - Si plusieurs apparitions de C dans le mapping de S_i, des instances distinctes de S_i sont ajoutées au bucket
  - Les buckets des clauses de Q:
    - Œuvre(t, d, n) : {S_1(t, d, n, dn'), S_2(t, d, n, p') }
    - Artiste(n, p', dn) : {S_1(t', d', n, dn), S_3(n, dn) }
      (S_2 ne fournit pas dn)

Page 31

Information Manifold : requêtes

Q(t, d, n, dn) : Œuvre(t, d, n), Artiste(n, p, dn), d > 1900

- Génération des combinaisons de sources des buckets
  Q(t, d, n, dn) : S_1(t, d, n, dn), d > 1900
  Q(t, d, n, dn) : S_1(t, d, n, dn'), S_3(n, dn), d > 1900
  Q(t, d, n, dn) : S_2(t, d, n, p'), S_3(n, dn), d > 1900
  Q(t, d, n, dn) : S_2(t, d, n, p'), S_1(t', d', n, dn), d > 1900

- Elimination des combinaisons redondantes
  - Hypothèses: sources complémentaires, correctes et sans valeurs NULL

  → la combinaison S_1-S_3 est plus restrictive que S_1 seule !

- Résultat

Q(t, d, n, dn) : S_1(t, d, n, dn), d > 1900 ∪
S_2(t, d, n, p'), S_3(n, dn), d > 1900 ∪
S_2(t, d, n, p'), S_1(t', d', n, dn), d > 1900

Page 32

Algorithme Minicon

- Amélioration de Bucket
  - A partir des clauses C de Q, on crée des MCD (Minicon descriptions)
  - Une MCD regroupe plusieurs clauses C auxquelles peut répondre une même source
  - Chaque MCD vérifie la compatibilité entre les variables de Q et celles des sources
  - Résultat: évite des combinaisons invalides que Bucket peut produire
  - La réécriture Q se fait à partir des MCD
- Exemple
  - Modèle global M constitué de 3 relations: U(a,b), R(c,d), T(e,f)
  - Deux sources:
    - $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
    - $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$
  - Requête: $Q(x) = U(y, z)\ R(x, z)\ T(z, y)\ R(s, x)$

Algorithme Minicon: comparaison avec Bucket

$Q(x) = U(y, z)\ R(x, z)\ T(z, y)\ R(s, x)$

- $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
- $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$

- Avec Bucket
  - Par exemple, U(y, z) peut être fourni par S_1 si v→y et w→z
    - $S_1(u, v)$ devient $S_1(v1, y)$, où v1 est une variable quelconque remplaçant u
  - Résultat pour toutes les clauses C de Q
    - U(y, z) : $\{S_1(v1, y), S_2(v2, y, v3)\}$
    - R(x, z) : $\{S_1(z, x), S_2(v4, v5, x)\}$
    - T(z, y) : $\{S_1(y, v6), S_2(y, v7, v8)\}$
    - R(s, x) : $\{S_1(x, s)\}$
      (pour S_2 on obtiendrait $S_2(v9, v10, s)$, à qui il manque la variable distinguée x)

  → 8 combinaisons dont la validité doit être vérifiée

Algorithme Minicon: construction MCD

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
- $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$

- Avec Minicon
  - Pour U(y, z) on commence par S_1
    - Avec v→y et w→z, on obtient S_1(v1, y), comme pour Bucket
    - w existentielle dans S_1 et w→z ⇒ on regarde les autres clauses de Q où apparait z, donc R(x, z) et T(z, y)
    - Pour R(x, z) on doit avoir v→x et u→z, incompatible avec v→y pour U(y, z)
    - Conclusion: S_1 ne peut pas former une MCD pour U(y, z)

Algorithme Minicon: construction MCD

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
- $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$

- Suite avec Minicon
  - Pour U(y, z) on continue avec S_2
    - Avec v→y et w→z, on obtient S_2(v2, y, v3), comme pour Bucket
    - Comme pour S_1, on regarde aussi R(x, z) et T(z, y)
    - Pour R(x, z) on a déjà w→z, auquel on doit rajouter t→x
    - Pour T(z, y) on a déjà w→z, auquel on rajoute u→y
    - Conclusion: les variables des trois clauses de Q et de S_2 sont compatibles et la variable distinguée x de Q est liée à la variable distinguée t de S_2

  $MCD_1 = (S_2(y, y, x), \{1, 2, 3\})$ (couvre les clauses 1, 2 et 3 de Q)
  - Pour la dernière clause R(s, x) on regarde S_1
    - Il faut v→s et u→x (u et x distinguées) ⇒ $MCD_2 = (S_1(x, s), \{4\})$

Algorithme Minicon: combinaison de MCD

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
- $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$

- Suite avec Minicon
  - Il n'y a pas d'autres MCD

  $MCD_1 = (S_2(y, y, x), \{1, 2, 3\})$

  $MCD_2 = (S_1(x, s), \{4\})$

- Résultat final:

$Q(x) = S_2(y, y, x)\ S_1(x, s)$

Page 37

Algorithme Inverse Rules

- Approche très différente de Bucket / Minicon
  - Idée: transformer les mappings LAV en mappings GAV et utiliser le dépliage GAV pour la réécriture des requêtes
  - Chaque clause à droite d'un mapping LAV produit une règle GAV
  - Le lien entre les règles GAV issues d'un même mapping LAV → identifiants (fonctions Skolem) pour les variables existentielles du mapping
- Exemple: le même que pour Minicon
  - Modèle global M constitué de 3 relations: U(a,b), R(c,d), T(e,f)
  - Deux sources:
    - $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
    - $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$
  - Requête: Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

Page 38

Algorithme Inverse Rules: mappings GAV

- Mappings LAV
  - $S_1(u, v) \subseteq T(w, u)\ U(v, w)\ R(v, u)$
  - $S_2(u, v, t) \subseteq T(w, u)\ U(v, w)\ R(t, w)$
- Mappings GAV (Inverse Rules)
  - $T(f1(u, v), u) \supseteq S_1(u, v)$
  - $U(v, f1(u, v)) \supseteq S_1(u, v)$
  - $R(v, u) \supseteq S_1(u, v)$
  - $T(f2(u, v, t), u) \supseteq S_2(u, v, t)$
  - $U(v, f2(u, v, t)) \supseteq S_2(u, v, t)$
  - $R(t, f2(u, v, t)) \supseteq S_2(u, v, t)$

Algorithme Inverse Rules: réécriture de requête

$Q(x) = U(y, z)\ R(x, z)\ T(z, y)\ R(s, x)$

- Inverse Rules
  - $T(f1(u, v), u) \supseteq S_1(u, v)$
  - $U(v, f1(u, v)) \supseteq S_1(u, v)$
  - $R(v, u) \supseteq S_1(u, v)$
  - $T(f2(u, v, t), u) \supseteq S_2(u, v, t)$
  - $U(v, f2(u, v, t)) \supseteq S_2(u, v, t)$
  - $R(t, f2(u, v, t)) \supseteq S_2(u, v, t)$
- Différence avec la réécriture GAV: il faut faire correspondre les identifiants fonction Skolem issus des inverse rules

Algorithme Inverse Rules: exemple

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- Inverse Rules
  1. T(f1(u, v), u) $\supseteq$ S_1(u, v)
  2. U(v, f1(u, v)) $\supseteq$ S_1(u, v)
  3. R(v, u) $\supseteq$ S_1(u, v)
  4. T(f2(u, v, t), u) $\supseteq$ S_2(u, v, t)
  5. U(v, f2(u, v, t)) $\supseteq$ S_2(u, v, t)
  6. R(t, f2(u, v, t)) $\supseteq$ S_2(u, v, t)
- Si pour U(y, z) on utilise la règle 2 : y→v, z→f1(u, v)
  - Q(x) = S_1(u, v) R(x, f1(u, v)) T(f1(u, v), v) R(s, x)
- Si pour R(x, f1(u, v)) on utilise la règle 3: R(x, f1(u, v)) → S_1(f1(u, v), x)
  - Q(x) = S_1(u, v) S_1(f1(u, v), x) T(f1(u, v), v) R(s, x) → impasse, pas de lien entre les deux instances de S_1
  - Si au lieu de la règle 3 on utilise la règle 6: impasse, car on ne peut pas faire correspondre f1 à f2 (fonctions Skolem différentes)

Page 41

Algorithme Inverse Rules: exemple

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- Inverse Rules
  1. T(f1(u, v), u) $\supseteq$ S_1(u, v)
  2. U(v, f1(u, v)) $\supseteq$ S_1(u, v)
  3. R(v, u) $\supseteq$ S_1(u, v)
  4. T(f2(u, v, t), u) $\supseteq$ S_2(u, v, t)
  5. U(v, f2(u, v, t)) $\supseteq$ S_2(u, v, t)
  6. R(t, f2(u, v, t)) $\supseteq$ S_2(u, v, t)
- Si pour U(y, z) on utilise la règle 5 : y→v, z→f2(u, v, t)
  - Q(x) = S_2(u, v, t) R(x, f2(u, v, t)) T(f2(u, v, t), v) R(s, x)
- Pour R(x, f2(u, v, t)) la règle 6: R(x, f2(u, v, t)) → S_2(u, v, x), avec t→x
  - Q(x) = S_2(u, v, x) S_2(u, v, x) T(f2(u, v, x), v) R(s, x)
    = S_2(u, v, x) T(f2(u, v, x), v) R(s, x)
  - L'utilisation de la règle 3 mène à une impasse: R(x, f2(u, v, t)) → S_1(f2(u, v, t), x)
    - Il ne faut pas avoir de fonction Skolem dans l'appel d'une source

Page 42

Algorithme Inverse Rules: exemple

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- Inverse Rules
  1. $T(f1(u, v), u) \supseteq S_1(u, v)$
  2. $U(v, f1(u, v)) \supseteq S_1(u, v)$
  3. $R(v, u) \supseteq S_1(u, v)$
  4. $T(f2(u, v, t), u) \supseteq S_2(u, v, t)$
  5. $U(v, f2(u, v, t)) \supseteq S_2(u, v, t)$
  6. $R(t, f2(u, v, t)) \supseteq S_2(u, v, t)$
- Pour T(f2(u, v, x), v) la règle 4: T(f2(u, v, x), v) → $S_2(v, v, x)$, avec u→v
  - Q(x) = $\underline{S_2(u, v, x)}$ $\underline{S_2(v, v, x)}$ R(s, x)
  - On peut éliminer $S_2(u, v, x)$, qui est moins restrictif que $S_2(v, v, x)$
  - Q(x) = $\underline{S_2(v, v, x)}$ R(s, x)
  - L'utilisation de la règle 1 mène à une impasse: f2 et f1 ne peuvent pas correspondre

Page 43

Algorithme Inverse Rules: exemple

Q(x) = U(y, z) R(x, z) T(z, y) R(s, x)

- Inverse Rules
  1. $T(f1(u, v), u) \supseteq S_1(u, v)$
  2. $U(v, f1(u, v)) \supseteq S_1(u, v)$
  3. $R(v, u) \supseteq S_1(u, v)$
  4. $T(f2(u, v, t), u) \supseteq S_2(u, v, t)$
  5. $U(v, f2(u, v, t)) \supseteq S_2(u, v, t)$
  6. $R(t, f2(u, v, t)) \supseteq S_2(u, v, t)$
- Pour R(s,x), seule la règle 3 n'introduit pas de fonction Skolem
  - R(s, x) → $S_1(x, s)$
- Résultat final

  $Q(x) = S_2(v, v, x)\ S_1(x, s)$

Page 44

Comparatif des algorithmes LAV

- Bucket
  - Le plus simple, mais génère de nombreuses combinaisons, dont il faut vérifier la validité
- Minicon
  - Plus complexe que Bucket, même si la création des MCD s'appuie sur des propriétés simplificatrices
  - Evite de nombreuses combinaisons inutiles
  - Le plus efficace des trois algorithmes
- Inverse Rules
  - Adopte la démarche GAV, mais la réécriture est complexe à cause des fonctions Skolem
  - Evite également les combinaisons inutiles, mais avec des vérifications plus complexes que Minicon

Sources

- ***Web Data Management***, S. Abiteboul, I. Manolescu, P. Rigaux, M.-C. Rousset, P. Senellart, *Cambridge University Press*, 2011, http://webdam.inria.fr/Jorge

- ***A Scalable Algorithm for Answering Queries Using Views***, R. Pottinger, A. Levy, *VLDB Conference*, 2000 https://www.vldb.org/conf/2000/P484.pdf