

Les Documents Structurés avec les langages XML et JSON

Principe de représentation de documents (qu'est-ce qu'un document structuré) et de représentation de données complexes.

Les données représentées dans des documents structurées sont des graphes.

Graphes => les données sont auto-décris c'est à dire qu'il y a un contenu d'information et une structure. Ces derniers sont associés pour créer ce qui est **clés-valeur**.

Les données sont auto-décris (le contenu est lié à sa propre description), on peut réaliser des structures élaborées à partir de listes et d'enregistrements qui sont imbriqués.

Remarques : Les données ne sont pas typées => aucune contrainte (mais attention, comparaison avec le mode relationnel).

Notion de sérialisation (code XML et JSON), l'objectif est la mise sous la forme d'une chaîne de caractères pour être transférer sur le réseau.

Organisation sous la forme d'une paire : clé/valeur donc il y a autodescription.

JSON : paire clé/valeur donc il y a autodescription.

Il existe des listes associatives de paires clés/valeur.

Par exemple :

Dans notre exemple, **Martin** qualifie le nom c'est-à-dire la clé qui représente le nom. Nous avons plusieurs **clés/valeur**.

```
{ "nom" :"Martin", "tel" : 0623142126, "email" :"stefan.martin@center.fr"}
```

On peut imbriquer **des listes dans d'autres listes**....

Par exemple :

```
{ "nom" : {"prenom" :"stefan", "famille" :"Martin"},  
  "tel" : 0623142126,  
  "email" :"stefan.martin@center.fr"}  
}
```

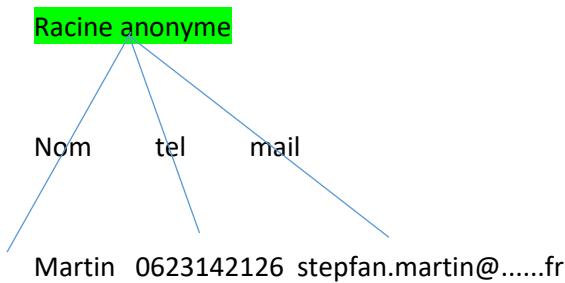
Le "**nom**" caractérise l'ensemble constitué du **prénom** et de **famille**.

Remarque : on a ici un codage du graphe sous la forme d'une chaîne de caractères c'est donc stockable sur un disque et échangeable => sérialisation

Le document lui-même est un graphe c'est à dire un arbre caractérisé par :

- **Une Racine Anonyme**
- Les feuilles sont donc **des valeurs**

On garde bien une unité d'information autonome



On a la forme arborescente c'est la forme sérialisée avec la structure représentée par la paire :

Clés/ Valeur

Remarque : En XML on a des nœuds.

Exemple : Si on compare le relationnel (SGBDR) et le JSON (document structuré)

Relationnel : table Administrateur

Administrateur (ID, Nom, Prenom)

ID	Nom	Prenom
167	Dupon	Jean
37	Martin	Stephane

Représentation JSON

(

```

Administrateur : { "ID" :37, "Nom" :"Martin ", "Prenom" :"Stephane"},  

Administrateur :{ "ID" :167, "Nom" :"Dupon ", "Prenom" :"Jean"},  

)

```

Représentation de données complexes

Si on prend le cas d'un film, on peut représenter des choses complexes grâce à l'imbrication.

En une seule unité d'information on a des valeurs imbriquées (directeur) (on ne sait organiser cette représentation en SGBD relationnel).

SGBD Relationnel :

Film (idfilm, title , year, idreal)

Artiste (idA, nom, prenom, annee)

Role (idfilm, idA, role)

Explication : On a 3 tables avec des informations concernant les films dans toutes les tables. On peut faire des jointures pour reconstituer (naviguer dans les tables en partant de la « table choisie » comme par exemple la table Film ou Artiste ou Role) les informations. Qd on ajoute un film on a une transaction dans toutes les tables. Cette représentation est à plat.

En NOSQL on a une seule unité on gère tout un film à un seul endroit.

```
{
  "_id": "1",
  "title": "le stratege",
  "year": "2020",
  "director": {
    "lastname": "Brad",
    "first-name": "Pitt"
  },
  "actors": [
    {"first-name": "Pitt", "lastname": "Brad"},  

    {"artist:27", "role": "Butch Coolidge"}
  ],
  "title": "Jacky Brown",
  ...
},
...
],
"films_jouées": [
{
  "title": "Pulp fiction",
  ...
},
{
  "title": "Reservoir Dogs",
  ...
},
...
]
}
```

Tout est centré sur le film (c'est le point de départ. Il aurait été possible de choisir une autre représentation mais cela aurait totalement modifié la structure du document à cause de sa forme arborescente) avec un metteur en scène en tant que directeur et on retrouve ensuite les acteurs (on descend dans l'arborescence), toute l'information est **rassemblée** (unique au même endroit mais pas au même niveau) mais on a 2 fois le même directeur et acteur c'est donc **redondant**, cela est totalement incohérent au sens Relationnel => espace de stockage perdu.

Si je fais une erreur sur Brad Pitt ce sera compliqué à gérer, donc le chemin d'accès est compliqué même si on a toute l'information en direct et rassemblée. D'autre part il faudra mettre à jour toutes les ignes

On a privilégié un chemin d'accès depuis le film ce qui est parfait en revanche pour les acteurs c'est plus difficile à gérer.

On doit utiliser ce type de représentation quand on peut se permettre la redondance, qd il n'y a pas de mise à jour comme pour un entrepôt de données, comme quand on enregistre les recherches d'un internaute, ses visites sur un site et qd on traite de très très gros volume => scalabilité. Les données ne sont jamais modifiées.

Donc pour résumer => structurer, flexible, complexe, auto décrite c'est la caractéristique des documents structurés soit en représentation en arbre, graphe avec une forme sérialisée (pas de référence d'un document à un autre donc pas de jointure).