RDBMS : what else?

Table de matières

- Approach NoSQL
- Données semi-structurées XML
- Web sémantique et données RDF

Relational database systems

- Developed in ‘70s
- Frequently unneeded complexity (i.e., ACID properties)
- Hardware has evolved, DBMS don’t
- No suitable for new large-scale applications

The value of RDBMS

- Getting persistent data
- Concurrency, recovery and consistency (ACID transactions)
- Shared database integration
- Mathematical background
- Standard query language
- Etc.

What's NoSQL (1/2)

- Not only SQL
- Term used first in 1998, then in 2009 in a NoSQL meeting at San Francisco
- Need to solve a problem that relational databases are bad fit for
- No to «one size fits all» thinking of traditional DBMS

What's NoSQL (1/2)

- The common characteristics of NoSQL databases
  - Not using the relational model
  - Running well on clusters
  - Open-source
  - Built for the 21st century web
  - Schema-less

- The most important result of the rise of NoSQL is **Polyglot Persistence**
  - Using different data stores in different circumstances

New data processing markets

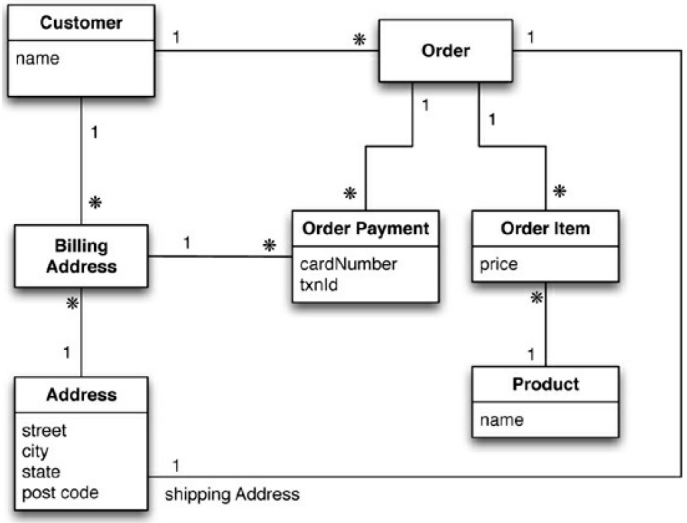
- Data warehouses
- Stream processing
- Text processing
- Scientific-oriented databases
- Semi-structured data

Aggregates vs Relations (1/2)

- The relational model
  - It is simple
  - Information is organized in tuples (rows)

- Aggregate
  - It is a collection of related objects
  - Can be complex things, e.g., lists, nested structures, heterogeneous records
  - It is a unit for data manipulation and management of consistency

A relational data model



Data instances

Customer

Id	Name
1	Martin

Orders

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

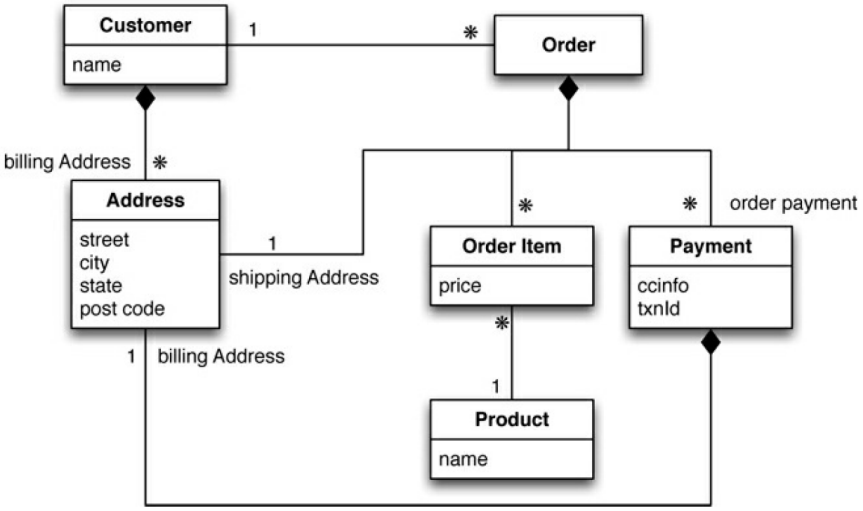
Address

Id	City
77	Chicago

OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

An aggregate data model (two aggregates)



Data in JSON

```
// in customers
{
"id":1,
"name":"Martin",
"billingAddress":[{"city":"Chicago"}]
}

// in orders
{
"id":99,
"customerId":1,
"orderItems":[
  {
  "productId":27,
  "price": 32.45,
  "productName": "NoSQL Distilled"
    }
  ],
"shippingAddress":[{"city":"Chicago"}]
"orderPayment":[
  {
    "ccinfo":"1000-1000-1000-1000",
    "txnId":"abelif879rft",
    "billingAddress": {"city": "Chicago"
  }
  ],
}
```

Aggregates vs Relations (2/2)

- The relational model
  - It is aggregate-ignorant
  - It is a better choice when there is no primary structure for manipulating data
  - Allows to manipulate any combination of rows from any tables in a single transaction

- Aggregates
  - Are hard to define if same data is used in different applications
  - Help with running on a cluster
  - Support atomic manipulation only in a single aggregate at a time
  - Transaction at inter-aggregate level are complicated and managed at application level

Types of NoSQL data stores (1/2)

- **Document stores.** The central concept of a document store is the notion of a "document" that encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, and JSON as well as binary forms like BSON.

- **Key-value stores.** In this model, data is represented as a collection of key-value pairs, such that each possible key appears at most once in the collection.The key-value model is one of the simplest non-trivial data models, and richer data models are often implemented on top of it. Key-value stores can use consistency models ranging from eventual consistency to serializability. Some support ordering of keys. Some maintain data in memory (RAM), while others employ solid-state drives or rotating disks

Types of NoSQL databases (2/2)

- Graph database. This kind of database is designed for data whose relations are well represented as a graph (elements interconnected with an undetermined number of relations between them). The kind of data could be social relations, public transport links, road maps or network topologies.

- Column-oriented database. It is a database management system (DBMS) that stores data tables as sections of columns of data rather than as rows of data. In comparison, most relational DBMSs store data in rows.

NoSQL implementations

- key-value stores
  - https://en.wikipedia.org/wiki/Key-value_database
- Document stores
  - http://en.wikipedia.org/wiki/Document-oriented_database
- Graph databases
  - http://en.wikipedia.org/wiki/Graph_database
- Column-oriented databases
  - https://en.wikipedia.org/wiki/Column-oriented_DBMS

XML

Objectifs de XML

- XML doit être facilement utilisable sur le Web
- XML doit supporter une grande variété d'applications
- Il doit être facile d'écrire des programmes qui traitent des documents XML
- Les documents XML doivent être lisibles et raisonnablement clairs
- La conception de XML doit être menée rapidement
- La description de XML doit être formelle et concise
- Les documents XML doivent être faciles à créer
- La concision du balisage XML est peu importante

Structure, contenu et présentation

- Trois aspects dans les documents
  - Le contenu
  - La structure logique
  - La présentation
- XML permet de représenter les contenus textuels et la structure logique
  - Les autres contenus sont des ressources externes (photos, vidéo, sons...)
  - La présentation est décrite par des moyens complémentaires (CSS, XSL)
  - La présentation peut changer, indépendamment des contenus et de la structure

Langage de balisage

- Le contenu est structuré en éléments qualifiés par des attributs avec des valeurs
- Chaque élément est représenté par une paire de balises et son contenu :

  <chapitre>...</chapitre>

- Les balises ouvrantes portent les attributs :

  <chapitre version="provisoire" date="16/06/03"/>

- L'imbrication et l'ordre des éléments reflètent la structure :

```
<ol xml:lang="fr">
    <li>Des balises décrivent la structure</li>
    <li>Structure arborescente</li>
</ol>
```

Contraintes syntaxiques

- Tout élément doit avoir une balise ouvrante et une balise fermante :

  <aaa>...</aaa>

- Raccourci d'écriture pour les éléments vides :

  <img src="image.png"></img> ou

- Les paires de balises ouvrantes/fermantes doivent être imbriquées :
  - correct
    <aaa>...<bbb>...</bbb>...</aaa>
  - Faux
    ....

- Un document possède une racine et une seule
- Tous les attributs doivent avoir une valeur
- Majuscules et minuscules sont différents

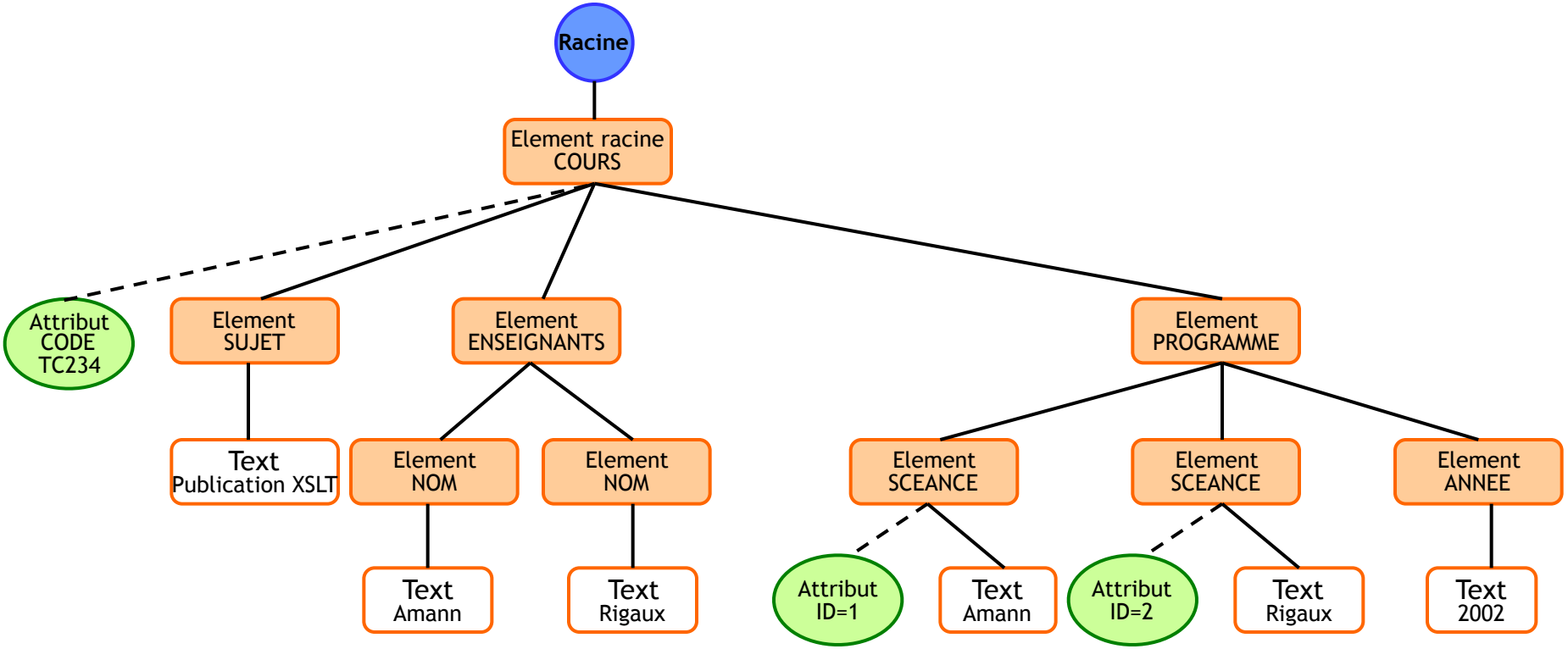
Un document XML

Fichier enseignants.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<COURS CODE="TC234">
    <SUJET>Publication XSLT</SUJET>
    <ENSEIGNANTS>
      <!-- Enseignant responsable -->
      <NOM>Amann</NOM>
      <NOM>Rigaux</NOM>
    </ENSEIGNANTS>
    <PROGRAMME>
      <SEANCE ID="1">Documents XML</SEANCE>
      <SEANCE ID="2">Programmation XSLT</SEANCE>
      <ANNEE>2002</ANNEE>
    </PROGRAMME>
</COURS>
```

L'arbre XPath du document XML



Structure et éléments

- Un document est essentiellement représenté en XML comme une structure logique arborescente
- Les éléments sont les constituants « logiques » du document
- Les utilisateurs peuvent définir leurs propres éléments :
  - Manuel, Titre, Auteur, Résumé, Chapitre, Section, Paragraphe, Note, Exemple, etc.
  - Pièce, Personnage, Acte, Scène, Réplique, etc.
  - LivreCuisine, Plat, Recette, Ingrédient, Temps, Préparation, Étape, etc.
  - etc.
- Les éléments ne sont pas prédéfinis, mais choisis en fonction du type de document à représenter
- Les éléments terminaux de la structure sont des éléments vides ou des chaînes de caractères
- L'ensemble de la structure est ordonné

Schéma

- Une Définition de Type de Document (DTD) spécifie
  - Les éléments utilisables dans les documents de ce type
  - Les noms et valeurs possibles des attributs utilisables
  - Quels attributs peuvent être associés à quels éléments
  - Les règles d'assemblage des éléments dans la structure
  - Les entités utilisables

- Une DTD définit les contraintes qui s'appliquent aux documents d'un type donné

Exemple de DTD

```
<!ELEMENT personne (nom, profession*)>
<!ELEMENT nom (prénom, nom_famille)>
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT nom_famille (#PCDATA)>
<!ELEMENT profession (#PCDATA)>
        <!ATTLIST profession
            depuis CDATA #IMPLIED
            enActivite (oui |no) #REQUIRED
        >
```

XMLSchema : autre manière de définir un schéma XML

Interrogation - XPath

- Langage d'interrogation des documents XML
- Permet de sélectionner parties d'un document XML : des sous arbres, des nœuds, des attributs, etc.
- Ce n’est pas un langage de la famille XML mais central dans le monde XML
- Intervient comme brique de base dans :
  - XML Schéma (expression des contraintes d'unicité et de clefs),
  - XSLT
  - XQuery
  - XLink
  - XPointer
  - etc.

Étape de localisation - XPath

- Nœud courant : c'est l'endroit dont on part.

  À partir de là, on considère trois éléments :
  - un axe : la direction vers laquelle on se dirige à partir du noeud courant (vers le père, vers les fils, vers les frères de gauche, etc.) ;
  - un filtre (déterminant, NodeTest) : le type de noeuds qui nous intéresse dans l'axe choisi (des noeuds quelconques, des éléments quelconques ou un élément précis, des commentaires, etc.) ; l'ensemble de nœuds qui ne répondent pas au critère indiqué sont éliminés
  - un prédicat optionnel : des conditions supplémentaires pour sélectionner des noeuds parmi ceux retenus par le filtre dans l'axe.

- Ces trois éléments constituent une *étape*
  - LocationStep = Axe, "::", Filtre, Predicat*

- L'enchaînement de plusieurs étapes constitue une *chemin XPath* :
  - axe1::filtre1[prédicat1]/axe2::filtre2[prédicat2]

    Exemple : parent::*/child::node()[position()=2]

Exemples

- child::*
- child::figure
- attribute::*
- child::text()
- child::comment()
- child::processing-instruction("play")
- child::node()

Exemples

- Avec prédicat
  - Utiliser l'indice de proximité avec la fonction position()
    - child::figure[position()=3]
    - child::figure[position()=last()]
  - Être constitué d'une étape de localisation
    - child::figure[attribute::type='gif']
  - Avoir la forme [node-set]
    - child::figure[attribute::scale]
    - child::figure[parent::paragraphe]
    - child::*[self::figure or self::image]
  - Avoir la forme [node-set=String]
    - child::figure[attribute::scale="0.5"]

Imbrication de données

- Les données relationnelles sont plates : lignes et colonnes
- Les données XML sont imbriquées et la profondeur peut être irrégulière et imprédictible
- Les relations peuvent représenter de données hiérarchiques grâce aux clés étrangères ou aux types de données structurées
- En XML il est naturel de chercher les objets à de niveaux inconnus dans la hiérarchie : "Trouve ce qui est rouge"
- XPath est un langage compact et bien adapté pour ce type de requête : //*[@color="Red"]

Métadonnées

- Les données relationnelles sont uniformes et répétitives
  - Tous le comptes de banque ont une structure similaire
  - Les métadonnées peuvent être factorisées dans un système de catalogue
- Les données XML sont très variables
  - Chaque page web est différente
  - Chaque objet XML a besoin d'une auto description
  - Les métadonnées sont réparties tout au long du document
  - Les requêtes peuvent accéder aussi bien aux métadonnées qu'aux données

    "trouve les éléments dont le nom est égal au contenu"

    //*[name(.)=string(.)]

Séquences hétérogènes

- Les requêtes relationnelles renvoient un ensemble uniforme de lignes
- Une requête XML peut renvoyer de types mixtes et de structures complexes
  - Ce qui est rouge : un drapeau, une cerise, une robe, etc.
  - Dans la réponse, les types « element » peuvent être mélangés aux valeurs atomiques
- Les requêtes XML doivent pouvoir faire de transformations structurelles

Ordonnancement

- Les lignes d'une relation ne sont pas ordonnées
  - Dans la réponse à une requête, n'importe quel ordre peut être dérivé à partir des valeurs des relations

- Les éléments dans un document XML sont ordonnés

Information manquante

- Les données relationnelles sont denses
  - Chaque ligne a une valeur dans chaque colonne
  - Une valeur nulle est nécessaire pour les données manquantes
- Les données XML peuvent être clairsemées
  - Les éléments manquantes ou inapplicables peuvent être vides ou tout simplement ne pas apparaître
  - Ceci donne à XML un degré de liberté qui n'est pas présent dans les bases de données relationnelles

XQuery

- Spécification W3C
- Langage permettant de
  - Sélectionner des éléments/attributs à partir de documents XML
  - Combiner de données à partir de multiples documents
  - Modifier les données
  - Calculer de nouvelles données
  - Rajouter de nouveaux éléments/attributs aux documents sortants
  - (Re)ordonner les résultats
  - Etc.

Exemple

- Livres publiés après 2000 ?

```
declare variable $bib := doc("bib.xml")//book;
declare function published-after($b, $y)
   {$b/@year > $y};
```

Prologue

```
<bib>
   {
   for $b in $bib
   where published-after($b, 2000)
   return
            <book year="{$b/@year}">
                     {$b/title}
            </book>
   }
</bib>
```

Corps

Web sémantique

et données RDF

En quoi ca consiste ?

- Le web « traditionnel »
  - Pages web liées entre elles
- Problème
  - Recherches comme : la sixième ville de France ?

  Ne marchent pas bien
  - On ne connaît pas la sémantique des liens entre les pages

Le web sémantique

- Vise à aider l'émergence de nouvelles connaissances en s'appuyant sur le web existant
- Web des données
  - Permet de lier et structurer l'information sur internet pour accéder simplement à la connaissance déjà existante
  - Les données sont décrites en RDF

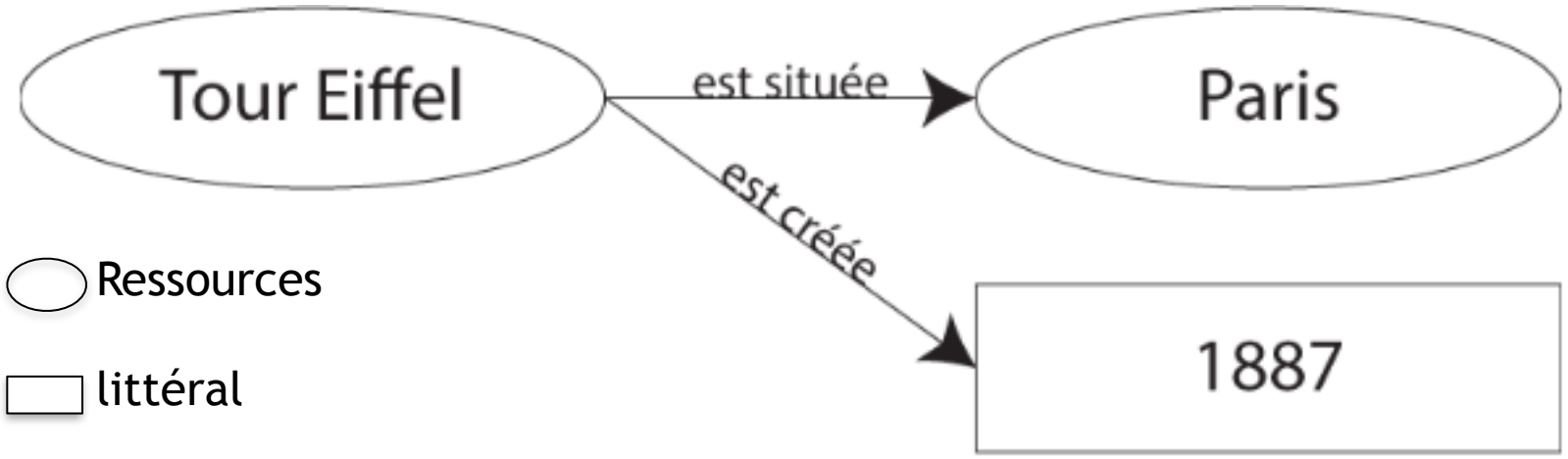
RDF (Resource Description Framework)

- RDF est un standard décrivant :
  - des ressources, e.g., personnes, lieux, animaux, documents, concepts, etc.
  - la description de ces ressources par des attributs et des relations ;
  - le framework contenant un modèle de données, des langages et des syntaxes

- Triplets comme modèle de données
  - Sujet, prédicat, objet
    - Tour Eiffel, est créée, 1887.
    - Tour Eiffel, est située, Paris.

Graphe de données

- La structure des données RDF est un graphe orienté



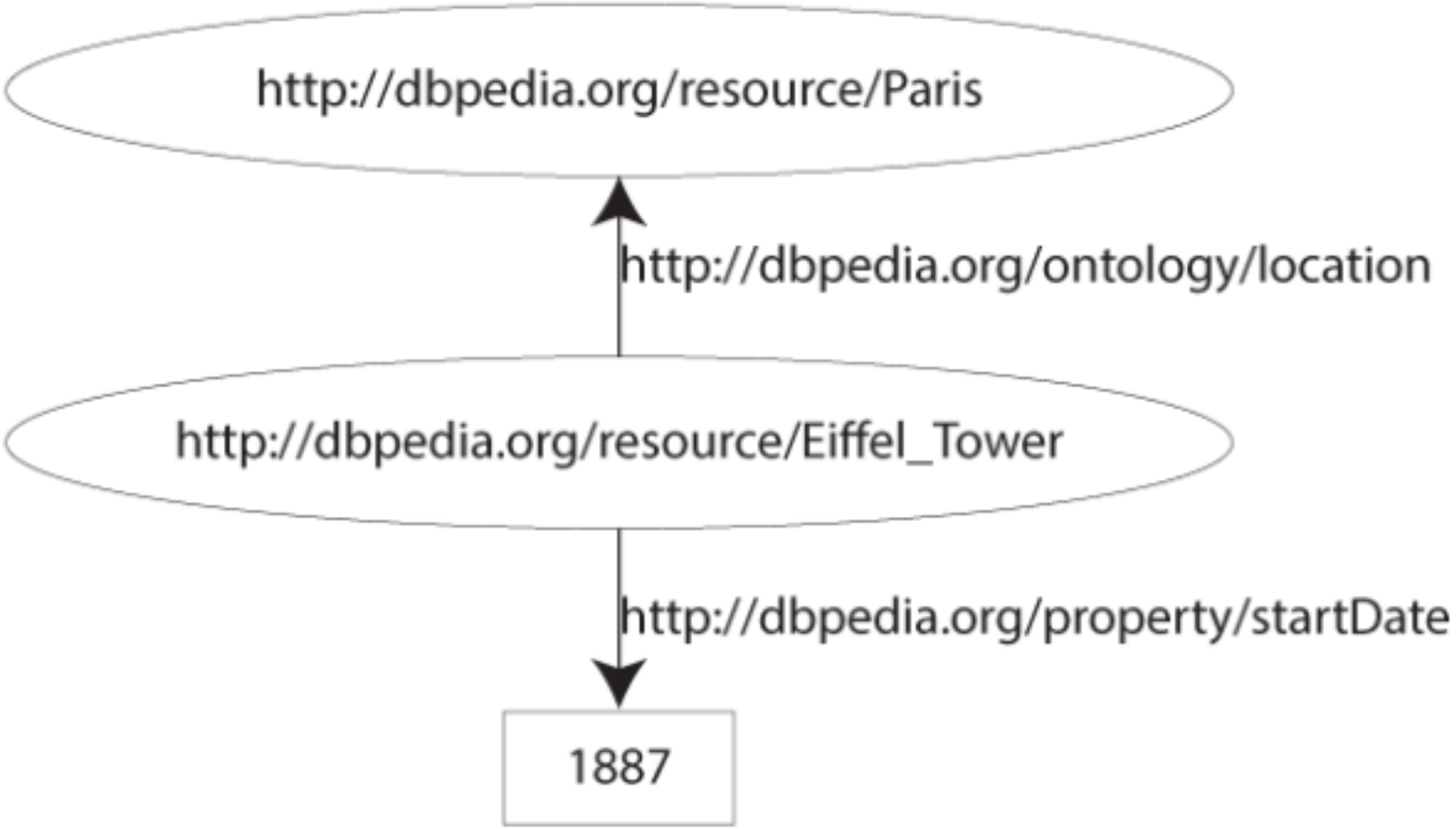
http://web-semantique.developpez.com/

URI (Uniform Resource Identifier)

- Un URL représente de manière unique une page web
- Un IRI représente de manière unique une ressource
- DBpedia
  - Version données liées de wikipédia
  - Source de ressources
  - Notre exemple
    - http://dbpedia.org/resource/Eiffel_Tower, http://dbpedia.org/ontology/location, http://dbpedia.org/resource/ Paris.

    - http://dbpedia.org/resource/Eiffel_Tower, http://dbpedia.org/property/startDate, 1887.

Graphe RDF


http://dbpedia.org/resource/Paris
http://dbpedia.org/ontology/location
http://dbpedia.org/resource/Eiffel_Tower
http://dbpedia.org/property/startDate
1887

Turtle ou XML ?

- XML très verbeux
  - http://www.w3.org/TR/rdf-syntax-grammar/

- Turtle plus compact et lisible par l'humain

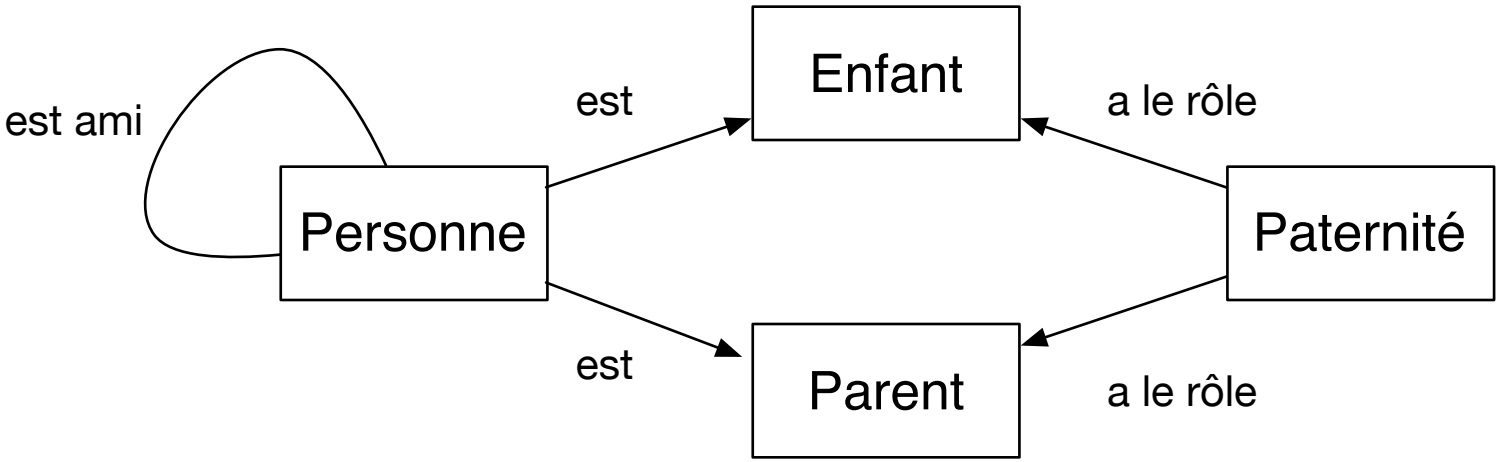
```
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbpedia:Eiffel_Tower dbpedia-owl:location dbpedia:Paris ;
                     dbpprop:startDate "1887"^^xsd:integer ;
                     dbpprop:name "Tour Eiffel"@fr, "Eiffel Tower"@en .
```

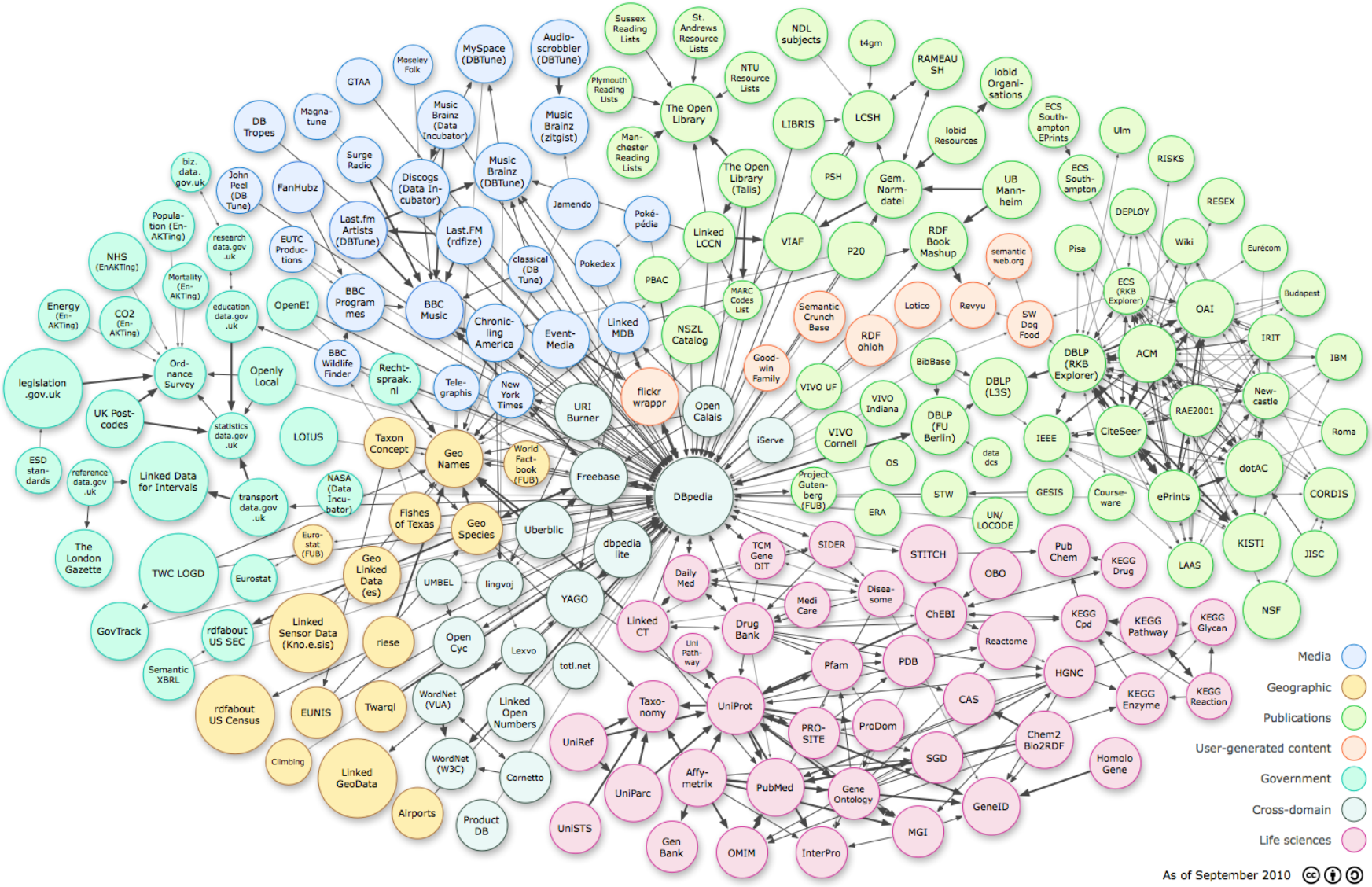
Ontologies

- Ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances.
- Les concepts sont organisés dans un graphe dont les relations peuvent être :
  - des relations sémantiques
  - des relations de subsomption
- décrivent généralement :
  - individus : les objets de base ;
  - classes : ensembles, collections, ou types d'objets ;
  - attributs : propriétés, fonctionnalités, caractéristiques ou paramètres que les objets peuvent posséder et partager ;
  - relations : les liens que les objets peuvent avoir entre eux ;
  - événements : changements subis par des attributs ou des relations.

Exemple

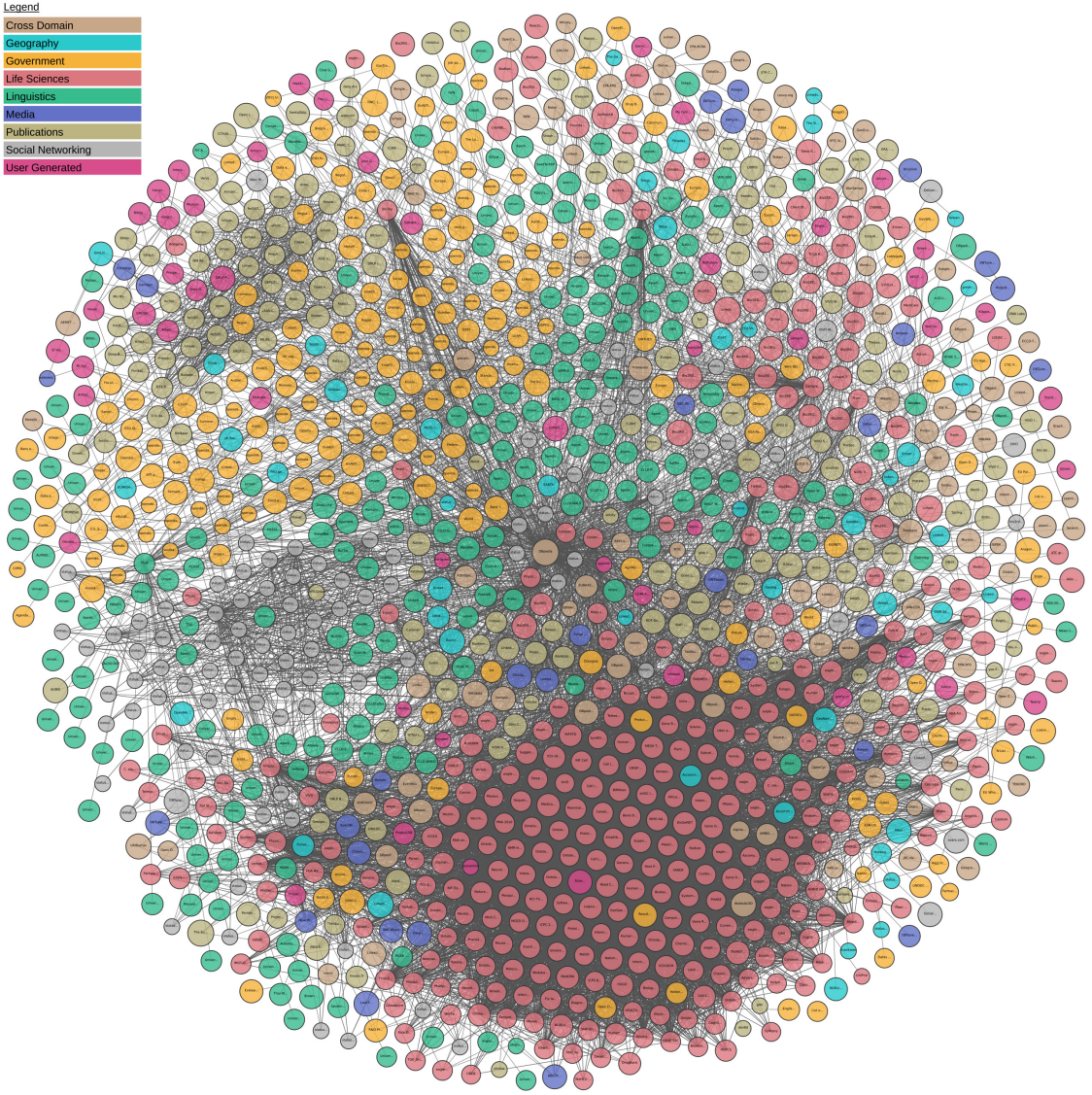

est ami
Personne
est
Enfant
a le rôle
Paternité
est
Parent
a le rôle

Linked open data world -2010



https://lod-cloud.net/

2019



The Linked Open Data Cloud from lod-cloud.net

History

	White	Colored	Graph file	Dataset list	Datasets
2019-03-29		png	svg json		1,239
2019-01-08		png	svg json		1,234
2018-11-26		png	svg json		1,231
2018-10-31		png	svg json		1,229
2018-08-28		png	svg json		1,224
2018-07-30		png	svg json		1,220
2018-06-28		png	svg json		1,205
2018-05-30		png	svg json		1,186
2018-04-30		png	svg json		1,184
2017-08-22		png	svg json	tsv	1,163
2017-02-20		png	svg		1,139
2017-01-26		png	svg		1,146
2014-08-30	png pdf svg	png pdf svg			570
2011-09-19	png pdf svg	png pdf svg			295
2010-09-22	png pdf svg	png pdf svg			203
2009-07-14	png pdf svg				95
2009-03-27	png pdf svg	png pdf svg			93
2009-03-05	png pdf svg	png pdf svg			89
2008-09-18	png pdf svg				45
2008-03-31	png pdf svg				34
2008-02-28	png pdf svg				32
2007-11-10	png pdf svg				28
2007-11-07	png				28
2007-10-08	png				25
2007-05-01	png				12

SPARQL

- Langage de requêtes et protocole pour chercher, modifier ou supprimer des données RDF disponibles sur le Web
- Standard du W3C
- Basé sur une algèbre à la SQL
- Une des couches pour la mise en œuvre du Web sémantique.

Exemple

http://query.linkeddatafragments.org/