



*Version expérimentale  
En cours de validation*



## RÉSUMÉ THÉORIQUE – FILIÈRE DÉVELOPPEMENT DIGITAL OPTION WEB FULL STACK M111 – GÉRER LES DONNÉES

Elaboré par :

**Widad JAKJOUR**

**Formatrice à ISTA TIC - SAFI**



**90 heures**



# Equipe de rédaction et de lecture

## Equipe de rédaction :

Mme Jakjoud Widad : Formatrice en développement digital

## Equipe de lecture :

Mme Laouija Soukaina : Formatrice animatrice au CDC Digital & IA



# SOMMAIRE

## 1. Exploiter les fonctionnalités avancées d'un SGBD relationnel

Maitriser le langage de programmation  
procédurale sous MySQL  
Optimiser une base de données MySQL  
Protéger la base de données MySQL

## 2. Exploiter les fonctionnalités des bases de données NoSQL MongoDB

Découvrir les bases de données NoSQL  
Mettre en place une base de données MongoDB  
Modéliser les documents  
Manipuler les données avec mongoDB  
Effectuer des requêtes depuis des programmes  
Python  
Sécuriser une base de données MongoDB

# MODALITÉS PÉDAGOGIQUES



1

## LE GUIDE DE SOUTIEN

Il contient le résumé théorique et le manuel des travaux pratiques



2

## LA VERSION PDF

Une version PDF est mise en ligne sur l'espace apprenant et formateur de la plateforme WebForce Life



3

## DES CONTENUS TÉLÉCHARGEABLES

Les fiches de résumés ou des exercices sont téléchargeables sur WebForce Life



4

## DU CONTENU INTERACTIF

Vous disposez de contenus interactifs sous forme d'exercices et de cours à utiliser sur WebForce Life



5

## DES RESSOURCES EN LIGNES

Les ressources sont consultables en synchrone et en asynchrone pour s'adapter au rythme de l'apprentissage



## PARTIE 2

### Exploiter les fonctionnalités des bases de données NoSQL MongoDB

#### Dans ce module, vous allez :

- Découvrir les bases de données NoSQL
- Mettre en place une base de données MongoDB
- Modéliser les documents
- Manipuler les données avec mongoDB
- Effectuer des requêtes depuis des programmes Python
- Sécuriser une base de données MongoDB



50 heures

# CHAPITRE 1

## Découvrir les bases de données NoSQL

### Ce que vous allez apprendre dans ce chapitre :

- Définir le concept de bases de données NoSQL,
- Comparer les bases de données traditionnelles et NoSQL,
- Recenser les caractéristiques des NoSQL
- Identifier les bases de données NoSQL,
- Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)
- Comparer les différents types de bases de données NoSQL



05 heures



# CHAPITRE 1

## Découvrir les bases de données NoSQL

1. Définir le concept de bases de données NoSQL,
2. Comparer les bases de données traditionnelles et NoSQL,
3. Recenser les caractéristiques des NoSQL,
4. Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe),
5. Comparer les différents types de bases de données NoSQL



## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?



#### Des SGBD Relationnels ..... au NoSQL

- Les défis majeurs des **SGBDs** étaient toujours le stockage des données et la recherche des données,
- Les **SGBDR** sont adaptés à gérer des données bien structurées de types simples (chaines de caractères, entier, ...) et représentables sous forme de tables (colonnes => propriétés et lignes => données),
- Ils reposent sur le modèle relationnelle d'Edgard Codd et ont prouvé leur **efficacité** pour des décennies grâce à:

Une séparation logique et physique

Une forte structuration des données et un fort typage

Une représentation tabulaire

Un langage déclaratif (SQL)

Un ensemble de contraintes permettant d'assurer l'intégrité des données

Et une forte cohérence transactionnelle

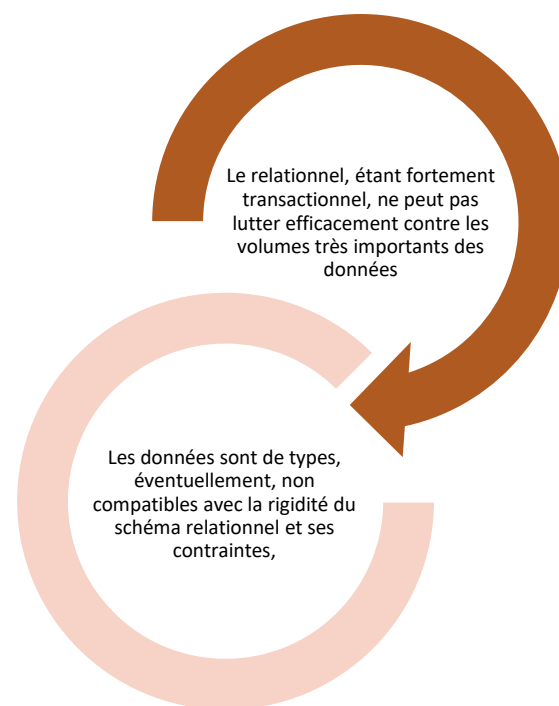


## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?

#### Des SGBD Relationnels ..... au NoSQL

Mais , les **SGBDR** ont montré leur limite face **aux 3V (*Volume, Velocity, Veracity*)** que caractérisent l'ère actuelle des données **(Big Data)**:



## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?



#### Des SGBD Relationnels ..... au NoSQL

- Le **NoSQL** (Not Only SQL) propose une nouvelle manière de gérer les données, sans respecter *forcement* le paradigme relationnel,
- Le **NoSQL** supporte de nouveaux types de données (xml, collections d'objets, triplets,... ) ,
- Cette approche propose de relâcher certaines contraintes lourdes du relationnel (structure des données, langage d'interrogation ou la cohérence) pour favoriser la distribution,
- Le **NoSQL** ne remplace pas les bases **SQL**, il les complète en apportant des avantages en terme de stockage réparti par exemple.



## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?

#### Définition

- Le **NoSQL** est un ensemble de technologies de **BD** reposant sur un modèle différent du modèle relationnel,
- Les Bases **NoSQL** sont le fruit du mouvement **NoSQL** apparu au milieu des années 2000,
- Le mouvement a initialement piloté les besoins Big Data des principaux acteurs du web **GAFA** ( Google, Amazone, Facebook, Apple,...):

Google avec sa base  
*Hbase*

Apple avec sa base ,,,,,,,

Facebook avec sa base  
*Cassandra*

Amazone avec sa base  
*DynamoDB*

- Les serveurs de données **NoSQL** se caractérisent par des architectures distribuées ce qui leur permettent de mieux répondre aux problématiques du big data.

## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?



#### Avantages du NoSQL

- Le format de la base **NoSQL** est basée essentiellement sur des paires clé-valeur beaucoup plus simple à mettre en œuvre,
- Il est possible de stocker directement des objets manipulés dans des langages de programmation comme des listes, des collections d'objets, des tableaux de valeurs,...
- Les bases de données **NoSQL** sont pour la plupart Open-source et ne possèdent pas de droits de licence,
- Il est très facile d'étendre une base de données **NoSQL** en rajoutant, tout simplement des serveurs,
- Les données sont regroupées par unités logiques et non dans des tables ce qui facilite la manipulation .
- **Par exemple**, pour avoir les informations d'un client qui a passé une commande donnée, on aura pas besoin de passer par des jointures entre les tables client et commande.

## 01 – Introduction aux Bases de données NoSQL

### C'est quoi une BD NoSQL?



#### Inconvénients du NoSQL

- Absence du concept de clé étrangère, ce qui veut dire qu'il n'y a pas de mécanisme pour vérifier la cohérence des données (il faut le faire au niveau de la programmation),
- **NoSQL** n'est pas adaptable aux applications basées sur des transactions sécurisées et fiables (Gestion bancaire par exemple),
- Les requêtes **SQL** et **NoSQL** ne sont pas compatibles.

# CHAPITRE 1

## Découvrir les bases de données NoSQL

1. Définir le concept de bases de données NoSQL,
2. **Comparer les bases de données traditionnelles et NoSQL,**
3. Recenser les caractéristiques des NoSQL
4. Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)
5. Comparer les différents types de bases de données NoSQL



### Comparaison entre les bases de données traditionnelles et NoSQL

#### Base de données SQL

Les données sont représentées sous forme de tables composées de n nombre de lignes de données, Elles respectent un schéma stricte et standard.

L'augmentation de la charge est gérée par l'augmentation du processeur, de la RAM, du SSD, etc. sur un seul serveur:  
Scalabilité (mise à l'échelle) verticale.

L'augmentation de la charge n'est pas pris en compte nativement, elle risque de compromettre l'intégrité transactionnelle de la BD.

#### Base de données NoSQL

les données sont représentées sous forme de collections de paires clé-valeur, de documents, de graphes, etc. Elles ne possèdent pas de définitions de schéma standard.

L'augmentation de la charge est gérée plutôt par l'ajout de serveurs supplémentaires : Scalabilité (mise à l'échelle) horizontale.

L'augmentation de la charge est automatique, si un serveur tombe en panne, il se remplace automatiquement par un autre serveur sans interruption du service.

## 01 – Introduction aux Bases de données NoSQL

### SQL ou NoSQL?



#### Comparaison entre les bases de données traditionnelles et NoSQL

##### Base de données SQL

Assure l'intégrité des données en assurant la conformité ACID (Atomicité, Cohérence, Isolation et Durabilité)

Recommandée par de nombreuses entreprises en raison de sa structure et de ses schémas prédéfinis.

Mais, ne convient pas au stockage de données hiérarchiques

La plus appropriée pour les applications transactionnelles à usage intensif étant plus stable et assurant l'atomicité, l'intégrité et la cohérence des données.

##### Base de données NoSQL

Repose sur les propriétés BASE (Basically Available, Soft state, Eventually Consistent) (voir le slide 23)

Recommandée pour les données semi structurées ou même non structurées

Hautement préférée pour les ensembles de données volumineux et hiérarchiques



## 01 – Introduction aux Bases de données NoSQL

### SQL ou NoSQL?



#### Résumons

### NoSQL

Offre des meilleures performances que **SQL** vu qu'il ne gère aucune règle de cohérence

Optimisé pour gérer d'énormes volumes de données avec performance

### SQL

Basé sur le langage de requête unifié (**SQL**) qui apporte une certaine uniformité entre les différentes bases **SQL**,

Offre une meilleure fiabilité et cohérence des données au détriment de la performance si les données deviennent volumineuses



Certes **NoSQL** et **SQL** permettent de stocker/rechercher de l'information, mais ils ne servent pas les mêmes objectifs ce qui rend toute comparaison subjective voir non justifiée.

# CHAPITRE 1

## Découvrir les bases de données NoSQL

1. Définir le concept de bases de données NoSQL,
2. Comparer les bases de données traditionnelles et NoSQL,
- 3. Recenser les caractéristiques des bases de données NoSQL**
4. Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)
5. Comparer les différents types de bases de données NoSQL

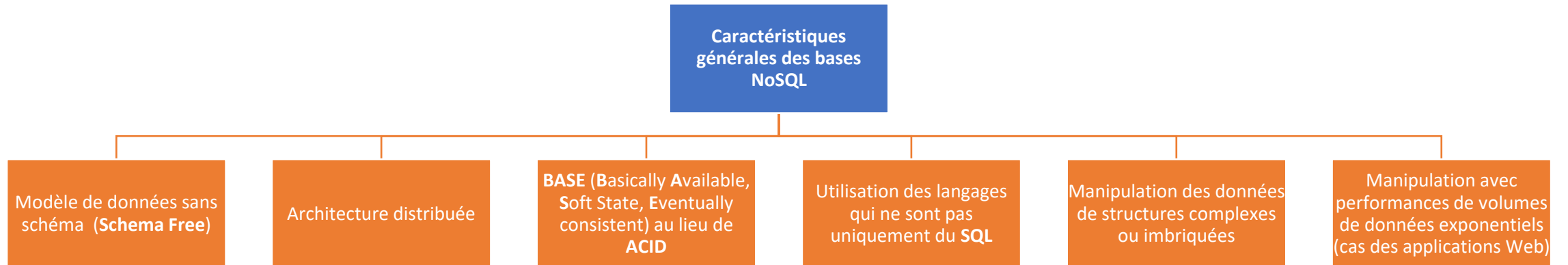


## 01 – Introduction aux Bases de données NoSQL

### Caractéristiques



### Caractéristiques générales des bases NoSQL



#### 1- Modèle Sans schéma (Schema Free)

- Dans un contexte relationnel, la création d'une base de données commence par la modélisation des entités et associations puis d'en déduire un schéma de la base,
- Cette démarche crée une rigidité dans la phase d'implémentation, puisqu'elle implique d'avoir une vision assez claire des évolutions de l'application dès le départ et au fil du temps, ce qui n'est pas souvent le cas de nos jours !!
- Les bases de données **NoSQL** s'appuient sur des données dénormalisées, non modélisées par des relations, mais plutôt par des enregistrements (ou documents) intégrés, il est donc possible d'interagir sans utiliser de langages de requêtes complexe.

# 01 – Introduction aux Bases de données NoSQL

## Caractéristiques



### 1- Modèle Sans schéma (Schema Free)

- Exemple

SQL

Posts(id,titre)  
Commentaires(id, #idPosts,texte)

Posts

Id	titre
P1	Titre1
P2	Titre2

Commentaires

Id	idPosts	texte
C1	P1	comment1
C2	P2	comment2
C3	P1	comment3

NoSQL

Posts(id,titre,commentaires)

Posts

P1	Titre1	Comment1 Comment3
P2	Titre2	Comment2

### 2- Architecture distribuée

- Le volume de données à stocker ainsi que les traitements demandés par les organismes modernes, ne peuvent plus être satisfaits sur une seule machine quelque soit sa performance, même en utilisant un réseau de machine l'interconnexion entre machines rendent les traitements très lents,



**Solution** : un patron d'architecture propose de distribuer les traitements (**le travail/la charge**) sur plusieurs machine puis regrouper les résultats de chaque machine et les agrège dans un résultat final → Apparition de MapReduce en 2003,

- **MAIS**, les bases de données traditionnelles ne permettent pas l'implantation d'un tel patron d'architecture,
- Les bases **NoSQL** sont conçues pour distribuer les données et traitements associés sur de multiple nœuds(serveurs) → partitionnement horizontal,



**Problème** : impossible d'avoir en même temps une disponibilité des données satisfaisante, une tolérance au partitionnement et une meilleure cohérence des données,

- Il faut toujours condamner un aspect en faveur des autres !

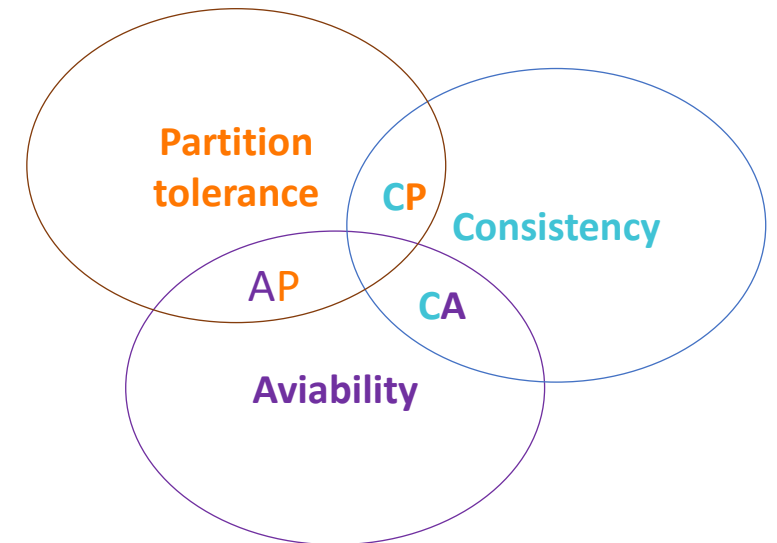
## 2- Architecture distribuée

### Théorème de CAP (Consistency, Availability, Partition tolerance)

Dans toute base de données, on ne peut respecter au plus que deux propriétés parmi les trois propriétés suivantes: la cohérence, la disponibilité et la distribution

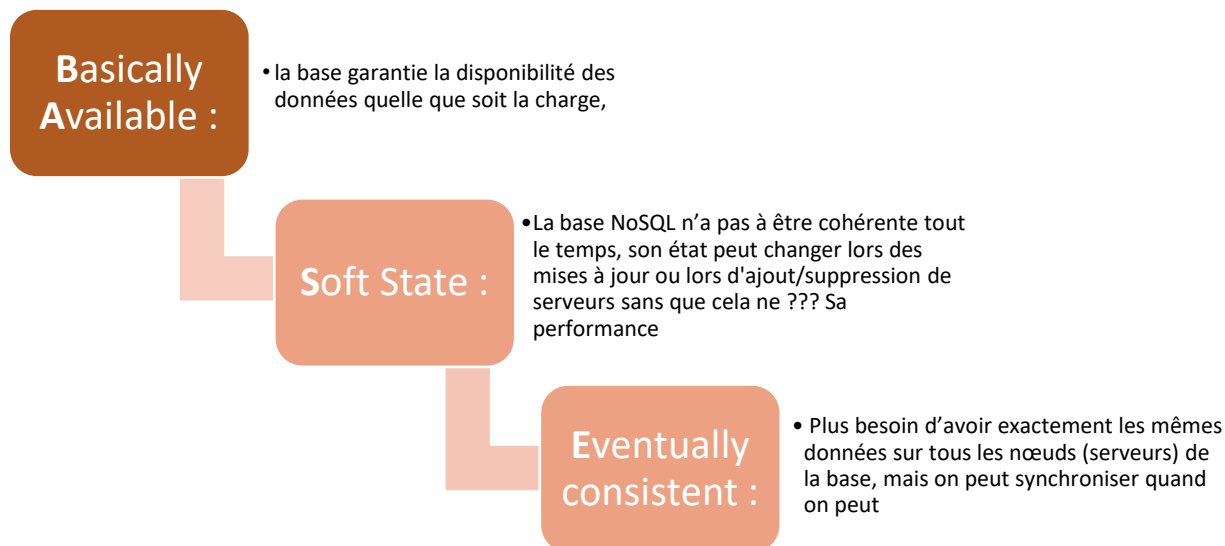
#### A savoir :

- **Consistency** (cohérence): tous les nœuds(serveurs) sont à jour sur les données au même moment,
- **Availability** (disponibilité): la perte d'un nœud(serveur) n'empêche pas le système de fonctionner et de servir l'intégralité des données,
- **Partition tolerance** (résistance au partitionnement): chaque nœud(serveur) doit pouvoir fonctionner de manière autonome,



### 3- BASE vs ACID

- Les propriétés **ACID** ne sont pas partiellement ou totalement applicables dans un contexte **NoSQL**,
- Les bases **NoSQL** reposent, par contre, sur les propriétés **BASE**:



Les bases **NoSQL** privilégient la disponibilité à la cohérence : **AP** (Availability + Partition tolerance) plutôt que **CP** (Consistency + Partition tolerance)



# CHAPITRE 1

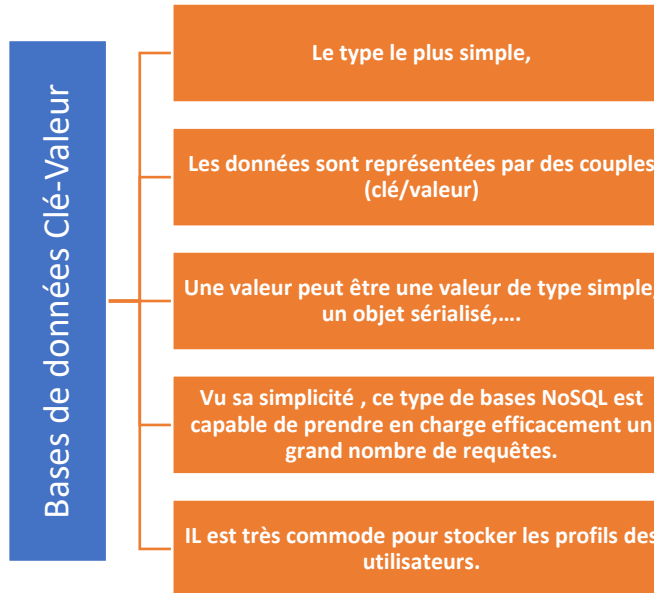
## Découvrir les bases de données NoSQL

1. Définir le concept de bases de données NoSQL,
2. Comparer les bases de données traditionnelles et NoSQL,
3. Recenser les caractéristiques des bases de données NoSQL
- 4. Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)**
5. Comparer les différents types de bases de données NoSQL



### Les quatre types des bases NoSQL

#### 1. Bases de données Clé-Valeur



#### Exemple :

Clé	Valeur
Ahmed	<b>type:</b> Formateur; <b>spec:</b> Dev digital; <b>modules :</b> M102, M104, M106, M203
Sanaa	<b>type:</b> Stagiaire; <b>filière:</b> Dev digital; <b>groupe :</b> DD203; <b>niveau :</b> 2A
Kamal	<b>type:</b> Stagiaire; <b>filière:</b> Infra digitale; <b>niveau :</b> 1A

## 01 – Introduction aux Bases de données NoSQL

### Types des bases NoSQL



#### Les quatre types des bases NoSQL

##### 1. Bases de données Clé-Valeur

##### Exemples de bases de données Clé/valeur

**Dynamo DB**  
Amazon



**Berkeley DB** ou **BDB** solution d'oracle  
GMAIL, RPM, SVN,...



**Voldemort** de LinkedIn  
(et pas le sorcier de Harry Potter 😊)

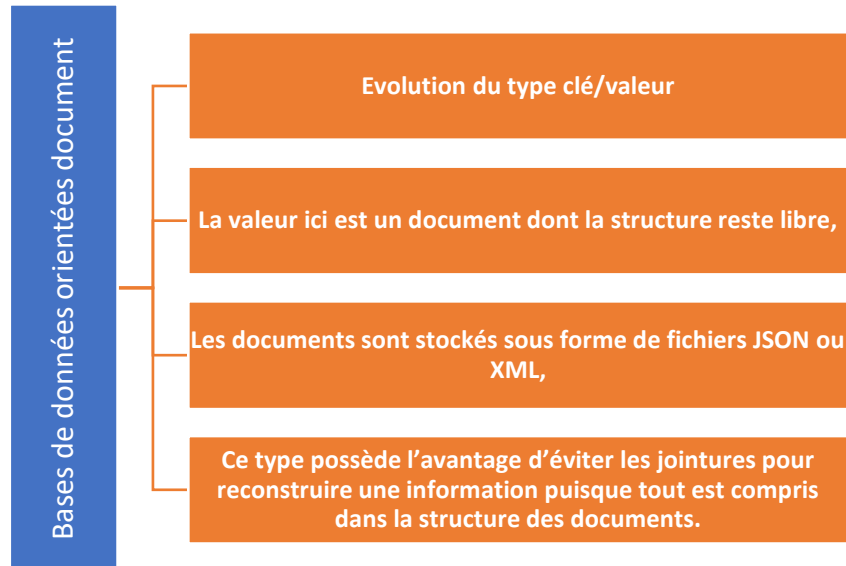


**Riak DB**  
Apache



### Les quatre types des bases NoSQL

#### 2. Bases de données orientées Document



#### Exemples

Clé

Valeur

Ahmed

```
{"type": "Formateur",  
"spec": "Dev digital",  
"modules": ["M102",  
"M104", "M106",  
"M203"]}
```

Sanaa

```
{"type": "Stagiaire",  
"filiere": "Dev digital",  
"groupe": "DD203",  
"niveau": "2A",  
"option": "Mobile"}
```

Sanaa

```
{"type": "Stagiaire",  
"filiere": "Infra digital",  
"niveau": "1A"}
```

## 01 – Introduction aux Bases de données NoSQL

### Types des bases NoSQL



#### Les quatre types des bases NoSQL

#### 2. Bases de données orientées Document

##### Exemples de bases de données orientées Document

**Mongo DB** de SourceForge  
Adobe, Bosch, Cisco, eBay,...



**CouchDB** d'Apache  
Disney, PayPal, Ryanair,....



**RavenDB**  
Plateformes .Net/Windows



**Cassandra** de FaceBook  
NY Times, eBay, Sky, Pearson Education



### Les quatre types des bases NoSQL

#### 3. Bases de données orientées Colonne

Bases de données orientées Colonne

Ce type change le paradigme traditionnel de la représentation des données en lignes,

Il rend possible de focaliser les requêtes sur les colonnes importantes sans avoir à traiter les données des autres colonnes (jugées alors inutile pour la requête),

Ce type est adapté aux systèmes avec de gros calculs analytiques ( comptage, moyenne, somme,...)

## 01 – Introduction aux Bases de données NoSQL

### Types des bases NoSQL



### Les quatre types des bases NoSQL

#### 3. Bases de données orientées Colonne

##### Exemple

- Représentation traditionnelle (représentation en ligne)

Id	Type	Spécialité	Niveau	Filière	Groupe	Option	Module
Ahmed	Formateur	Dev Digital					M102, M104, M106, M202
Sanaa	Stagiaire		2A	Dev Digital	DD203	Mobile	
Kamal	Stagiaire		1A	Infra Digitale			
Laila	Formateur	Infra Digitale					M105,M107,M201

## 01 – Introduction aux Bases de données NoSQL

### Types des bases NoSQL



### Les quatre types des bases NoSQL

#### 3. Bases de données orientées Colonne

##### Exemple

Id	Type	Spécialité	Niveau	Filière	Groupe	Option	Module
Ahmed	Formateur	Dev Digital					M102, M104, M106, M202
Sanaa	Stagiaire		2A	Dev Digital	DD203	Mobile	
Kamal	Stagiaire		1A	Infra Digitale			
Laila	Formateur	Infra Digitale					M105,M107,M201

- Exemples de représentations par colonnes

Id	Type
Ahmed	Formateur
Sanaa	Stagiaire
Kamal	Stagiaire
Laila	Formateur

Id	Filière
Sanaa	Stagiaire
Kamal	Stagiaire

Id	Module
Ahmed	M102
Ahmed	M104
Ahmed	M106
Ahmed	M202
Laila	M105
Laila	M107
Laila	M201

Id	Option
Sanaa	Mobile



### Les quatre types des bases NoSQL

#### 3. Bases de données orientées Colonnes

##### Exemples de bases de données orientées Colonnes

**BigTable DB de Google**



**HBase d'Apache**



**SparkSQL d'Apache**



**Elasticsearch db**



### Les quatre types des bases NoSQL

#### 4. Bases de données orientées Graphe

##### Bases de données orientées Graphe

Il est basé sur la théorie des graphes et ses fondements mathématiques,

Ce type est conçu principalement pour les données fortement interconnectées (comme les données des réseaux sociaux) avec un nombre indéterminé de relations entre elles,

En d'autres termes, c'est le type le mieux approprié pour modéliser le monde réel

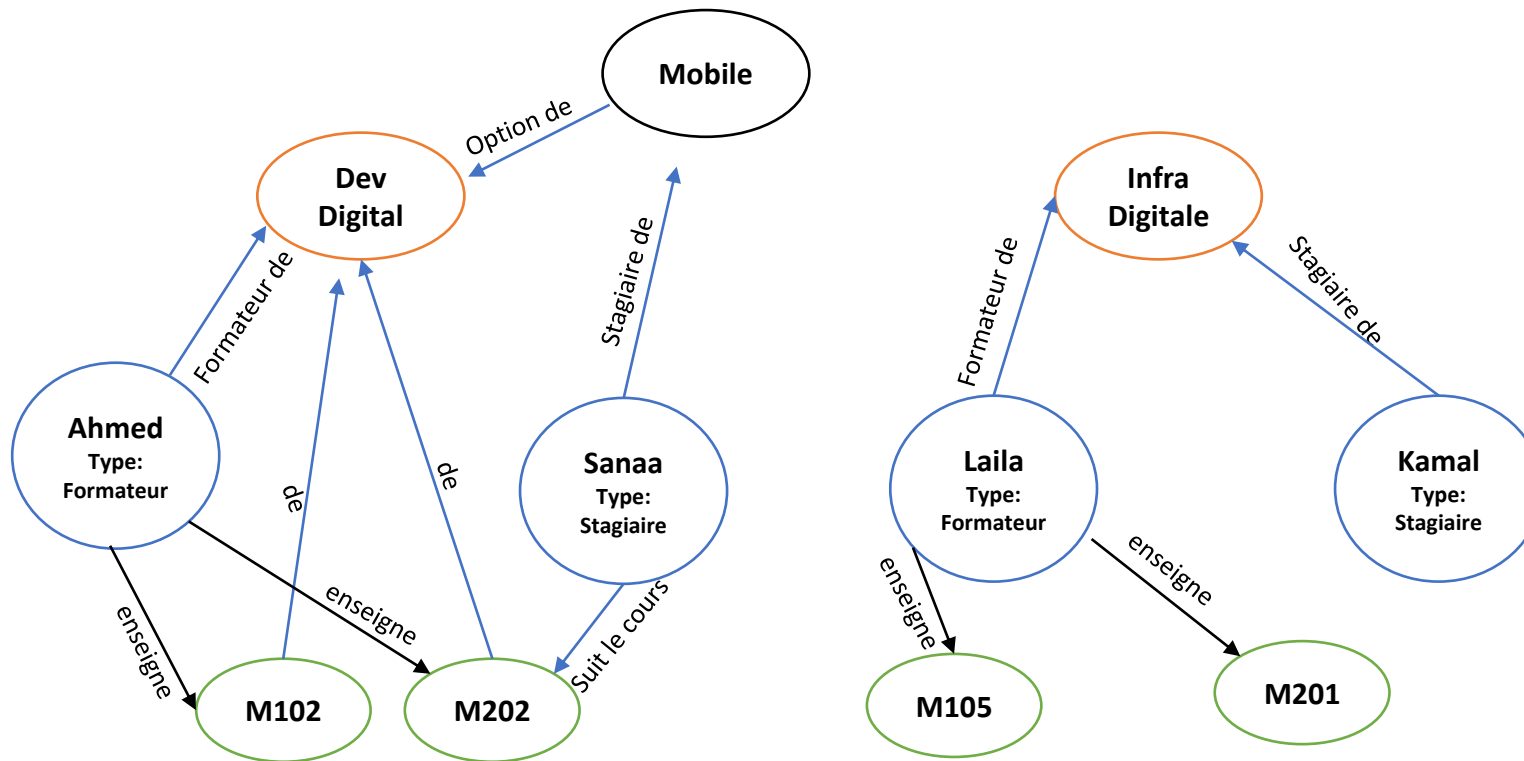
**Exemple** : la principale solution est Neo4j



### Les quatre types des bases NoSQL

#### 3. Bases de données orientées Graphe

##### Exemple



# CHAPITRE 1

## Découvrir les bases de données NoSQL

1. Définir le concept de bases de données NoSQL,
2. Comparer les bases de données traditionnelles et NoSQL,
3. Recenser les caractéristiques des bases de données NoSQL
4. Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)
5. **Comparer les différents types de bases de données NoSQL**



## 01 – Introduction aux Bases de données NoSQL

### Comparaison des types de bases NoSQL



	<b>Riak (Clé/Valeur)</b>	<b>MongoDB (Document)</b>	<b>Cassandra (Document)</b>	<b>HBase (Colonne)</b>
<b>Cout</b>	+	++	++	++
<b>Cohérence</b>	+	++	+	+
<b>Disponibilité</b>	++	+	++	++
<b>Langages d'Interrogation</b>	++	++	+	++
<b>Fonctionnalités</b>	Solution hautement disponible avec un langage de requêtes performant Approprié au stockage dans le cloud	La solution la plus populaire, structure souple et bonnes performances Favorise la cohérence à la disponibilité	solution mature, populaire, Excellente solution pour grands volumes de données besoins de bases distribuées Mais langage trop réduit	Destinée aux données volumineuses, Privilégie le langage et la disponibilité à la cohérence des données



### Pourquoi choisir MongoDB

**MongoDB** vient du terme **Humongous DB** qui veut dire une base de données **gigantesque**

C'est la base **NoSQL** la plus populaire grâce aux points forts suivants :

Open-source

Orientée  
Document

Assure une  
haute  
performance et  
disponibilité

Une mise à  
échelle  
automatique  
(scalabilité  
automatique)

Se base sur CP  
du théorème  
CAP = Opte  
pour la  
cohérence et la  
résistance au  
partitionnement

Offre une  
interface facile  
avec des  
langages  
courants (Java,  
Javascript, PHP,  
etc.)

Fonctionne sur  
plusieurs  
plateformes  
(machine  
virtuelle, cloud,  
etc.)

Conserve et  
assure les  
fonctionnalités  
essentielles des  
Bases de  
données  
Relationnelles

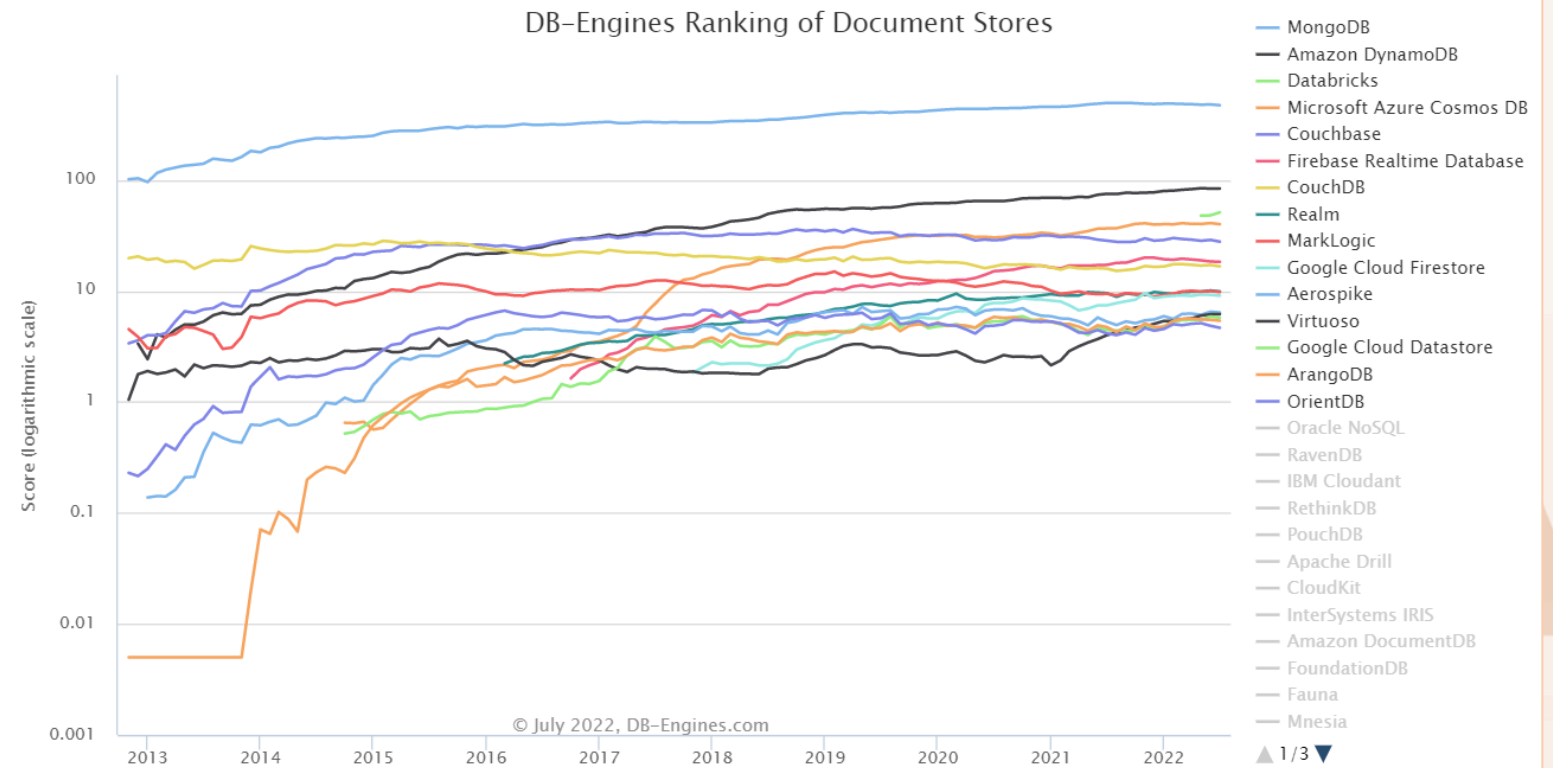
## 01 – Introduction aux Bases de données NoSQL MongoDB



### Pourquoi choisir MongoDB

- La popularité de **MongoDB** vient du fait qu'elle est fortement utilisée par les développeurs partout dans le monde,
- Elle est appréciée également d'être facilement intégrable dans toute application gérant des documents/objets.

### DB-Engines Ranking - Trend of Document Stores Popularity



[https://db-engines.com/en/ranking\\_trend/document+store](https://db-engines.com/en/ranking_trend/document+store)

## CHAPITRE 3

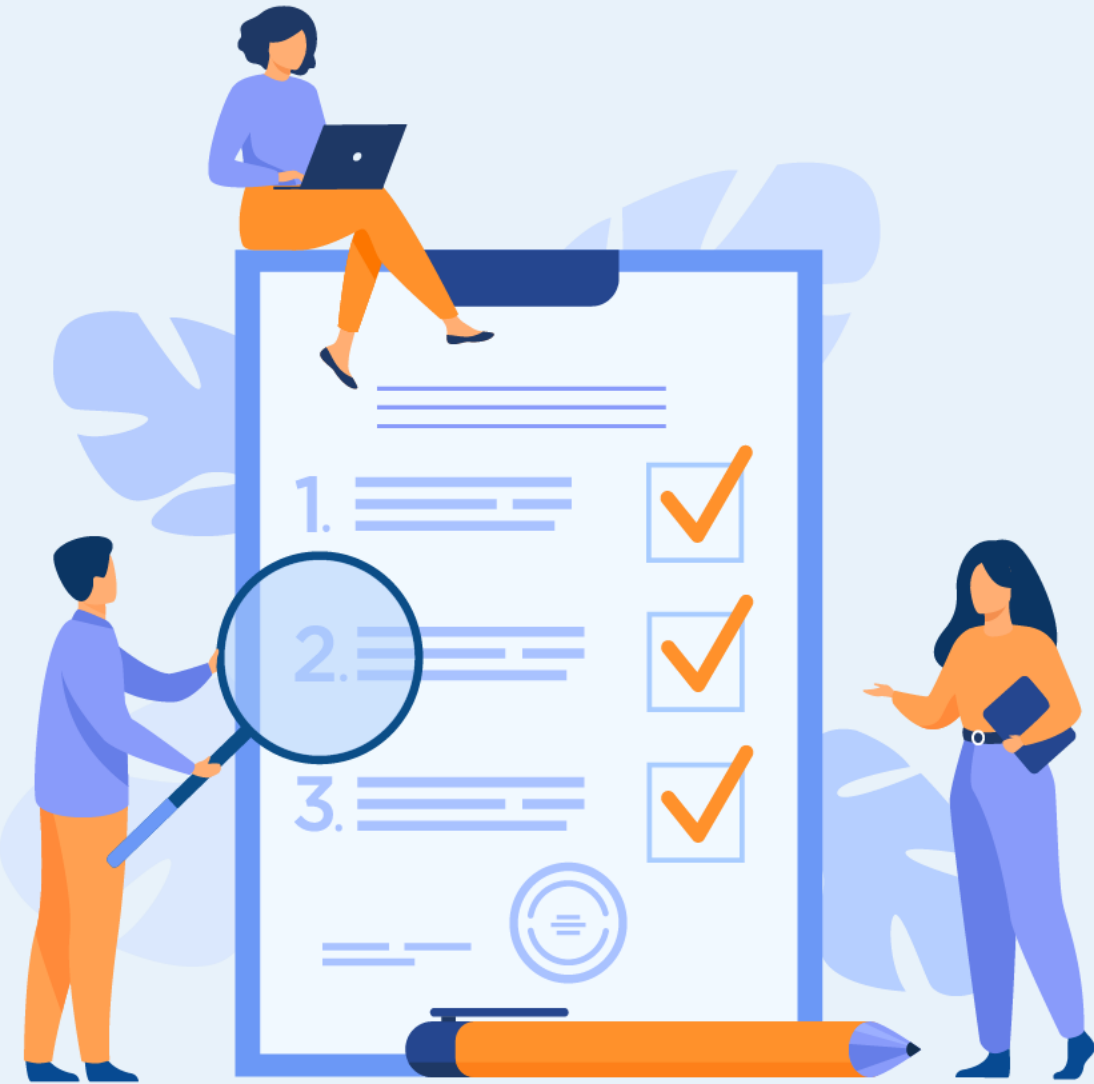
### Modéliser les documents

#### Ce que vous allez apprendre dans ce chapitre :

- Structurer un document JSON,
- Recenser les différences entre modéliser pour MongoDB versus une base de données relationnelles,
- Modéliser les liens,
- Utiliser des espaces de noms, des collections et des documents,



07,5 heures





## CHAPITRE 3

### Modéliser les documents

1. Structurer un document JSON,
2. Identifier la différences entre la modélisation pour MongoDB versus une base de données relationnelles,
3. Modéliser les liens,
4. Utiliser des espaces de noms, des collections et des documents



## 01 – Structure d'un document JSON

### Définition



JSON

JSON (JavaScript Object Notation) est un format standard de représentation logique de données, hérité de la syntaxe de création d'objets en JavaScript

Utilisé pour structurer et transmettre des données sur des sites web (par exemple, envoyer des données depuis un serveur vers un client ou vice versa)

C'est un format réputé texte léger (pas trop de caractères de structuration), lisible par les humains avec l'extension **.json**

Bien que JSON puise sa syntaxe du JavaScript, il est indépendant de tout langage de programmation. Il peut ainsi être interprété par tout langage à l'aide d'un parser

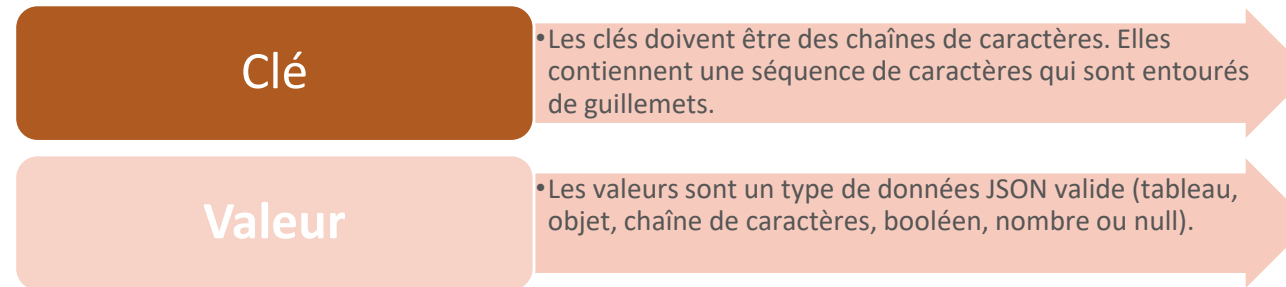
# 01 – Structure d'un document JSON

## Syntaxe de base



### Objet JSON

- Un objet **JSON** se base sur deux éléments essentiels : **Les clés** et **les valeurs**.



- Un objet **JSON** commence et se termine par des accolades {}.
- Il peut contenir plusieurs paires **clé/valeur**, séparées par une virgule., La clé est suivie de « : » pour la distinguer de la valeur.

# 01 – Structure d'un document JSON

## Syntaxe de base



### Types de valeurs JSON

- **JSON** supporte en principe trois types de valeurs
  - **Primitif** : nombre, booléen, chaîne de caractères, null,
  - **Objet** : liste de paires "**clé**": valeur entrés entre accolades, séparés par des virgules.

Exemple :

```
"stagiaire" : {"prenom": "Amina", "filiere": "Dev Digital ", "niveau": "1A" }
```

- **Tableau (Array)**: ensemble ordonné de valeurs, entouré de crochets [] ces valeurs sont séparées par une virgule,

Exemple:

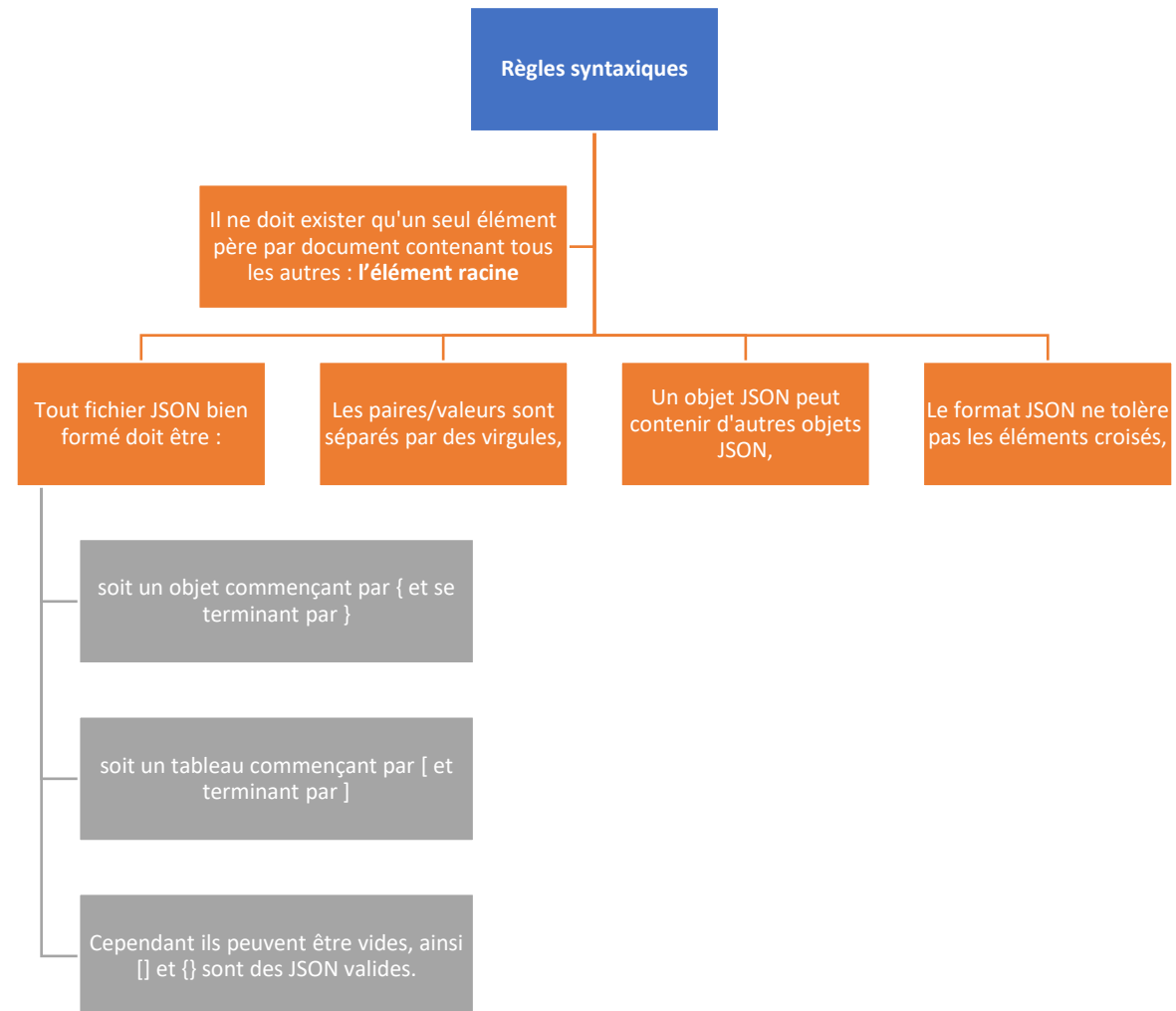
```
"stagiaires" : [  
    {"prenom": "Amina", "filiere": "Dev Digital ", "niveau": "1A" },  
    {"prenom": "Kamal", "filiere": "Infra Digitale ", "niveau": "1A" },  
    {"prenom": "Sanaa", "filiere": "Infra Digitale ", "niveau": "1A" }  
]
```

# 01 – Structure d'un document JSON

## Syntaxe de base



### Règles syntaxiques



## 01 – Structure d'un document JSON

### Exemple



#### Exemple de tableau JSON

```
{  
  [  
    { _id: 12,  
      "prenom": "Kamal",  
      "type": "Stagiaire",  
      "filier": "Dev Digital",  
      "groupe": "DD203",  
      "niveau": "2A",  
      "option": "Mobile"},  
    {  
      _id: 17,  
      "prenom": "Amina",  
      "filier": "Infra Digitale",  
      "niveau": "1A",  
    }  
  ]  
}
```

# 01 – Structure d'un document JSON

## Usages de JSON



### Chargements Asynchrones

- Avec la montée en flèche des chargements asynchrones tels que l'**AJAX** le format JSON s'est montré adapté que XML,

### APIs

- Des sociétés telles que Twitter, Facebook ou LinkedIn, offrent essentiellement des services basés sur l'échange, d'informations, et font preuve d'un intérêt grandissant envers les moyens possibles pour distribuer ces données à des tiers,
- JSON domine le domaine des APIs au détriment du format XML qui avait été pionnier,

### Bases de données

- Très utilisé dans le domaine des bases de données NoSQL (MongoDB, CouchDB, Riak...),
- Il est également, possible de soumettre des requêtes à des SGBDR et de récupérer une réponse en JSON

## CHAPITRE 3

### Modéliser les documents

1. Structurer un document JSON,
2. **Identifier la différences entre la modélisation pour MongoDB versus une base de données relationnelles,**
3. Modéliser les liens,
4. Utiliser des espaces de noms, des collections et des documents





## 02 – Modélisation MongoDB vs BDR

### Normalisation vs Dénormalisation



Dans le cas des bases de données relationnelles, la modélisation des données repose sur la normalisation des structures de données afin d'éviter toute duplication de l'information,

Une fois les structures des données sont normalisées, on procède à la création des requêtes se basant essentiellement sur les jointures engendrées par la normalisation,

Les bases de données **NoSQL**, se caractérisent par l'abandon des jointures et transactions au profit d'un temps de réponse court et des performances optimales,



*Comment, alors, concevoir un schéma de données approprié à ce contexte afin de mieux interroger les données?*



**Réponse :** La dénormalisation

## 02 – Modélisation MongoDB vs BDR

### Normalisation vs Dénormalisation



#### Définition et objectif

---

La **dénormalisation** consiste à regrouper plusieurs tables liées par des références, en une seule table, en éliminant les jointures, elle favorise la redondance des données

---

Son objectif est d'améliorer les performances de la base de données en recherche sur les tables considérées

---

Elle vise également à améliorer les performances de la base de données en recherche sur les tables considérées

---

**Dénormaliser** consiste à dupliquer les données (ou une partie de données )d'une structure de données dans une autre

---

## 02 – Modélisation MongoDB vs BDR

### MongoDB et la dénormalisation



- Les bases de données relationnelles s'appuient sur le modèle relationnel,
- **La normalisation** est une contrainte obligatoire pour la validation du modèle des données,
- **La normalisation** au niveau conceptuel impose la structure des données de la base ( Modèle conceptuel des données)
- Le passage du Modèle Conceptuel des Données au Modèle Logique des Données détermine la nature des relations entre les données (liaison par clé primaire au niveau de la table et par clé étrangère entre les tables),
- Les requêtes de la manipulation des données (Recherche opérations CRUD) doivent respecter le modèle logique préétabli,
- Le modèle des données impose la manière d'écriture des requêtes sur les données

## 02 – Modélisation MongoDB vs BDR

### MongoDB et la dénormalisation



- La différence majeure au niveau de la modélisation des données entre les bases de données relationnelles et celles orientées **Document** est la dénormalisation des données,
- En effet, les données ne sont pas soumises aux contraintes de normalisation:
  - **Attributs non atomiques** : première forme normale non respectée,
  - **Données redondantes** : deuxième forme normale non respectée,
  - ...
- Dans l'absence totale ou partielle d'un modèle des données, c'est la nature des relations qui exige plutôt le type de requêtes qu'on désire élaborer sur les données,
- On détermine le schéma des données suivant l'utilisation des données par l'application c.-à-d. les requêtes

## 02 – Modélisation MongoDB vs BDR

### Récapitulation



Base de données relationnelle	MongoDB (Base de données orientée documents)
Base de données	Base de données
Table	Collection
Enregistrement	Document
Schéma de données fixe	Schéma de données flexible
Les enregistrements de la table doivent avoir le même ensemble de champs	Les documents d'une même collection n'ont pas besoin d'avoir le même ensemble de champs
Le type de données d'un champ est fixe pour tous les enregistrements de la table	Le type de données d'un champ peut différer d'un document à l'autre d'une collection.
Les requêtes sur les données doivent respecter un modèle des données logique fixe	Le type d'utilisation des données (les requêtes) détermine le schéma des données

## CHAPITRE 3

### Modéliser les documents

1. Structurer un document JSON,
2. Identifier la différences entre la modélisation pour MongoDB versus une base de données relationnelles,
- 3. Modéliser les liens,**
4. Utiliser des espaces de noms, des collections et des documents



### Les types de relations entre les données sous MongoDB

- MongoDB détermine deux types de relations entre les données :
  - **Les relations d'enchâssement (embedding) :**
    - L'imbrication d'un (ou d'une partie) d'un document dans un autre, on parle de document autonome,
    - L'imbrication permet d'éviter de faire des jointures: inutile de faire des jointures pour restituer l'information qui n'est pas dispersée sur plusieurs entités (tables en relationnel),
    - On utilise l'imbrication des documents (**embedding**) quand les documents sont très petits et n'ont pas tendance à grandir dans le futur. La taille des documents ne doit pas dépasser 16Mb,
    - Adéquates pour les contextes qui privilégient la recherche à la mise à jour,

- MongoDB détermine deux types de relations entre les données :
  - **Les relations de liaisons (Linking) :**
    - La duplication de l'identifiant d'un document dans un autre document,
    - Reprend, en quelque sorte, le concept de jointure entre les tables relationnelles,
    - N'est privilégiée que dans le contexte de relations plusieurs-plusieurs:

**Exemple :** Commande  $\leftrightarrow$  Produit

Un produit peut être commandé plusieurs fois et une commande peut contenir plusieurs produits,

Une imbrication des Produits dans la commande aura de gros impacts sur les mises à jour (tous les produits à mettre à jour !)



### L'enchassement (embedding)

- Pour une relation entre deux documents A et B, cela consiste à imbriquer partiellement ou totalement le document B dans le document A.
- Ce modèle de relations dénormalisés permet aux applications de récupérer et de manipuler des données associées en une seule opération de base de données,
- **Exemple:**

```
{  
  "_id": "1234",  
  "prenom": "Amina",  
  "filier":  
    { "_id": "DD",  
      "intitule": "Développement digital"  
    }  
}
```

} Document imbriqué

### La liaison (Linking)

- Les relations de liaison permettent d'inclure les liens ou des références d'un document dans un autre,
- Ce modèle de relations récupère les données en deux étapes:
  - une première requête pour récupérer l'identifiant,
  - une deuxième requête pour récupérer les données de l'autre côté de la relation.
- **Exemple:**

```
{ "_id": "DD", "intitule": "développement digital" }  
{  
  "_id": "1234",  
  "prenom": "Amina",  
  "filiere": "DD"  
}
```

### Enchâssement vs liaison

- On utilise l'imbrication des documents (**embedding**) quand les documents sont très petits et n'ont pas tendance à grandir dans le futur. La taille des documents ne doit pas dépasser **16Mb**,
- Si la taille de la collection ou les documents va augmenter dans le futur, il vaut mieux opter pour la liaison des documents.

## CHAPITRE 3

### Modéliser les documents

1. Structurer un document JSON,
2. Identifier la différences entre la modélisation pour MongoDB versus une base de données relationnelles,
3. Modéliser les liens,
4. **Utiliser des espaces de noms, des collections et des documents**



#### Définition

- Les bases de données sont des groupes de collections stockées sur le disque à l'aide d'un seul ensemble de fichiers de données,
- Un espace de nom **namespace** est la concaténation du nom de la base de données et des noms de collection, séparés par un point,

#### Exemple:

**myDB.Stagiaires** → le nom de la base de données(myDB) suivi du nom de la collection

- Les collections sont des conteneurs pour les documents

#### Définition

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:

- String** : chaîne de caractères, les strings BSON sont en UTF-8,

Exemple :

```
{"_id": "1234", "prenom": "Amina"}
```

- Integer**: entier qui peut être stocker le type de données entier sous deux formes : entier signé 32 bits et entier signé 64 bits.

Exemple :

```
{"_id": "1234", "prenom": "Amina", "age": 19}
```

#### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:

- Double** : utilisé pour stocker les valeurs à virgule flottante. ,

Exemple :

```
{"_id":"1234","prenom":"Amina","moyBaccalaureat":14.25}
```

- Boolean**: utilisé pour stocker vrai ou faux

Exemple:

```
{"_id":"1234","prenom":"Amina","moyBaccalaureat":14.25,"admis":true}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - **Date :**
    - stocke la date sous forme de millisecondes (entier 64bits),
    - Le type de données BSON prend généralement en charge la date et l'heure UTC et il est signé, les valeurs négatives représentent les dates antérieures à 1970,
    - La date peut être exprimée sous forme de string : **Date()** ou d'objet date **new Date()**



### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:

- Date :**

#### Exemple

```
{  
  "prenom":"Ahmed",  
  "niveau":"1A",  
  "filier": "Dev digital",  
  "DateInscription_1":Date(),  
  "DateInscription_2":new Date()  
}
```

#### Vue JSON

```
{  
  "_id" : ObjectId("62dd1dff1a19f3d7ecc66252"),  
  "prenom" : "Ahmed",  
  "niveau" : "1A",  
  "filier" : "Dev digital",  
  "DateInscription_1" : "Sun Jul 24 2022 11:25:03 GMT+0100",  
  "DateInscription_2" : ISODate("2022-07-24T10:25:03.613+0000")  
}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans BSON, le format codé binaire de JSON, elle prend en charge divers types de données à savoir:
  - **Null** : utilisé pour stocker la valeur null,

Exemple :

```
{  
  "_id": "1234",  
  "prenom": "Amina",  
  "telephone": null  
}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - **Données binaires** : utilisé pour stocker les données binaires,

Exemple :

```
{  
  "_id": "1234",  
  "prenom": "Amina",  
  "telephone": null,  
  "BinaryValues": "10010001",  
}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - **Array**: Ensemble des valeurs pouvant être de types de données identiques ou différentes. Dans **MongoDB**, le array est créé à l'aide de crochets ([]).

Exemple:

```
{
  "_id": "1234",
  "prenom": "Kamal",
  "niveau": "2A",
  "option": "Mobile",
  "skills": [
    "python",
    "javascript",
    "php" ]
}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - ObjectId (Id d'objet)**: pour chaque nouveau document crée dans une collection, MongoDB crée automatiquement un identifiant d'objet unique `_id` s'il n'est pas crée explicitement,

**Exemple: Soient les deux documents suivants:**

```
{"prenom":"Ahmed","niveau":"1A","filiere":"Dev digital"}  
{"_id":1234,"prenom":"Alaa","niveau":"2A","filiere":"Dev digital"}
```

Vue table

Stagiaire > filiere			
_id	prenom	niveau	filiere
1234	Alaa	2A	Dev digital
62dd137d1a19f3d7ecc66250	Ahmed	1A	Dev digital

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - ObjectId (Id d'objet)**: pour chaque nouveau document crée dans une collection, MongoDB crée automatiquement un identifiant d'objet unique `_id` s'il n'est pas crée explicitement,

**Exemple: Soient les deux documents suivants:**

```
{"prenom":"Ahmed","niveau":"1A","filier":"Dev digital"}  
{"_id":1234,"prenom":"Alaa","niveau":"2A","filier":"Dev digital"}
```

Vue JSON

```
{  
  "_id" :  
  ObjectId("62dd128a1a19f3d7ecc6624f"),  
  "prenom" : "Ahmed",  
  "niveau" : "1A",  
  "filier" : "Dev digital"  
}  
  
{  
  "_id" : 1234.0,  
  "prenom" : "Alaa",  
  "niveau" : "2A",  
  "filier" : "Dev digital"  
}
```

### Types de données

- Dans **MongoDB**, les documents sont stockés dans **BSON**, le format codé binaire de **JSON**, elle prend en charge divers types de données à savoir:
  - **ObjectId (Id d'objet)**: pour chaque nouveau document crée dans une collection, MongoDB crée automatiquement un identifiant d'objet unique `_id` s'il n'est pas crée explicitement,

- **Exemple:**

```
{
  "_id" :
  ObjectId("62dd128a1a19f3d7ecc6624f"),
  "prenom" : "Ahmed",
  "niveau" : "1A",
  "filier" : "Dev digital"
}
{
  "_id" : 1234.0,
  "prenom" : "Alaa",
  "niveau" : "2A",
  "filier" : "Dev digital"
}
```

← Identifiant généré automatiquement par MongoDB

← Identifiant crée explicitement par MongoDB

## CHAPITRE 5

### Effectuer des requêtes depuis des programmes Python

Ce que vous allez apprendre dans ce chapitre :

- Présentation et installation de pymongo,
- Connexion des bases de données avec le serveur MongoDB,
- Création des requêtes :
  - Requêtes simples,
  - Création des indexs,
  - Requêtes d'agrégation,
  - Requêtes de modifications.

 07,5 heures





## CHAPITRE 5

# Effectuer des requêtes depuis des programmes Python

1. **Présentation et installation de pymongo,**
2. Connexion des bases de données avec le serveur MongoDB,
3. Création des requêtes :
  - Requêtes simples,
  - Création des indexs,
  - Requêtes d'agrégation,
  - Requêtes de modifications.

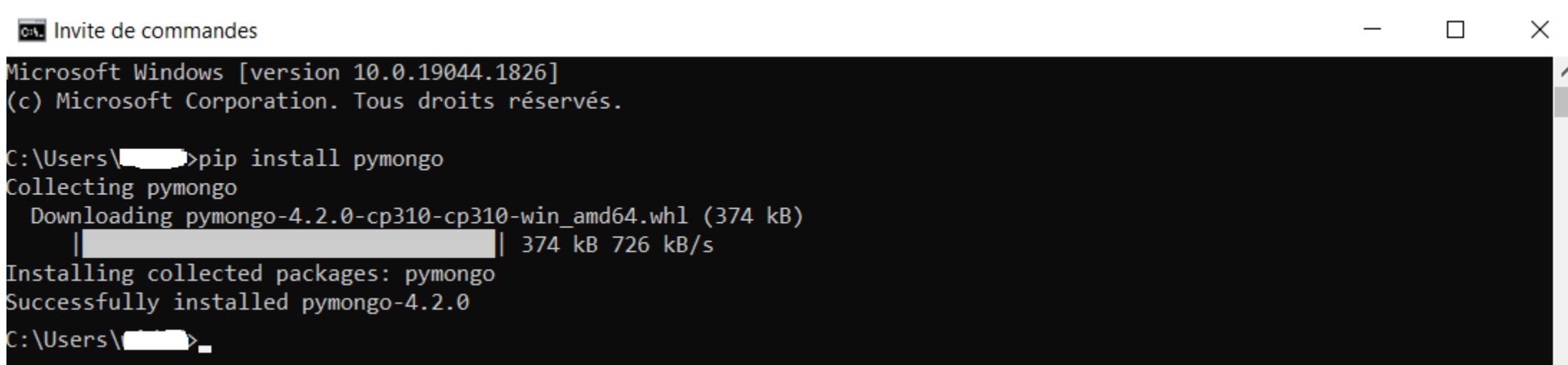


## 05 – Effectuer des requêtes depuis des programmes Python

### Présentation et installation de pymongo,

#### Installation de pymongo

- **PyMongo** est une librairie Python native contenant des outils pour travailler avec MongoDB,
- **PyMongo** est maintenue par les développeurs de MongoDB officiel ce qui en fait la référence dans Python<sup>1</sup>,
- Pour installer la librairie, il faut saisir la commande suivante dans un terminal (invité de commande par exemple)



```
C:\> Invite de commandes

Microsoft Windows [version 10.0.19044.1826]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\> pip install pymongo
Collecting pymongo
  Downloading pymongo-4.2.0-cp310-cp310-win_amd64.whl (374 kB)
    | 374 kB 726 kB/s
Installing collected packages: pymongo
Successfully installed pymongo-4.2.0
C:\Users\>
```

1: <https://pymongo.readthedocs.io/en/stable/>

## CHAPITRE 5

### Effectuer des requêtes depuis des programmes Python

1. Présentation et installation de pymongo,
2. **Connexion des bases de données avec le serveur MongoDB,**
3. Création des requêtes :
  - Requetes simples,
  - Création des indexs,
  - Requetes d'agrégation,
  - Requetes de modifications.



## 05 – Effectuer des requêtes depuis des programmes Python

### Connexion au serveur MongoDB

#### Connexion au serveur MongoDB

- La première étape consiste à créer une connexion avec le serveur **MongoDB**,
- Pour effectuer cette connexion on utilise **MongoClient**, qui se connecte, par défaut, à l'instance **MongoDB** s'exécutant sur localhost:**27017** si aucune chaîne de connexion n'est spécifiée,
- On commence par importer la classe **MongoDB** du module **pymongo**,
- Puis on affiche l'objet :

```
C:\Users\>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from pymongo import MongoClient
>>> client = MongoClient()
>>> print(client)
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
>>>
```

## 05 – Effectuer des requêtes depuis des programmes Python

### Connexion au serveur MongoDB



#### Connexion à une base de données

- On peut spécifier une chaîne de connexion au serveur:

```
>>> client = MongoClient(host="localhost", port=27017)
>>> print(client)
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
>>> _
```

- L'objet **client** est une instance de la classe **pymongo.mongo\_client.MongoClient** où on retrouve les informations de la connexion comme le host le port, etc...
- Pour se connecter à la base de données *Exemples* du serveur objet de la connexion:

```
>>> db = client["Exemples"]
>>> print(db)
Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'Exemples')
>>> _
```

- Ainsi, l'objet **client** retourne un objet de classe **pymongo.database.Database**.

## 05 – Effectuer des requêtes depuis des programmes Python

### Connexion au serveur MongoDB



#### Recapitulons

- La librairie **pymongo** permet d'utiliser l'IDE python pour manipuler les objets suivants:

L'interface cliente (la connexion au serveur),

Les bases de données du serveur,

et les collections d'une base de données spécifique.

## CHAPITRE 5

### Effectuer des requêtes depuis des programmes Python

1. Présentation et installation de pymongo,
2. Connexion des bases de données avec le serveur MongoDB,
3. **Création des requêtes :**
  - Requetes simples,
  - Création des indexs,
  - Requetes d'agrégation,
  - Requetes de modifications.



# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes simples:

- **Pymongo** utilise la même syntaxe que l'interface **MongoDB**:

```
Client.baseDonnees.nomDeLaCollection.requete()
```

Requête simple	Explications
db.Stagiaires.find()	<pymongo.cursor.Cursor object at 0x000001D9D93BACE0> <ul style="list-style-type: none"><li>• Retourne un curseur (type Cursor) sur les données</li><li>• <b>Pymongo</b> ne retourne pas les résultats sous forme de liste par souci de mémoire</li></ul>
<pre>&gt;&gt;&gt; db.Stagiaires.find() &lt;pymongo.cursor.Cursor object at 0x000001D9D93BACE0&gt;</pre>	



# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes simples:

- **Pymongo** utilise la même syntaxe que l’interface MongoDB:

Client.baseDonnees.nomDeLaCollection.requete()

Requête simple	Explications
<pre>resultat = db.Stagiaires.find() for r in resultat[:2]:     print(r)</pre>	<ul style="list-style-type: none"> <li>• On récupère le curseur dans la variable <b>resultat</b> (objet itérable en Python),</li> <li>• Pour accéder au contenu de la requête il faut parcourir l’objet renvoyé</li> </ul>
<pre>&gt;&gt;&gt; resultat = db.Stagiaires.find() &gt;&gt;&gt; for r in resultat[:2]: ...     print(r) ... {'_id': ObjectId('62e0f02cd55ee92da665d213'), 'nom': 'Alami', 'prenom': 'Amina', 'filieres': {'_id': 'DD', 'intitule': 'D eveloppement digital'}, 'moy1A': 14.5, 'niveau': '2A', 'option': 'Mobile'} {'_id': ObjectId('62e0f02cd55ee92da665d214'), 'nom': 'Ennaïm', 'prenom': 'Nidal', 'filieres': {'_id': 'ID', 'intitule': ' Infrastructure digitale'}, 'moy1A': 17.25, 'niveau': '2A', 'option': 'Cyber Security'} &gt;&gt;&gt;</pre>	

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes simples:

- Afin de rendre les requêtes plus lisibles, il est possible de créer des variables Python qui correspondent aux conditions de selection ou/et de projection quand utilise dans la requête

Requête simple	Explications
<pre>selection={} selection["option"]="Mobile" resultat = db.Stagiaires.find(selection) for i in resultat[:]:     print(i)</pre>	<ul style="list-style-type: none"> <li>On crée une variable <b>selection</b> de type dictionnaire ,</li> <li>On y ajoute la clé et la valeur correspondant à la sélection voulue,</li> <li>On attribue la variable à la requête</li> <li>On récupère le curseur dans la variable <b>resultat</b> (objet itérable en Python),</li> <li>Pour accéder au contenu de la requête il faut parcourir l’objet renvoyé</li> </ul>

```
>>> selection={}
>>> selection["option"]="Mobile"
>>> resultat = db.Stagiaires.find(selection)
>>> for i in resultat[:]:
...     print(i)
...
{'_id': ObjectId('62e0f02cd55ee92da665d213'), 'nom': 'Alami', 'prenom': 'Amina', 'filiere': {'_id': 'DD', 'intitule': 'Developpement digital'}, 'moy1A': 14.5, 'niveau': '2A', 'option': 'Mobile'}
>>>
```

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes simples:

- Afin de rendre les requêtes plus lisibles, il est possible de créer des variables **Python** qui correspondent aux conditions de sélection ou/et de projection quand utilise dans la requête.

Requête simple	Explications
<pre>projection={"nom":1,"prenom":1,"_id":0} selection={} selection["filiere.intitule"]="Developpement digital" resultat = db.Stagiaires.find(selection,projection) for i in resultat[:]:     print(i)</pre>	<ul style="list-style-type: none"> <li>On ajoute une variable projection de type dictionnaire avec les elements correspondants aux champs de projection de la requête</li> <li>On attribue les deux variables à la requête</li> <li>On récupère le curseur dans la variable <b>resultat</b> puis on parcourt l'objet renvoyé</li> </ul>

```
>>> projection={"nom":1,"prenom":1,"_id":0}
>>> selection={}
>>> selection["filiere.intitule"]="Developpement digital"
>>> resultat = db.Stagiaires.find(selection,projection)
>>> for i in resultat[:]:
...     print(i)
...
{'nom': 'Alami', 'prenom': 'Amina'}
{'nom': 'Alami', 'prenom': 'Salim'}
>>>
```

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes simples:

- Pour utiliser certaines méthodes sur la requête, c’est sensiblement toujours la même syntaxe de MongoDB

Client.baseDonnees.nomDeLaCollection.requete().methode()

Requête simple	Explications
<pre>r = db.Stagiaires.find().sort("nom",-1) for i in r[:]:     print(i)</pre>	<ul style="list-style-type: none"> <li>• On récupère le curseur dans la variable <b>resultat</b> (objet itérable en Python),</li> <li>• Pour accéder au contenu de la requête il faut parcourir l’objet renvoyé</li> </ul>

```
>>> r = db.Stagiaires.find().sort("nom",-1)
>>> for i in r[:]:
...     print(i)
...
{'_id': ObjectId('62e0f02cd55ee92da665d214'), 'nom': 'Ennaim', 'prenom': 'Nidal', 'filiere': {'_id': 'ID', 'intitule': 'Infrastructure digitale'}, 'moy1A': 17.25, 'niveau': '2A', 'option': 'Cyber Security'}
{'_id': ObjectId('62e0fe5fd55ee92da665d216'), 'nom': 'Dalil', 'prenom': 'Karima', 'filiere': {'_id': 'DDesign', 'intitule': 'Digital Design'}, 'niveau': '1A'}
{'_id': ObjectId('62e0f02cd55ee92da665d213'), 'nom': 'Alami', 'prenom': 'Amina', 'filiere': {'_id': 'DD', 'intitule': 'Developpement digital'}, 'moy1A': 14.5, 'niveau': '2A', 'option': 'Mobile'}
{'_id': ObjectId('62e0f02cd55ee92da665d215'), 'nom': 'Alami', 'prenom': 'Salim', 'filiere': {'_id': 'DD', 'intitule': 'Developpement digital'}, 'moy1A': 12.75, 'niveau': '2A', 'option': 'FullStack'}
>>>
```

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Utilisation des index:

- Définition:**
  - Des structures de données spéciales qui stockent une petite partie de l'ensemble de données de la collection sous une forme facile à parcourir,
  - L'index stocke la valeur d'un champ spécifique ou d'un ensemble de champs, triés par la valeur du champ.

- Syntaxe:**

```
Client.BasedeDonnee.Collection.requete()
```

- Avec :**

Requête	Explication	Exemples
index_information()	Retourne la liste des index de la collection	for k,v in db.Stagiaires.index_information().items(): print("index:" ,k, "valeur :",v,"\n")
create_index()	Création d'un index	db.Stagiaires.create_index("ind1")
drop_index()	Suppression d'un index	db.Stagiaires.drop_index("ind1")

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes d'agrégation

- Les requêtes d'agrégation ont pour but de faire des calculs simples (agrégats) sur toute la collection ou seulement sur certains groupes,
- La syntaxe reste la même que l'interface **MongoDB**

```
Client.BasedeDonnee.Collection.aggregate()
```

Exemple	Explication
<pre>resultat = db.Stagiaires.aggregate([ {"\$group":{"_id":"\$filiere.intitule",            "moyenne":{"\$avg":"\$moy1A"}} }) for i in resultat:     print(i["moyenne"]," moyenne de la filiere ",i["_id"])</pre>	<ul style="list-style-type: none"> <li>• On récupère le curseur dans la variable <b>resultat</b>,</li> <li>• Pour accéder au contenu de la requête il faut parcourir l'objet renvoyé:</li> </ul>

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes d'agrégation

- Les requêtes d'agrégation ont pour but de faire des calculs simples (agrégats) sur toute la collection ou seulement sur certains groupes,
- La syntaxe reste la même que l'interface MongoDB

Client.BasedeDonnee.Collection.aggregate()

Exemple	Explication
<pre>resultat = db.Stagiaires.aggregate([ {"\$group":{"_id":"\$filiere.intitule", "moyenne":{"\$avg":"\$moy1A"}} }]) for i in resultat:     print(i["moyenne"]," moyenne de la filiere ",i["_id"])</pre>	<ul style="list-style-type: none"> <li>• On récupère le curseur dans la variable <b>resultat</b>,</li> <li>• Pour accéder au contenu de la requête il faut parcourir l'objet renvoyé:</li> </ul>

```
>>> resultat = db.Stagiaires.aggregate([
... {"$group":{"_id":"$filiere.intitule","moyenne":{"$avg":"$moy1A"}}}])
>>> for i in resultat:
...     print(i["moyenne"]," moyenne de la filiere ",i["_id"])
...
None moyenne de la filiere Digital Design
17.25 moyenne de la filiere Infrastructure digitale
13.625 moyenne de la filiere Developpement digital
>>>
```

# 05 – Effectuer des requêtes depuis des programmes Python

Création des requêtes avec pymongo

## Requêtes de modification

- La syntaxe reste la même que l’interface **MongoDB**

```
Client.BaseeDonnee.Collection.requete()
```

Requête pymongo	Fonctionnement
insert_one()	Insertion d’un seul document
insert_many()	Insertion d’une liste de documents
delete_one()	Suppression d’un document
delete_many()	Suppression d’une liste de documents
update_one()	Modification d’un document
update_many()	Modification d’une liste de documents



# 05 – Effectuer des requêtes depuis des programmes Python

## Création des requêtes avec pymongo

### Requêtes de modification

- Exemple d'insertion :

```

Invite de commandes - python
>>> db.Stagiaires.insert_one({"_id":"12345678","nom":"Jobs","prenom":"Steve","filiere":{"_id":"ID","intitule":"Infrastructure digitale"},"moy1A":14.25,"niveau":"2A","option":"Cloud"})
<pymongo.results.InsertOneResult object at 0x000001D9D93BB4F0>
>>> resultat = db.Stagiaires.find()
>>> for i in resultat[:]:
...     print(i)
...
{'_id': ObjectId('62e0f02cd55ee92da665d213'), 'nom': 'Alami', 'prenom': 'Amina', 'filiere': {'_id': 'DD', 'intitule': 'Developpement digital'}, 'moy1A': 14.5, 'niveau': '2A', 'option': 'Mobile'}
{'_id': ObjectId('62e0f02cd55ee92da665d214'), 'nom': 'Ennaïm', 'prenom': 'Nidal', 'filiere': {'_id': 'ID', 'intitule': 'Infrastructure digitale'}, 'moy1A': 17.25, 'niveau': '2A', 'option': 'Cyber Security'}
{'_id': ObjectId('62e0f02cd55ee92da665d215'), 'nom': 'Alami', 'prenom': 'Salim', 'filiere': {'_id': 'DD', 'intitule': 'Developpement digital'}, 'moy1A': 12.75, 'niveau': '2A', 'option': 'FullStack'}
{'_id': ObjectId('62e0fe5fd55ee92da665d216'), 'nom': 'Dalil', 'prenom': 'Karima', 'filiere': {'_id': 'DDesign', 'intitule': 'Digital Design'}, 'niveau': '1A'}
{'_id': '12345678', 'nom': 'Jobs', 'prenom': 'Steve', 'filiere': {'_id': 'ID', 'intitule': 'Infrastructure digitale'}, 'moy1A': 14.25, 'niveau': '2A', 'option': 'Cloud'}
>>>
  
```