
Architectures web pour la gestion de données

Dan VODISLAV

Université de Cergy-Pontoise

Plan

- Le Web
- Intégration de données
- Architectures distribuées

Le Web

- **Internet** = réseau *physique* d'ordinateurs
- **Web** (WWW, World Wide Web) = réseau *logique* de "hyper-documents" (documents avec des liens)
 - Documents statiques (fixes) ou dynamiques (produits par un programme)
 - Web publique et webs privés
 - Chaque document (page web) identifié par une URL

- **URL** (Uniform Resource Locator)

`https://www.exemple.org:443/chemin/vers/page?p1=v1&p2=v2#part`

- Protocole: http, https, ftp, ...
- Nom serveur: ici `www.exemple.org`
- Port: par défaut 80 pour http, 443 pour https, ...
- Chemin et nom du document: ici `chemin/vers/page`
- Paramètres requête (documents dynamiques): ici `?p1=v1&p2=v2`
- Fragment: partie du document, ici `#part`

Le web classique

- Données: *documents texte* avec mise en page et *liens*
 - Format HTML (le plus souvent HTML 4.01, recommandation du W3C)
 - Conçu pour la présentation de texte/images/liens dans un navigateur web
 - XHTML 1.0: XML-isation de HTML 4.01
 - Programmes: scripts pour générer du contenu dynamique
 - Exécutés par le serveur web
 - Communication: protocole HTTP
 - HTTP: protocole client-serveur pour le web, basé sur TCP/IP
 - Format texte, utilisation "manuelle" à travers un navigateur web
 - HTTPS: version sécurisée de HTTP (cryptage, authentification, ...)
 - Outils: navigateur web
 - Interface d'utilisation informelle: paramètres de type texte, résultat HTML
- ***Web homme - machine***

Les limitations du web classique

- Le web = la plus grande *base de données* jamais vue
 - Milliards de documents fournis par des millions d'acteurs
 - Répartition sur des millions de machines
 - Principales difficultés
 - Hétérogénéité des données/systèmes, faible structuration, sémantique cachée
 - Très large échelle
 - Automatiser l'accès à cette base de données
- Vers un ***web machine-machine***
- Données: XML
 - Format d'échange général, flexible
 - Programmes: services web, fournis par chaque machine
 - Appel automatisable de programmes à distance
 - Web ***dynamique***: chaque site exécute des programmes qui interagissent
 - Communication: SOAP, pour l'appel à distance de programmes (services web)
 - Interface formalisée: WSDL, UDDI, OWL
 - Paramètres/résultat typés XML Schema, annuaires de services, annotations sémantiques

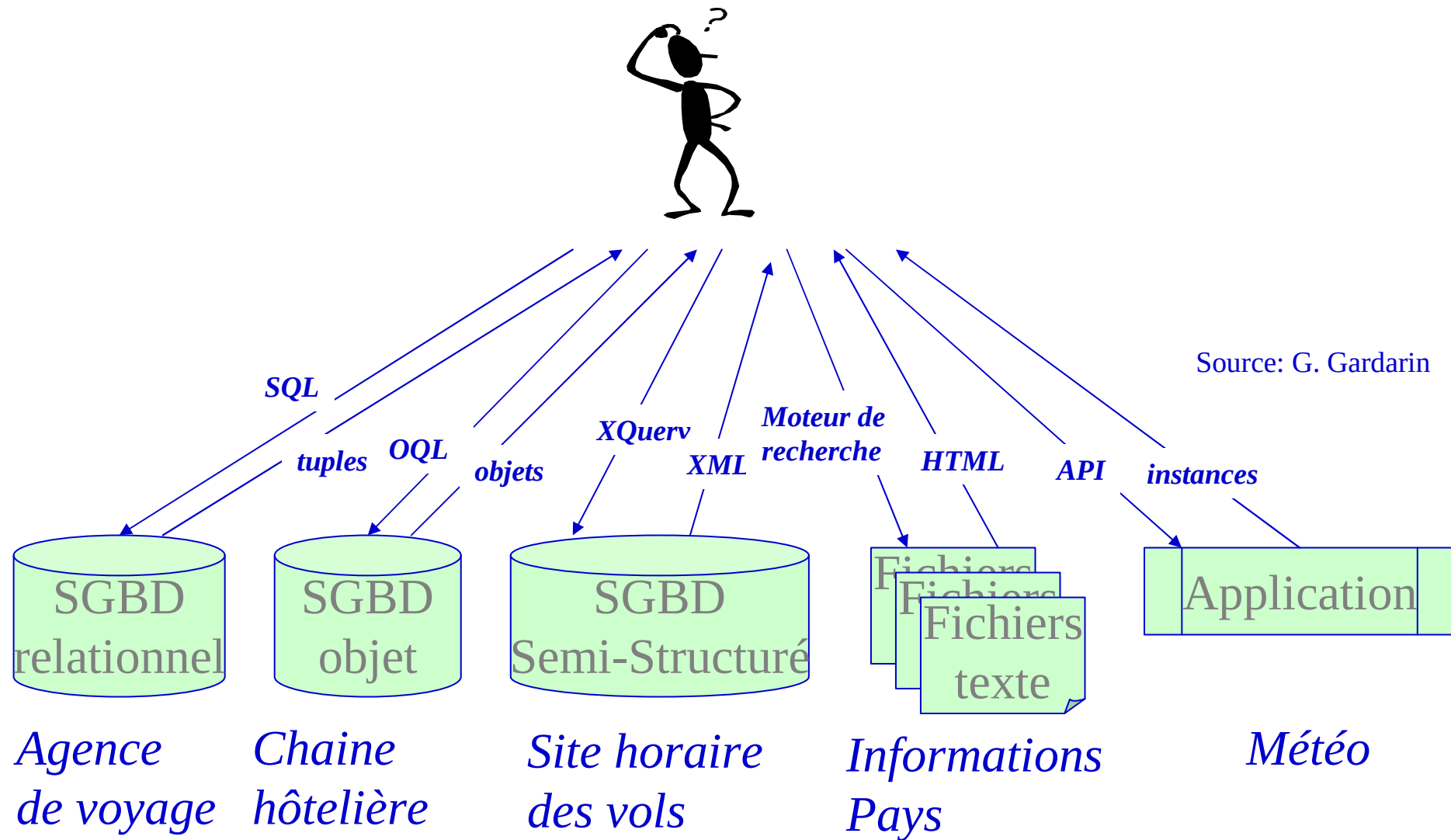
Gestion de données sur le web

- Exploitation (intégration) des données
 - Cacher l'hétérogénéité, la répartition
 - Exploiter les données comme si elles constituaient une base de données homogène
 - Accéder de façon transparente à des données sur plusieurs sites
 - Élément unificateur: *le format XML*
- Collaboration entre sites pour réaliser des applications
 - Répartition des données et des services → passage à l'échelle
 - Architectures de répartition et collaboration
 - Élément unificateur: *les services web*

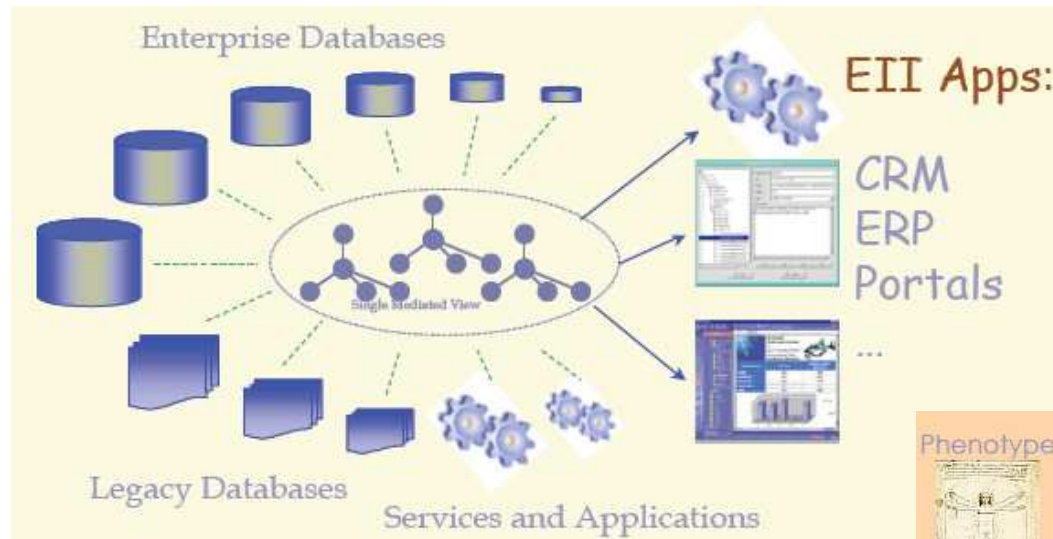
Intégration de données

- Exploitation 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
- 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)

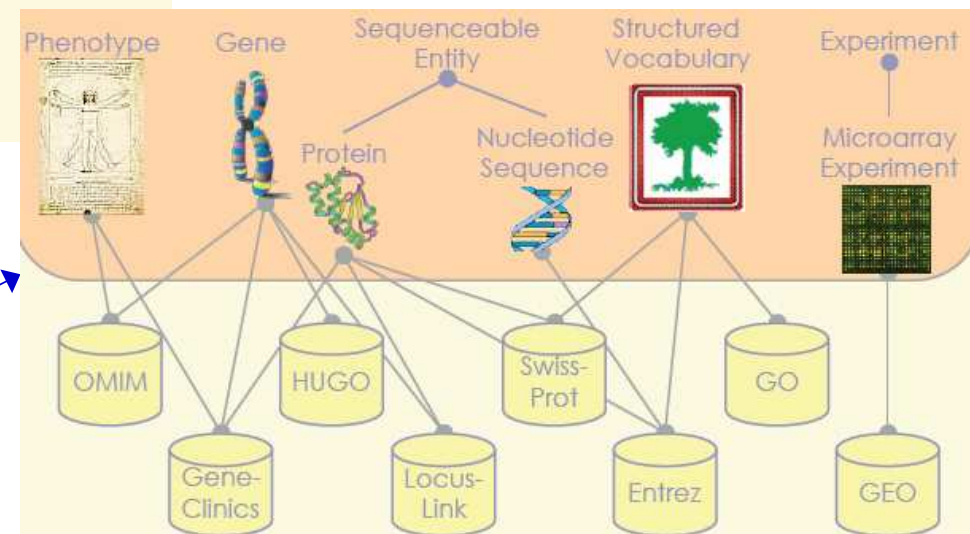
Exemple



Applications



Données scientifiques

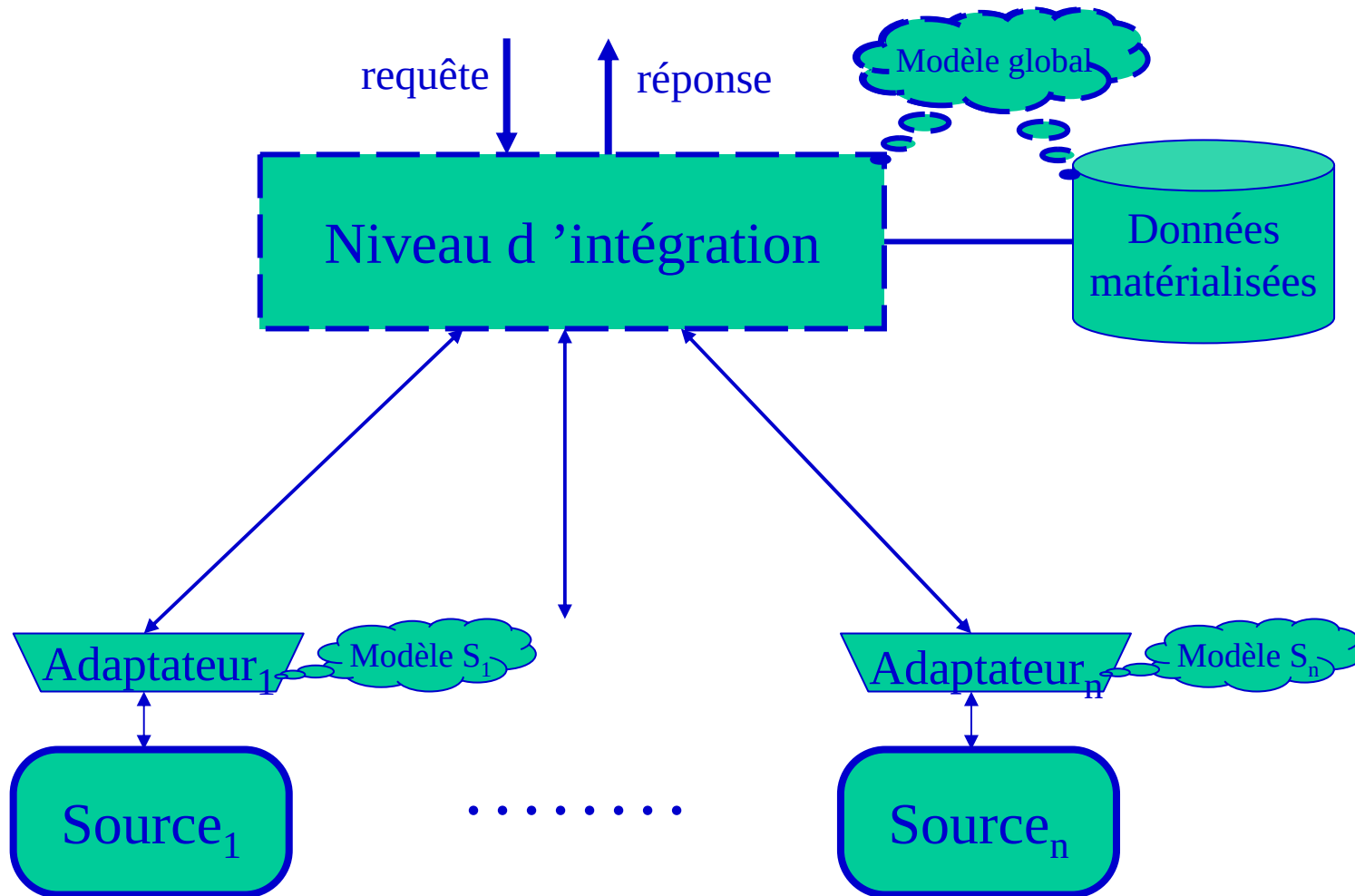


+ **le Web !** → centaines de milliers de sources de données

Caractéristiques des sources de données

- *Distribution*
 - Données réparties sur plusieurs sites, facteur important: *l'échelle*
 - Avantages: disponibilité, partage de charge
 - Inconvénients: temps de communication, localisation des sources
- *Autonomie*
 - Les sources décident de la façon d'accéder à leurs données, de leur disponibilité, etc.
- *Hétérogénéité*
 - De format et structure des données, de langage d'accès, de sémantique
 - De capacité de traitement

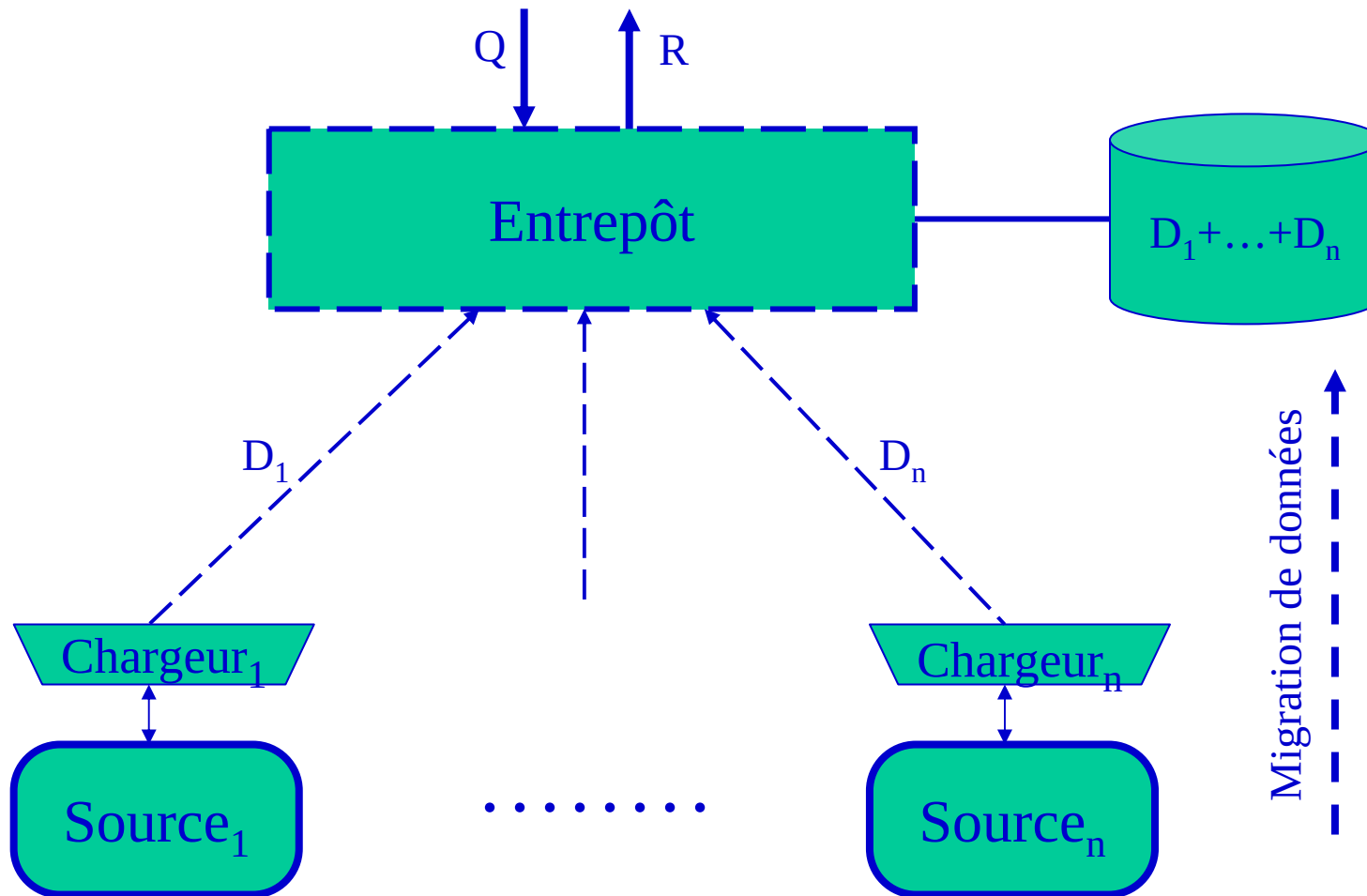
Architecture générale d'intégration



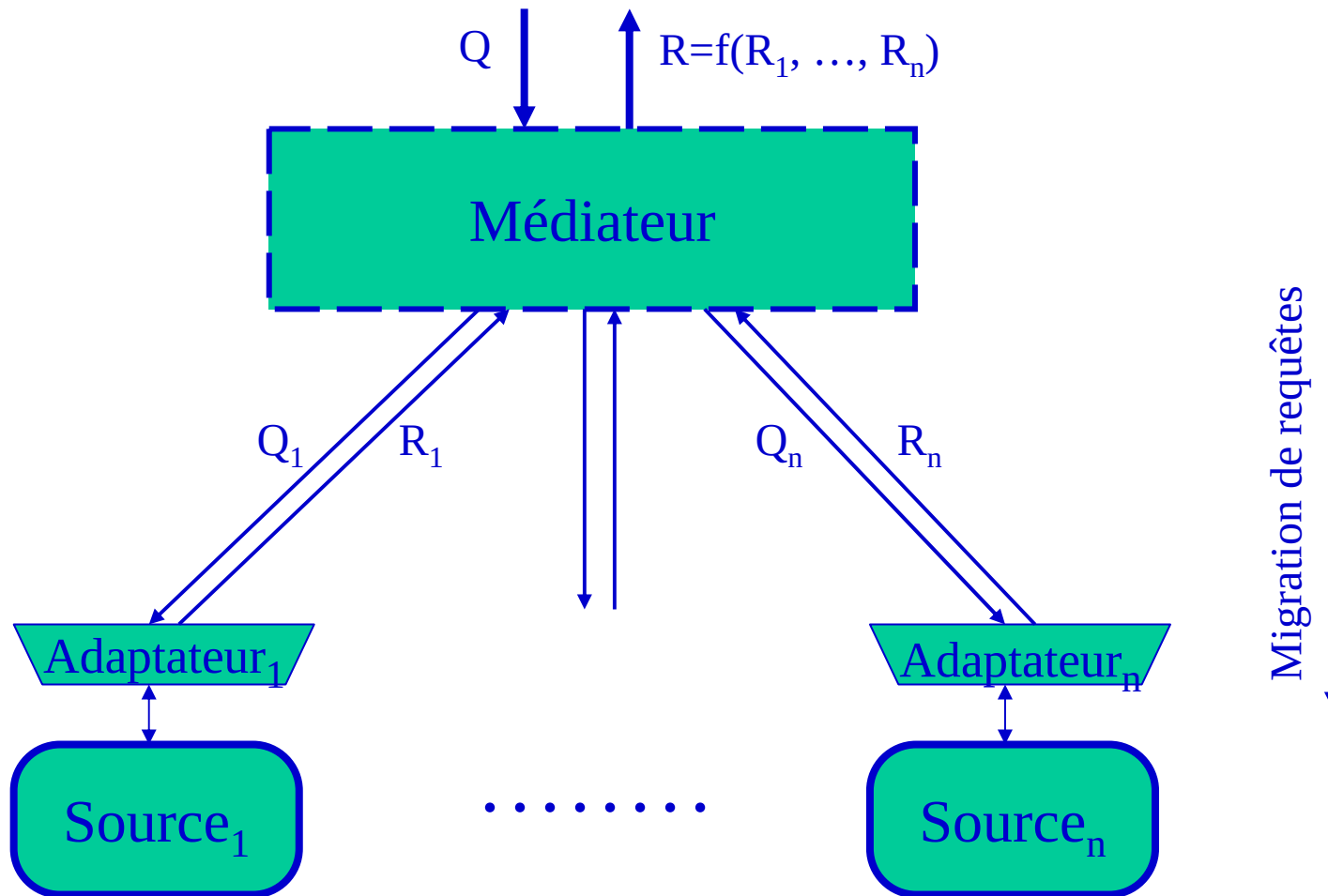
Deux approches

- 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

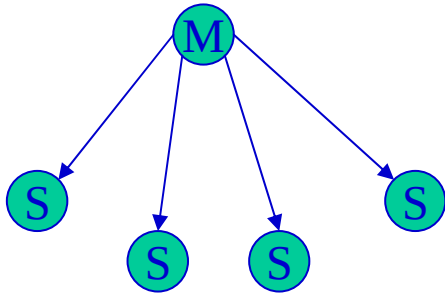
En pratique

- Entrepôt: 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 »)
- Médiateurs: moins utilisés, mais 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, IBM Websphere, Information Integrator
- XML: format idéal d'échange
 - Pour les adaptateurs des sources, pour le niveau d'intégration

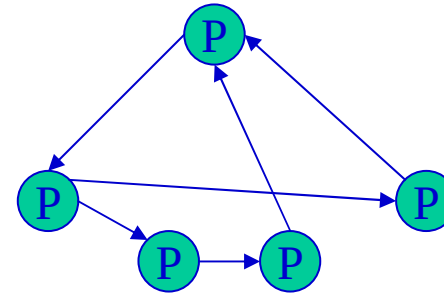
Architectures distribuées

- 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: seules des fonctionnalités d'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 sur les données que l'interrogation et qui collaborent → applications réparties de gestion de données basés sur ces services

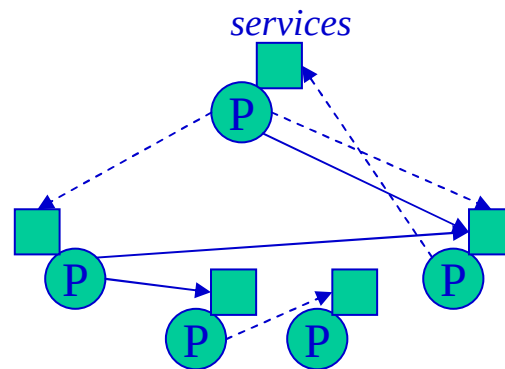
Médiateur, pair à pair, réparti



Médiateur



Pair à pair

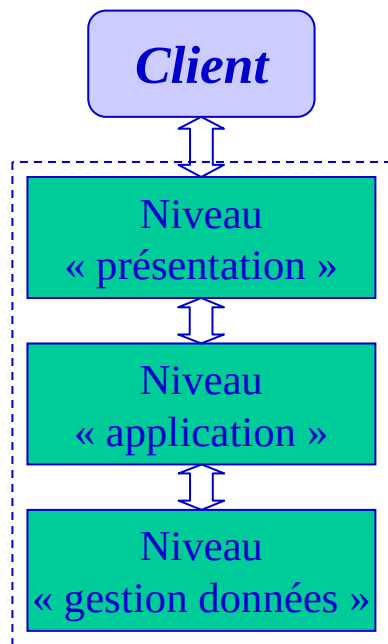


Réparti général

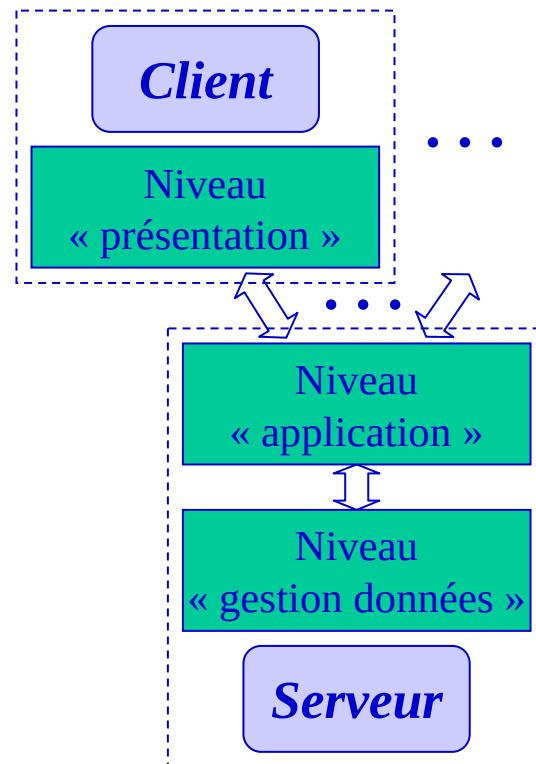
Applications réparties

- 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

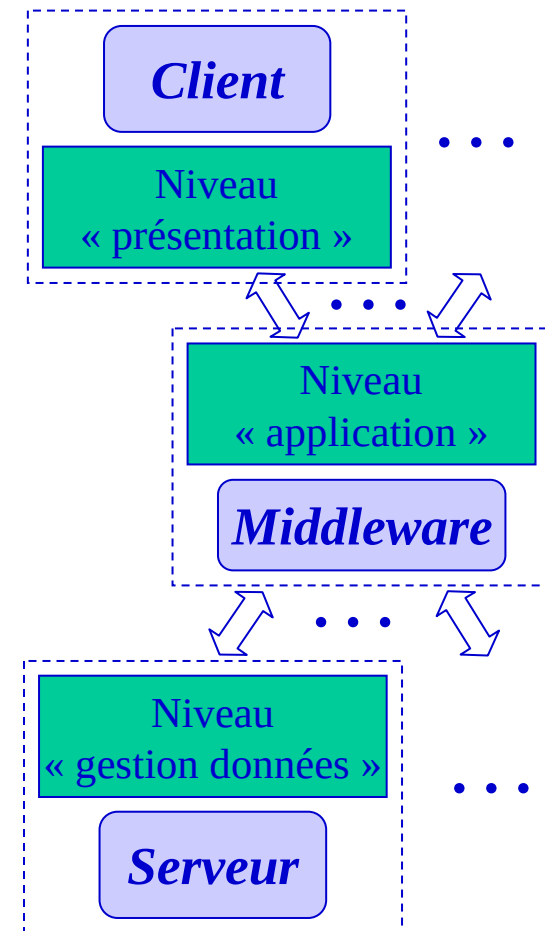
Architectures



1-tier



2-tiers



3-tiers

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 du comportement des sites
 - Débit faible
 - Clients légers
 - Interaction, présentation 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 »

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)