BUT Réseaux & Télécommunications

Traitement des données

S.A.É. R1.05

Olivier ECKLE

Introduction

Les professionnels du monde de l'informatique sont confrontés à des masses de données toujours plus importantes…

Exemples :

- « Logs » des équipements réseaux (routeurs, switch, firewall, …).
- « Logs » d'erreurs, d'accès serveurs, de connections échouées.
- « Logs » d'évènements ou de sessions Windows, MacOS, Linux.
- « Dump » mémoire ou analyse de malware.
- Taux d'utilisation d'espace HD.
- Données IoT issues de capteurs connectés.
- Images/Vidéos issue de télésurveillance.
- Historique de navigations.
- Etc…

Introduction

Ces masses de données ne sont pas exploitables sans la mise en œuvre d'outils de traitement de leurs contenus…
Les professionnels de l'informatique qualifient ces scripts de traitement : outils de « ***reporting*** ».
De nombreuses société proposent des quantités d'outils de « ***reporting*** » pour les traitements courants.
Dans certains cas, ils est cependant nécessaire de développer soi-même son propre utilitaire de « ***reporting*** »…

La typologie des données

On distingue TROIS types de données :

1. Les données ***structurées*** :
   - Tableaux, Bases de données relationnelles…
   - CSV, TSV & XLS/XLSX – SQL Dump, SQLite …
2. Les données ***semi-structurées*** :
   - « Markup language », Données statistiques & scientifiques, Données réseaux & « Logs »…
   - JSON – XML, YAML – HDF5, NetCFD – SysLOG, PCAD…
3. Les données ***non structurées*** :
   - Fichiers texte & « Logs » bruts, Documents bureautique, Multimédia, Email & SMS, Archives binaires…
   - TXT – DOC, PDF – JPG, PNG, GIF, MP4, AVI, MP3, WAV – MBOX – ZIP, RAR, EXE, BIN…

La typologie des données

Bien évidemment, plus les données sont structurées,
plus l'outils de « ***reporting*** » est simple à mettre en œuvre…
Lorsque les données sont structurées, on va pouvoir retrouver une cohérence entre les différents enregistrements de données !

```
{
  "timestamp": "2025-01-06T10:00:00Z",
  "level": "INFO",
  "source": "webserver",
  "message": "Request processed successfully",
  "details": {
    "method": "GET",
    "endpoint": "/api/users",
    "status_code": 200,
    "response_time_ms": 123
  }
}
```

Exemple de « log » structuré…

Exemple de « log » NON structuré…

```
2025-01-06 10:00:00 INFO: webserver - Request processed successfully:
GET /api/users, status 200, response time 123ms
```


Pourquoi traiter les données ?

Le traitement de données permet de prendre des décisions éclairées basées sur des faits mesurables et non des supposition… Cela est essentiel dans des domaines tels que le commerce, l'ingénierie, la gestion de projets, ou encore la santé !

Exemple en entreprise :
Une entreprise collecte des données sur ses ventes, ses stocks et ses performances marketing. En analysant ces données :
- → Elle peut identifier quels produits se vendent le mieux selon les régions.
- → Elle peut ajuster ses campagnes publicitaires pour maximiser le retour sur investissement.
- → Elle peut optimiser ses chaînes d'approvisionnement pour éviter des ruptures de stock.

Exemple en ingénierie :
Une équipe de maintenance dans le domaine des télécommunications peut analyser « logs » pour identifier des pannes récurrentes. Grâce à cette analyse, elle peut :
- → Planifier une maintenance préventive pour éviter des interruptions de service.
- → Optimiser l'utilisation des ressources humaines et matérielles.

Exemple en médecine :
Les données médicales collectées sur un patient (historique médical, résultats d'analyses, imagerie médicale, etc.) sont analysées pour identifier des problèmes spécifiques ou adapter les traitements :
En analysant les résultats d'une imagerie médicale (IRM, scanner), associée à des bases de données contenant des milliers de cas similaires, les outils d'IA peuvent aider les médecins à :
- → Identifier les tumeurs à un stade précoce.
- → Différencier un kyste bénin d'une tumeur maligne avec une grande précision.
- → Recommander des traitements adaptés en fonction du profil du patient (âge, antécédents, etc…)

Pourquoi traiter les données ?

Si de plus, on dispose de plusieurs traitements de données similaire sur un intervalle de temps significatif, il est possible d'appliquer une « dérivée » et de faire apparaitre ainsi des tendances d'évolutions. Tendances qui permettent éventuellement d'anticiper des évènements futurs.

Exemple en ingénierie :
En étudiant les données d'utilisation du réseau (bande passante, temps d'appel, etc.), un opérateur peut :
- → **Identifier une augmentation progressive de la consommation de données mobiles.**
- → **Prendre des mesures pour augmenter la capacité des antennes dans les zones concernées avant que des saturations n'apparaissent.**

Le RGPD

- Le Parlement Européen a adopté le 27 avril 2016 un texte de loi applicable dans les 27 états membres intitulé RGPD (Règlement Général sur la Protection des Données).
- Les principaux objectifs du RGPD sont d'accroître à la fois la protection des personnes concernées par un traitement de leurs données à caractère personnel et la responsabilisation des acteurs de ce traitement (société qui utilisent ces données). Autrement dit, ce texte permet de « ***redonner aux citoyens le contrôle de leurs données personnelles*** ».
- Les sociétés hors UE doivent respecter ce texte pour tous les citoyens de UE !!!

Le RGPD

Conséquences du RGPD :

1. *Lorsqu'un organisme de démarchage téléphonique (ex : changement de fenêtres) vous contacte, vous avez le droit d'exiger que l'opérateur modifie ou supprime au niveau de sa base de données, les informations qu'il dispose sur vous.*
2. *Vous devez avoir le droit de modifier ou supprimer une partie ou la totalité des informations personnelles qui sont stockées sur les différents réseaux sociaux.*
3. *Vous n'avez pas le droit d'intégrer dans les données que vous mettez en ligne (notamment les photos) d'autres personnes sans leur demander leur accord.*

Ce que le RGPD ne permet plus :

D'utiliser des données personnelles pour mettre en œuvre des influences politiques… C'est le cas de la société « Cambridge Analytica » qui a utilisé les données « facebook » de 87 millions d'utilisateurs pour apporter des éléments à :

- *La campagne publicitaire du « Brexit » au RoyaumeUni.*
- *La 1ère campagne présidentielle de Donald TRUMP.*

La divulgation au grand public des activités de « Cambridge Analytica » a créé un véritable scandale mondial !!!

Voir le documentaire « The Great Hack » sur netflix…

De la durée de vie des données…

Les données ont le cycle de vie suivant :

L'absence de Suppression ou un Archivage non optimisé est consommateur de place dans les datacenters et donc d'énergie…
C'est un des points les plus perfectible à l'heure actuelle.
Il est cruciale de progresser en la matière !!!

9000 milliards de To

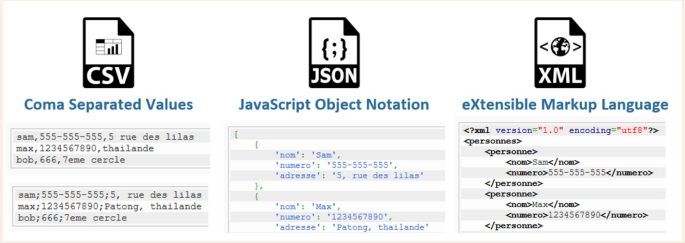
Consommation des Datacenters de la planète en 2026 : 1000 TWh

X 5 en 8 ans !!!

120 centrales nucléaires comme celle de Fessenheim…

Les formats de données ouverts…

- Un format de données peut être :
  - « ***Propriétaire*** » c'est à dire chiffré et donc utilisable uniquement par un logiciel payant.
  - « ***Public*** » donc connu et utilisable par tous. Les données sont généralement clairement lisibles par l'être humain .
- Les trois formats publics les plus utilisés sont :



CSV

Le format CSV (*Comma-Separated Values*) est un format de fichier qui doit sa notoriété à « Excel ». Il est largement utilisé pour représenter des données tabulaires sous forme de texte.

Caractéristiques fondamentales :

- **Structure** : Chaque ligne représente un enregistrement.
  Les colonnes sont séparées par un délimiteur, souvent une virgule « , », mais cela peut être un point-virgule « ; » ou une tabulation « \t ».
- **Portabilité** : Compatible avec de nombreux outils : tableurs (Excel, Google Sheets), bases de données, langages de programmation (Python, Java, etc.).
  Peut être ouvert avec un éditeur de texte simple ou un tableur.
  Facile à compresser, manipuler, et transmettre.

Exemples CSV :

L'usage veux que l'on évite les caractère spéciaux (é,è,à, etc…) ainsi que les caractères de ponctuation dans les noms des « ***clés*** »…

```
Nom,Prenom,Origine EOL
AMIDALA,Padmé,Naboo EOL
QUI-GON,Jinn,Coruscant EOL
ORGANA,Leia,Polis Massa EOL
SECURA,Aayla,Ryloth EOL
TANO,Ahsoka,Shili EOL
```

Un caractère non imprimable et non visible à la fin de chaque ligne : EOF (« ***End Of Line*** ») qui se note aussi « \n »

Le dernier EOL est optionnel…

Cas particuliers CSV

Lorsque la donnée contient une virgule « , » ou bien un EOL, elle est placée entres guillemets: « " " »…

Et si une donnée contient un guillemet, ce dernier est doublé…

```
Prenom,Adresse EOL
Gaspar,"10, Grand'Rue" EOL
```

Pour intégrer une « , » dans la donnée !!!

```
Personne,Message EOL
Charlotte,""Hello World"" EOL
```

Pour intégrer un « " » dans la donnée!!!

```
Prenom,Adresse EOL
Gaspar,"10, Grand'Rue EOL
68000 COLMAR" EOL
```

Pour intégrer un « EOL » dans la donnée!!!

Création d'un fichier CSV en Python

A partir d'une liste :

```python
import csv

donnees = [
    ['Nom', 'Prenoms', 'Age', 'Email'],
    ['Dupont', 'Marie,Anne', 30, 'marie.dupont@example.com'],
    ['Durand', 'Robert ("Bob")', 25, 'robert.durand@example.com']]

with open('fichier.csv', mode='w', encoding='utf-8', newline='') as file:
    fichier = csv.writer(file)
    fichier.writerows(donnees)
```

Important pour ne pas avoir de doubles sauts de lignes dans le fichier CSV…

```
Nom,Prenoms,Age,Email
Dupont,"Marie,Anne",30,marie.dupont@example.com
Durand,"Robert (""Bob"")",25,robert.durand@example.com
```


Création d'un fichier CSV en Python

A partir d'un dictionnaire :

```
Nom,Prenoms,Age
Dupont,"Marie,Anne",30
Durand,"Robert (""Bob"")",25
```

```python
import csv

donnees = [
    {'Nom': 'Dupont', 'Prenoms': 'Marie,Anne', 'Age': 30},
    {'Nom': 'Durand', 'Prenoms': 'Robert ("Bob")', 'Age': 25}
]

with open('fichier.csv', mode='w', encoding='utf-8', newline='') as file:
    champs = ["Nom", "Prenoms", "Age"]
    fichier = csv.DictWriter(file, fieldnames=champs)
    fichier.writeheader()
    fichier.writerows(donnees)
```

Important pour ne pas avoir de doubles sauts de lignes dans le fichier CSV…

Lecture d'un fichier CSV en Python

A partir du fichier :

```
Nom,Prenoms,Age
Dupont,"Marie,Anne",30
Durand,"Robert (""Bob"")",25
```

```python
import csv

with open('fichier.csv', mode='r', encoding='utf-8') as file:
    fichier = csv.reader(file)
    for ligne in fichier:
        print(ligne)
```

```
['Nom', 'Prenoms', 'Age']
['Dupont', 'Marie,Anne', '30']
['Durand', 'Robert ("Bob")', '25']
```


Avantages & Inconvénients du CSV

Avantages du format CSV :
1. **Simplicité** : Facile à créer et manipuler.
2. **Compatibilité universelle** : Prend en charge plusieurs logiciels.
3. **Taille réduite** : Idéal pour les transferts de données.

Inconvénients du format CSV :
1. **Pas de hiérarchie** : Ne convient pas pour des structures complexes.
2. **Pas de métadonnées** : Ne stocke pas d'informations sur les types de données.
3. **Problèmes d'échappement** : Peut devenir complexe avec des caractères spéciaux.

JSON

JSON (*JavaScript Object Notation*) est un format léger mais structuré d'échange de données. Il est facile à lire et à écrire pour les humains et peut également être analyser (« ***parser*** ») et générer (« ***stringification*** ») par la plupart des langages de programmation actuels.

Il est couramment utilisé pour transmettre des données entre un client et un serveur, les API et le stockage de données utilisateurs.

Caractéristiques fondamentales :

- **Basé sur le texte** : JSON est écrit sous forme de texte brut, ce qui le rend indépendant de la plateforme (Windows / MacOS / Linux).
- **Structuré et hiérarchique** : Il permet de représenter des objets complexes avec des relations imbriquées et prend en compte le type des données.
- **Standardisé** : JSON suit une spécification bien définie par la norme RFC 8259.

JSON

Le format JSON comprend DEUX structures principales qui peuvent être utilisées indépendamment ou être combinées en fonction de l'organisation des données à stocker :

- **Les « Objets »** : Collections de paires « clé-valeurs ».
- **Les « Listes » (ou « Tableaux »)** : Collections ordonnées d'« **Objets** » ou directement de « **valeurs** ».

Les « OBJETS » JSON

Un « **OBJET** » est toujours entouré d'accolades « { } » et contient une suite NON ordonnée de paires « **clé-valeur** ».

- La **« clé »** est toujours une chaîne de caractères encadrés par des guillemets « " " » ou pas !!!
- La **« valeur »** peut être de différents types parmi :

Type JSON	Equivalent Python
`string`	Chaîne de caractères
`number`	Entier / Flottant
`boolean`	Booléen (`true`/`false`)

Type JSON	Equivalent Python
`object`	Dictionnaire {…}
`array`	Liste […]
`null`	*Absence de valeur*

Les « LISTES » JSON

Une « **LISTE** » est toujours entouré de crochets « **[ ]** » et contient une liste pouvant être directement des **« valeurs »** ou bien des « **OBJETS** » voire même des « **LISTES** ».

Les éléments d'une « **LISTE** » sont séparés par une virgule « **,** » tout comme dans les « **OBJETS** »…

L 'usage veux que les « **OBJETS** » qui se trouvent dans une « **LISTE** » aient la même structure (même collection de « **clés** »). Mais ce n'est pas une obligation formelle…

Exemples JSON :

« object »

```
{
  "nom": "Yoda",
  "prenom": none,
  "age deces": 900,
  "job": "Maître JEDI"
}
```

Séparés par des virgules

Pas de virgule au dernier !!!

« array »

```
[
  "Tano",
  "Asoka",
  16,
  "Padawan"
]
```

« array » d' « object »

```
[
  {
    "nom": "Palpatine",
    "prenom": "Sheev",
    "age deces": 88,
    "job": "Empereur Galactique"
  },
  {
    "nom": "Skywalker",
    "nom": "Anakin",
    "age deces": 45,
    "job": "Seigneur Sith"
  }
]
```

Séparés par une virgule

Pas de virgule au dernier !!!

Exemples JSON :

```
{
  "nom": "WEBER",
  "prenom": "Alice",
  "age": 25,
  "isEtudiante": true,
  "competences": ["Python", "Réseaux", "Sécurité"],
  "adresse": {
    "rue": "25, Grand'rue",
    "codePostal": 68000,
    "ville": "COLMAR",
  }
}
```

Donnée : type « **string** »

Donnée : type « **number** »

Donnée : type « **boolean** »

Donnée : type « **array** »

Donnée : type « **object** »

Création d'un fichier JSON en Python

A partir d'un dictionnaire :

```
import json

donnees = {
    'nom': 'WEBER',
    'prenoms': 'Alice',
    'age': 35,
    'competences': ['Python', 'Réseaux']
}

with open('fichier.json', mode='w', encoding='utf-8') as file:
    json.dump(donnes, file, indent=4)
```

```
{
    "Nom": "WEBER",
    "Prenom": "Alice",
    "age": 35,
    "competences": [
        "Python",
        "R\u00e9seaux"
    ]
}
```

Le « é » est remplacé par son code UNICODE (utf-8) : « \u00e9 »…

On impose une indentation pour rendre le fichier JSON plus lisible…

Création d'un fichier JSON en Python

A partir d'une liste :

```
import json

donnees = ['WEBER', 'Alice', 30, ['Python', 'WCS']]

with open('fichier.json', mode='w', encoding='utf-8') as file:
    json.dump(donnes, file, indent=4)
```

```
[
    "WEBER",
    "Alice",
    30,
    [
        "Python",
        "WCS"
    ]
]
```

Dans ce cas, il n'y a pas de « clé »…

Lecture d'un fichier JSON en Python

A partir du fichier :

```
import json

with open('fichier.json', mode='r') as file:
    fichier = json.load(file)

print(type(fichier))
print(fichier)
```

```
{
    "prenom": "Alice",
    "age": 25,
    "spe": [
        "Python",
        "R\u00e9seaux"
    ]
}
```

```
<class 'dict'>
{'prenom': 'Alice', 'age': 25, 'spe': ['Python', 'Réseaux']}
```


Cas particuliers JSON

Lorsque la donnée contient un guillemet « " » ou bien le caractère d'échappement de Python (et Java) : « \ », on doit le faire précéder par un caractère d'échappement « \ »…

```
data = {
  'fichier': 'Image.png',
  'chemin': "C:\\Images\\"
}
```

Pour intégrer un « \ » dans la donnée !!!

```
{
  "fichier": "Image.png",
  "chemin": "C:\\Images\\"
}
```

```
data = {
  'orateur': 'Barack OBAMA',
  'message': '"Yes, We can."'
}
```

```
{
  "orateur": "Barack OBAMA",
  "message": "\"Yes, We can.\""
}
```

Pour intégrer un « " » dans la donnée !!!

XML

XML (*eXtensible Markup Language*) est un langage de balisage conçu pour stocker et transporter des données dans un format structuré très flexible. XML est largement utilisé pour l'échange de données entres systèmes et applications.

Il a été créé par W3C en se basant sur HTML… Son fonctionnement est donc similaire à HTML.

Caractéristiques fondamentales :

- **Lisibilité** : Les fichiers XML sont lisibles par l'être humain.
- **Flexibilité** : Il permet de définir ses propres balises, adaptées à ses besoins.
- **Portabilité** : XML est totalement indépendant du langage utilisé et de la plateforme (Windows / MacOS / Linux).

Exemple XML :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<bibliotheque>
    <livre id="1">
        <titre>Xml</titre>
        <auteur>Alexandre BRILLANT</auteur>
        <annee>2010</annee>
    </livre>
    <livre id="2">
        <titre>Xml pour l'édition</titre>
        <auteur>Bernard PROST</auteur>
        <annee>2011</annee>
    </livre>
</bibliotheque>
```

Un attribut peut compléter une balise : « id="1" ».

Règles syntaxiques de XML

Le format XML doit respecter les règles suivantes :

- **Racine unique** : Un document XML doit avoir un élément racine (balise) unique.
- **Sensibilité à la casse** : Les balises `<Bonjour>` et `<bonjour>` sont différentes…
- **Nesting** : Les balises doivent s'imbriquer en « *Matriochkas* » (poupées russes) tout comme en HTML.
  Ainsi : `<a><b>…</b></a>` est correcte mais `<a><b>…</a></b>` ne l'est pas.
- **Attributs** : Les attributs doivent être entourés de guillemets « " " » ou bien d'apostrophes « ' ' »...

Les attributs les plus courants en XML

En XML, les attributs ne sont pas prédéfinis (contrairement à des langages comme HTML), car XML permet de définir des balises et attributs personnalisés en fonction des besoins. Cependant, certains attributs couramment utilisés dans divers contextes ou standards XML sont les suivants :

Attribut	Descriptions	Attribut	Descriptions
`id`	Identifiant UNIQUE d'un élément.	`version`	Spécifie une version de l'élément.
`name`	Nom explicite d'un élément.	`href`	Pour inclure un lien (idem HTML)
`type`	Spécifie le type du contenu.	`title`	Donne un titre à l'élément.
`lang`	Indique la langue utilisée (fr, en, …)	`class`	Définit une CLASS pour du style…

On peut associer un fichier CSS à un XML…

Création d'un fichier XML en Python

```python
import xml.etree.ElementTree as XML

# Ajouter un élément
racine = XML.Element("bibliotheque")
# Ajouter un élément
livre1 = XML.SubElement(racine, "livre", id="1")
XML.SubElement(livre1, "titre").text = "Xml »
XML.SubElement(livre1, "auteur").text = "Alexandre BRILLANT »
XML.SubElement(livre1, "annee").text = "2010"
# Convertir en arbre XML et écrire dans un fichier
arbre = XML.ElementTree(racine)
with open("livres.xml", "wb") as file:
    arbre.write(file, encoding="utf-8", xml_declaration=True)

print("Fichier XML créé avec succès.")
```


Création d'un fichier XML en Python

```xml
<?xml version='1.0' encoding='utf-8'?>
<bibliotheque>
    <livre id="1">
        <titre>Xml</titre>
        <auteur>Alexandre BRILLANT</auteur>
        <annee>2010</année>
    </livre>
</bibliotheque>
```


Lecture d'un fichier XML en Python

```python
import xml.etree.ElementTree as XML

# Charger et parser le fichier XML
arbre = XML.parse("livres.xml")
racine = arbre.getroot()
# Parcourir les éléments
for livre in racine.findall("livre"):
    id_livre = livre.get("id")
    titre = livre.find("titre").text
    auteur = livre.find("auteur").text
    annee = livre.find("annee").text

    print(f"Livre ID: {id_livre} \n  Titre: {titre}\n  Auteur: {auteur}\n  Année: {annee}")
```

```
Livre ID: 1
   Titre: Xml
   Auteur: Alexandre BRILLANT
   Annee: 2010
```


Bilan

Ce CM de la SAE R1.05 ne constitue qu'une petite introduction au vaste monde du traitement des données…

Chaque « niche » du monde de l'informatique a ses propres usages en matière de stockage et traitements des données.
A vous de vous adapter à ces usages.

Plus généralement, l'usage de formats libres est toujours préférable à l'usage de formats propriétaires. Ces formats libres n'interdisent pas d'intégrer des données chiffrées si des besoins de confidentialités sont nécessaires…