Hashtag Twitter #datatuesday

contact@datatuesday.com

www.data-tuesday.com

La gestion de données à l'heure de la Toile

Serge Abiteboul
INRIA Saclay
& Collège de France

Fernando Velez
SAP

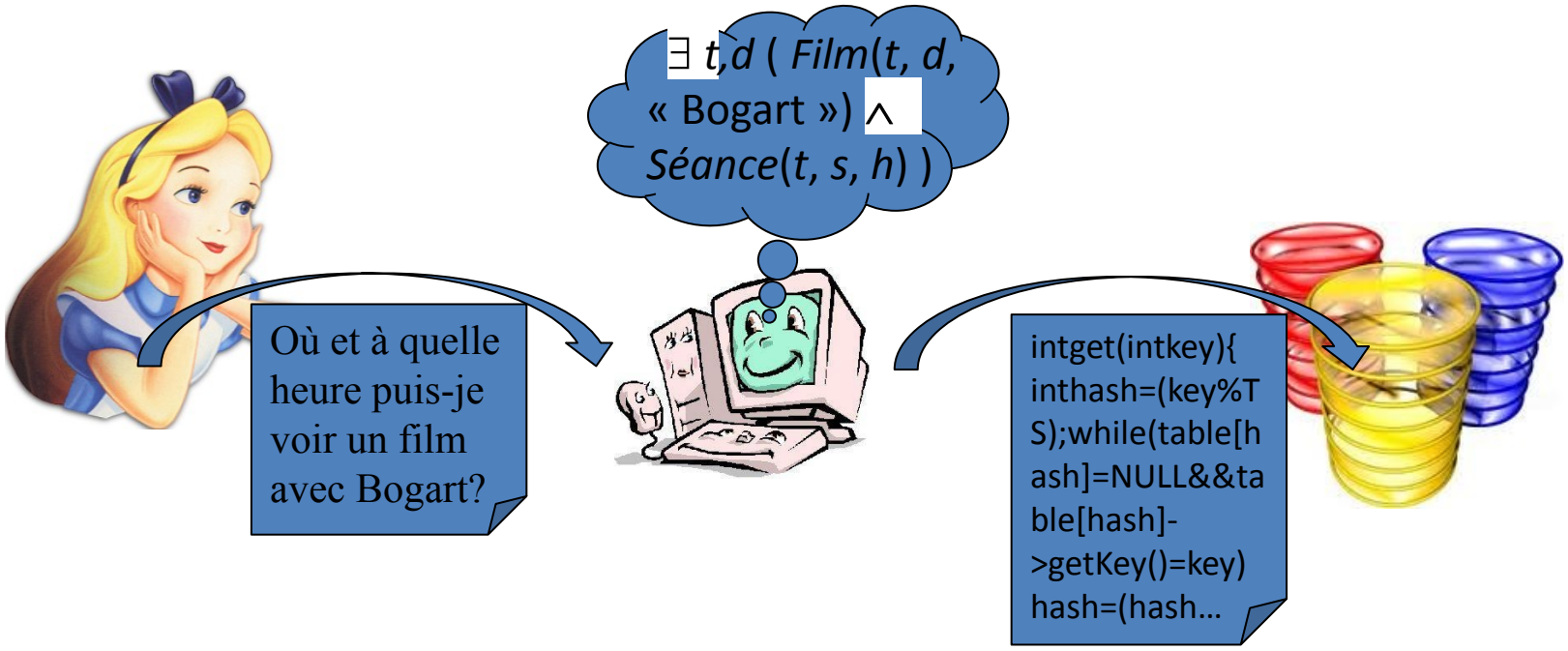
Organisation

- La gestion de données
  - Grands principes
  - Tendances
- Les principaux modèles de données
  - Modèle relationnel, OLAP, modèle objet, modèles hiérarchiques et XML, NoSQL, Web sémantique
- Les nouvelles architectures pour la gestion de données
  - Architectures de gestion de données
  - MapReduce
  - P2P indexing
- Conclusion

La gestion de données

1er principe: médiation et abstraction

Le système de gestion de données joue le rôle de médiateur entre des utilisateurs intelligents et des objets qui stockent l'information



Les questions sont traduites en **logique du premier ordre** puis en **programme**

- Syntaxe et sémantique précise, non ambiguë
- Alice n'a pas envie d'écrire ce programme; elle n'a pas à le faire

1er principe: abstraction des données & langages de manipulation de données

- Structure de données simple et langage de requêtes puissant
- Les premiers modèles
  - Hiérarchie d'enregistrements: IMS
  - Graphe d'enregistrement: CODASYL
  - Langages procéduraux: getFirstRecord, getNext, getFirstChild…
- Le système relationnel: Codd 1970
  - Données sous forme de table
  - Requêtes « déclaratives » en calcul relationnel
  - En pratique un langage plus riche: SQL
  - Un grand succès scientifique et industriel
    - Systèmes commerciaux comme Oracle, DB2
    - Logiciels libres comme mySQL
    - SGBD sur PC, MS Access

2ème principe: universalité

- Les SGBD visent à capturer

  toutes les données du monde

  pour tout type d'applications

  - Langages puissants
  - Fonctionnalités très riches: voir plus loin
  - Eviter de multiplier les développements
- En réalité
  - Données moins structurées restent dans des fichiers
  - Applications trop intenses demandent des logiciels spécialisés
  - Aujourd'hui: de plus en plus de systèmes spécialisés

Pour l'universalité

- Il faut des services comme
  - Concurrence et transactions
  - Fiabilité et sécurité
  - Données distribuées
- Passage à l'échelle
  - Volume de données
  - Volume de requêtes
- Performance
  - Temps de réponse
  - Débit

Concurrence et transactions - ACID

- *atomique* : la suite d'opérations est indivisible, en cas d'échec en cours d'une des opérations, la suite d'opérations doit être complètement annulée
- *cohérente* : le contenu de la base de données à la fin de la transaction doit être cohérent sans pour autant que chaque opération durant la transaction donne un contenu cohérent.
- *isolée* : lorsque deux transactions A et B sont exécutées en même temps, les modifications effectuées par A ne sont ni visibles par B, ni modifiables par B tant que la transaction A n'est pas terminée et validée (*commit*).
- *durable* : Une fois validé, l'état de la base de données doit être permanent, et aucun incident technique ne doit pouvoir engendrer une annulation des opérations effectuées durant la transaction.

Fiabilité et sécurité

- Les SGBD doivent résister aux pannes
- Toute une gamme de techniques: journal, copies back-up, pages ombrées
- « Hot-standby »: second système fonctionnant en simultanée

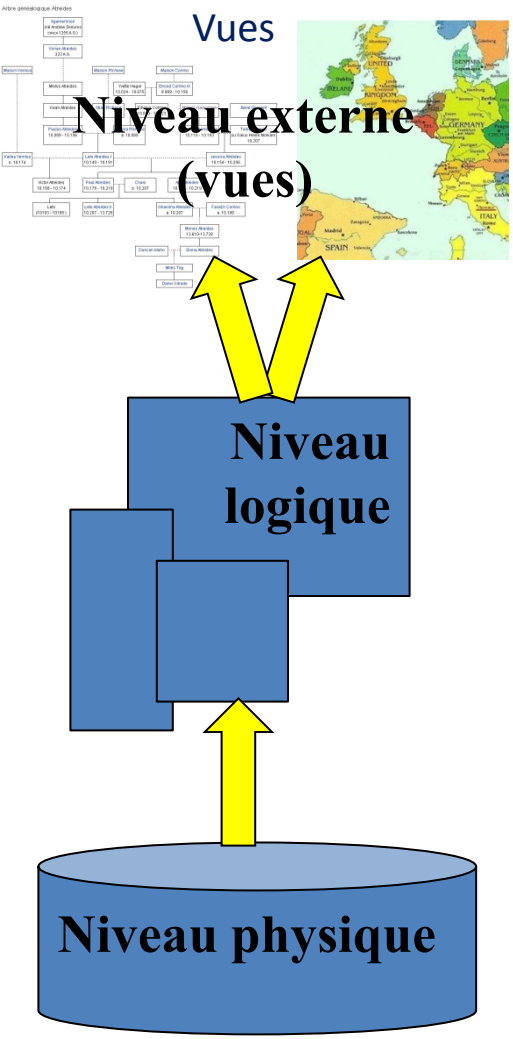
- Aussi gestion
  - de la confidentialité: control d'accès, authentification, autorisation
  - de l'intégrité: protection des données

Données distribuées

- Fragmentation de données
  - Typiquement partitionnement horizontal
- Gestion de requêtes sur des données distribuées
  - Localisation des données & optimisation globale
- Transaction distribuées
  - Commit à deux phases

- Typiquement trop lourd pour des applications Web

3ème principe: indépendance physique/logique/externe



- Séparation en trois niveaux
  - Niveau physique: organisation physique des données sur disque, gestion disque, schémas, indexes, transaction, journal
  - Niveau logique: organisation logique dans un schéma des données; évaluation de requêtes et maj
  - Niveau externe: schémas des vues

    API, environnements de programmation
- Indépendance
  - Physique : On peut modifier l'implantation physique sans modifier le niveau logique
  - Logique : On peut faire évoluer le niveau logique sans changer les applications
  - Externe : On peut modifier ou rajouter des vues sans toucher au niveau logique

Deux classes d'applications avec des besoins aigus de gestion de données

- OLTP: On Line Transaction Processing – transactionnel
  - Gestion des commandes, e-commerce, banque, etc.
  - Transactions très simples et connues d'avance
  - Très forte charge en nombre de transactions par seconde
- OLAP: On Line Analytical Processing – décisionnel
  - Requête de business intelligence
  - Requêtes souvent très complexes impliquant des fonctions agrégat
  - Requêtes multidimensionnelles: e.g., date, pays, produit

Tendance: beaucoup d'information sur la Toile

Google: gestion des pages du Web
Facebook: informations personnelles et communautés
Wikipedia: dictionnaire
Amazon, eBay: catalogues de vente sur le Web
YouTube, Dailymotion : vidéos
Twitter: communication, news
Flickr: base de données de photos
iTunes, AresGalaxy, Kazaa, Emule, Batanga, BearShare, etc: musique en ligne
Myspace: pages Web
Meetic: fiches individuelles
Wikileaks: secrets d'états

Quel est leur point commun ?

Gestion d'information

Tendances

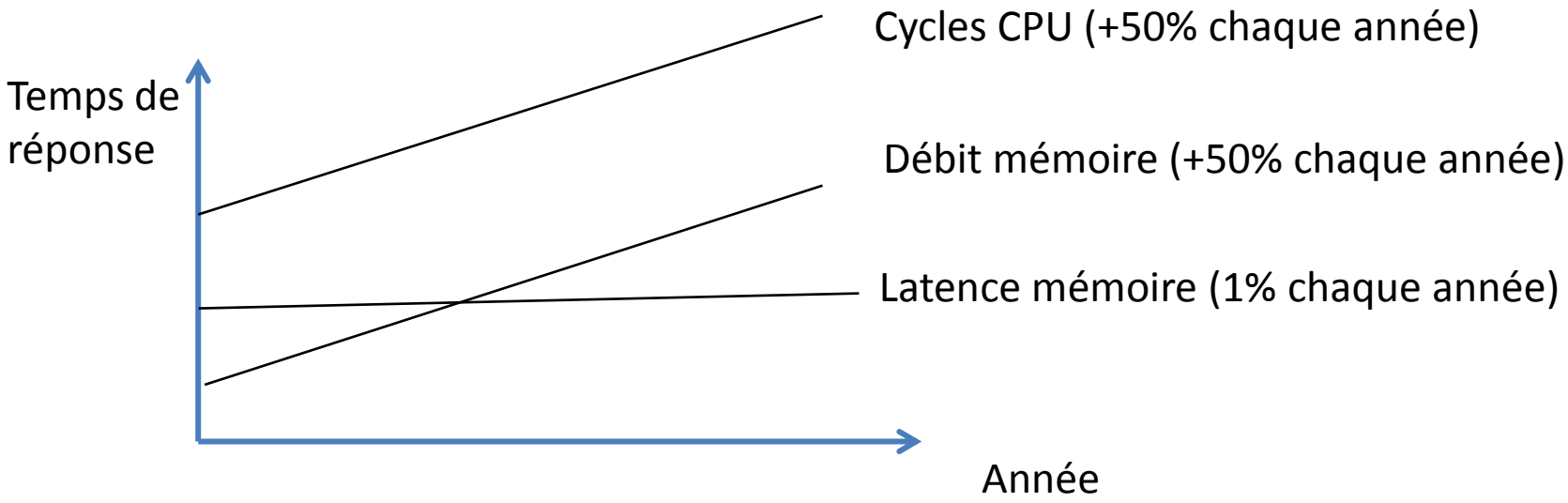
- Accès uniforme et universel à l'information par la Toile
  - Standards universels pour échanger de l'information
- Gestion d'information en nuage (*on the cloud*)
  - Pour les applications standards
- Parallélisme massif et utilisation de volumes massifs de mémoire
  - pour les applications extrêmes
- Emergence de nouveaux modèles de données
- Logiciels libres

Le matériel

- Support par ordre croissant en volume de stockage persistant
  - la mémoire flash
  - le disque dur (ou disque magnétique)
  - le disque optique (qui inclut les CD et DVD)
  - la bande magnétique.
- Les données que nous utilisons sont de moins en moins sur le disque local, de plus en plus, sur le cloud

Support de stockage	Temps d'accès	Taille
Mémoire vive	Microsecondes	Gigaoctets
Disque dur	Millisecondes	Quelques centaines de gigaoctets au téra
Réseau local	Millisecondes	Téraoctets (10^{12})
La Toile	Secondes	Virtuellement ∞

Le matériel: tendances



- Densité de transistors sur une puce de silicium double tous les deux ans (Loi de Moore)
- Capacités de stockage, la densité de mémoire pour les disques durs double chaque année (Loi de Kryder)

Les principaux modèles de données

Les principaux modèles de données

- Modèle relationnel
- OLAP
- Modèle objet
- Modèles hiérarchiques et XML
- NoSQL
- Web sémantique

Les principaux modèles de données

Modèle relationnel

Les données sont organisées en relations

Film		
Titre	Directeur	Acteur
Casablanca	M. Curtiz	Humphrey Bogart
Les quatre cents coups	F. Truffaut	Jean-Pierre Léaud
Star wars	G. Lucas	Harrison Ford

Séance		
Titre	Salle	Heure
Casablanca	Lucernaire	19 :00
Casablanca	Studio	20 :00
Star Wars	Sel	20 :30
Star Wars	Sel	22 :15

Les questions sont exprimées en calcul relationnel

- q_{HB} = { $s, $h | ∃ $d, $t (*Film*($t, $d, « Humphrey Bogart ») ∧ *Séance*($t, $s, $h) }

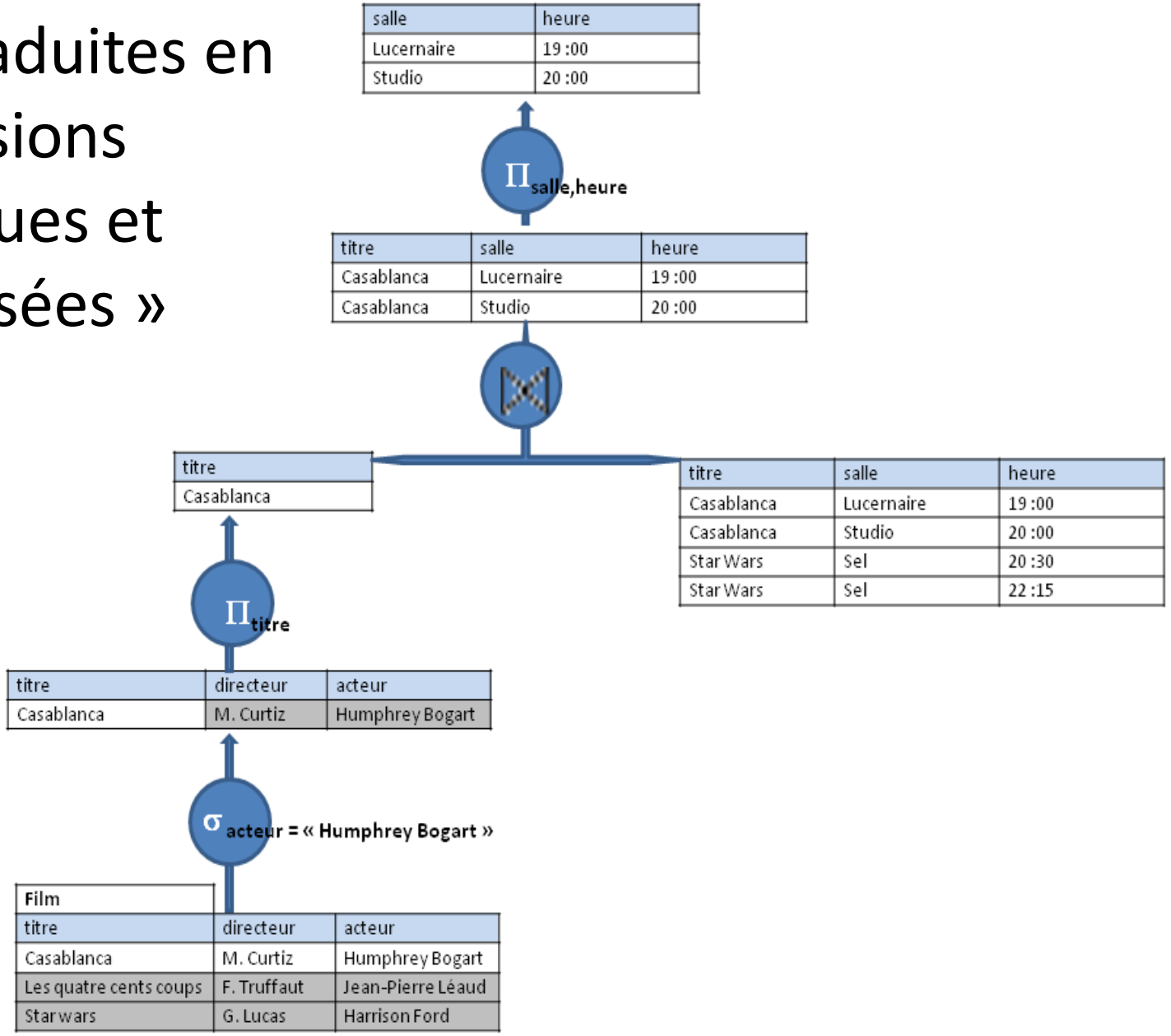
- En pratique avec une syntaxe plus simple à comprendre:
- SQL:

  select salle, heure

  from Film, Séance

  where Film.titre = Séance.titre **and** acteur= «Humphrey Bogart»

Elles sont traduites en expressions algébriques et « optimisées »



Optimisation

- Utilisation de structure d'accès
  - Hachage
  - Arbre-B
- Utilisation d'algorithme sophistiqué
  - Jointure
- Evaluation des coûts pour choisir un plan d'exécution
- Problème: espace de recherche est trop grand
- Technique: Réécriture de requêtes basée sur des heuristiques pour n'en explorer qu'une partie

Les raisons du succès de ces systèmes

- Les requêtes sont fondées sur le calcul relationnel, un langage logique, simple et compréhensible par des individus surtout dans des variantes comme SQL
- Une requête du calcul est facilement traduisible en une expression de l'algèbre simple à évaluer (Th. de Codd)
- L'algèbre relationnelle est un modèle de calcul limité (elle ne permet pas de calculer n'importe quelle fonction). C'est pour cela qu'il est possible d'optimiser l'évaluation d'expressions de l'algèbre.
- Enfin, pour ce langage, le parallélisme permet de passer à l'échelle de très grandes bases de données (classe AC0).

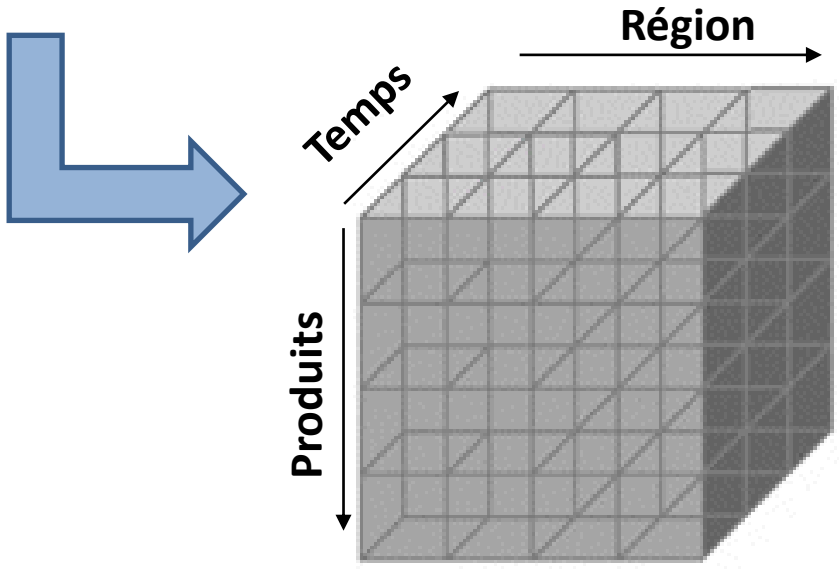
Les principaux modèles de données

Le modèle OLAP multidimensionnel

Les données sont organisés en cubes

Mars

	USA	Canada	France	UK
Pain	12	25	14	86
Fromage	23	68	45	25
Yaourt	12	95	65	42
Chocolat	44	22	33	18



+ d'autres dimensions :

- Canal de vente
- Type de client
- …

Nature, besoins et techno

Gestionnaires de données fournissant des vues multi-dimensionnelles de données quantitatives pour l'analyse

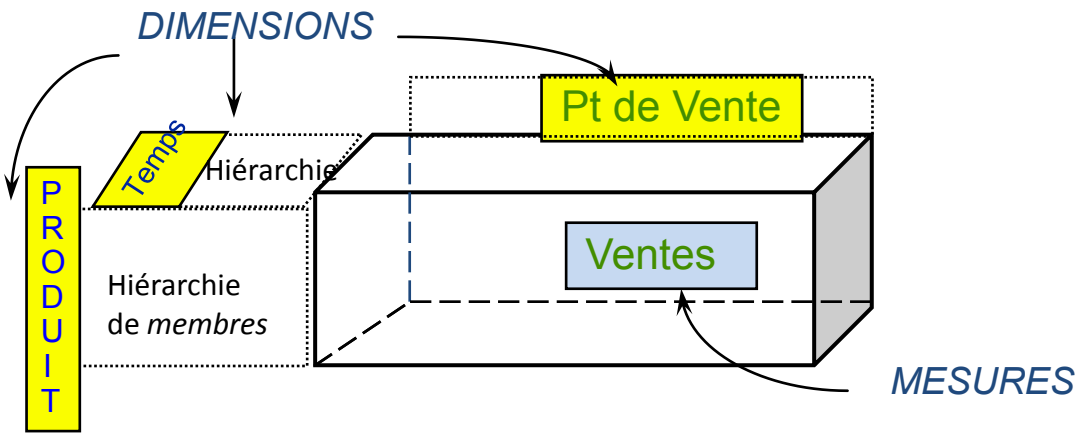
- Evolution à partir du tableur
- Règles OLAP de E. F. Codd: 1993
- Besoin: réponse rapide à des requêtes du genre
  - 5 premiers groupes démographiques achetant mes vélos
  - Produits vendus en France où le taux de rejet a diminué

Agrégations des quantités à différents niveaux d'analyse

  - ventes par mois, trimestre et année

Moteur, mais aussi outils, langages, APIs

Outils: navigation dans des *cubes OLAP*



Drop Filter Fields Here

			Store State							
			CA		OR		WA		Grand Total	
Product Family	Product Department	Product Category	Store Sales	Unit Sales	Store Sales	Unit Sales	Store Sales	Unit Sales	Store Sales	Unit Sales
Drink	Alcoholic Beverages		$4,011.37	1,936.00	$3,460.32	1,680.00	$6,557.39	3,222.00	$14,029.08	6,838.00
	Beverages	Carbonated Beverages	$1,696.84	923.00	$1,511.01	858.00	$3,028.50	1,626.00	$6,236.35	3,407.00
		Drinks	$1,710.08	766.00	$1,476.12	632.00	$2,456.09	1,071.00	$5,642.29	2,469.00
		Hot Beverages	$2,626.93	1,208.00	$2,378.87	1,078.00	$4,255.94	2,015.00	$9,261.74	4,301.00
		Pure Juice Beverages	$1,972.68	989.00	$1,581.13	817.00	$3,054.34	1,590.00	$6,608.15	3,396.00
		Total	$8,006.53	3,886.00	$6,947.13	3,385.00	$12,794.87	6,302.00	$27,748.53	13,573.00
	Dairy		$2,185.34	1,280.00	$1,729.84	1,041.00	$3,143.42	1,865.00	$7,058.60	4,186.00
	Total		$14,203.24	7,102.00	$12,137.29	6,106.00	$22,495.68	11,389.00	$48,836.21	24,597.00
Food	Baked Goods		$4,619.81	2,150.00	$4,088.95	2,013.00	$7,746.67	3,707.00	$16,455.43	7,870.00
	Baking Goods		$11,217.84	5,799.00	$9,224.88	4,810.00	$18,227.69	9,636.00	$38,670.41	20,245.00
	Total		$115,193.17	53,656.00	$102,564.67	48,537.00	$191,277.75	89,747.00	$409,035.59	191,940.00
Non-Consumable			$29,771.43	13,990.00	$27,575.11	13,016.00	$50,019.79	23,230.00	$107,366.33	50,236.00
Grand Total			$159,167.84	74,748.00	$142,277.07	67,659.00	$263,793.22	124,366.00	$565,238.13	266,773.00

Langage de Requête standard: MDX (MSFT, 1997)

SQL

- *select, from, where, group-by*
- Produit une table (2-dim)
- Sélectionne des colonnes d'une ou plusieurs tables
- Filtre les lignes avec des prédicats dans la clause *where*
- Agrège avec *group by*

MDX

- *with, select, from, where*
- Produit un cube (N-dim)
- Sélectionne des *axes* : ensembles de membres des dimensions d'un cube
- Spécifie des membres des dimensions non sélectionnées (*where*) ou des dims sélectionnées (*filter* dans select)
- Agrégation implicite

```
with member Measures.profit as Measures.StoreSales – Measures.cost
select
   {Measures.StoreSales, Measures.Profit} on columns,
    non empty filter(Product.ProductDepartment.members,
            (Product.currentMember, Measures.StoreSales) > 20000.0) on rows
from [Sales]
where ([Time].[1997])
```

Types de moteur

OLAP Multidimensionnel (MOLAP)

- Stockage en vecteur spécialisé favorisant le temps de réponse
- Agrégats pré-calculés, compression, déconnexion de la source

OLAP Relationnel (ROLAP)

- Schéma d'entrepôt (*warehouse*) relationnel spécialisé
- Données plus fraiches, moins gourmand en espace
- SGBD peut traiter plus de volume, mais temps de réponse plus lent
- Des vues matérialisées peuvent être crées pour accélérer les réponses

OLAP Hybride (HOLAP)

- Cube pour les agrégations, tables pour les détails ; compromis

Extensions analytiques de SQL

- sous-totaux: group-by *cube* | *rollup*, fonctions analytiques

Utilisations et proposition de valeur

Utilisations

- Source de type *Business Intelligence* par excellence
  - Comprendre le passé: requêtes complexes, gros volumes
  - Deviner le futur (analyse prédictive): algorithmes statistiques
- Support pour des applications d'entreprise: planification budgétaire, profitabilité, gestion de la performance, …
- Modèle de données adopté par la BI, même sur des SGBD-R

Proposition de valeur

- Rapidité et prédictibilité du temps de réponse
- Analyse indépendant du moteur transactionnel
- Manipulation intuitive
- Sécurité

Acteurs principaux

Microsoft

SAP

IBM

Oracle

Logiciel libre

- Pentaho Mondrian

SAP HANA

Machine SGBD en mémoire

Transactionnel (SQL) et OLAP (SQL et MDX)

Base pour la suite applicative SAP et pour la BI (2012)

Choix entre stockage ligne ou colonne pour une table

Des *vues analytiques* permettent de

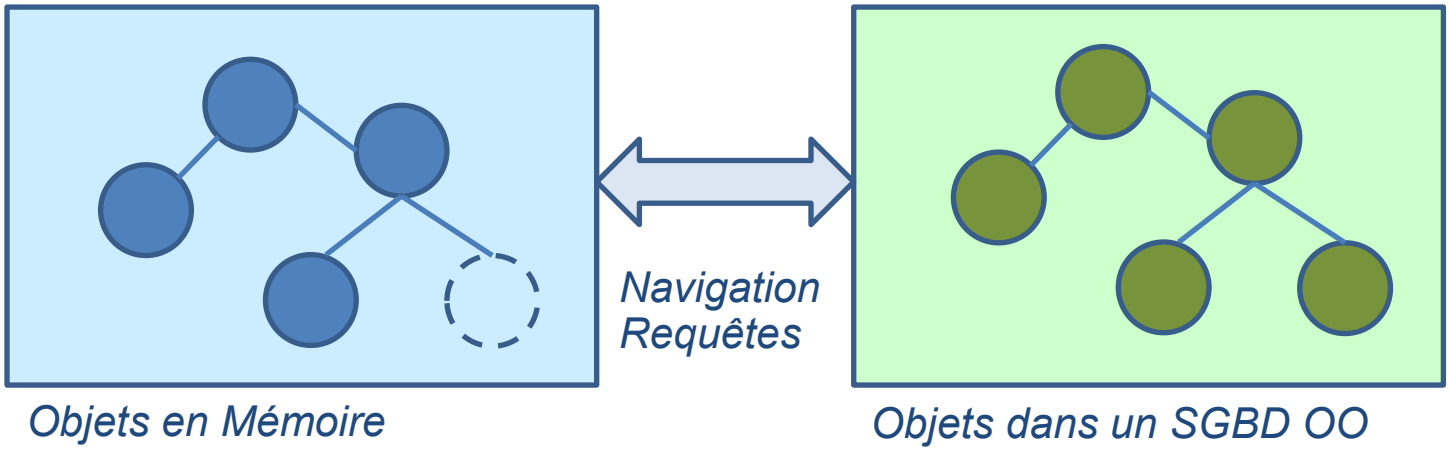
- modéliser des tables en tant que dimensions et cubes (mesures et calculs de même dimensionnalité)
- matérialiser des indexes

Les principaux modèles de données

Le modèle objet

Les données ce sont des objets

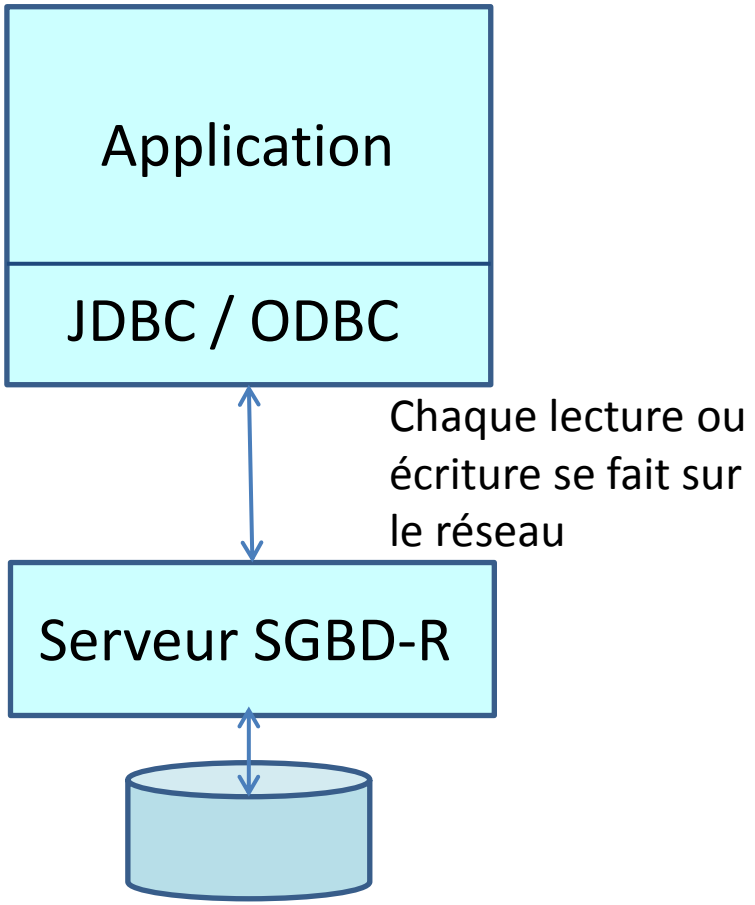
Objet = donnée + comportement

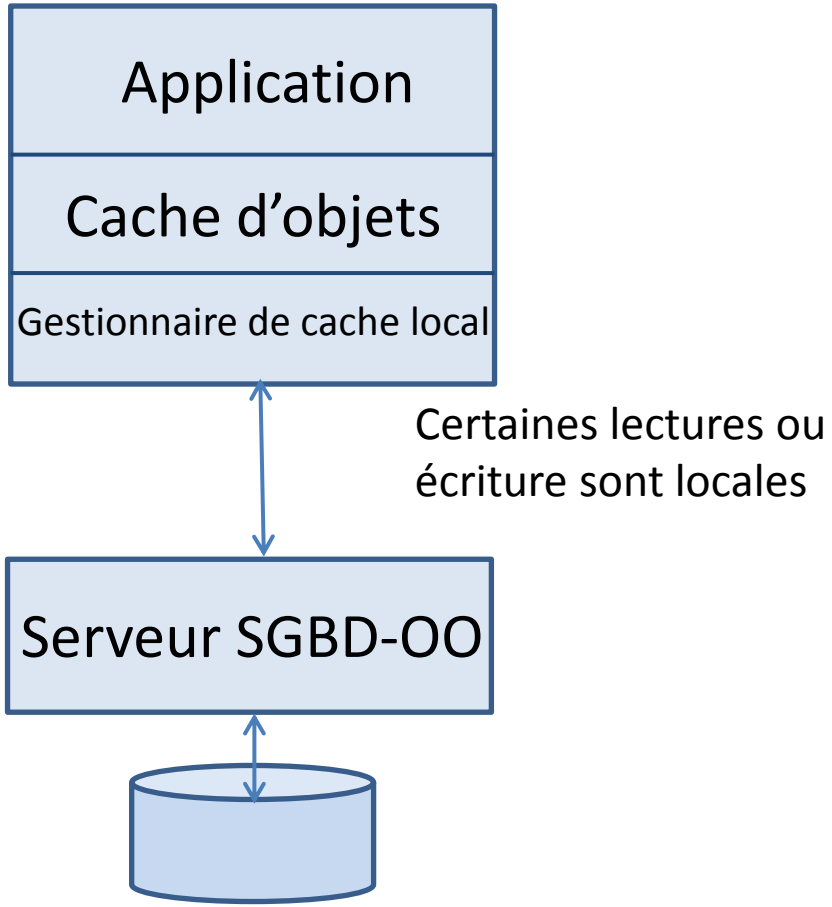


SGBD-OO = LP OO + BD

- Modèle de données objet (identité, objets complexes, encapsulation, classes, héritage…)
- Extension d'un langage de programmation objet avec des fonctionnalités bases de données (transactions, requêtes, etc.)

Architecture SGBD: R versus OO



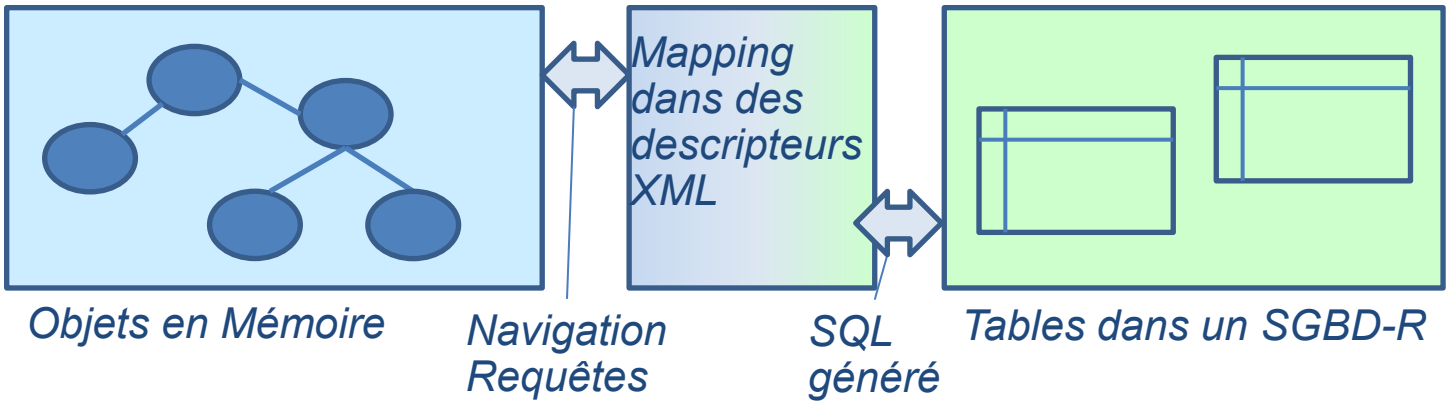


Histoire, acteurs et standardisation

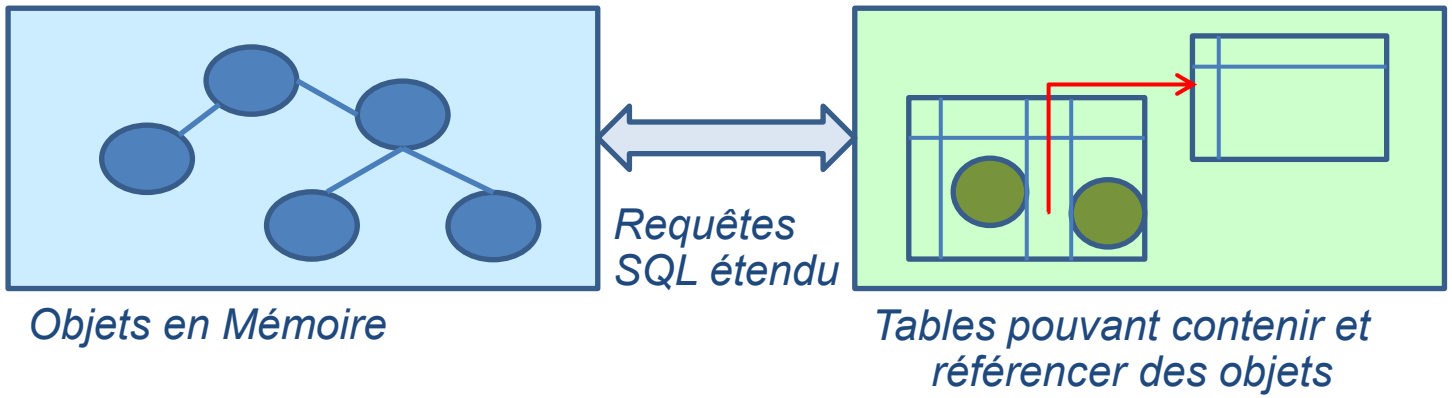
- 1987 – 1993: Produits initiaux
  - GemStone, O2, ObjectStore, Objectivity, Versant, Poet
- 1989: Object Database Manifesto
  - Atkinson, Bancilhon et al; SGBD-OO = LP OO + DB
- 1993 – 2000: Standard ODMG 1.0 → 3.0
  - Modèle Objet, OQL, *bindings* SmallTalk, C++, Java
- 2000 – 2010
  - Standard post-ODMG autour de Java: JDO
  - Logiciel libres (p.ex., Db4o, implémentations JDO)
  - *Frameworks* de persistance langage (p.ex., JPA, DataObjects.NET)

Deux autres variantes

- *Mapping* Objet-Relationnel (p.ex., TopLink, Hibernate)



- Modèle Objet-Relationnel (p.ex. Informix, Oracle)



Utilisations et proposition de valeur

Simplicité pour le développeur d'application

- Un seul modèle de données (plus riche), intégration avec langage de programmation objet, persistance orthogonale, extensibilité, langage de requêtes objet

SDBD-OO

Performances pour la manipulation d'*objets complexes*

- Jointures entre plusieurs tables remplacés par de la navigation inter-objet, souvent sur cache local

Mapping et modèle OR

- Réutilisation de données existantes, faciliter conversion OO→R
- migration R→OR aisée

Les principaux modèles

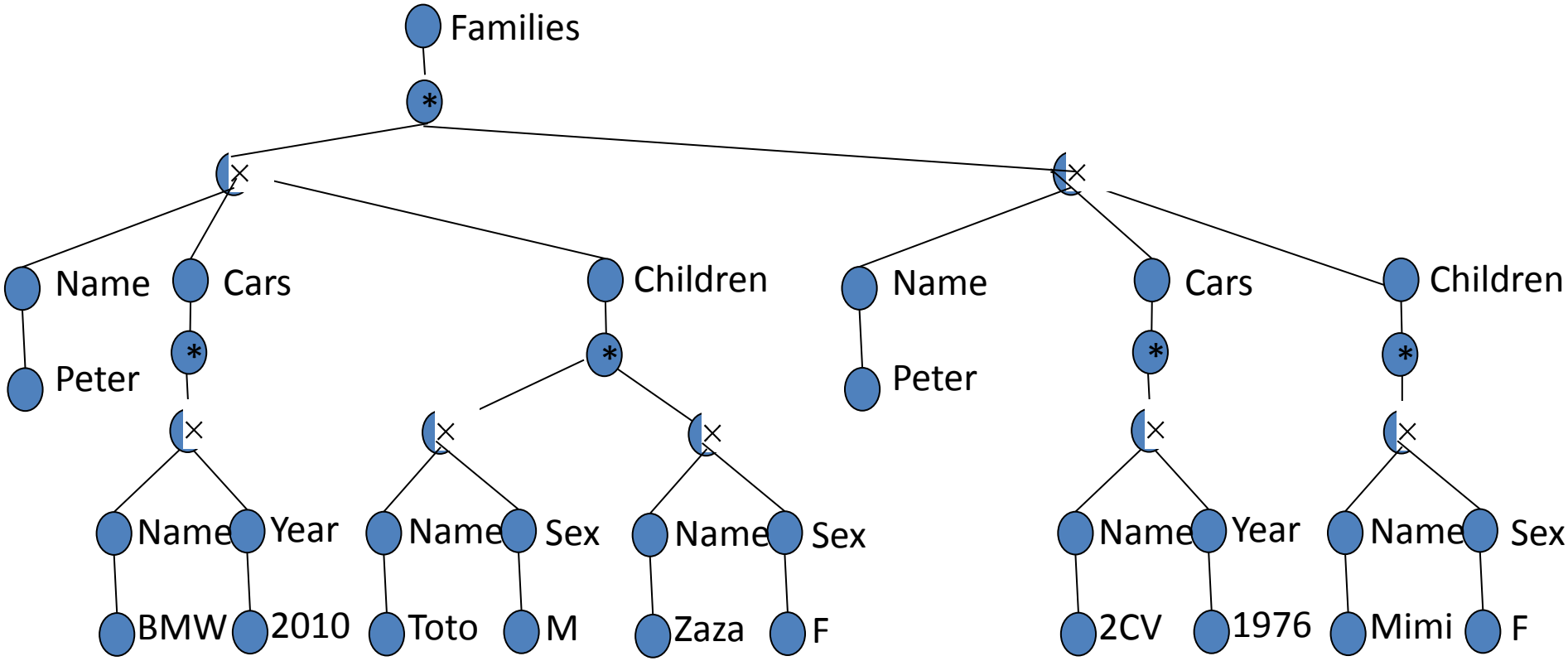
Les arbres de données et XML

Représenter les données sous forme d'arbre: une vieille idée

- On va ignorer les premiers modèles hiérarchiques comme IMS des années 70 qui n'avaient pas de langages déclaratifs
- Tout a commencé dans les années 80 avec les modèles N1NF
  - François Bancilhon en France et Hans Schek en Allemagne

Name	Child	Car
Alice	Toto Lulu	Jaguar 2CV
Bob	Mimi Zaza	Mustang Prius

Modèle d'objets complexes: constructeur n-uplet et ensemble



Logique et algèbre pour objets complexes

- **Logique**: principale nouveauté – variable dénote des ensembles
- Exemple: Requête AbouBanat
- { T.Father | Families(T) $\wedge \forall$ X $\in$ T.Children (X.Sex = F) }

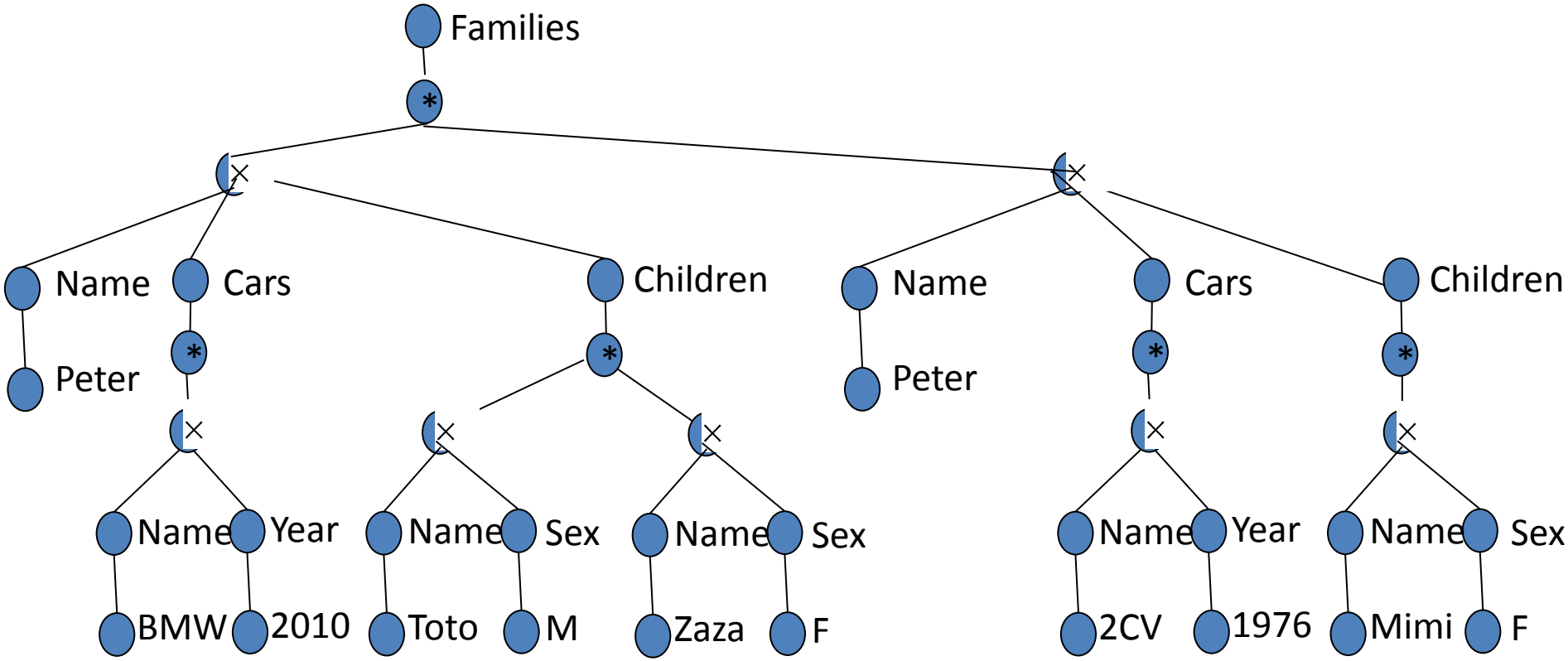
- **Algèbre** : opérations ensemble des parties, unnest/nest

Name	Child	Car
Alice	Toto	
Bob	Mimi Zaza	Mustang
Bob	Lulu	Prius

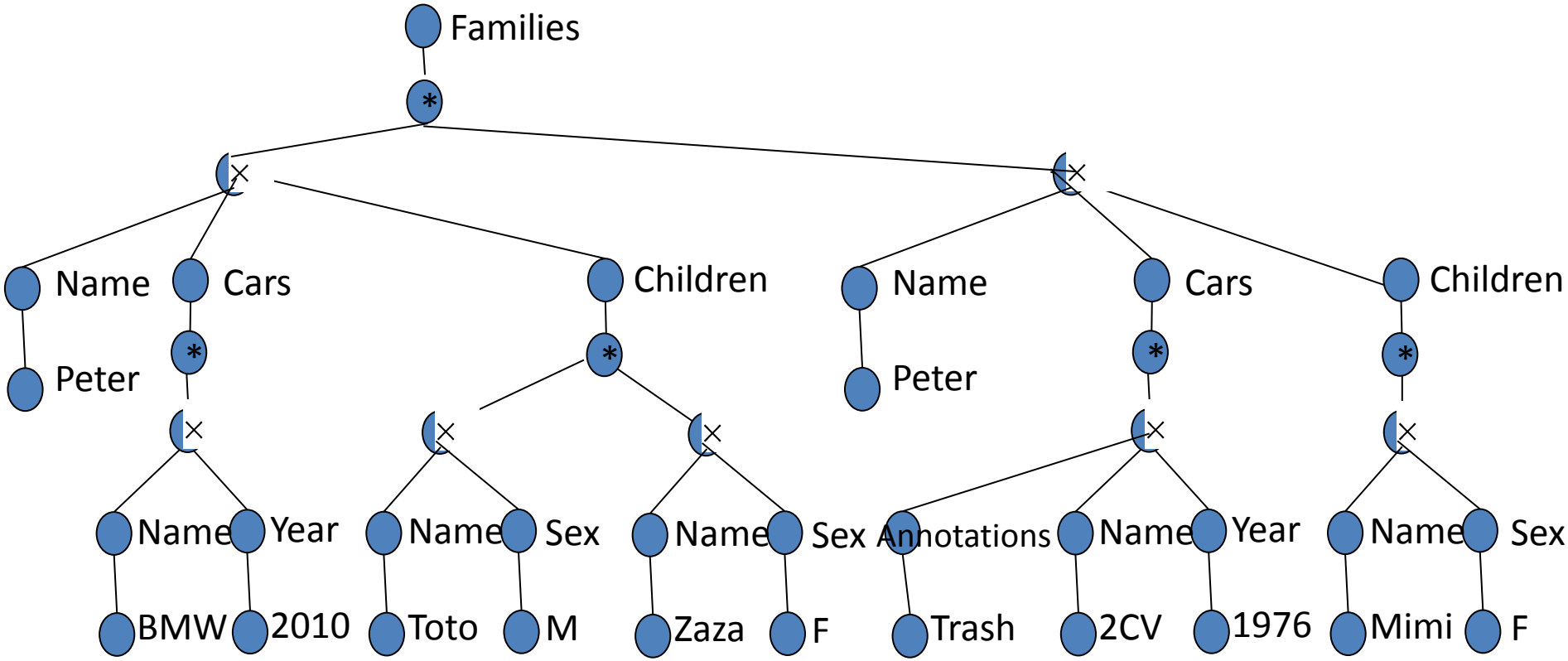
Name	Child	Car
Bob	Mimi	Mustang
Bob	Zaza	Mustang
Bob	Lulu	Prius

Name	Child	Car
Bob	Mimi Zaza Lulu	Mustang Prius

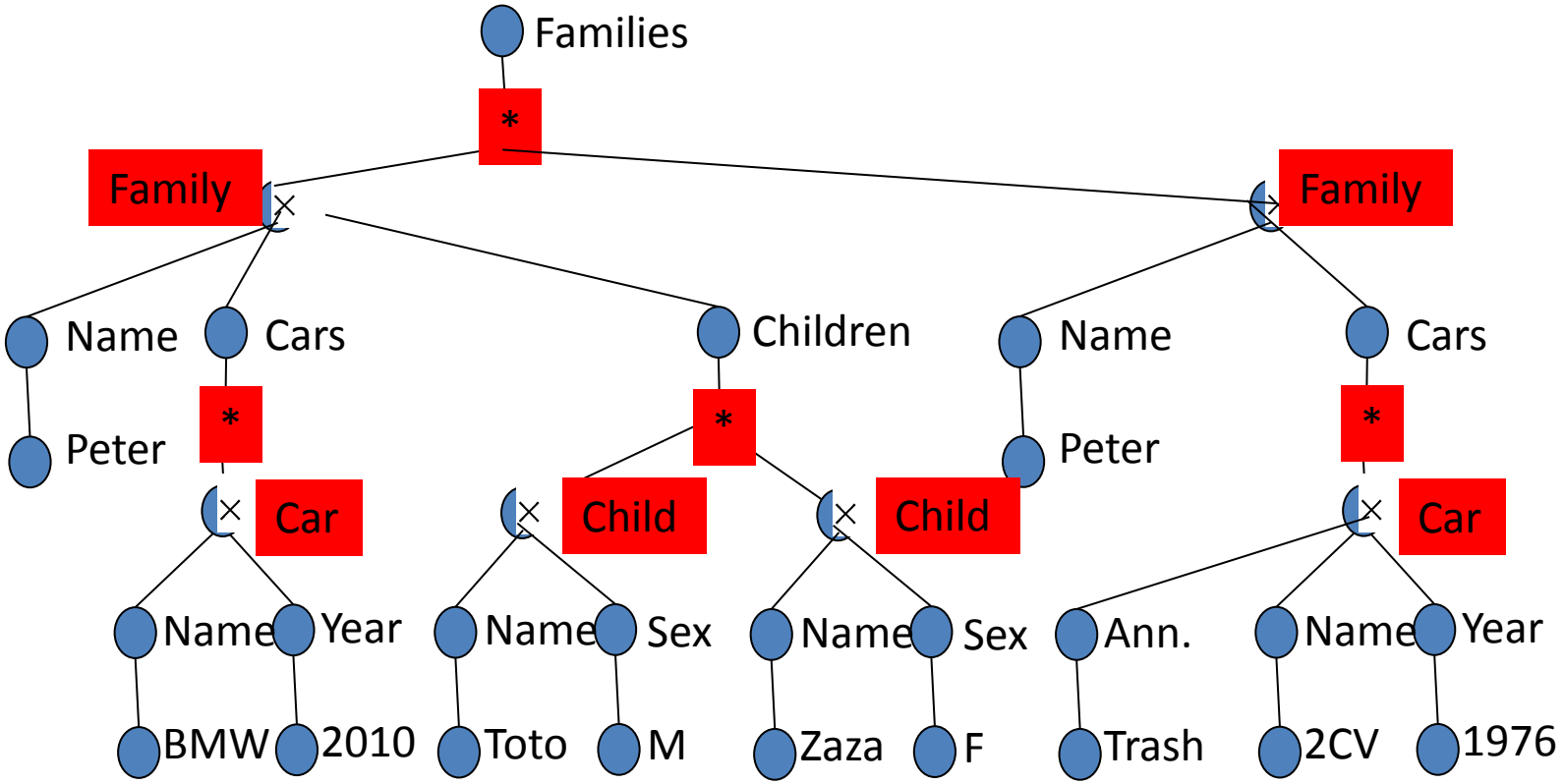
Des objets complexes aux données semistructurées et XML



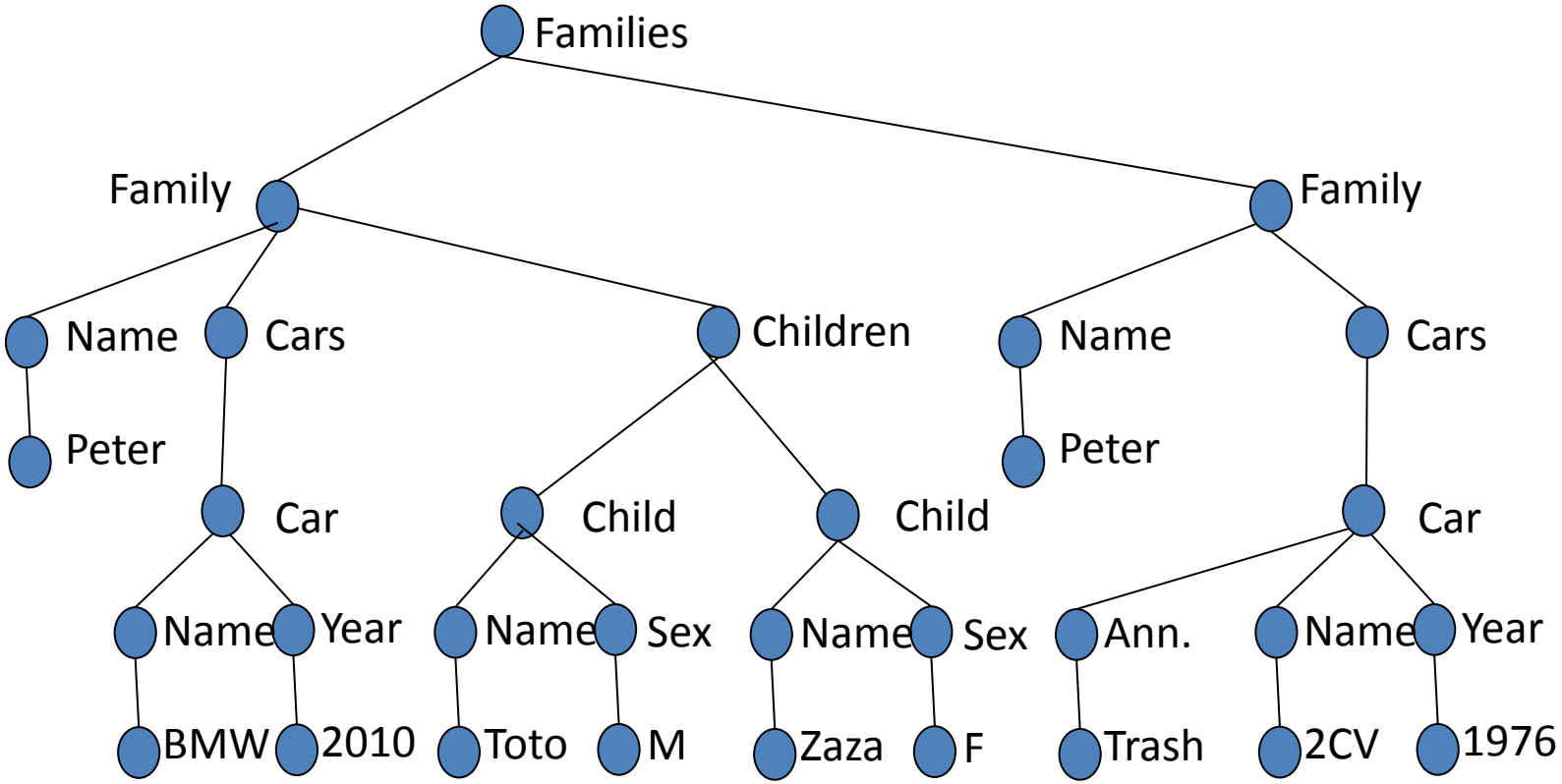
Révolution 1: plus de flexibilité



Révolution 2: virer les nœuds « * » & nommer tout



XML =
arbres *ordonnés*, étiquetés, non bornés



Données directement publiables sur le web

- Une forme sérialisée pour les échanges – syntaxe verbeuse

  <families><family><name>Peter</Name><Cars><Car><Name>BMW</Name><Year>2010</Year></Car></Cars><Children><Child> …
- Données autodescriptives: plus de séparation données/schéma
- Flexibilité
- Ordre: sérieux problème pour l'optimisation (automates d'arbres)

Plus ça change,
plus c'est la même chose

oubliez la syntaxe
pensez aux arbres

Le monde XML

- Langages de requête:
  - XPATH
  - Xquery

```
FOR $p IN document("bib.xml")//publisher
LET $b := document("bib.xml)//book[publisher = $p]
WHERE count($b) > 100
RETURN $p
```

- Langage de transformation: XSLT
- Typage: DTD, XML Schema, relax NG, automates d'arbre
- Autres standards autour de XML
  - SOAP, DOM
  - XML dialects: RSS, WML, SVG, XLINK, MathML

Stockage XML

- Dans de simples fichiers
  - Un répertoire devient maintenant une « BD interrogeable »
- Dans des SGBD natifs XML
  - Logiciel Libre: eXist, MonetDB
- Dans des SGBD relationnels
  - Oracle
- Plusieurs types d'API
  - XQJ XQuery API for Java specification (XQJ)
  - XML:DB JDBC pour bases de données XML

Tendance: atténuer la séparation entre SGBD et systèmes de fichiers

Les principaux modèles

NoSQL

Systèmes de gestion de données spécialisés

- Applications « extrêmes »
- OLTP
  - Millions de transactions par seconde
  - Transactions distribuées à l'échelle mondiale
  - Besoin de disponibilité permanente même en cas de panne
  - E.g.: Amazon
- OLAP
  - Analyse de téraoctets de données
  - Décisionnel sur des données d'entreprise
  - Analyse de données du Web: recommandation
    - Par exemple, des livres dans Amazon ou des films dans Netflix

Motivations du NoSQL

- Les SGBD paient un fort overhead pour leur universalité
- Eviter cet overhead pour des applications très exigeantes
- Overheads principaux à éviter
  1. Gestion de buffers pour « cacher » les blocs de disques en mémoire
  2. Locking: pour la gestion de la concurrence. Des transactions doivent attendre la libération des verrous
  3. Latching: des verrous à courte durée utilisés pour les structures d'accès qui sont très partagées comme arbre-B
  4. Logging: chaque mise-à-jour est écrite dans le journal qui est forcé sur disque

- Analyse d'applications OLTP [Harizopoulos&al08] :

  35% gestion de buffer — 21% locking

  19% latching — 17% logging

Systèmes de gestion de données spécialisés

- Spécialisés pour certains types de requêtes
- Spécialisés pour certains aspects comme le passage à l'échelle
- En contrepartie: sacrifient l'universalité
  - Sacrifient certains types de requêtes comme la jointure
  - Sacrifient certaines fonctionnalités comme la concurrence
- No SQL
  - Systèmes non-standard pour la gestion de données
  - Typiquement modèles de données plus simple

  (Supportent parfois SQL)

Attention: Certains utilisent aussi le terme NoSQL pour des systèmes au contraire basés sur des modèles plus complexes: objets/XML/RDF

NoSQL : différents parfums

- Performances extrêmes
  - Passage à l'échelle massif
  - Distribution massive
  - Disponibilité totale
- Spécialisation
  - Taux de transaction élevé
  - Requêtes OLAP simples sur très gros volumes

- Pas universalité
- Moins d'indépendance
  - Pas 3 niveaux
- Moins d'abstraction
  - Pas relationnel et SQL
  - Données simples: clé/valeur
  - Requêtes simples
- Perte de fonctionnalités
  - Pas ACID (strict)
  - Moins de typage et d'intégrité
  - Structures d'accès plus simples
  - API simpliste - pas de JDBC

Exemples

- Stockage clé/valeur avec faible cohérence
  - Cassandra (Apache), Dynamo (Amazon)
- Stockage clé/valeur sur disque
  - Hadoop HBase (Apache), BigTable (Google)
- Stockage document/N1NF
  - MongoDB (logiciel libre)
- Bases de données en mémoire et monothread pour OLTP
  - VolTDB

- Et beaucoup d'autres…

Les principaux modèles

Le Web sémantique

Le Web sémantique

Ajouter des indications sémantiques

Sur un document

auteur = Serge Abiteboul ; titre = Gestion de données sur le Web

nature = présentation ; date = Mars 2012 ; langue = français

A l'intérieur d'un document

Woody Allen <dbpedia:Woody_Allen> était à Cannes <geo:ville_France> pour la première de …

Ces connaissances se basent sur des *ontologies*

Ontologies

- Des phrases logiques comme :
  - ***classes*** *sa: Personne, sa:-Réalisateur, sa:Cinéaste*
  - *sa:-Réalisateur* ***sous classe de*** *sa:Personne*
  - *sa:-Réalisateur* ***synonyme de*** *sa:Cinéaste*
  - *sa:Woody_Allen* ***est un*** *sa:Réalisateur*
  - ***relation*** *sa:a_réalisé*
  - *sa:Woody-_Allen sa:a_réalisé sa:movie_Manhattan*
- Réponse plus fine aux recherches d'information

  Le document *d* contenant *sa:Woody_Allen*

  est une réponse à la question « *cinéaste Woody Allen Manhattan* »

  même s'il ne contient ni le mot *cinéaste* ni le mot *Manhattan*

Permet aussi d' « intégrer » plusieurs sources d'information

A terme, intégrer toutes les informations de la Toile?

Langages pour le Web

- RDF: modèle de graphe proposé par le W3C
  - Sérialisation en XML
  - Langage de requête SPARQL
  - Très simple – triplets – < sujet, prédicat, objet >
- Linked open data
  - publier des collections de données ouvertes liées entre elles sur la Toile
  - September 2011, 31 billions RDF triplets
- Plus riche: RDFS (RDF Schema), OWL
- Ça sert à quoi?
  - Mieux répondre aux requêtes en utilisant les ontologies
    - Requête « Sèvres céramique »
    - Si je sais que « *porcelaine isa céramique* »
      un document avec les mots Sèvres et porcelaine est une réponse
  - Intégrer des données venant de plusieurs sources

Les nouvelles architectures pour la gestion de données

Les nouvelles architectures pour la gestion de données

Architectures système

Architectures système

Architectures de déploiement

- Centralisé
- Client-serveur
- Internet

Architecture du serveur

- Serveur de requêtes vs. serveur de données
- Architectures parallèles
- Stockage colonne vs. stockage en ligne
- Serveur orienté mémoire vs. orienté disque

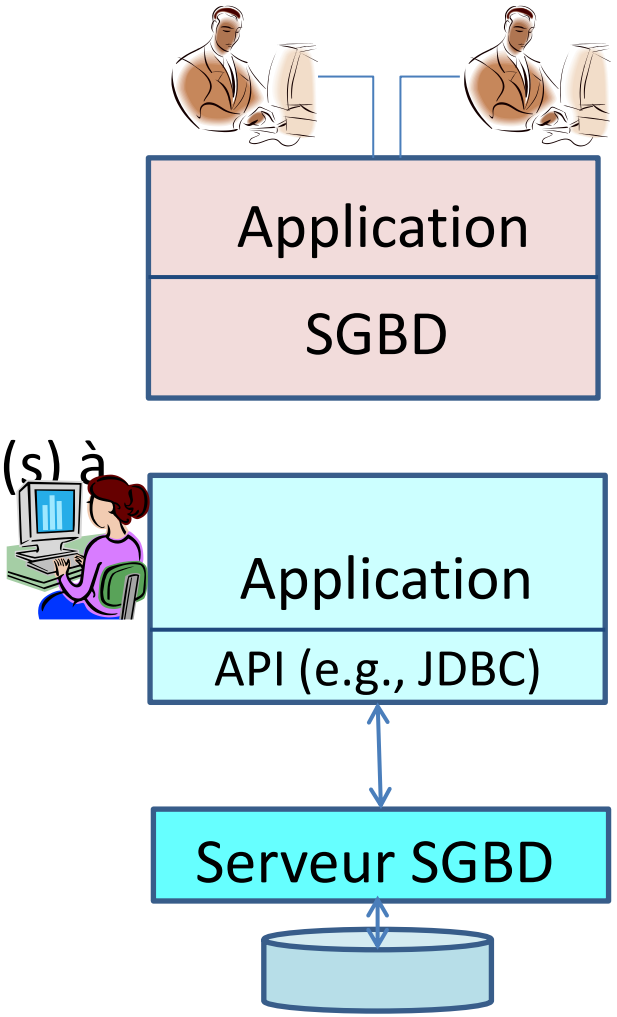
Architectures de déploiement

Centralisé

- Mainframes multi-utilisateur, terminaux
- Systèmes embarqués

Client-Serveur

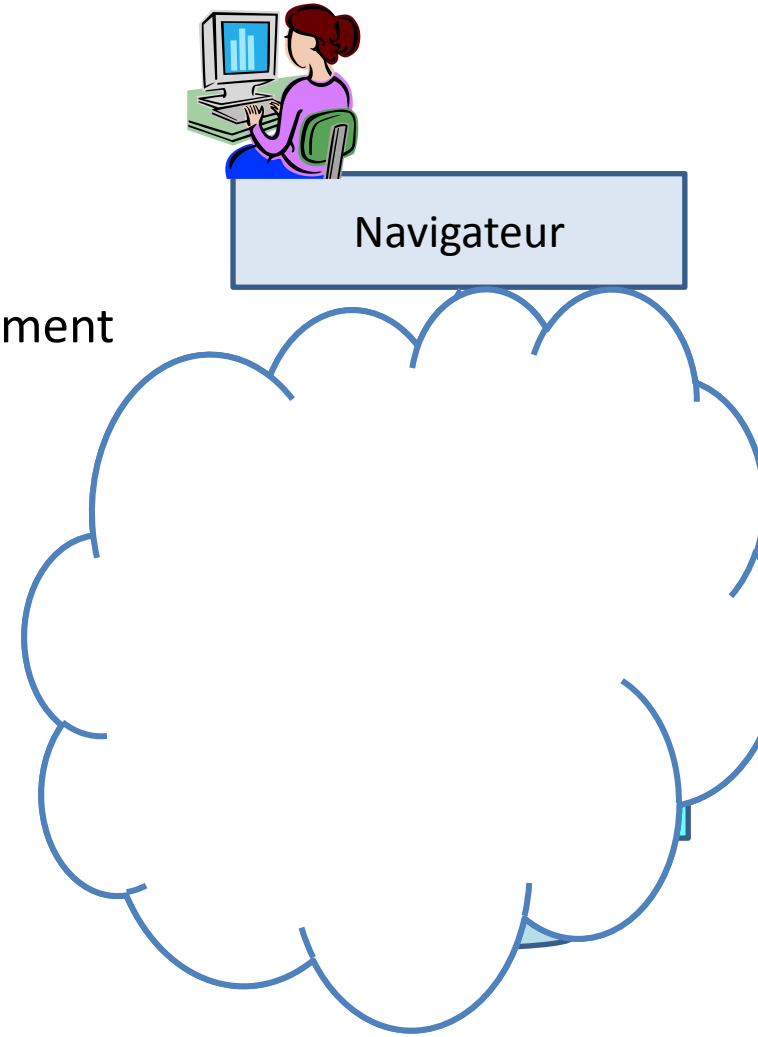
- PCs/stations connectés à un (des) serveur(s) à travers un réseau
- Client fait tourner la présentation (IHM)
  - Éventuellement la logique applicative
- Serveur fait tourner le SGBD et stocke les données
  - Procédures stockées: logique app serveur



Architectures de déploiement (cont.)

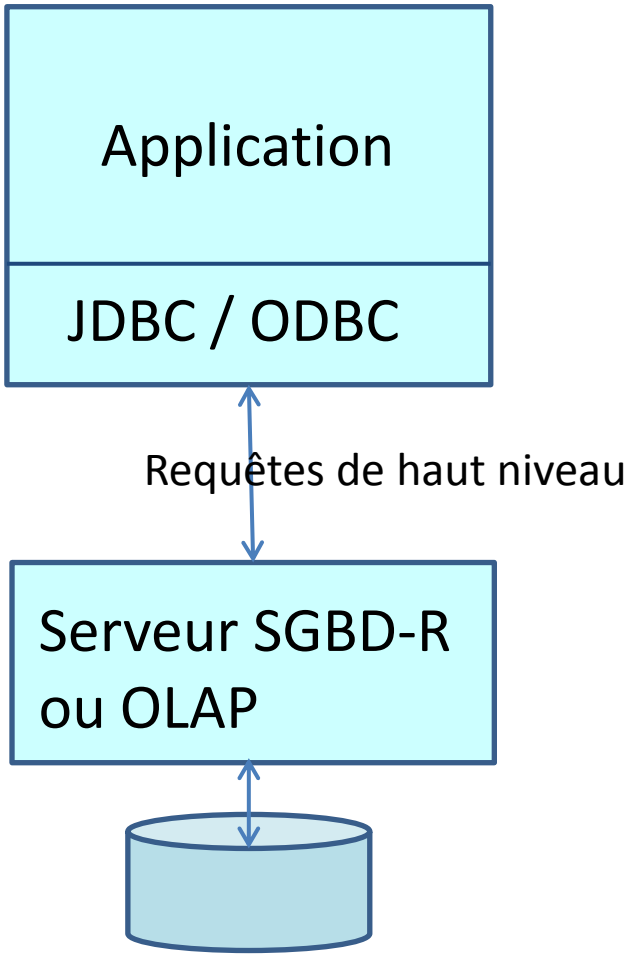
Internet

- Client est un navigateur qui
  - rend du contenu décrit en HTML et
  - communique avec le tier central (typiquement en HTTP)
- Tier Central
  - génère le contenu à rendre,
  - fait tourner la logique applicative, et
  - communique avec la base de données
- Tier serveur de données
  - Comme le serveur en client-serveur

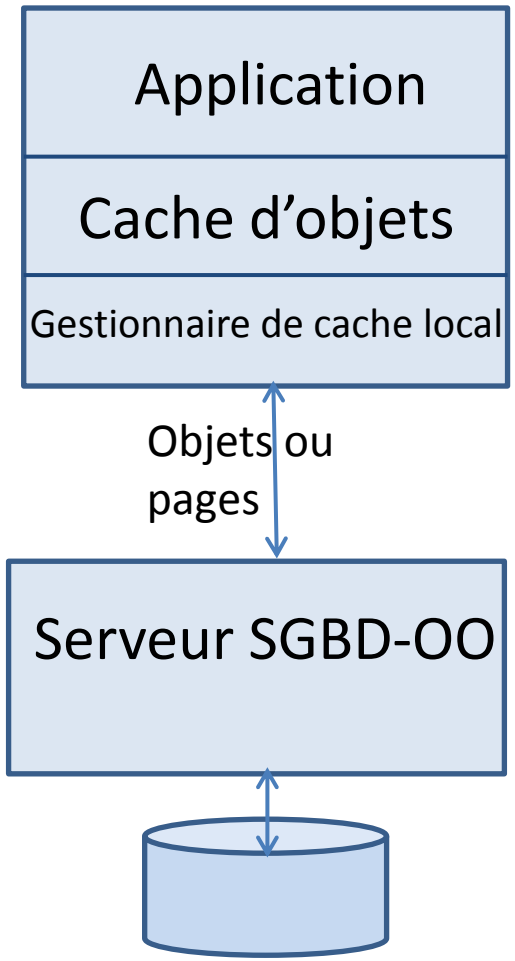


Architecture SGBD: requêtes versus données

Serveur de requêtes



Serveur de données



Décisions :

- Grain de verrouillage
- Cache inter-transaction

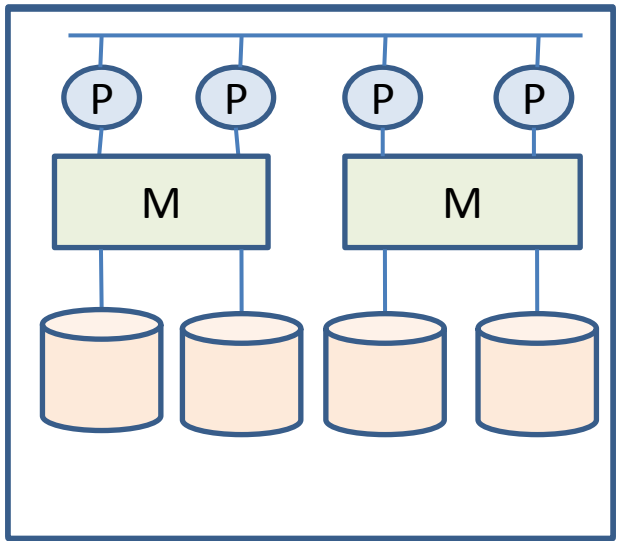
Architecture parallèle

Lorsque le tier serveur tourne sur une architecture

- multi-CPU et multi-disque
- Connexion réseau rapide

Architectures

- Mémoire partagée
- Disque partagé
- Pas de partage (*shared nothing*)
- Hybride (*NUMA*)



Comparatif

Mémoire partagée

- Le bus devient le goulot d'étranglement au-delà de 32-64 proc.
- Utilisé dans la pratique dans des machines de 4 à 8 processeurs

Disque partagé

- Communication inter-CPU plus lente, architecture plus tolérante aux pannes
- Goulot d'étranglement repoussé (centaine de processeurs)

Pas de partage

- Coût de communication plus élevé, surtout en environnement transactionnel
- Réel passage à l'échelle (milliers de processeurs) pour le décisionnel
  - Succès dépend d'un bon schéma de partitionnement
  - Architecture préférée des SGBD-R parallèles, Hadoop

Stockage ligne vs. colonne

Lignes

- Lire/écrire des attributs d'un n-uplet
  - Tous: rapide
  - Un: lent
- Compression limitée
- Agrégations plus lentes
- Adéquat pour le transactionnel

Colonnes

- Lire/écrire des attributs d'un n-uplet
  - Tous: lent
  - Un: rapide
- Plusieurs techniques de compression (10x typique)
- Agrégations rapides
- Adéquat pour le décisionnel

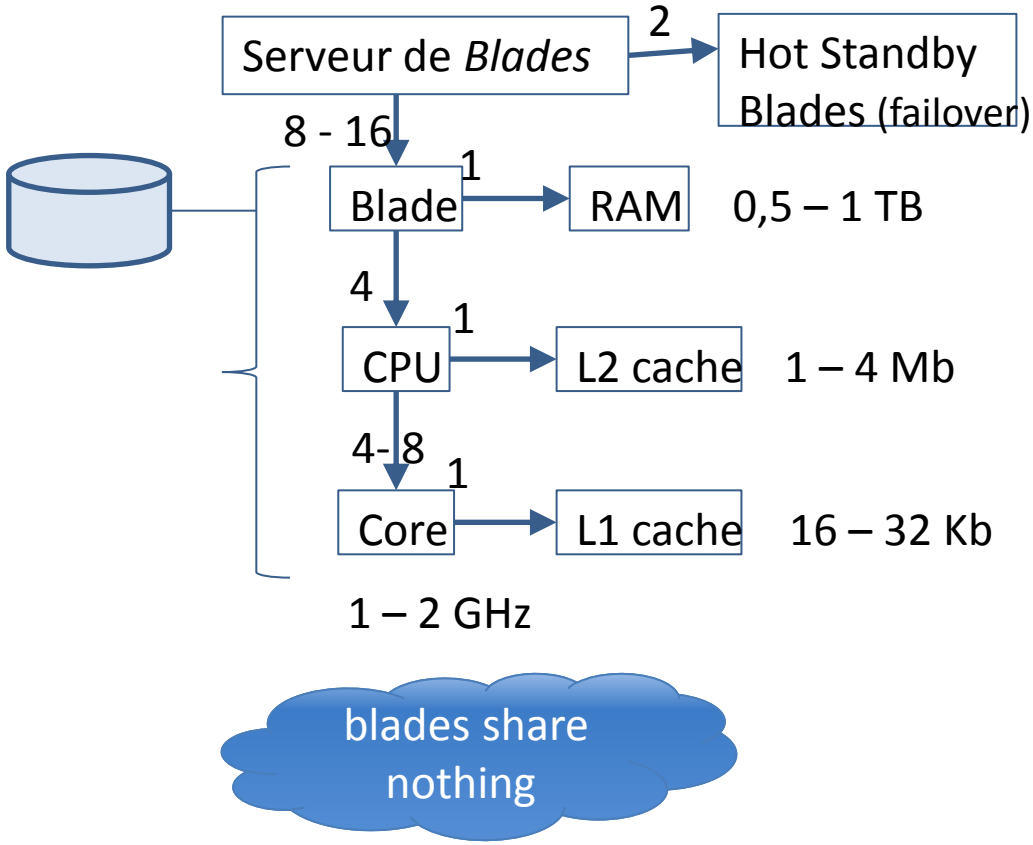
Serveur orienté mémoire vs. orienté disque

Au delà de 100 *cores*

Au delà de 10 Tb de mémoire

SGBD en mémoire devient attractif

- Algorithmique parallèle et stockage colonne permettent par exemple des gains très importants sur des requêtes de type décisionnel
- Il faut privilégier les accès séquentiels à la mémoire



Rappel: puissance de calcul et débit mémoire augmentent, la latence augmente très lentement

Parallèlisme massif, stockage colonne

Parallelisme

SGBD-R : Teradata
Neteeza(IBM)
DATAllegro (Microsoft)

Logiciel libre, NoSQL : Hadoop MR

(dans 1 minute…)

Stockage colonne

Sybase IQ
Kickfire (Teradata)
Logiciel Libre: MonetDB

Parallelisme & stockage colonne

Exasol
Vertica
Greenplum (EMC)
Logiciel libre, NoSQL: Hadoop HBase

Les nouvelles architectures pour la gestion de données

MapReduce

MapReduce

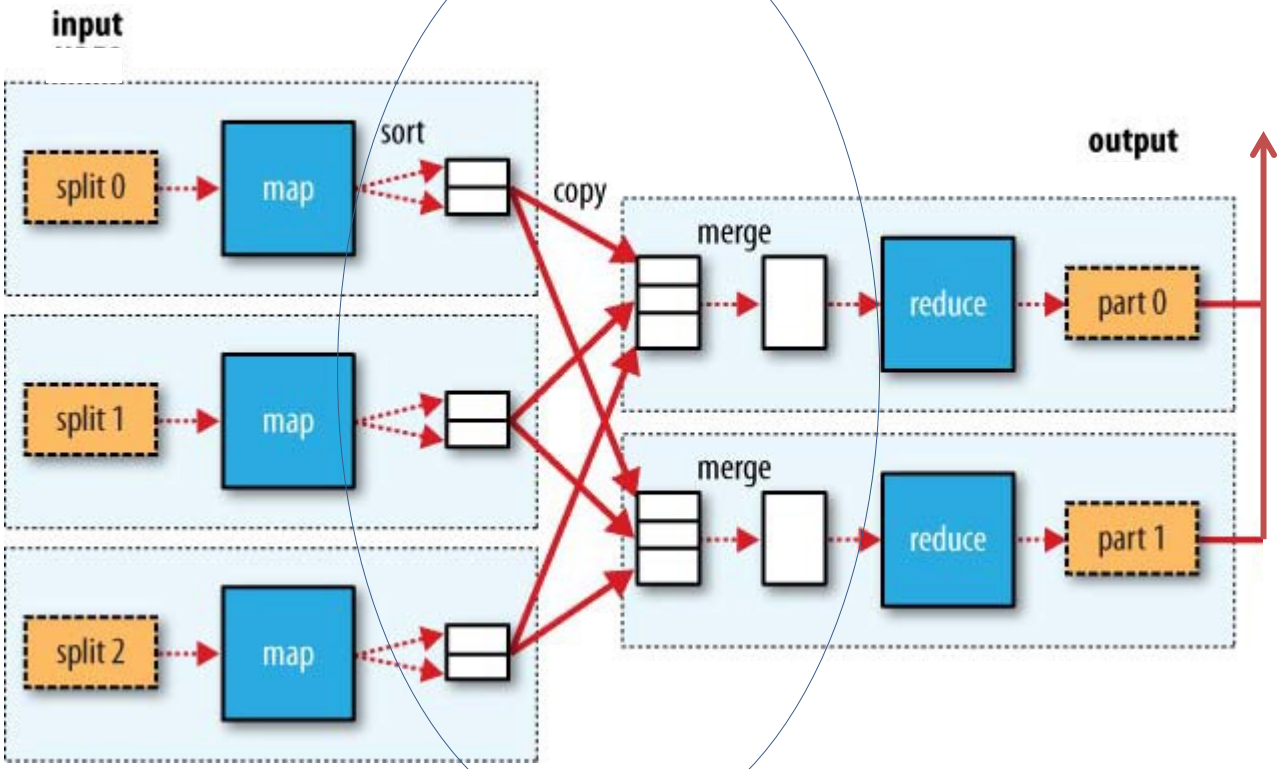
MapReduce : un modèle de calcul basé sur une distribution massive qui permet l'exécution efficace de traitements sur de gros volumes de données

- 2004 : publication Google
- 2006: implémentation open source : Hadoop

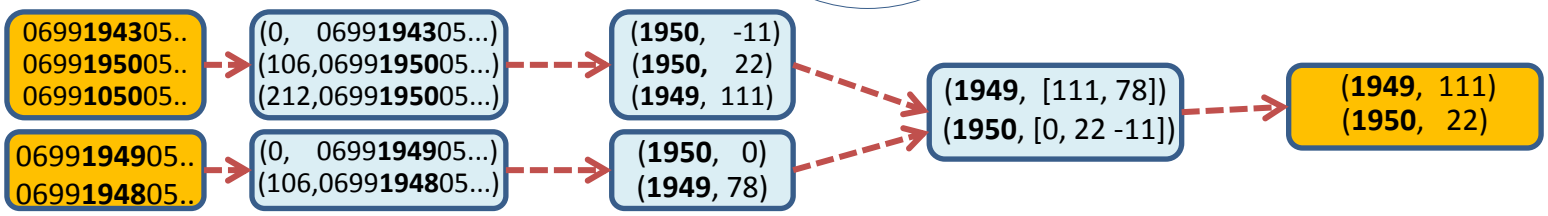
Principes

- Données distribuées sur un grand nombre de machines (*shared nothing*)
- On pousse le traitement près des données, exécution parallèle
- L'environnement est en charge de la distribution et de la tolérance aux pannes
- Traitement basé sur 2 fonctions : *map*, *reduce*

MapReduce: Flux de Données



Input | **map** | ***Shuffle*** | **reduce** > **Output**



Map et Reduce

Map: (K, V) → list (K', V') ; typiquement :

- Filtre, sélectionne une (nouvelle) clé, projette/transforme
- Partitionne ses résultats sur M fichiers, pour M réducteurs

Shuffle: list (K', V') → list (K', list (V'))

- Système groupe les paires intermédiaires ayant la même clé

Reduce: (K', list (V')) → list (K'', V'') ; typiquement :

- Agrège (COUNT, SUM, MAX)
- Concatène, combine et/ou filtre (p.ex., jointure « interne »)

Optimisation optionnelle: *combine: list(V') → V'*

- Exécutée sur le nœud mappeur pour combiner des paires avec la même clé en un seul pair (réduit trafic réseau)

Caractéristiques

Pas de modèle de données, pas de schéma

- Données typiquement semi-structurés et variables (p.ex. *web logs*)
- Traitement tient compte des irrégularités des données

Langages de plus haut niveau et schéma construits au-dessus (voir Hadoop)

Pas de mises à jour, encore moins des transactions (au sens ACID)

Traitement par lots (*batch*)

- pas de *reduce* avant que les *map* aient fini, pas d'indexes

Tuning Complexe

Modèle de calcul indépendant du langage de programmation

Peut marcher sur plusieurs systèmes de stockage

- HDFS Hadoop, Amazon S3, MongoDB, AsterData, etc

Hadoop

Implémentation open source Apache, écrit en Java

Yahoo a été le contributeur principal

Composants

- Hadoop filesystem (HDFS)
- MapReduce (MR, sur cluster HDFS et fichier local)
- Pig (langage dataflow pour l'exploration de données, sur HDFS et MR)
- Hive (JDBC, interface de type SQL pour *warehousing* sur HDFS et MR)
- HBase (SGBD clé-valeur à stockage orienté colonne, NoSQL, sur HDFS)
- Zookeeper (service de coordination pour des applications distribuées)

APIs pour Java et C++, streaming API pour tout autre langage

Plusieurs distributions

- Cloudera, Microsoft, IBM, Oracle, Greenplum, AmazonEMR,Hortonworks

Pig Latin

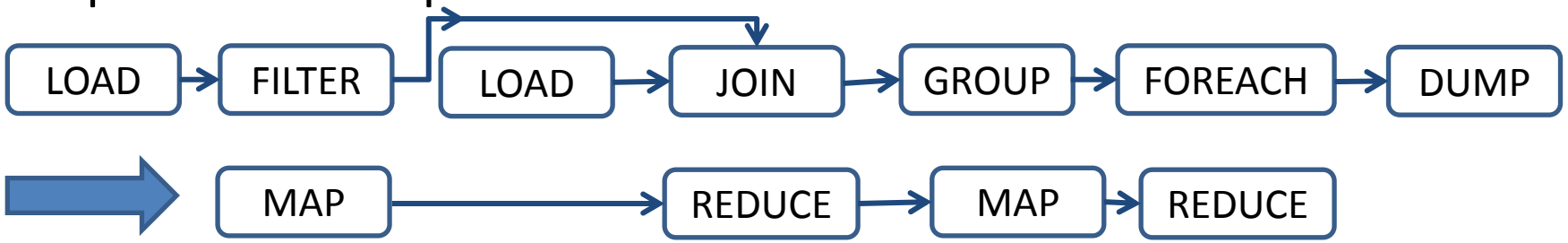
Modèle relationnel N1NF

LOAD – (transformation)* – STORE | DUMP

Exemple

```
Livres = LOAD'livre.txt' AS (titre: chararray, auteur: chararray,…);
Abiteboul = FILTER Livres BY auteur == 'Serge Abiteboul';
Edits = LOAD 'editeurs.txt' AS(titre: chararray, editeur: chararray);
jointure = JOIN Abiteboul BY titre, Edits BY titre;
groupe = GROUP jointure BY Abiteboul::author;
compteur = FOREACH groupe GENERATE group, COUNT(jointure.editeur);
DUMP compteur
```

Compilation en MapReduce



Utilisations et proposition de valeur

Proposition de valeur

- Réel passage à l'échelle en termes de volume
  - Utilisation batch plutôt que temps réel
  - Traitement nécessitant un gros volume des données
- Robustesse vis-à-vis des pannes
- Modèle de traitement supportant des langages de haut niveau

Types d'utilisation aujourd'hui (sources: TDWI, Gartner)

- Marketing et gestion de clients via les web logs / media sociaux
- Meilleure perspicacité dans les affaires (*business insights*)

Place de Systèmes de type MapReduce

Dans l'entreprise:

- L'entrepôt lui-même
  - Tous les distributeurs de Hadoop
- Gestionnaire de données pour des outils BI
  - Vendeurs de BI (IBM, MSFT, SAP, Pentaho, Tableau…)
- Outil de support d'un ETL à destination d'un entrepôt
  - Informatica, Talend, Oracle, Microsoft, SAP …

Dans la technologie des bases de données

- Fait partie d'un *entrepôt virtuel* incluant d'autres SGBD
- Technologie intégrée dans un SGBD (servant d'entrepôt)

Conclusion

Leçons

- Pour des requêtes complexes, une structure riche est essentielle pour un bon temps de réponse
  - OLAP > relationnel > MapReduce
- Pour des données massives, le parallélisme et une structure simple favorise un bon temps de réponse
  - Clé/valeur > relationnel ou OLAP
- Pour le transactionnel, le parallélisme coûte cher
  - MapReduce n'est pas adapté
- Pour des données de structure flexible et complexes, XML et/ou objets sont bien adaptés

Tendances

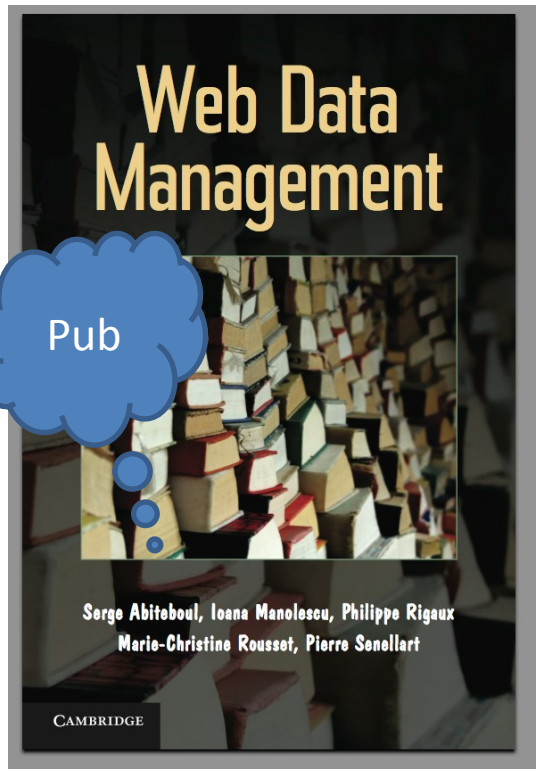
- La toile, le cloud, le parallélisme massif, les SGBD en mémoire, le logiciel libre
- De plus en plus de publication de données interconnectées entre elles
  - Utilisation de référentiels communs (p.ex. ontologies)
- De plus en plus d'applications impliquant des données d'organisations indépendantes
  - Intégration réelle (entrepôt) ou virtuelle (fédération) de bases de données disponibles sur le réseau à des fins d'interrogation et d'analyse
  - Workflow/choréographie sur des données distribuées avec aspects transactionnels à des fins opérationnelles multi-entreprises

Challenge

Construire la prochaine génération de systèmes de gestion de données

qui répondront aux exigences des applications extrêmes et

qui ne sacrifieront aucun des trois grands principes.


Web Data Management
Pub
Serge Abiteboul, Ioana Manolescu, Philippe Rigaux
Marie-Christine Rousset, Pierre Senellart
CAMBRIDGE


Merci !
FRAPAR.


Merci !
FRAPAR.