NFE204 – Bases de données avancées

Bases de données documentaires et NOSQL

Philippe Rigaux, Nicolas Travers

Conservatoire National des Arts et Métiers

October 8, 2013

Outline

Retour sur la notion de “base de données”

Par définition, une base de données c’est :

1. Un ensemble **structuré** d’informations,
2. stocké de manière **persistante** pour être préservé à long terme.

Indissociable d’un système informatique (le SGBD) qui assure des services

1. D’accès aux fichiers (sur un disque)
2. De recherche
3. de sécurisation
4. de concurrence d’accès,
5. et beaucoup d’autres...

Bases relationnelles

Les **bases relationnelles** sont utilisées dans toutes les applications gérant des informations **structurées** et **régulières**.

- applications de “gestion”,
- applications Web type commerce électronique,
- applications mobiles.

Les SGBD relationnels apportent de très nombreuses fonctionnalités: stockage, interrogation SQL, concurrence d’accès, reprise sur panne, droits d’accès, optimisation, environnement de développement, etc.

Ces aspects sont supposés maîtrisés ici ! Sinon révisions nécessaires.

Problématique 1 : La notion de “document”

Document = unité d’information autonome ou quasi-autonome.

- Peu ou pas de référence à d’autres documents.
- Peu ou pas de structure; ou une structure très flexible.
- Un contenu souvent à orientation multimédia.

Examples (1): documents textuels, types documents Web.

Examples (2): images, documents audios, des vidéos ; pas de structure explicite, production de descripteurs synthétiques pour tenter de les indexer.

Examples (3): jeux en ligne : artifacts graphiques, objets 3D, actions utilisateur.

Défi

Impose de repenser la notion de schéma et de représentation.

Problématique 2 : données à très grande échelle

On atteint facilement des **volumes** de données extrêmement importants.

- les moteurs de recherche qui collectent des **documents** disponibles sur le Web.
- les applications utilisées à l'échelle du Web ; commerce électronique (Amazon); réseaux sociaux (Facebook).
- données gérées par les jeux en ligne.

Les collections occupent typiquement des centaines de Gigaoctets, voire des Téraoctets.

Solution

Nouveaux systèmes, dits "NoSQL" pour gérer de vastes collections de documents et **passer à l'échelle**.

Problématique 3 : la recherche

Dans les bases documentaires, peu ou pas de structure fixe, la recherche s'effectue souvent **par similarité**

- on fournit un document "requête"
- le système recherche les documents **proches** du document-requête.
- implique une notion de distance, et donc un **classement** du résultat.

Par exemple, quand on recherche sur le Web:

- on fournit un ensemble de mots-clés: c'est le document requête
- le moteur de recherche trouve les documents les plus proches (on verra comment)
- le classement (et sa pertinence) sont des éléments essentiels.

Contenu du cours

Le cours couvre les aspects essentiels de la gestion de grandes bases documentaires.

On se limite aux documents “textuels” (pas de données multimédia).

Trois parties :

- Représentation : structuration de documents textuels ; annotations (méta données).
  Application: XML, JSON
- Stockage, gestion : les systèmes NoSQL.
  Application: MongoDB, Hadoop, et autres.
- Recherche : moteurs de recherches, index, algorithmes.
  Application: Lucene, SolR

On reprend pas à pas...

Outline

Représentation de documents textuels

Modèle de données basé sur des **graphes**, permettant de représenter des données régulières ou irrégulières.

Idées de base

Les données sont auto-décrites. Le contenu vient avec sa propre description.

Structures riches. Le contenu se décrit avec des listes, des enregistrements imbriqués, des ensembles.

Typage flexible. Les données peuvent être typées (“c’est un entier”), et/ou structurées (“cette partie du graphe **doit** avoir telle forme”), et/ou ni l’un ni l’autre. Dans ce dernier cas c’est à l’application d’effectuer le contrôle.

Sérialisation. Structure **et** contenu doivent pouvoir être transformés ensemble en une chaîne de caractères autonome.

Exemple avec JSON

Point de départ : **listes associatives**, i.e., des enregistrements avec des paires (clé, valeur).

```
{nom: "Alan", tél: 2157786,  email: "agb@abc.com"}
```

Extension naturelle : les valeurs sont elles-mêmes d'autres structures.

```
{nom: {prénom: "Alan",  famille: "Black"},
      tél: 2157786,
      email: "agb@abc.com"}
```

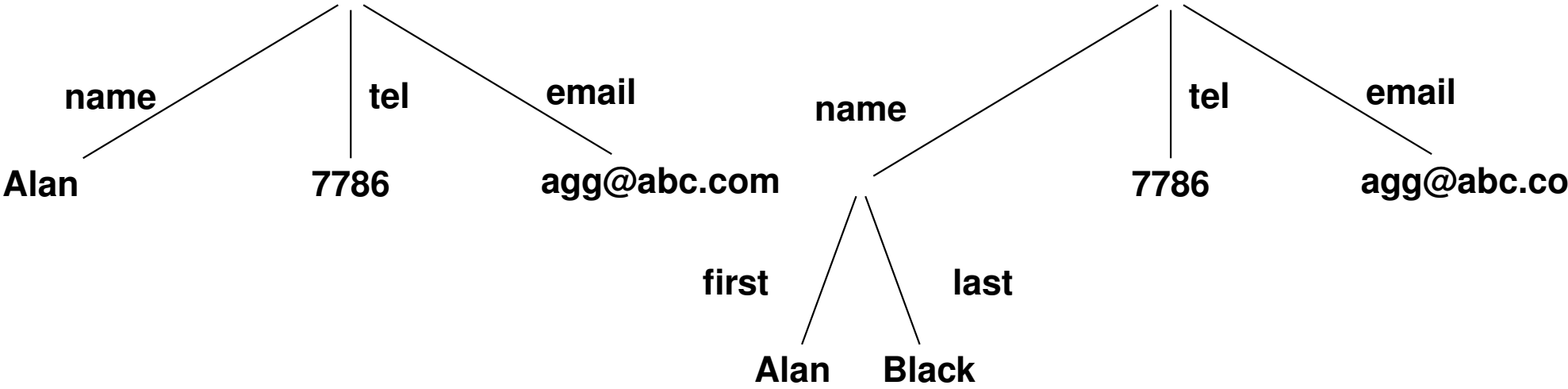
Autre exemple: imbrication de listes.

```
    {name: "Alan", téls: [2157786, 2498762] }
```

NB : c'est la version sérialisée, un chaîne stockable sur disque.

Vue conceptuelle: modélisation sous forme arborescente

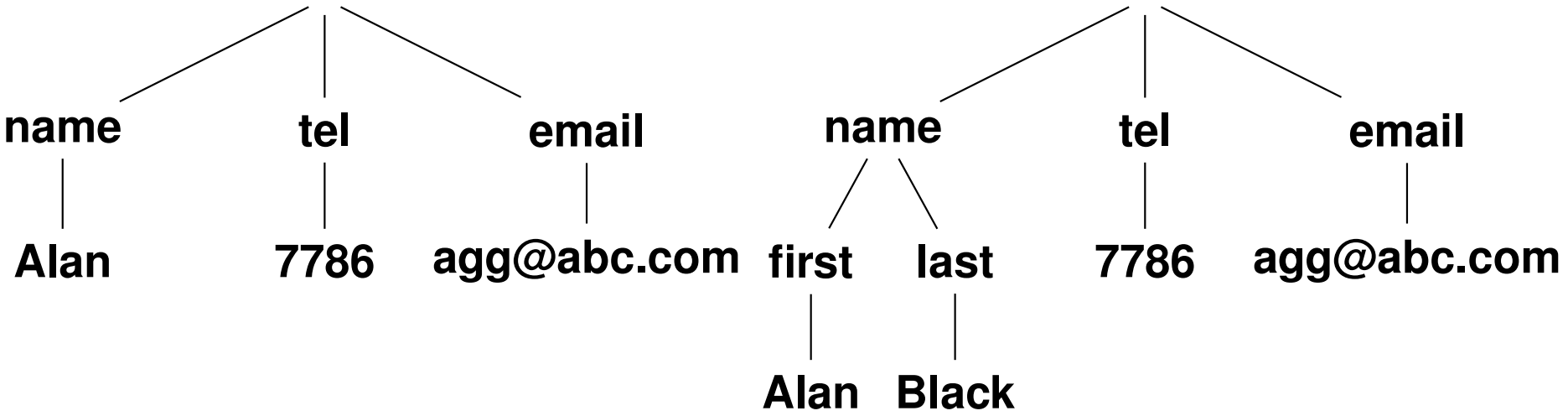
Données représentées par des arbres. Les étiquettes sont sur les arêtes, les valeurs sur les feuilles.



Base du **raisonnement** pour opérations (de recherche, de navigation, ...)

Autre possibilité : avec nœuds-étiquettes

On peut représenter à la fois les étiquettes et les valeurs comme des nœuds.



Remark

Choix de représentation du langage XML.

Représentation de données régulières

Facile de représenter des données régulières:

```
{ person:  {name: "alan", phone: 3127786, email: "alan@abc.com"},
  person:  {name: "sara", phone: 2136877, email: "sara@xyz.edu"},
  person:  {name: "fred", phone: 7786312, email: "fd@ac.uk"} }
```

Remark

1. on peut représenter des données régulières
2. probablement pas du tout efficace dans ce dernier cas.

$\Rightarrow$ peut être utile pour échanger des informations ; pas pour les stocker nativement.

Flexibilité de la représentation

Quel degré de varation accepte-t-on dans une base documentaire (données manquantes, doublons, structures variables, etc.)?

Exemple extrême,

```
{person: {name: "alan", phone: 3127786, email: "agg@abc.com"},
 person: &314
         {name: {first: "Sara", last: "Green"                  },
          phone: 2136877,
          email: "sara@math.xyz.edu",
          spouse: &443                                         },
 person: &443
         {name: "fred", Phone: 7786312,  Height: 183,
          spouse: &314                                         }}
```

Identité des nœuds

En se donnant le moyen d'identitifer des nœuds, on peut y faire référence, et décrire des cycles et des modèles objet.

XML: données “semi-structurées”

XML est d’abord un standard du World-Wide-Web Consortium (W3C).

- La sérialisation de documents XML est normalisée, ce qui permet des échanges réseau sans perte d’information.
- XML est un format générique, qui se spécialise ensuite en **“dialectes”** pour des domaines spécificque (e.g., XHTML, affichage dans un navigateur).
- Le W3C a défini des standards associés: DOM (le “modèle”), XSchema (le typage), XPath (navigation), XSLT (restructuration), XQuery (requêtes), et beaucoup d’autres.

Remark

1. XML est une version simplifiée de **SGML**.
2. HTML, jusqu’à la version 4.0, est **aussi** une variante de SGML. XHTML est un dialecte XML.

Documents XML: modèle et sérialisation

Modèle Un document XML est un arbre, chaque nœud a un type particulier.

Sérialisation tout document XML peut être transformé en une chaîne de caractères représentant l'arobrescence du document. each node.

La norme XML définit une **syntaxe**. Aucune interprétation n'est définie à priori.

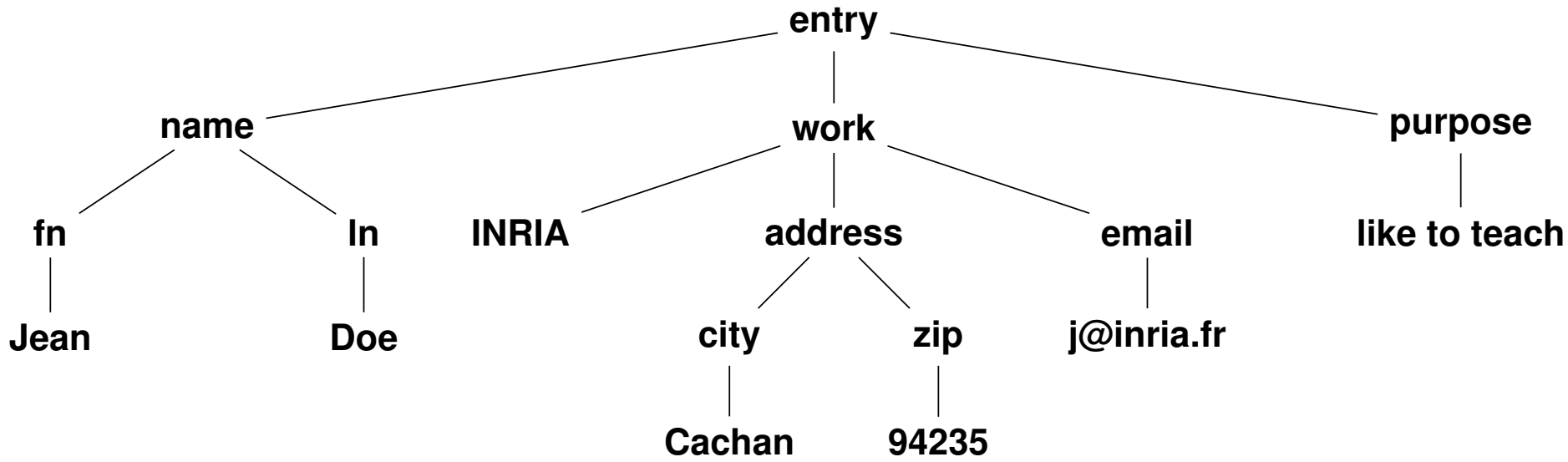
Les dialectes XML imposent des contraintes, et définissent une interprétation.

Le dialecte XHTML

Un document XHTML est un document XML. Il obéit à des contraintes de structure, et s'interprète sous forme de règles d'affichage à l'écran.

Modèle: un document est un arbre

Une application manipule un document XML comme un arbre.



Exemple: interfaces de programmation (DOM); langages de navigation (XPath), restructuration (XSLT), recherche (XQuery)

Remark

Une exception: l'API SAX s'applique à la forme sérialisée d'un document XML.

Forme sérialisée

Exemple :

```
<entry><name><fn>Jean</fn><ln>Doe</ln></name>INRIA<adress><city>
Cachan</city><zip>94235</zip></adress><email>j@inria.fr</email>
</job><purpose>like to teach</purpose></entry>
```

avec une petite mise en forme :

```
<entry>
   <name>
      <fn>Jean</fn>
      <ln>Doe</ln> </name>
   <work>
      INRIA
      <adress>
         <city>Cachan</city>
         <zip>94235</zip> </adress>
      <email>j@inria.fr</email> </work>
   <purpose>like to teach</purpose>
</entry>
```

XML permet de structurer des documents textuels

Un document textuel difficile à interpréter pour un logiciel.

```
The book ``Fundations of Databases'', written by Serge Abiteboul,
Rick Hull and Victor Vianu, published in 1995 by Addison-Wesley
```

Avec structuration XML :

```
<bibliography>
  <book>
    <title> Foundations of Databases </title>
    <author> Abiteboul </author>
    <author> Hull </author>
    <author> Vianu </author>
    <publisher> Addison Wesley </publisher>
    <year> 1995 </year> </book>
  <book>...</book>
</bibliography>
```

Une application peut construire l'arbre XML, en extraire des parties, renommer, restructurer, recherche selon certains critères, etc.

Qui utilise XML?

XML est LE standard pour la représentation de données textuelles.

Exemples : XHTML, DocBook, RSS, ...

XML est AUSSI utilisé pour sérialiser des données applicatives, par exemple pour les données gérées par vos applications personnelles.

Exemples : Traitements de texte (Word), tableurs (Excel), calendriers, graphiques (en SVG), etc.

Dans tous les cas

La représentation en XML permet de **réutiliser** le contenu, de l'échanger avec d'autres applications.

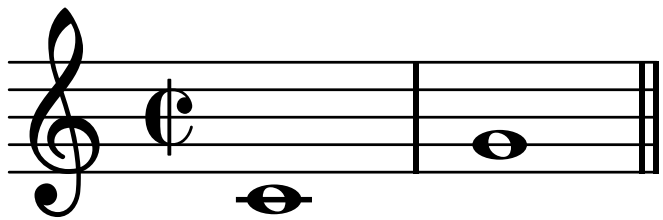
```
<?xml version="1.0" encoding="UTF-8" ?>

<rss version="2.0">
<channel>
<title>Webdam Project</title>
<atom:link href="http://webdam.inria.fr/wordpress/?feed=rss
            rel="self" type="application/rss+xml" />
<link>http://webdam.inria.fr/wordpress</link>
<description>Web Data Management</description>
<pubDate>Wed, 26 May 2010 09:30:54 +0000</pubDate>

  <item>
    <title>News for the beginning of the year</title>
    <description>...</description>
    <link>http://webdam.inria.fr/wordpress/?p=475</link>
    <pubDate>Fri, 15 Jan 2010 08:48:45 +0000</pubDate>
    <dc:creator>Serge</dc:creator>
    <category>News</category>
  </item>
</channel>
</rss>
```

Scalable Vector Graphics (SVG)

```
<?xml version="1.0" encoding="UTF-8" ?>
<svg xmlns="http://www.w3.org/2000/svg">

   <polygon points="0,0  50,0  25,50"
        style="stroke:#660000;"/>

  <text x="20" y="40">Some SVG text</text>
</svg>
```

Music XML

```
<?xml version="1.0" encoding="UTF-8"?>

<score-partwise version="2.0">
  <part id="P1">
      <attributes>
        <divisions>1</divisions>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
      </note>
  </part>
</score-partwise>
```



Un modèle issu de la sérialisation d'objets (Javascript, ou autre)

JSON (Javascript Object Notation) est le format de sérialisation des objets Javascript. Il est très utilisé dans le cadre des applications Ajax.

Avantage principal : il est possible de le parser directement sous la forme d'un objet du langage (javascript, mais aussi tous les langages de scripts, voire Java ou C++)

Simple, intuitif, léger : une alternative à XML en tant que modéle semi-structuré.

Modèle JSON

Une **valeur** est soit une chaîne de caractère, un numérique, un Booléen, un **objet** ou un **tableau**.

Exemple: "Joe", 12.3, true, 0, 1.

Un **objet** est une liste non-ordonnée de paires `(clé, valeur)`.

Exemple: `{"first_name": "Joe", "last_name": "Doe"`

Un **tableau** est une liste ordonnée de valeurs.

Exemple: `["doe@cnam.fr", "john.doe@cnam.fr", "doej@cnam.fr"]`

Structures imbriquées

```
var book = {
  title :  "Web Data Management",
  year : 2010,
  // Authors is an array
  authors : [  { "firstName" : "Serge",  // First author
                 "lastName"  : "Abiteboul"},
               { "firstName" : "Ioana",  // Second
                 "lastName"  : "Manolescu",
                  "email" : "ioana@inria.fr"}
             ],
 "publisher"         : "Cambridge University Press",
 "content"  : <binary representation>
}
```

Note: pas (encore) de **schéma** JSON, pas (encore) de langage de navigation ou de recherche.

Outline

une base de données de documents?

Question : si on a des milliers ou des millions de documents (textuels), comment les gérer?
Dans un système de fichiers? Infernal.

Avec un système de GED ? Pourquoi pas.

Avec un SGBD adapté?

- Basé sur XML : nombreuses extensions de bases relationnelles; quelques SGBD natifs (eXist).
- Basé sur JSON : en plein dévelopement (CouchDB, MongoDB)

$\Rightarrow$ demo.

Bases JSON

Exemple with mongoDB:

```
mongo> j = { name : "mongo" };
{"name" : "mongo"}
mongo> t = { x : 3 };
{ "x" : 3 }
mongo> db.things.save(j);
mongo> db.things.save(t);

mongo> db.things.find();
{ "_id" : ObjectId("4c2209f9"), "name" : "mongo" }
{ "_id" : ObjectId("KJHKd84b"), "x" : 3 }

mongo> db.things.find({name:"mongo"}).forEach(printjson);
{ "_id" : ObjectId("4c2209f9"), "name" : "mongo" }
```

. . . and CouchDB.

Gestion de données à grande échelle

Systèmes qui **abandonnent** certainent propriétés des SGBD relationnels.

- le langage d'interrogation
- le contrôle du schéma
- la concurrence d'accès.

Pourquoi ?. Pour la **flexibilité** et le **passage à l'échelle**.

Comment,?

- Par des techniques de distribution dans des fermes de serveurs
- Par une adaptation "élastique" à la charge et au volume (*Clouds*)

Les systèmes “No SQL”

Tendance récente (et fourre-tout), désignant des systèmes qui abandonnent le modèle relationnel pour :

- un modèle de données plus intuitif, plus flexible,
- exploitation de ce modèle de données pour distribuer plus facilement,
- abandon de certaines contraintes fortes des systèmes relationnels, et notamment la concurrence stricte.

Key-value stores

A l'extrême: la collection est vue comme un ensemble de paires $< key, value >$, et distribuée à très grande échelle dans un *Cloud* de serveurs.

Quelques systèmes représentatifs

Key-value stores: un objet sérialisé (XML, JSON, binaire, . . .) est associé à un identifiant et stocké dans un espace distribué.

Simple Storage Service (S3, Amazon EC2), Project Voldemort, CouchDB.

Web-scale storage systems: conçus pour gérer de très grandes masses de données

BigTable (Google), Hadoop (Open-Source + Yahoo!), Cassandra (Apache)

BD documenatires: bien adaptées à la gestion de documents et de données flexibles.

eXist (XML) et de nombreuses implantations XML des grands éditeurs (ORACLE, DB2, etc.); CouchDB et mongoDB (JSon).

Voir le site `no-sql.database.org`.

Outline

Qu'est-ce que la RI? (*Information Retrieval, IR*)

Déf.: "étant donnée une grande collection de documents, la recherche d'information consiste à identifier et classer ceux coorespondant à un besoin donné".

La recherche peut se faire :

- sur les méta-données (annotations des documents);
- sur le texte lui-même (*full text search*)

Différence essentielle avec SQL :

- SQL renvoie une réponse **exacte** constituée des objets qui satisfont des **critères**.
- En RI, on renvoie une liste **classée** des objets les plus pertinents (ou les plus proches) de la requête.

Recherche d'information, principes généraux

L'utilisateur soumet une **requête**. Ce peut être

- des mots-clés ou un document complet (recherche texte),
- une image, un son, tout objet (recherche multimédia)

Le système

- produit une **description** synthétique de la requête, d_q.
- **compare** cette description avec les descriptions des documents de la base.

Le résultat est donné par ordre décroissant de proximité avec d_q

Les moteurs de recherche

Systèmes permettant d'indexer des collections et d'effectuer des recherches d'information.

Nous étudierons :

- Lucene, un index pour recherche plein texte.
- SolR, un moteur de recherche complet.

Travaux pratiques organisés pour ces systèmes.

Outline

Organisation du cours

Organisation d'une séance :

- 1h30 à 2h de cours magistral sur le sujet du jour.
- 1h d'atelier sur préparation d'un projet, avec rapport et exposé final.

Quatorze séances de cours décomposées de la manière suivante :

- 6 séances sur modèles de données semi-structurés -> XML.
- 4 séances sur la gestion de données à grande échelle.
- 4 séances sur la recherche d'information.

Dernière séance: contrôle des connaissances + exposés.

Les projets

Objectif: vous prenez un sujet qui vous intéresse, en liaison avec les techniques avancées de gestion de bases de données, et vous préparez un rapport que vous présenterez.

Suggestions:

Documents structurés. Etudier un dialecte XML complet (DocBook?), étudier les évolutions des outils XML (...) ou JSON (schéma?).

Bases documentaires. Etudier les systèmes de GED (Alfresco?), étudier un des innombrables systèmes NoSQL; réaliser un petit projet.

Moteurs de recherche. Appliquer Lucene ou SolR à **vos** données ; étudier les moteurs de recherche alternatifs, ou pour d'autres types de documents.

Vous êtes invité(e) à proposer quelque chose ! **Et faites ce qui vous intéresse avant tout.**

Infos pratiques

Polycopié : sur le web.

- Site de P. Rigaux: `http://deptinfo.cnam.fr/~rigaux` pour ce qui est spécifique au cours.
- Site `http://www.bdpedia.fr` pour toutes les recources, code, polycopiés, etc.

The end for today

Le but : que vous appreniez des choses nouvelles, intéressantes, utiles, en remettant en perspective ce que vous savez déjà (ou ce que vous croyez déjà savoir!)

Merci