Gestion de Masse de Données (GMD)

Cours 3 : Utilisation des index et mapping

I. Accéder et extraire des données

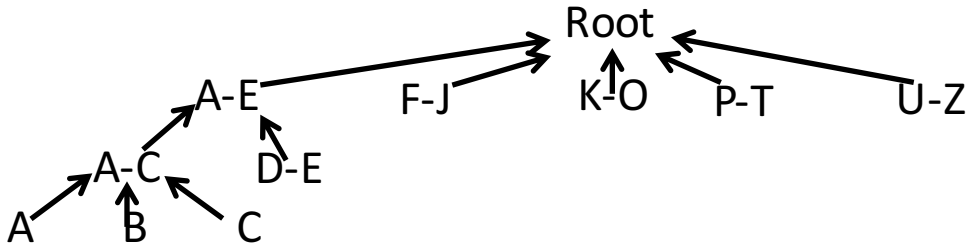
④ avec les index

a) index de SGBD

- un index
  - une structure de données
  - qui permet un accès plus rapide (sub-linear time, e.g. $O(\log_2 N)$
  - qu'une *recherche linéaire* = une examination un à un des n-uplets, $O(N)$
  - aux données d'une table

- Coûts
  - ralenti l'écriture
  - augmente la taille de la base

- Exemple : parcourir un arbre permet un accès plus rapide (**en moyenne**) que de lire les lignes 1 à 1



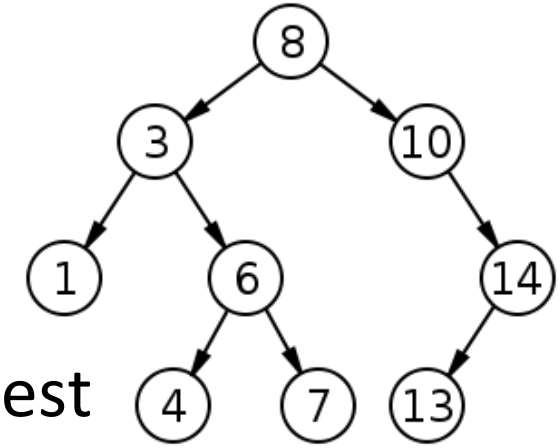
- un index = un ensemble de clées uniques,
  organisées dans une structure
  qui pointent vers des valeurs (i.e.
  qui sont associées soit à l'adresse
  d'un document, soit à l'adresse
  d'un block)

- Index BitMap
  - Principe : utilise des tableaux de bits (bitmap) pour faire référence à une valeur d'attribut
  - Exemple

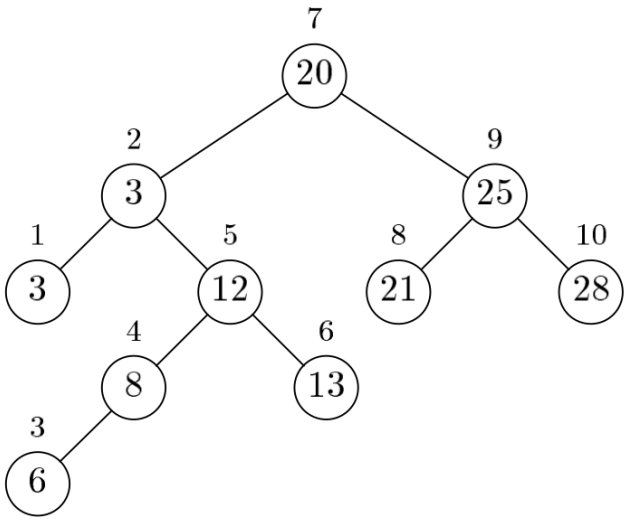
Identifier	Gender	Bitmaps	
		F	M
1	Female	1	0
2	Male	0	1
3	Male	0	1
4	Unspecified	0	0
5	Female	1	0

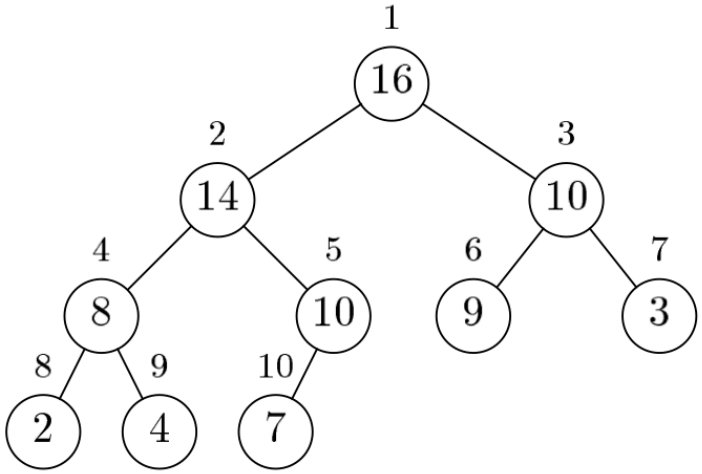
  - Adapté aux attributs qui ne prennent qu'un petit nombre de valeurs différentes, *ex : le genre*
  - Ne prend pas beaucoup d'espace mémoire

- Binary Search Tree (BST) / Arbre de recherche binaire
  - Nœud = une clé et une valeur
  - Arbre binaire
  - Pas de duplication
  - La valeur de la clé de chaque nœud est
    - > aux clés du SAG
    - < aux clés du SAD
  - Il existe un chemin unique de la racine à une valeur quelconque



- Arbre équilibré
  - contrainte et facteur d'équilibre : un arbre est équilibré si la différence entre la profondeur du SAG et du SAD <= 1 && >=-1





attention : ces 2 arbres ne sont ni des BST ni des BTree

  - Conserver l'équilibre doit limiter les problèmes si insertions, suppressions

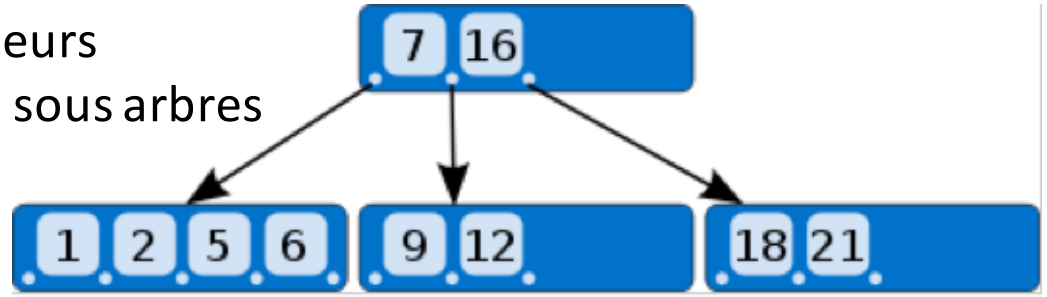
I. Accéder et extraire des données

④ avec les index

a) index de SGBD (1/3)

- Index BTree
  - le plus couramment utilisé
  - Btree ≠ Binary Tree ; étymologie inconnue, peut être *Balanced Tree*
  - Les nœuds peuvent avoir plus de 2 enfants
  - les valeurs de l'attribut indexé sont stockées dans un Btree pour accélérer leur recherche
  - Structure de l'arbre = arbre équilibré
  - Un nœud = une séquence de valeurs

séparées par des pointeurs vers les sous arbres



  - Principe de recherche :
    - récursif, départ à la racine

```
if valeur_recherchée == valeur_noeud
    nœud_recherché = nœud_courant
else if valeur_recherchée < valeur_noeud
    explore SAG
else
    explore SAD
```

    - ex1 recherche valeur 21 dans arbre ci contre
    - ex2 recherche de 12 dans arbre ci-contre

```
CREATE TABLE Patient (
  id INT PRIMARY KEY,
  ssid INT,
  INDEX USING BTREE (ssid),
  lastname VARCHAR
) ENGINE = InnoDB;
```

```
CREATE INDEX IDX_SSID_LASTN(
  on Patient(ssid,lastname));
```

```
SELECT title, author
FROM Amazone.livre
WHERE summary LIKE '%hobbit%'
```

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Apache **`Lucene`**
  - projet open source apache
  - librairie Java (au départ, maintenant porté à d'autres langages)
  - Création et utilisation d'index *full text* (càd d'index sur des attributs textuels)
- Notion de *Document* et de *Field*
  - *Document* est l'unité d'indexation et de recherche
  - Un *Document* est un ensemble de *Fields*
  - Chaque *Field* a un nom et une valeur textuelle
  - Un *Field* peut être indexé ou non (on peut faire une recherche sur ses valeurs)
  - Un *Field* peut être stocké (ou non) et ainsi retourné lorsqu'une recherche trouve le document associé
  - Un *Field* indexé peut être analysé par un *analyseur syntaxique*
  - Un document doit donc avoir au moins un *Field* indexé et un *Field* stocké

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Exemple
  - on a un fichier .csv avec des info sur des études cliniques
  - 81 MB, 103785 lignes (càd études cliniques)
  - Créer un index d'études cliniques

NCT ID	Title	Recruitment	Study Results	Conditions	Interventions	Sponsors	Gender	Age Groups
NCT0000020	Cocaine Effects in Humans	Completed	No Results Available	Cocaine-Rela	Drug: Methadone	National In	Both	Adult
NCT0000030	Feasibility Study of Take-H	Completed	No Results Available	Opioid-Relat	Drug: LAAM	National In	Both	Adult|Senior
NCT0000060	Vitamin E and C to Slow Pr	Terminated	No Results Available	Cardiovascul	Drug: Vitamin E|Drug: Vitamin C	National He	Both	Adult
NCT0000040	Alendronate and/or Parath	Completed	No Results Available	Osteoporosis	Drug: Human parathyroid hormone	National In	Both	Adult|Senior

  - on veut
    - stocker le champ *NCT ID* sans l'indexer
    - indexer et stocker le champ *Title*
    - Indexer sans stocker le champ *Interventions*

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Exemple (suite)

```
public class CreateClinicalTrialSimpleIndex {

  private CreateClinicalTrialSimpleIndex() {}

  static final File INDEX_DIR = new File("clinical_trial_index");

  /** Index all lines of a text file (path of the file is args[0]). */
  public static void main(String[] args) {

    if (INDEX_DIR.exists()) {
      System.out.println("Cannot save index to '" +INDEX_DIR+ "' directory, please delete it first");
      System.exit(1);
    }

    final File file = new File(args[0]);
    if (!file.exists() || !file.canRead()) {
      System.out.println("File '" +file.getAbsolutePath()+ "' does not exist or is not readable, please check the path");
      System.exit(1);
    }

    Date start = new Date();
    try {
      IndexWriter writer = new IndexWriter(FSDirectory.open(INDEX_DIR), new StandardAnalyzer(Version.LUCENE_30), true, IndexWriter.MaxFieldLength.LIMITED);
      System.out.println("Indexing to directory '" +INDEX_DIR+ "'...");
      indexDoc(writer, file);
      System.out.println("Optimizing...");
      writer.optimize();
      writer.close();

      Date end = new Date();
      System.out.println(end.getTime() - start.getTime() + " total milliseconds");

    } catch (IOException e) {
      System.out.println(" caught a " + e.getClass() +
       "\n with message: " + e.getMessage());
    }
  }
```

- Exemple (suite)

```
static void indexDoc(IndexWriter writer, File file) throws IOException {
  int eltCount = 0;
  if (file.canRead() && !file.isDirectory()) {
    // each line of the file is a new document
    try{
      InputStream       ips  = new FileInputStream(file);
      InputStreamReader ipsr = new InputStreamReader(ips);
      BufferedReader    br   = new BufferedReader(ipsr);
      String line;
      while ((line=br.readLine())!=null){
          String fields[] = line.split("\t");
          // make a new, empty document
          Document doc = new Document();
          //add 3 fields to it
          doc.add(new Field("id",     fields[1], Field.Store.YES, Field.Index.NO));
          doc.add(new Field("title",  fields[2], Field.Store.YES, Field.Index.ANALYZED));
          doc.add(new Field("interv", fields[6], Field.Store.NO,  Field.Index.ANALYZED));
          writer.addDocument(doc);
          eltCount++;
      }
      br.close();
    }catch (Exception e){
      System.out.println(e.toString());
    }
  }
  System.out.println(eltCount+" elts have been added to the index " + System.getProperty("user.dir")+ "/" + INDEX_DIR);
}
```

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Exemple (suite)

```
static void indexDoc(IndexWriter writer, File file) throws IOException {
  int eltCount = 0;
  if (file.canRead() && !file.isDirectory()) {
    // each line of the file is a new document
    try{
      InputStream       ips  = new FileInputStream(file);
      InputStreamReader ipsr = new InputStreamReader(ips);
      BufferedReader    br   = new BufferedReader(ipsr);
      String line;
      while ((line=br.readLine())!=null){
          String fields[] = line.split("\t");
          // make a new, empty document
          Document doc = new Document();
          //add 3 fields to it
          doc.add(new Field("id",     fields[1], Field.Store.YES, Field.Index.NO));
          doc.add(new Field("title",  fields[2], Field.Store.YES, Field.Index.ANALYZED));
          doc.add(new Field("interv", fields[6], Field.Store.NO,  Field.Index.ANALYZED));
          writer.addDocument(doc);
          eltCount++;
      }
      br.close();
    }catch (Exception e){
      System.out.println(e.toString());
    }
```

```
Indexing to directory 'clinical_trial_index'...
103785 elts have been added to the index /Users/coulet/workspace/EsialGMD/clinical_trial_index
Optimizing...
4548 total milliseconds
```

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Exemple (suite)

```
static void indexDoc(IndexWriter writer, File file) throws IOException {
  int eltCount = 0;
  if (file.canRead() && !file.isDirectory()) {
    // each line of the file is a new document
    try{
      InputStream       ips  = new FileInputStream(file);
      InputStreamReader ipsr = new InputStreamReader(ips);
      BufferedReader    br   = new BufferedReader(ipsr);
      String line;
      while ((line=br.readLine())!=null){
          String fields[] = line.split("\t");
          // make a new, empty document
          Document doc = new Document();
          //add 3 fields to it
          doc.add(new Field("id",     fields[1], Field.Store.YES, Field.Index.NO));
          doc.add(new Field("title",  fields[2], Field.Store.YES, Field.Index.ANALYZED));
          doc.add(new Field("interv", fields[6], Field.Store.NO,  Field.Index.ANALYZED));
          writer.addDocument(doc);
          eltCount++;
      }
      br.close();
    }catch (Exception e){
      System.out.println(e.toString());
    }
```

```
Indexing to directory 'clinical_trial_index'...
103785 elts have been added to the index /Users/coulet/workspace/EsialGMD/clinical_trial_index
Optimizing...
4548 total milliseconds
```

```java
public static void main(String[] args) throws Exception {
  String usage =
    "Usage:\tjava esial.gmd.SimpleSearch [-index dir] [-field f] [-query query]";
  if (args.length > 0 && ("-h".equals(args[0]) || "-help".equals(args[0]))) {
    System.out.println(usage);
    System.exit(0);
  }

  String index      = "clinical_trial_index";
  String field      = "title";
  String userQuery  = null;
  for (int i = 0; i < args.length; i++) {
    if ("-index".equals(args[i])) {
      index = args[i+1];i++;
    } else if ("-field".equals(args[i])) {
      field = args[i+1];i++;
    } else if ("-queries".equals(args[i])) {
      userQuery = args[i+1];i++;
    }
  }
  // only searching, so read-only=true
  IndexReader reader = IndexReader.open(FSDirectory.open(new File(index)), true);
  Searcher searcher  = new IndexSearcher(reader);
  Analyzer analyzer  = new StandardAnalyzer(Version.LUCENE_30);
  Query    query     = new QueryParser(Version.LUCENE_30, field, analyzer).parse(userQuery);

  System.out.println("Searching for: " + query.toString(field));
  int hitsPerPage = 10;// result is ordered with lucene scored then true
  TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
  searcher.search(query, collector);
  int numTotalHits = collector.getTotalHits();
  ScoreDoc[] results = collector.topDocs().scoreDocs;
  //display results
  System.out.println("Found " + numTotalHits + " hits.");
  for(int i=0;i<results.length;++i) {
    int docId = results[i].doc;
    Document d = searcher.doc(docId);
    System.out.println((i + 1) + ". " + d.get("id")+ ", title= " + d.get("title"));
  }
}
```

```java
public static void main(String[] args) throws Exception {
  String usage =
    "Usage:\tjava esial.gmd.SimpleSearch [-index dir] [-field f] [-query query]";
  if (args.length > 0 && ("-h".equals(args[0]) || "-help".equals(args[0]))) {
    System.out.println(usage);
    System.exit(0);
  }

  String index      = "clinical_trial_index";
  String field      = "title";
  String userQuery  = null;
  for (int i = 0; i < args.length; i++) {
    if ("-index".equals(args[i])) {
      index = args[i+1];i++;
    } else if ("-field".equals(args[i])) {
      field = args[i+1];i++;
    } else if ("-queries".equals(args[i])) {
      userQuery = args[i+1];i++;
    }
  }
  // only searching, so read-only=true
  IndexReader reader = IndexReader.open(FSDirectory.open(new File(index)), true);
  Searcher searcher  = new IndexSearcher(reader);
  Analyzer analyzer  = new StandardAnalyzer(Version.LUCENE_30);
  Query    query     = new QueryParser(Version.LUCENE_30, field, analyzer).parse(userQuery);
```

```
java ./SimpleSearch -query aspirin
```

```
Searching for: aspirin
Found 178 hits.
1. NCT01072604, title= Pharmacokinetic Study Comparing Aspirin and Aspirin Granules
2. NCT01081353, title= Pharmacokinetic Study Comparing Aspirin and Effervescent Aspirin
3. NCT00671021, title= Duration of Platelet Inhibition by Aspirin
4. NCT00004728, title= Aspirin Or Warfarin To Prevent Stroke
5. NCT01113528, title= Omega-3 and Aspirin in Periodontal Regeneration
6. NCT01102439, title= Clopidogrel/Aspirin Interaction Study
7. NCT01118546, title= Physiopathology of Rapid Aspirin Desensitization
8. NCT01113060, title= Aspirin Effectiveness Study
9. NCT00467363, title= The Effects of Aspirin in Gestation and Reproduction
10. NCT01039480, title= Aspirin and Clopidogrel Resistance Study
```

```java
public static void main(String[] args) throws Exception {
  String usage =
    "Usage:\tjava esial.gmd.SimpleSearch [-index dir] [-field f] [-query query]";
  if (args.length > 0 && ("-h".equals(args[0]) || "-help".equals(args[0]))) {
    System.out.println(usage);
    System.exit(0);
  }

  String index      = "clinical_trial_index";
  String field      = "title";
  String userQuery  = null;
  for (int i = 0; i < args.length; i++) {
    if ("-index".equals(args[i])) {
      index = args[i+1];i++;
    } else if ("-field".equals(args[i])) {
      field = args[i+1];i++;
    } else if ("-queries".equals(args[i])) {
      userQuery = args[i+1];i++;
    }
  }
  // only searching, so read-only=true
  IndexReader reader = IndexReader.open(FSDirectory.open(new File(index)), true);
  Searcher searcher  = new IndexSearcher(reader);
  Analyzer analyzer  = new StandardAnalyzer(Version.LUCENE_30);
  Query    query     = new QueryParser(Version.LUCENE_30, field, analyzer).parse(userQuery);

  System.out.println("Searching for: " + query.toString(field));
  int hitsPerPage = 10;// result is ordered with lucene scored then true
```

```
java ./SimpleSearch -query aspirin -field interv
```

```
Searching for: aspirin
Found 227 hits.
1. NCT01268917, title= The Effects of Preoperative Aspirin on Graft Patency and Cardiac Events in Off-pump Coronary Artery Bypass
2. NCT00725127, title= Chronotherapy With Low-dose Aspirin for Primary Prevention
3. NCT00812032, title= Evaluation of Antiplatelet Effects of Different Dosages of Aspirin in Type 2 Diabetic Patients
4. NCT00753935, title= Aspirin Resistance in Coronary Artery Disease
5. NCT00110448, title= Japanese Primary Prevention of Atherosclerosis With Aspirin for Diabetes (JPAD) Trial
6. NCT00761891, title= Validation of an Assay to Measure Cyclooxygenase-1 Activity
7. NCT00272311, title= Aspirin Dose and Atherosclerosis in Patients With Metabolic Syndrome
8. NCT00405613, title= Aspirin Use and Postoperative Bleeding From Dental Extractions in a Healthy Population
9. NCT00942617, title= Measurement of Platelet Dense Granule Release in Healthy Volunteers
10. NCT00578721, title= Trial of Aspirin and Arginine Restriction in Colorectal Cancer
```

```
public static void main(String[] args) throws Exception {
  String usage =
    "Usage:\tjava esial.gmd.SimpleSearch [-index dir] [-field f] [-query query]";
  if (args.length > 0 && ("-h".equals(args[0]) || "-help".equals(args[0]))) {
    System.out.println(usage);
    System.exit(0);
  }

  String index      = "clinical_trial_index";
  String field      = "title";
  String userQuery  = null;
  for (int i = 0; i < args.length; i++) {
    if ("-index".equals(args[i])) {
      index = args[i+1];i++;
    } else if ("-field".equals(args[i])) {
      field = args[i+1];i++;
    } else if ("-queries".equals(args[i])) {
      userQuery = args[i+1];i++;
    }
  }
  // only searching, so read-only=true
  IndexReader reader = IndexReader.open(FSDirectory.open(new File(index)), true);
  Searcher searcher  = new IndexSearcher(reader);
  Analyzer analyzer  = new StandardAnalyzer(Version.LUCENE_30);
  Query    query     = new QueryParser(Version.LUCENE_30, field, analyzer).parse(userQuery);

  System.ou
  int hitsP
  TopScoreD
  searcher.
  int numTo
  ScoreDoc[
  //display results
  System.out.println("Found " + numTotalHits + " hits.");
  for(int i=0;i<results.length;++i) {
    int docId = results[i].doc;
    Document d = searcher.doc(docId);
    System.out.println((i + 1) + ". " + d.get("id")+ ", title= " + d.get("title"));
  }
}
```

```
java ./SimpleSearch -query "aspirin AND effectiveness"
```

```
Searching for: +aspirin +effectiveness
Found 1 hits.
1. NCT01113060, title= Aspirin Effectiveness Study
```

```
public static void main(String[] args) throws Exception {
  String usage =
    "Usage:\tjava esial.gmd.SimpleSearch [-index dir] [-field f] [-query query]";
  if (args.length > 0 && ("-h".equals(args[0]) || "-help".equals(args[0]))) {
    System.out.println(usage);
    System.exit(0);
  }

  String index      = "clinical_trial_index";
  String field      = "title";
  String userQuery  = null;
  for (int i = 0; i < args.length; i++) {
    if ("-index".equals(args[i])) {
      index = args[i+1];i++;
    } else if ("-field".equals(args[i])) {
      field = args[i+1];i++;
    } else if ("-queries".equals(args[i])) {
      userQuery = args[i+1];i++;
    }
  }
  // only searching, so read-only=true
  IndexReader reader = IndexReader.open(FSDirectory.open(new File(index)), true);
  Searcher searcher  = new IndexSearcher(reader);
  Analyzer analyzer  = new StandardAnalyzer(Version.LUCENE_30);
  Query    query     = new QueryParser(Version.LUCENE_30, field, analyzer).parse(userQuery);

  System.ou
  int hitsP
  TopScoreD
  searcher.
  int numTo
  ScoreDoc[
  //display results
  System.out.println("Found " + numTotalHits + " hits.");
  for(int i=0;i<results.length;++i) {
    int docId = results[i].doc;
    Document d = searcher.doc(docId);
    System.out.println((i + 1) + ". " + d.get("id")+ ", title= " + d.get("title"));
  }
}
```

```
java ./SimpleSearch -query "aspirin AND effectiveness"
```

```
Searching for: +aspirin +effectiveness
Found 1 hits.
1. NCT01113060, title= Aspirin Effectiveness Study
```

I. Accéder et extraire des données

④ avec les index

b) index de texte : Lucene

- Java Doc :

http://lucene.apache.org/core/4_0_0/core/index.html

- Syntax de requètes :

http://lucene.apache.org/core/4_0_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html#package_description

ex1 : `title:"aspririn effectiveness" AND cancer`

ex2 : `te?t`

- Lucene est super puissant
- si vous avez compris, vous êtes super puissants

II. Transformer les données

① par rapport à un schéma global ou une ontologie

a) Notion de shémas local vs shéma global

- **Schéma local** : schéma d'une source de données considérée dans un processus d'intégration
- **Schéma global** : schéma construit pour un processus d'intégration qui permet de réconcilier différents schémas locaux

en pratique il s'agit

- soit du schéma d'un entrepôt de données ou
- soit du schéma de médiation

selon l'approche d'intégration choisie (cf. introduction du module)

- **Mises en correspondance** ou *mapping*

Définition de correspondances entre les entités (les tables/relations et les attributs par exemple) d'un schéma local et du schéma global

Ces correspondances doivent permettre la **transformation**

de données exprimées avec les termes du schéma local

=> en des données compatibles avec les termes du schéma global

La définition de ces mappings est la clé de voute d'un système d'intégration de données. Leur définition peut être obtenue manuellement ou de façon semi-automatique.

Exemple de mapping 1 / Linder et Petitjean

Nom du médicament

myDrugBankLexicon

Nom générique	Synonyme	Marque
Lepirudin	Hirudin variant-1	Refludan
Cetuximab	Anti EGFR;IMC-C225;cetuximab	Erbitux
Dornase Alfa	DNase;DNase I;Deoxyribonuclease I;Deoxyribonuclease-1	
Denileukin diftitox	DT;Diphtheria toxin precursor;NAD(+--diphthami	
Etanercept	CD120b;TNF-R2;Tumor necrosis factor receptor 2;Tumor n	
Bivalirudin	Not Available	Angiomax;Angiox

DrugBank

```
<drugbank-id>DB00001</drugbank-id>
<name>Lepirudin</name>
<description>Lepirudin is identical
</description>
<cas-number>120993-53-5</cas-number>
...
<targets>
<enzymes>
<transporters>
<transporter partner="1588" position="1">
<actions>
...
<partner id="1588">
<name>Multidrug resistance protein 1</name>
<general-function>Defense mechanisms and drug
<specific-function>Energy-dependent efflux pu
<gene-name>ABCB1</gene-name>
```

CTD

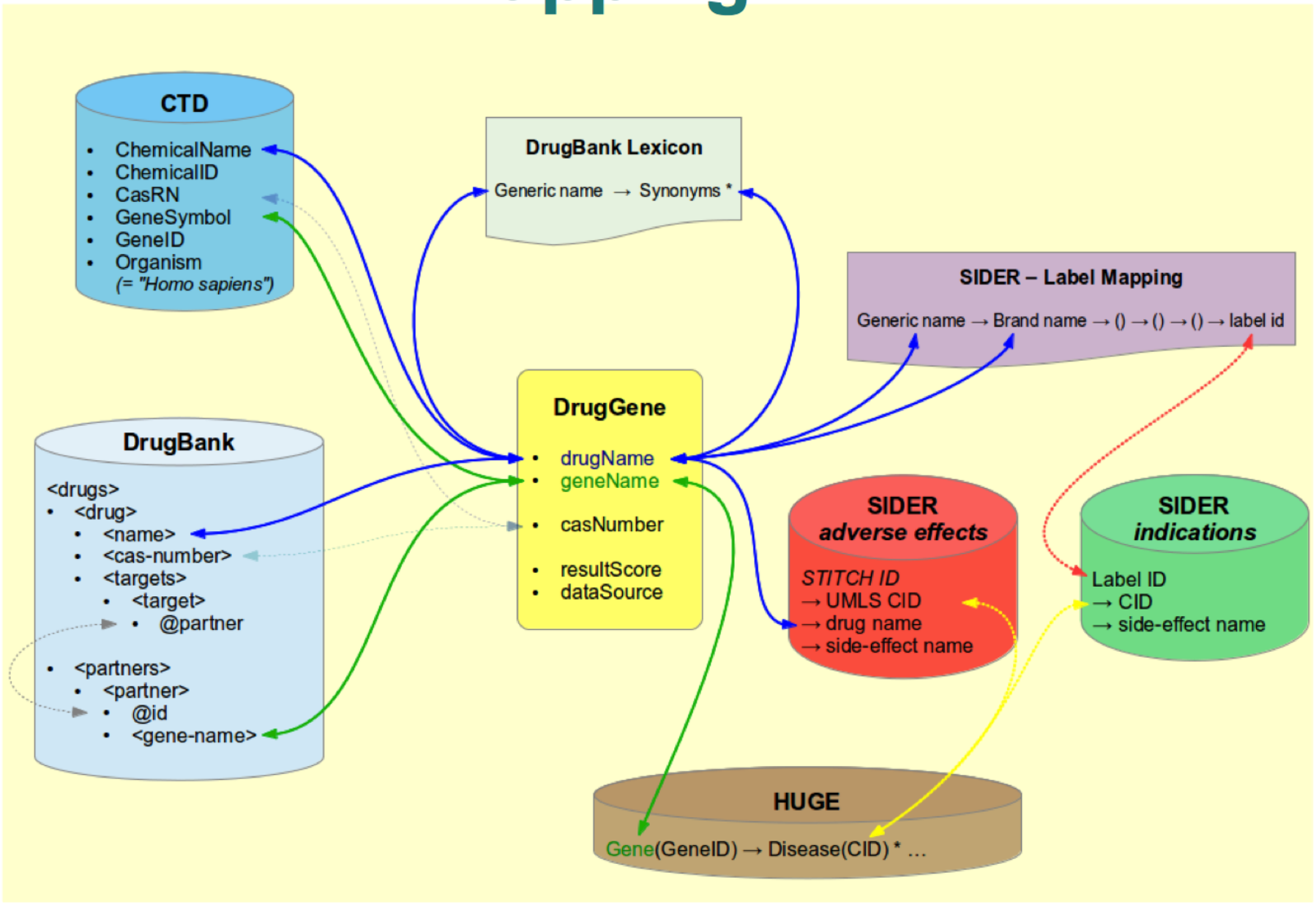
Select * from CTD_chem_gene
where ChemicalName = 'lepirudin'
AND Organism = 'Homo sapiens';

ChemicalName	ChemicalID	CasRN	GeneSymbol	GeneID	Organism	OrganismID
lepirudin	C083544		F2	2147	Homo sapiens	9606

LISTE 1

Exemple de mapping 2 / Siebert et Vitale

Mapping



II. Transformer les données

① par rapport à un schéma global ou une ontologie

b) Exemple

- Exemples de schémas locaux

Expedia_Flight(f_num, dep_date, dep_time, meal, seat) # seat is the nb of available seats

Opodo_Vol(num_vol, date_dep, h_dep, repas, places) #meal is a boolean

Orbitz_Flight(f_num, departure, meal) # departure is date+" "+time ; meal is an enum # {none, regular, regular+vegetarian}

Bing_Flight(f_number, dep_time, meal, seat) # seat is the position in the plane, eg 24 A

- Exemple de schéma global

N'importe lequel des précédents ou un intermédiaire pratique

Coulet_Flight(f_num,dep_date, dep_time, meal)

- Représentation des mappings : **pas de standard malheureusement**

$$\text{Coulet\_Flight.f\_num} = \text{Bing\_Flight.f\_number}$$
$$\text{Coulet\_Flight.dep\_date} = f(\text{Orbitz\_Flight.departure})$$
$$\text{Coulet\_Flight.dep\_time} = g(\text{Orbitz\_Flight.departure})$$
$$\text{Coulet\_Flight.meal} = h(\text{Orbizt\_Flight.meal})$$

II. Transformer les données

① par rapport à un schéma global ou une ontologie

c) Problèmes d'hétérogénéité

- Challenges pour la définition de mapping : toujours cette hétérogénéité de:
  - **syntaxe**

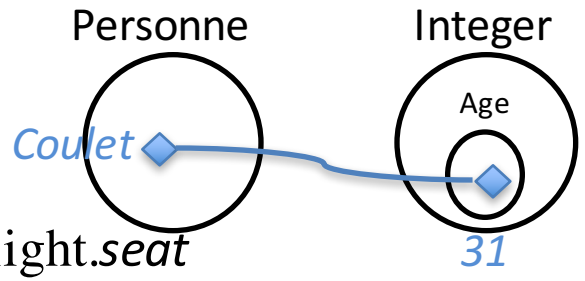
    Opodo_Flight(*AF84, 14/04/2011, 10h40, 1, 412*)

    Expedia_Flight("*AF 84*", *04-14-2011, 10:40am, true, 412*)
  - **structure** (ou format)

    Opodo_Flight(*AF84, 14/04/2011, 10h40, 1, 412*)

    Orbitz_Flight(*AF84*, "*14/04/2011 – 10h40*", "*regular+vegetarian*")
  - mode de **représentation** *ex : passage BD relationnelle vs représentation par objet*

nom	age
Coulet	31



  - **sémantique :** Bing_Flight.*seat* & Expedia_Flight.*seat*

    ex : *nombre de places disponibles vs. position du siège dans l'avion*
- Souvent plusieurs sources d'hétérogénéité en même temps
- Le tout automatique est déconseillé de façon unanime
- Pas de standard … mais … les techno du **Web sémantique**

II. Transformer les données

① par rapport à un schéma global ou une ontologie

d) Notion d'ontologie et d'alignement

- Ontologie
  - "Formalisation d'une conceptualisation" ie une représentation des connaissances

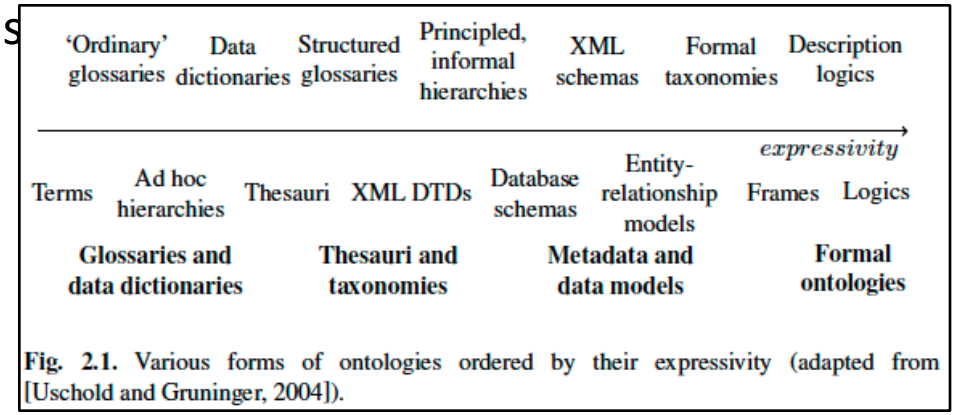
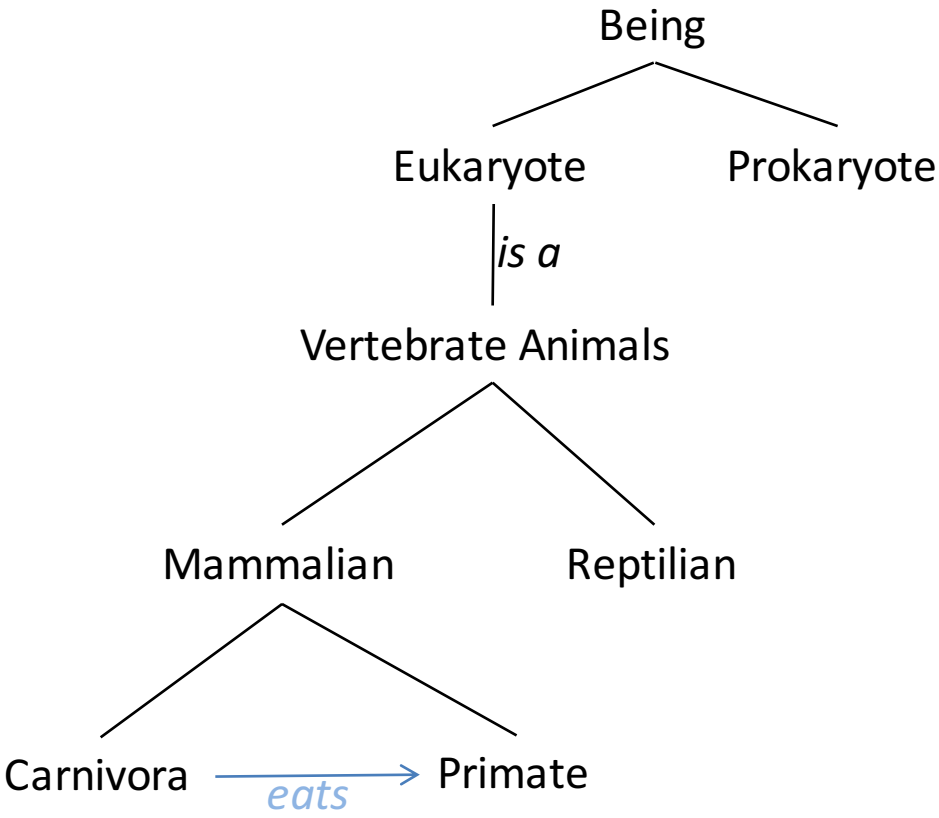
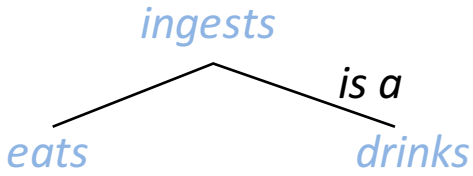


Fig. 2.1. Various forms of ontologies ordered by their expressivity (adapted from [Uschold and Gruninger, 2004]).

figure extraite de *Euzenat et Schvaiko :*Ontology Matching

  - Plus formellement $O = \{S_c, S_r, \mathcal{H}, \mathcal{A}\}$
  - Exemple : slide suivante
  - Peut être utilisée pour représenter un schéma local, un schéma global, des connaissances pour définir des mises en correspondances (alignement)
  - OWL est le langage standard soutenu par le W3C pour représenter des ontologies





(Axiome1) Carnivora $\sqsubseteq$ Mammlian

(Axiome2) Carnivora $\equiv \exists$ eats.Meat

II. Transformer les données

① par rapport à un schéma global ou une ontologie

d) Notion d'ontologie et d'alignement

- Alignement d'ontologies

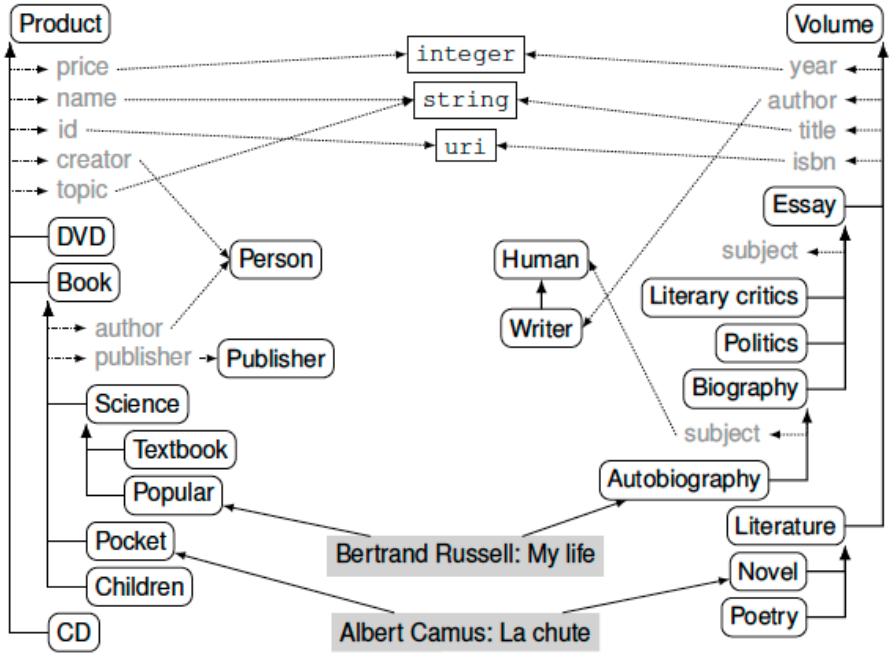


Fig. 2.7. Fragments of two ontologies.

figure extraite de *Euzenat et Schvaiko* :Ontology Matching

- Base du Web sémantique (Module PLBC de 3A, pour les IL)
  - standards : RDF, SPARQL, SKOS, OWL (*Description Logics embedded*)
- Très utilisées en intégration de données partagées sur le Web (ex : facebook)

II. Transformer les données

② Pour gérer les données manquantes ou bruitées

a) Données manquantes

- La **qualité** des données : impacts sur l'analyse, la fouille, etc.
- Identifier le type des données manquantes
  - données manquantes pour des variables dépendantes VS. indépendantes
  - apparition des données manquantes de façon aléatoire VS. non aléatoire
    - l'absence de valeur d'une variable indépendante est aléatoire
    - c'est rarement le cas en pratique : l'apparition de données manquantes est souvent associée à un phénomène particulière. *ex: les données sont manquantes en raison de : localisation géographique (ex US State hors des Etats Unis), éducation, âge, le manque d'information, etc.*
  - données manquantes de façon "voulue"
    - non demandé, non applicable (ex : sous contraceptif oral pour les hommes ? enceinte ?)
    - ce sont des cas particuliers de données manquante de façon non-aléatoire
  - proportion de données manquantes et répartition de ces données
    - quelle proportion ? 1%, 40%
    - quelle répartition ?
      - bcp de données manquantes pour peu de n-uplets ?

      ex 20 individus avec 40% de données manquantes et 2000 avec 8% de données manquantes
      - bcp de données manquantes pour certains attributs ?

      ex 2 attributs avec 80% de données manquantes et 10 avec 2% de données manquantes

II. Transformer les données

② Pour gérer les données manquantes ou bruitées

a) Données manquantes

- Méthodes pour traiter les données manquantes
  - Abandonner un attribut : dans le cas ou bcp de ses valeurs sont manquantes
    évidemment il faut espérer que cet attribut n'est pas trop important !
  - Abandonner des n-uplets
    réduit le nombre de cas (d'une étude par exemple)
    l'échantillon final peut ne pas être représentatif ou ne pas être significatif
  - créer une catégorie pour les données manquantes (Attributs nominaux *ONLY*)
    ex : la variable *gender* peut prendre les valeurs *male, female, unspecified*
  - Imputer une valeur aux valeurs manquantes
    ex : remplacer par la moyenne de l'ensemble des valeurs présentes
    ex : la moyenne d'un sous ensemble des valeurs (ex : homme/femme)
    ex : remplacer par une valeur attendue (si des règles *x->y si x->?* on peut remplacer *?* par *y*)
    ex : méthodes plus sophistiquées : les moindres carrés, régression, etc.

Il n'y pas de bonne méthode … ce sont des méthodes existantes qui sont utilisées…

Exemple

Case #	Y	X1	X2	X3
1	30	2	Missing	12
2	37	2	1	Missing
3	41	3	1	20
4	42	1	Missing	16
5	45	3	2	Missing
6	49	1	2	27
7	51	Missing	1	30
8	55	3	2	33
9	58	Missing	2	19
10	60	2	Missing	24

II. Transformer les données

② Pour gérer les données manquantes ou bruitées

b) Données bruitées

- … et les données bruitées ?
  - C'est également un problème à considérer surtout si vous avez des données
    - mesurées par un appareil
    - entrées par un humain : le taux d'erreur est à prendre en considération
  - Il existe des outils pour réduire le biais causé par le bruit
  - idées souvent similaires au traitement des données manquantes
  - à **vous** de voir si vous en avez besoin