Architectures d'intégration de données

Dan VODISLAV

Université de Cergy-Pontoise

Master Informatique M1

Cours IED

Plan

- Intégration de données
  - Objectifs, principes, caractéristiques
- Architectures type d'intégration de données
  - Médiateur
  - Entrepôt de données
- Schémas d'intégration
  - Intégration de schéma
  - Mapping
- Architectures distribuées
  - Architectures k-tiers
  - Services web

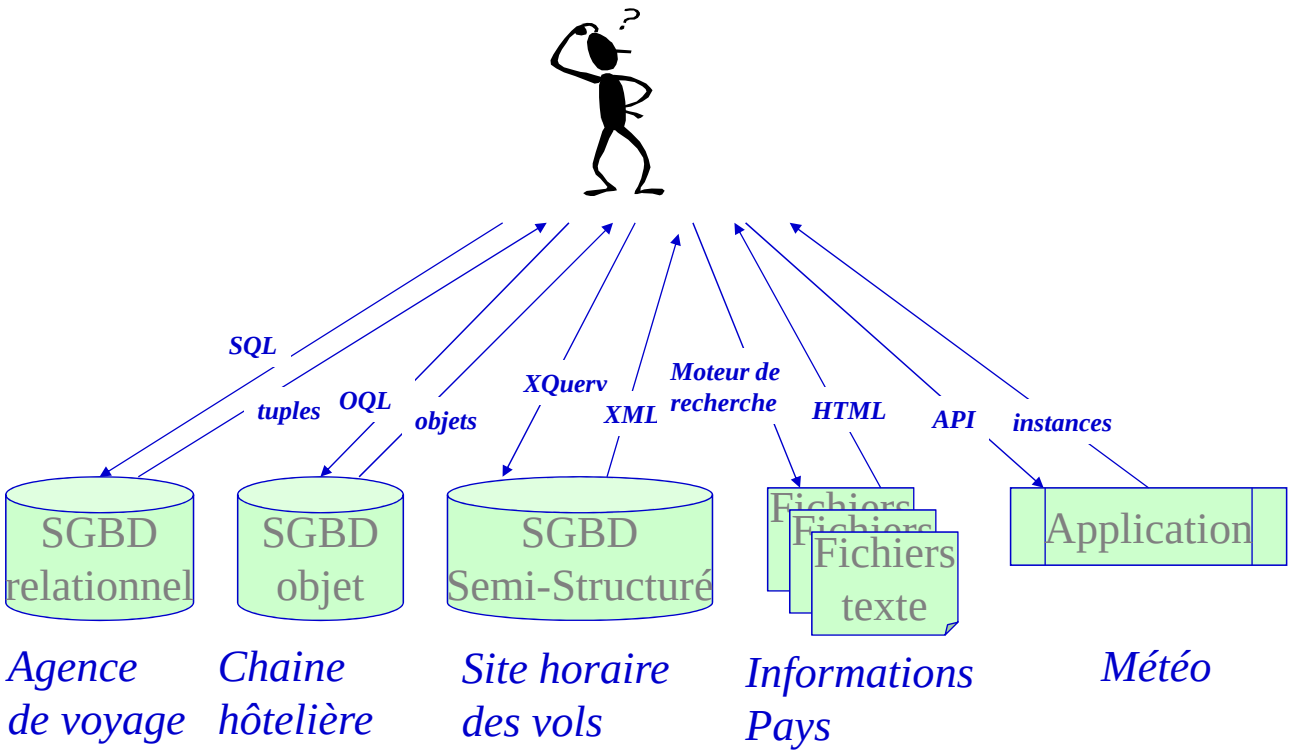
Intégration de données

- Gestion de données traditionnelle
  - Bases de données homogènes (modèle/schéma uniques)
  - Architecture centralisée ou distribuée, transparente au niveau logique
- Évolution vers le "mélange" de données en provenance de plusieurs sources
  - Sources d'information nombreuses et variées
    - SGBD relationnels/XML, pages Web HTML, LDAP, tableurs, fichiers, applications, formulaires, services web, …
  - 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 ces données comme si elles constituaient une seule base de données homogène

Objectif

- Plus particulièrement, ***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)

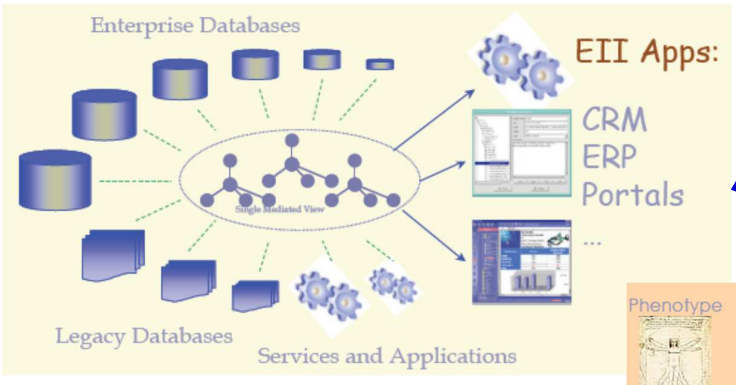
Exemple



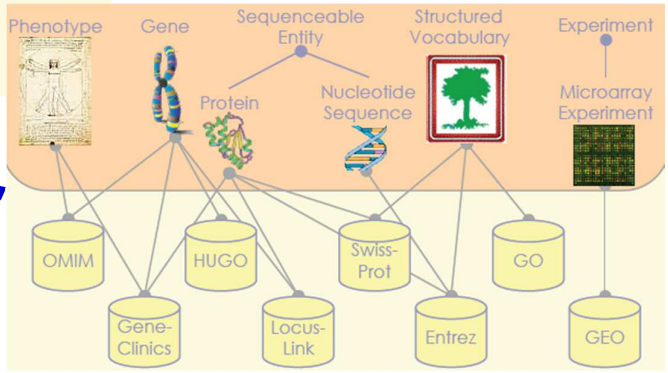
Enjeux

- 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:
    - IBM: « pour 1$ dépensé pour une application, 5-9$ sont dépensés pour assurer son intégration »
    - Gartner: « plus de 40% des budgets IT sont dépensés en intégration »
    - Morgan Stanley: « l'intégration de données est devenue la priorité n°1 des entreprises avant le e-business et le CRM »
- 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



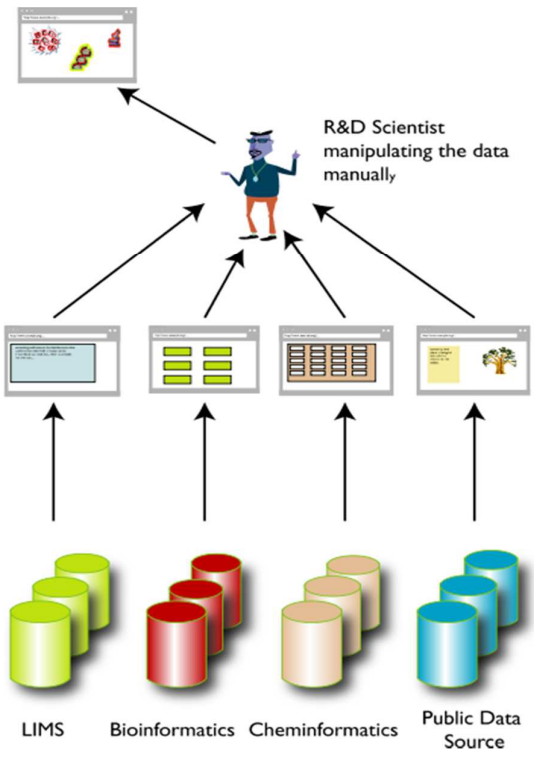
Données d'entreprise

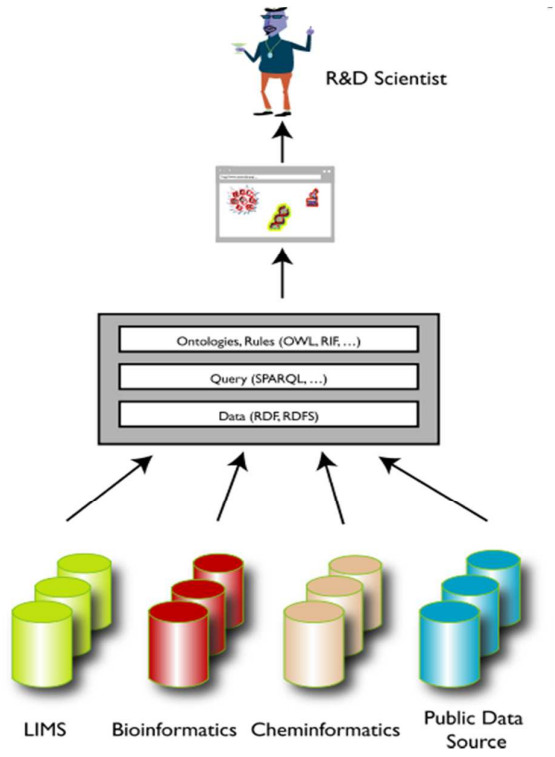


Données scientifiques

+ **le Web !** → centaines de milliers de sources de données

La différence





Caractéristiques des sources de données

- … qui rendent l'intégration de données difficile
  - → *Distribution*
  - → *Autonomie*
  - → *Hétérogénéité*

Distribution

- Données stockées sur des supports répartis sur plusieurs sites
  - Caractéristique importante: *l'échelle*
- Avantages
  - Disponibilité: ne tombent pas en panne en même temps
  - Temps d'accès: partage de la charge, parallélisme
- Problèmes
  - Les temps de communication
  - Localisation des sources contenant les données pertinentes
  - Hétérogénéité en termes de puissance de traitement et de charge
  - Les sources peuvent être temporairement indisponibles

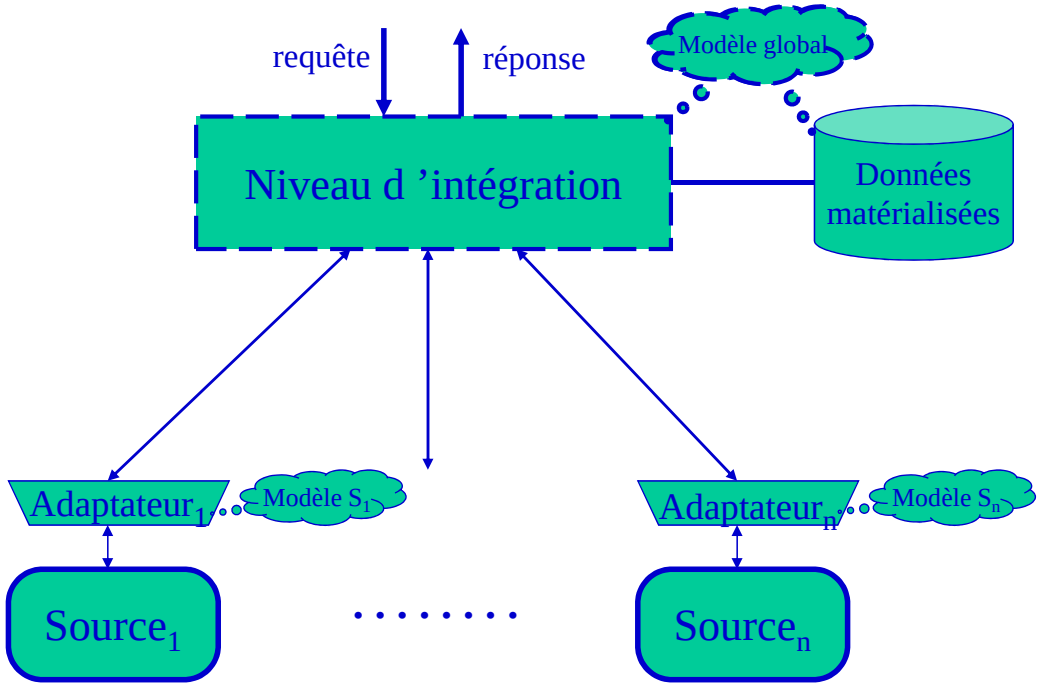
Autonomie

- Conception : les sources décident de leur propre
  - modèle de données,
  - langage d'interrogation,
  - sémantique des données.
- Communication : les sources décident quand et comment répondre aux questions d'autres sources
- Exécution : les sources décident de l'ordre d'exécution des transactions locales ou des opérations externes
  - Peu ou pas d'informations fournies sur les détails internes d'exécution
- Association des sources :
  - connexion et déconnexion des sources
  - partage de données et des fonctions

Hétérogénéité

- Concerne les données, les modèles, les langages, …
- Système homogène :
  - même logiciel gérant les données sur tous les sites
  - même modèle de données / langage d'accès
  - même univers de discours / sémantique
- Système hétérogène : qui n'est pas homogène sur au moins un critère
  - Divers niveaux et degrés d'hétérogénéité
- Aussi: hétérogénéité de la *puissance* / des *capacités de traitement* des sites

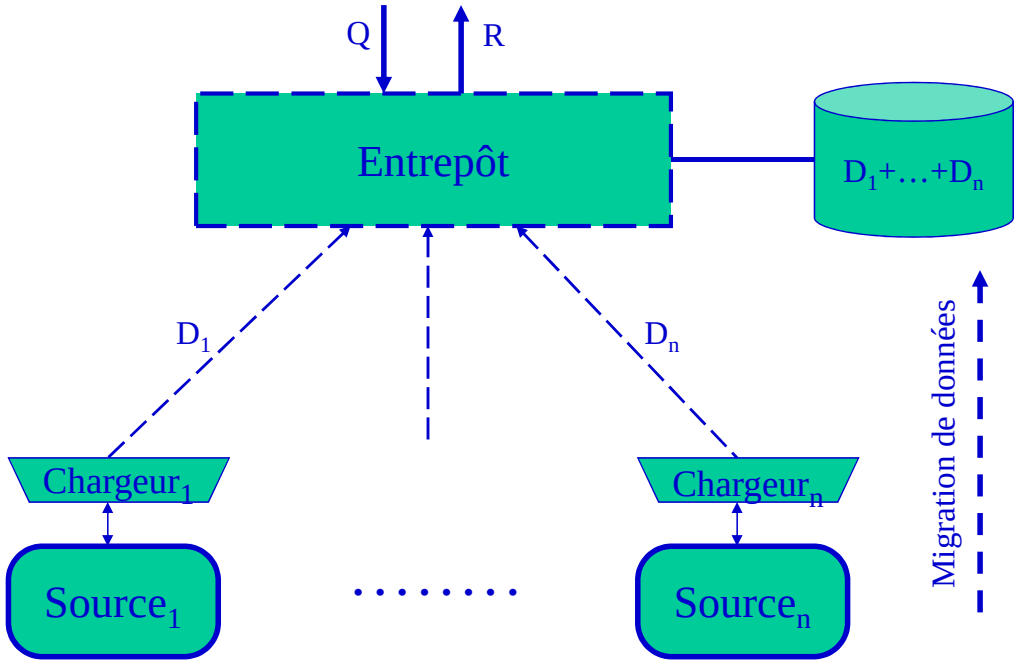
Architecture générale d'intégration



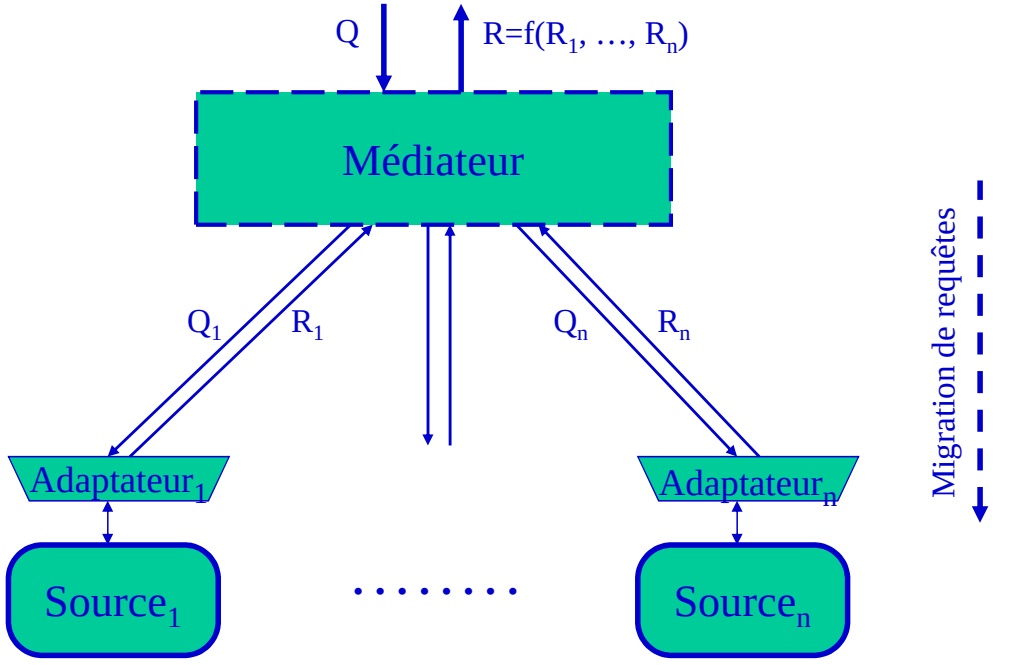
Deux approches

- 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

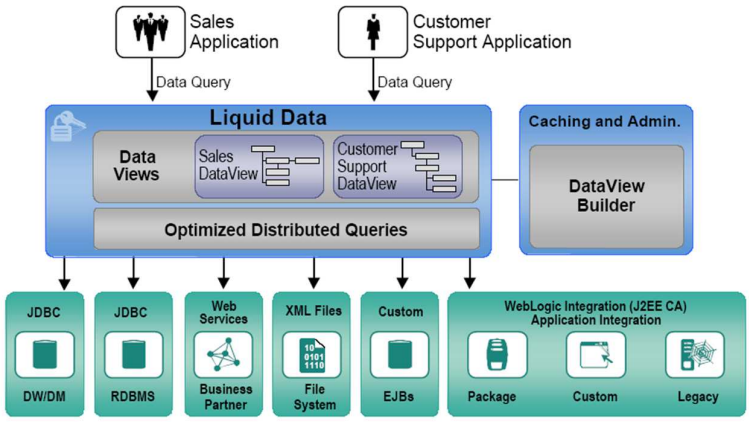
- L'approche la plus populaire d'intégration de données
  - Gros avantage: performances
  - Autre gros avantage: contrôle plus facile de l'hétérogénéité des données
- Utilisation pour les systèmes décisionnels OLAP
- Transformation de données pour alimenter l'entrepôt
  - Chargeurs = *systèmes ETL* («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

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 »
  - BEA Liquid Data (AquaLogic Data), IBM InfoSphere Federation Server, …



Schémas d'intégration

- Problèmes
  - *Intégration de schéma*: comment définir un schéma (modèle) global d'intégration à partir des schémas (modèles) 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 schéma global et les schémas des sources?

- Modèle de données global
  - *Relationnel*: mieux maîtrisé
  - *XML*: plus riche et flexible
  - *Sémantique* (ontologie): intégration sémantique
  - *Mixte*

Intégration de schéma

- Le lien entre schéma global et schémas locaux est défini à travers un *mapping* (correspondance structurelle)
  - Schéma global **M**
  - Schéma des sources $\mathbf{S_i}$
- 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

      $\mathbf{M = V(S_1, \ldots, S_n)}$
    - Avantages: approche naturelle, traduction des requêtes simple
  - 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

      $\mathbf{S_i \subseteq V_i(M)}$
    - Avantage: facile à rajouter de nouvelles sources

Mapping

- Mapping = correspondance entre le schéma global et les schémas des sources
  - utilisé pour la traduction des requêtes et la structuration des résultats
- Diversité
  - les schémas : relationnel, XML, orienté-objet, entité-association
  - le mapping : couples d'éléments correspondants, fonctions, contraintes, degrés de similarité
- Objectifs contradictoires
  - mapping complexe : précision, pouvoir d'expression
  - mapping simple : découverte automatique, composition, maintenance simplifiée
- Intégration : sources nombreuses, hétérogènes, ajouts de sources
  - besoin de calcul (semi-)automatique du mapping
  - un mapping simple, structuré est plus facile à maintenir

  $\Rightarrow$ *solutions de compromis, privilégiant le mapping simple*

Exemples de mapping

S_1 : Client — Numéro
— Société
— Nom
— Prénom

S_2 : Acheteur — ID
— Compagnie
— Contact
— Téléphone

M_a : Client → Acheteur
Client.Numéro → Acheteur.ID
Client.Société → Acheteur.Compagnie
Client.Nom → Acheteur.Contact
Client.Prénom → Acheteur.Contact

M_b : Client → Acheteur
Client.Numéro → Acheteur.ID
Client.Société → Acheteur.Compagnie
concat(Client.Nom, Client.Prénom) →
Acheteur.Contact

M_c : create view Client
select ID as Numéro,
Compagnie as Société,
getNom(Contact) as Nom,
getPrenom(Contact) as Prénom
from Acheteur

M_d: create view Acheteur
select Numéro as ID,
Société as Compagnie,
concat(Nom, Prénom) as Contact
from Client

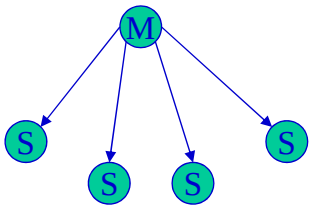
La difficulté de définir le mapping

- En pratique la création du mapping prend > 50% de l'effort d'intégration!
  - Il est important d'avoir des outils de génération (semi-)*automatique*
  - … mais la *précision* du mapping est très importante pour l'intégration
- Les difficultés de la génération automatique du mapping
  - Hétérogénéité des schémas et des modèles
  - Les schémas ne captent jamais complètement la sémantique → il faut la chercher partout, dans les données, les commentaires
  - Il faut combiner plusieurs critères, proposer plusieurs variantes avec des degrés de confiance différents
- Technique générale: deux étapes
  - Mise en correspondance d'éléments individuels du schéma («matching»)
  - Utilisation des correspondances individuelles pour définir le mapping
    - Union, jointure, composition, filtrage, agrégation, …

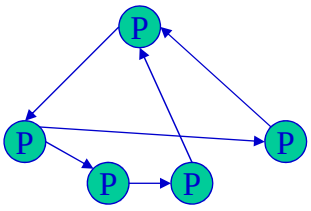
Architectures distribuées

- Intégration de données → architectures distribuées
  - Les sources = serveurs de données, le médiateur = client
  - Médiateur = serveur de données, l'application = client
- Médiateur = architecture distribuée très simple
  - Un client, plusieurs serveurs
  - Les sources: limitées en général à l'interrogation de données
- En principe, on pourrait avoir:
  - Des rôles mélangés client/serveur pour les sites → *architectures distribuées pair à pair*
  - Des sites qui offrent d'autres services que l'interrogation des données et qui collaborent → *applications réparties de gestion de données* basés sur ces services

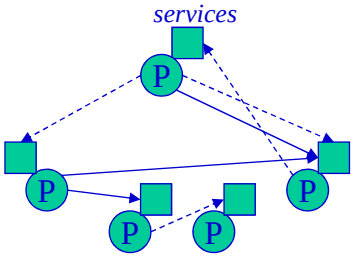
Médiateur, pair à pair, réparti



Médiateur



Pair à pair



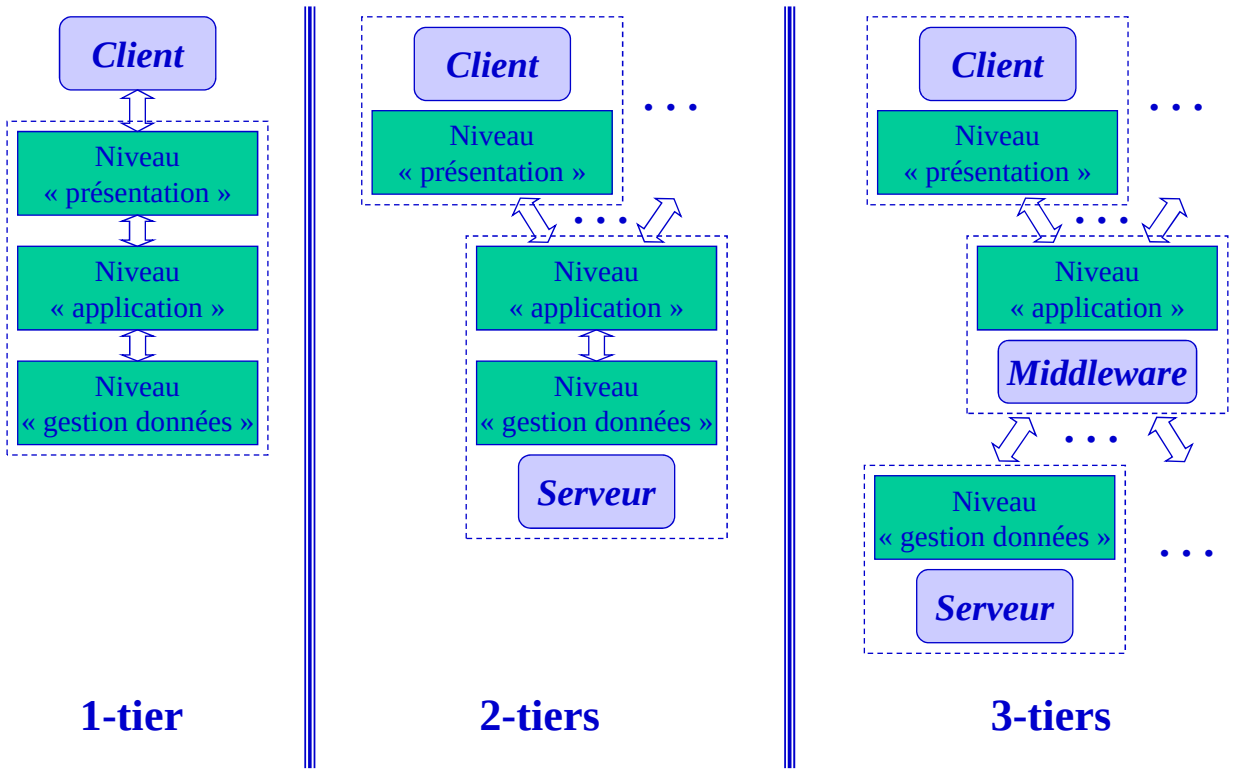
Réparti général

Applications réparties

- Applications réparties
  - Accès à plusieurs ressources / applications individuelles
  - Séparation entre « clients » et « serveurs »
- Architectures *k*-tiers
  - *1-tier* : centralisé
  - *2-tiers*: un serveur, plusieurs clients (client - serveur)
  - *3-tiers*: plusieurs serveurs, plusieurs clients (avec middleware)
  - *n-tiers*: spécifique à la diffusion sur le web

    Ex: serveurs web avec architecture 3-tiers + clients web
    - Clients n-tiers → serveurs (n+1) - tiers

Architectures



Communication

- Application répartie → communication entre les « tiers » qui réalisent des traitements
- Moyens de communication traditionnels
  - Middleware
    - RPC (« Remote Procedure Call »): appel de fonctions à distance
    - Moniteurs transactionnels (« TP monitors »): bases de données
    - « Object brokers »: RPC en orienté-objet (ex. CORBA, DCOM)
    - Moniteurs d'objets (« object monitors »): « object broker » + « TP monitor »
    - Middleware orienté-messages: asynchronisme, files d'attente
  - EAI (« Enterprise Application Integration »)
    - Communication entre systèmes plus hétérogènes (ex. entre systèmes 3-tiers)

    Ex: WebSphere MQ, BEA WebLogic Integration, webMethods, etc.
    - Visent souvent aussi des aspects « workflow » (séquence de traitements)

Services web

- Sur le web: contraintes qui n'apparaissent pas dans les environnements d'entreprise
  - Contrôle limité des sites
  - Débit faible
  - Clients légers
  - Présentation/Interaction moins riches (HTML)
- Objectif:
  - Réaliser des applications distribuées (architectures k-tiers) avec les contraintes imposées par le web

→ ***services web***

Caractéristiques des services web

- Demande de service adressée par un client à un serveur
  - Appel d'une fonction distante
  - Utilise les protocoles web: TCP/IP, HTTP
- Données transportées sur le web
  - Généralement du texte (HTTP: pages HTML)
  - Services web → texte en format XML
    - Format d'échange flexible
    - Standardisation
- Services web: évolution des architectures
  - Architectures distribuées classiques → web
    - RPC, RMI, CORBA, DCOM → HTTP, XML, services
  - Web « homme-machine » → web « machine-machine »

Web « machine-machine »

- Web dynamique « homme-machine »
  - HTML + HTTP + scripts
    - Scripts: tâches exécutées par un serveur web
    - HTML: contenu généré dynamiquement par les scripts
    - HTTP: utilisation « manuelle » à travers un navigateur web
  - Interface informelle
    - Paramètres de type texte
    - Résultat: HTML
- Web « machine-machine »
  - XML + SOAP + code
    - Code: programme/fonction appelé à distance
    - XML: format d'échange général
    - SOAP: utilisation par des programmes (automatique)
  - Interface formalisée (WSDL)
    - Paramètres et résultat typés XML Schema

Avantages des services web

- Flexibilité
  - Indépendance du langage et du système
  - Données XML
- Interopérabilité dans des environnements distribués
  - Interfaces formalisées
  - Communication entre services, composition
  - Automatisation
- Adaptés à la communication sur le web
  - Protocoles web bien connus et acceptés (HTTP, SMTP, …)
  - Invocation à travers des pare-feux (à la différence de CORBA)

REST vs. SOAP

- SOAP (Simple Object Access Protocol)
  - Principal standard W3C pour les services web
  - Associé à WSDL pour la description du service
  - Protocole de communication → échange de messages XML
  - Services appelables à travers des points d'accès sur le web
- REST (Representational State Transfer)
  - Appel de services web directement en HTTP
  - Messages HTTP : POST, GET, PUT, DELETE
    - Utilisation codes d'erreur, options d'appel HTTP, caching
  - Description WADL (peu utilisée)
  - Tout objet (ressource) manipulé par le service doit avoir une URI
  - Services sans état: exécution indépendante des appels précédents