Bases de données documentaires et distribuées
Cours NFE04
Documents structurés

Auteur : Philippe Rigaux

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

Représentation de documents textuels

La représentation des données est basée sur des **graphes**, permettant de représenter des données régulières ou irrégulières.

Notions essentielles

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.

Commençons avec JSON

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

```
{"nom": "Philippe",
 "tel": 2157786,
 "email": "philippe@cnam.fr"}
```

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

```
{"nom": {"prenom": "Philippe",  "famille": "Rigaux"},
 "tel": 2157786,
 "email": "philippe@cnam.fr"}
```

Autre extension : imbrication de listes.

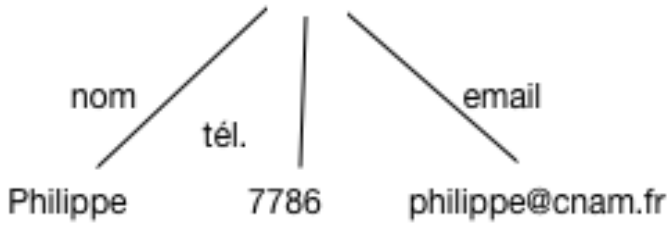
```
{"nom": "Philippe", "tels": [2157786, 2498762] }
```

Remarque

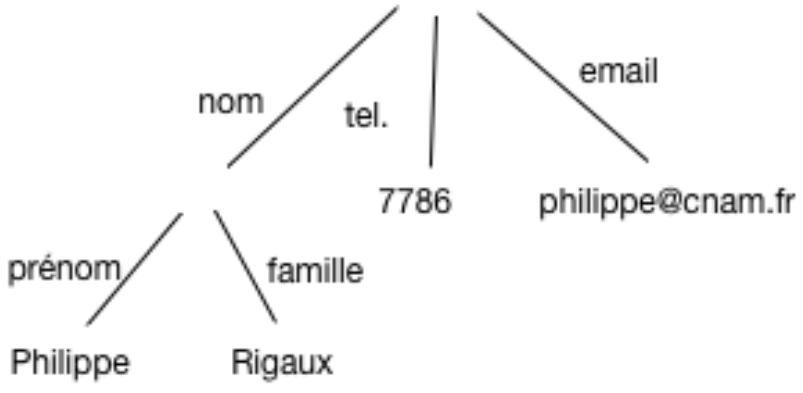
Les exemples ci-dessus montrent un **codage de l'information** sous forme d'une chaîne de caractères stockable sur disque et échangeable par réseau.

Vue conceptuelle : modélisation sous forme arborescente

Le codage représente en fait un **arbre**. Les étiquettes sont sur les arêtes, les valeurs sur les feuilles.



a. Arbre représentant un agrégat



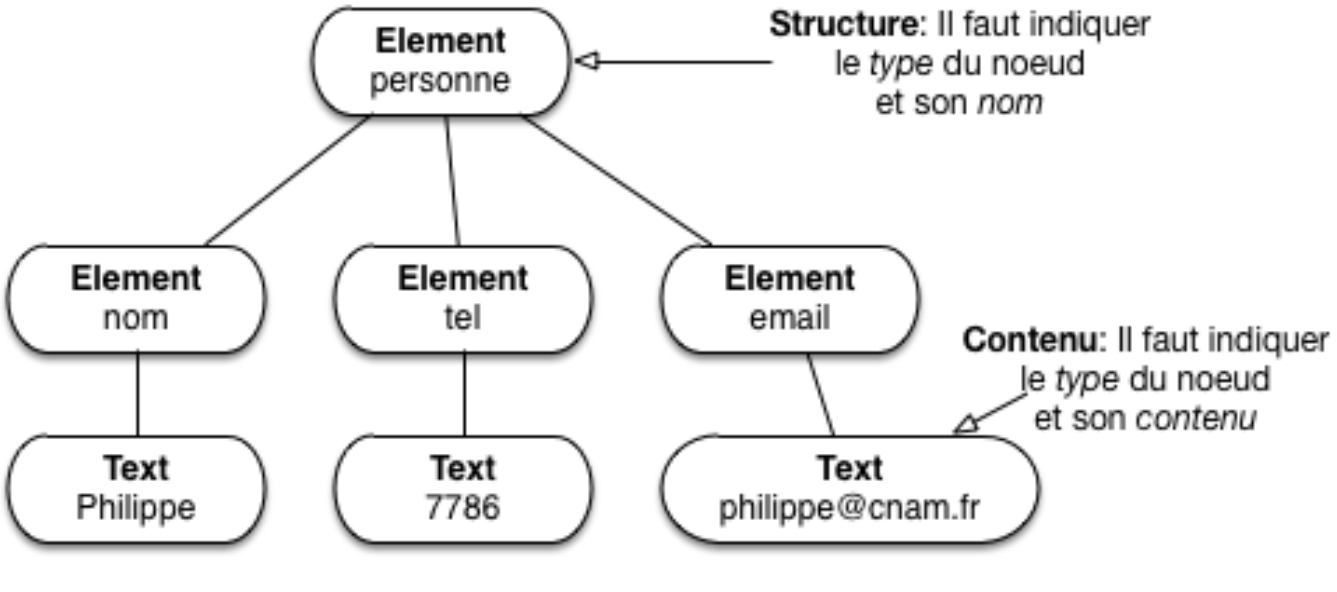
b. Arbre représentant une composition d'agrégats

Important

Le codage distingue bien une **structure** (l'arbre) et un **contenu** (le texte dans les feuilles). Toutes les **opérations** (de recherche, de navigation, ...) s'appuient sur la structure.

Autre possibilité : les étiquettes sont des nœuds

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



Remarque

C'est le choix de représentation du langage XML : toute l'information (structure et contenu) est stockée sous forme de nœud.

JSON, Javascript Object Notation

- JavaScript Object Notation ;
- Initialement créé pour la sérialisation et l'échange d'objets JavaScript ;
- Langage pour l'échange de données semi-structurées (et éventuellement structurées) ;
- Format texte indépendant du language de programmation utilisé pour le manipuler.

Utilisation première : échange de données dans un environnement Web (par exemple applications Ajax)

Extension : sérialisation et stockage de données

Les bases de JSON

Structure de base : paire *clef-valeur* (*key-value*)

```
"title": "The Social network"
```

Qu'est-ce qu'une valeur ? On distingue les valeurs **atomiques** et les valeurs **complexes** (construites)

Valeurs atomiques : chaînes de caractères (entourées par les classiques guillemets anglais (droits)), nombres (entiers, flottants) et valeurs booléennes (`true` ou `false`).

```
"year": 2010
```

```
"oscar": false
```

Les bases de JSON (suite)

Valeurs complexes : les **objets**.

Un *objet* est un ensemble de paires clef-valeur.

Au sein d'un ensemble de paires, une clef apparait au plus une fois (NB : les types de valeurs peuvent être distincts).

```
{"last_name": "Fincher", "first_name": "David"}
```

Un objet peut être utilisé comme valeur (dite *complexe*) dans une paire clef-valeur.

```
"director": {
  "last_name": "Fincher",
  "first_name": "David",
  "birth_date": 1962
}
```

Les bases de JSON (suite)

Valeurs complexes : les **tableaux**.

Un *tableau* (*array*) est une liste de valeurs (dont le type n'est pas forcément le même).

```
"actors": ["Eisenberg", "Mara", "Garfield", "Timberlake"]
```

Imbrication sans limite (vous vous souvenez d'XML ?) : tableaux de tableaux, tableaux d'objets contenant eux-mêmes des tableaux, etc.

Les bases de JSON (suite)

Un *document* est un objet. Il peut être défini par des objets et tableaux imbriqués autant de fois que nécessaire.

```
{
  "title": "The Social network",
  "summary": "On a fall night in 2003, Harvard undergrad and \n
        programming genius Mark Zuckerberg sits down at his \n
        computer and heatedly begins working on a new idea. (...)",
  "year": 2010,
  "director": {"last_name": "Fincher",
               "first_name": "David"},
  "actors": [
       {"first_name": "Jesse", "last_name": "Eisenberg"},
       {"first_name": "Rooney", "last_name": "Mara"}
    ]
}
```

Flexibilité de la représentation

Prenons un exemple plus proche de la notion « intuitive » de document : un rapport écrit.

```
{
    "author": "Philippe Rigaux"
    "date": "04/10/2065",
    "title": "BD documentaires",
    "content": {
       "para": "...",
       "para": "...",
       "section": {
          "title": "...",
          "para": "...",
          "figure" {"caption": "...", "content:"},
          "para": "...",
          "section": {...}
       }
    }
}
```

- Imbrication très libre d'éléments représentant des composants d'un rapport.
- Pas ou peu de **référence** (mais on pourrait l'envisager pour l'auteur).

Ce qu'il faut retenir

Les documents semi-structurés se caractérisent par les aspects suivants.

- Une structure **flexible** (présence/absence, variations), **complexe** (imbrication), et **auto-décrite**.
- Une double représentation : un vue conceptuelle (arbre) et son codage (forme sérialisée).
- **Valable** pour les documents "multimédia" au sens large : rapports, images, vidéos.
- La notion d'autonomie (pas d'association entre documents) est essentielle.
  - Autonomie ? On peut mettre en œuvre un **passage à l'échelle par distribution**. Typique des systèmes NoSQL.
  - Pas d'autonomie : un jour ou l'autre il faudra faire des jointures ; les systèmes NoSQL ne sont pas bon pour ça.

JSON vs. XML

- JSON plus léger et intuitif que XML,
- Facile à parser pour n'importe quel langage de programmation,
- JSON n'a pas (encore) de langage de spécification de schéma associé,
- JSON n'a pas (encore) de langage de requête associé.

Voir le chapitre du cours consacré à XML pour une présentation de ce dernier sous l'aspect "gestion de données".

Quelques sites pour aler plus loin

- Tout sur JSON : `http://json.org/`
- Un validateur de documents JSON : `http://jsonlint.com/`
- Une proposition de **schéma** pour JSON : `http://json-schema.org/`
- Un langage de requêtes JSON : JAQL, `http://code.google.com/p/jaql/`