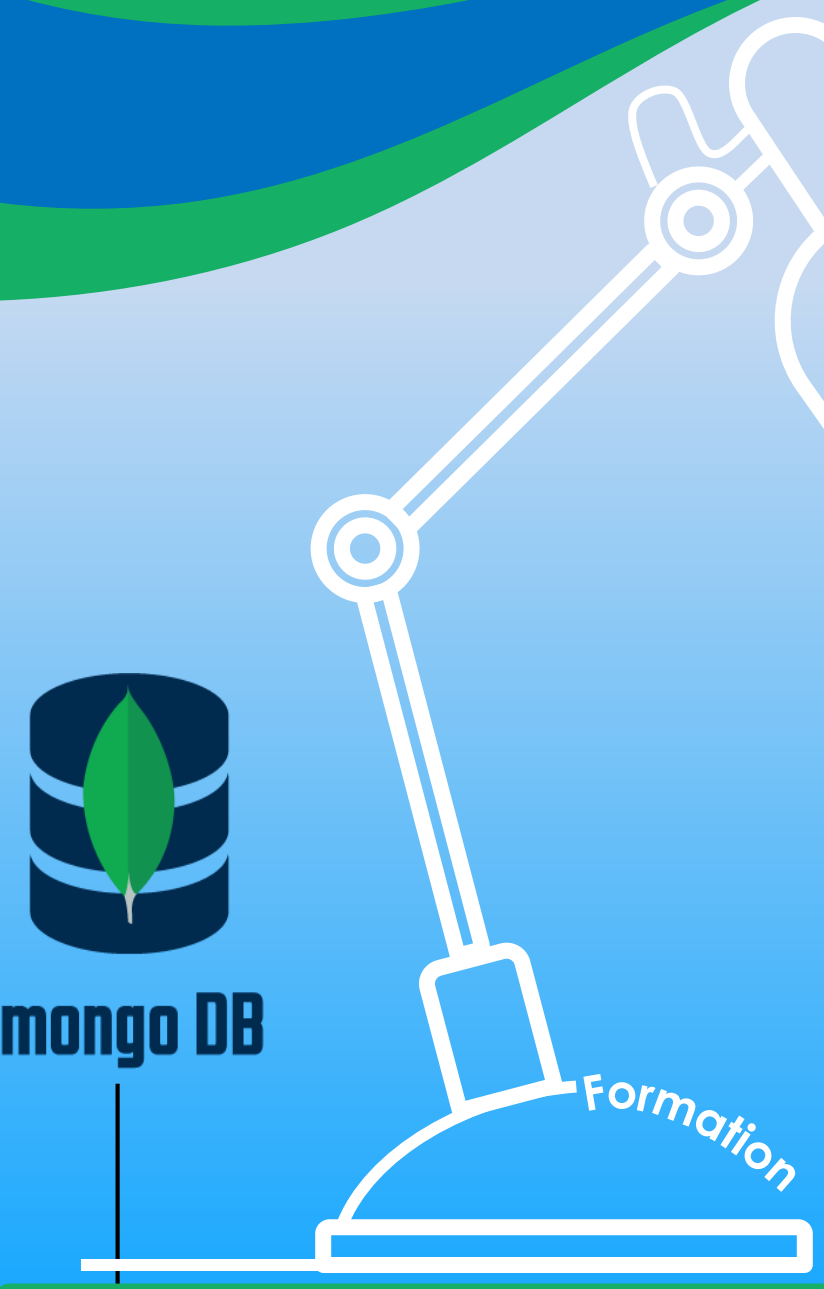
OFPPT
La Voie de l'avenir

Les bases de données NoSQL



Formateur : GAHI Said
Filière : Développement Digital
said.gahi@ofppt.ma

Les bases de données NoSQL

Développement Digital
(Option :Web FullStack)

SEANCE 01

Plan de Formation

- Définir le concept de bases de données NoSQL
- Comparer les bases de données traditionnelles et NoSQL
- Recenser les types de bases de données NoSQL (document, clé / valeur, colonne, graphe)
- Outils de travail pour gérer les base de données NoSQL

Module 203 : Gérer les bases de données

Présenté par : Said

Définir le concept de bases de données NoSQL

Formateur : Said GAHI

Définition

C’est quoi une BD NoSQL?

- Le **NoSQL** est un ensemble de technologies de **BD** reposant sur un modèle diffèrent 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,…):

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

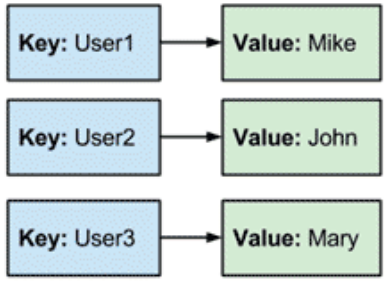
Définir le concept de bases de données NoSQL

Formateur : Said GAHI

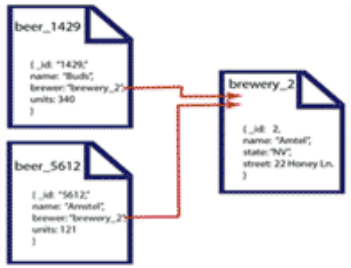
Définition

4 types de bases de données NoSQL

Clé-valeur



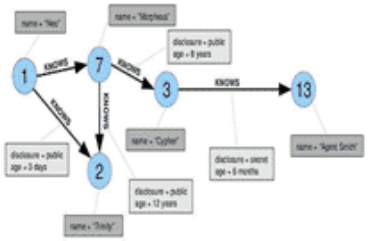
Document



Colonnes

Bases de données orientées aggrégats (BDOA)

Graphes



Bases de données orientées graphes (BDOG)

Résumé de Cours

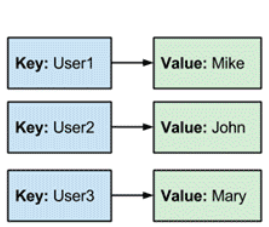
Définir le concept de bases de données NoSQL

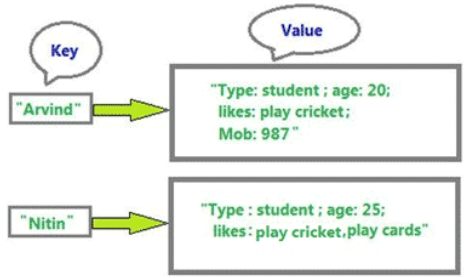
Formateur : Said GAHI

4 types de bases de données NoSQL

Définition

Type 1 : Entrepôts clé-valeur (ECV)





Facilement scalable

Temps de réponse en écriture / lecture très bas

Mises à jour compliquées

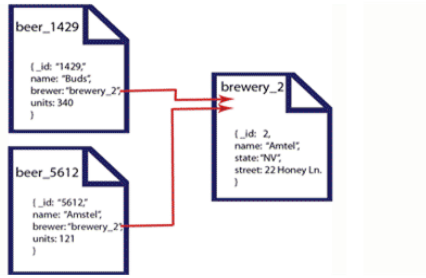
Requêtes rudimentaires

Exemples d'implémentation

Amazon DynamoDB (Beta) ORACLE BERKELEY DB 11g redis

Les données sont stockées en clé-valeur : une clé plus un BLOB (dans lequel on peut mettre : nombre, date, texte, XML, photo, vidéo, structure objet)

Type 2 : Bases orientées documents



Requétage plus complet

Flexibilité

Evolutif au cours du temps

Duplication des données

Cohérence pas forcément assurée

Exemples d'implémentation

CouchDB mongoDB terrastore

Ces bases de données stockent des données semi-structurées : le contenu est formaté JSON ou XML, mais la structure n'est pas contrainte.

Résumé de Cours

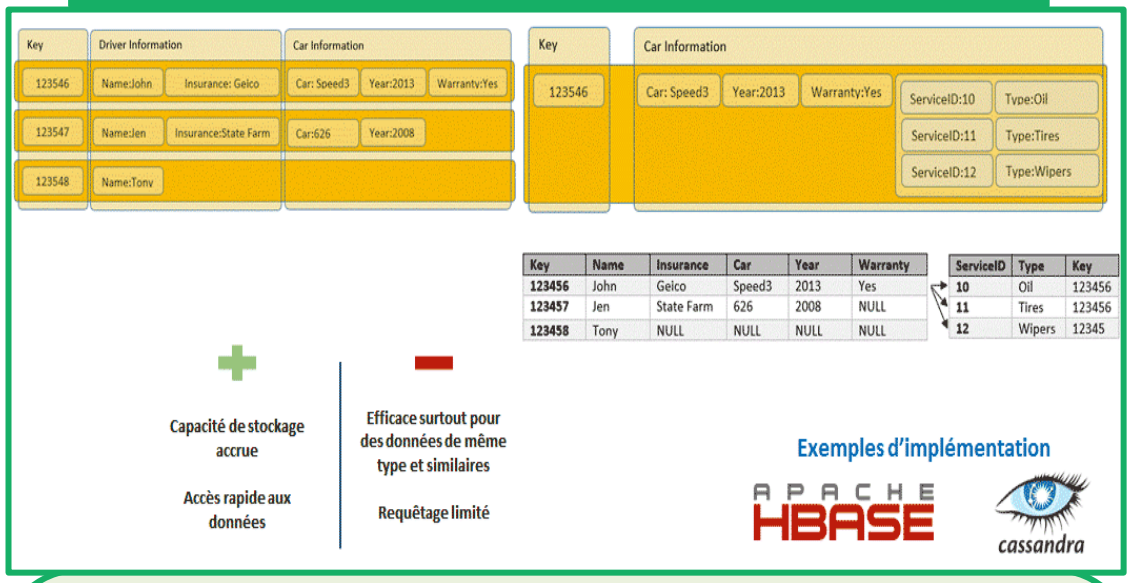
Définir le concept de bases de données NoSQL

4 types de bases de données NoSQL

Formateur : Said GAHI

Définition

Type 3 : Bases orientées colonnes



\+ Capacité de stockage accrue

\+ Accès rapide aux données

\- Efficace surtout pour des données de même type et similaires

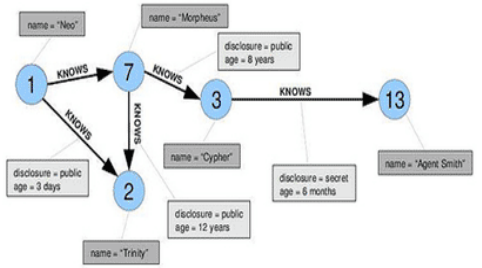
\- Requêtage limité

Exemples d'implémentation

APACHE HBASE

cassandra

Type 4 : Bases de données orientées graphes



\+ Adapté à la gestion de données relationnelles

Architecture modelable

\- Architecture limitée à certains cas

Exemple d'implémentation

TITAN

Neo4j

Ces bases de données se rapprochent des bases de données relationnelles, à ceci près qu'elles permettent de remplir un nombre de colonnes variable

Ces bases de données, basées sur la théorie des graphes, sont gérées par noeuds, relations et propriétés. Elles gèrent des données spatiales, sociales ou financières (dépôts/retraits).

Définir le concept de bases de données NoSQL

Formateur : Said GAHI

SQL vs NoSQL

MongoDB

```
{
"_id": 1,
"student_name": "Jasmin Scott",
  "school": {
    "school_id": 226,
    "name": "Tech Secondary",
    "address": "100 Broadway St",
    "city": "New York",
    "state": "NY",
    "zipcode": "10001"
  },
"marks": [98, 93, 95, 88, 100],
}
```

```
mongo
> db.students.find({“student_name”: “Jasmin Scott”})
```

SQL

students

id	name	school_id
1	Jasmin Scott	226
...	...	...

marks

id	student_id	mark
10	1	98
...	...	...

school_details

id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...	...	...	...	...	...

Results

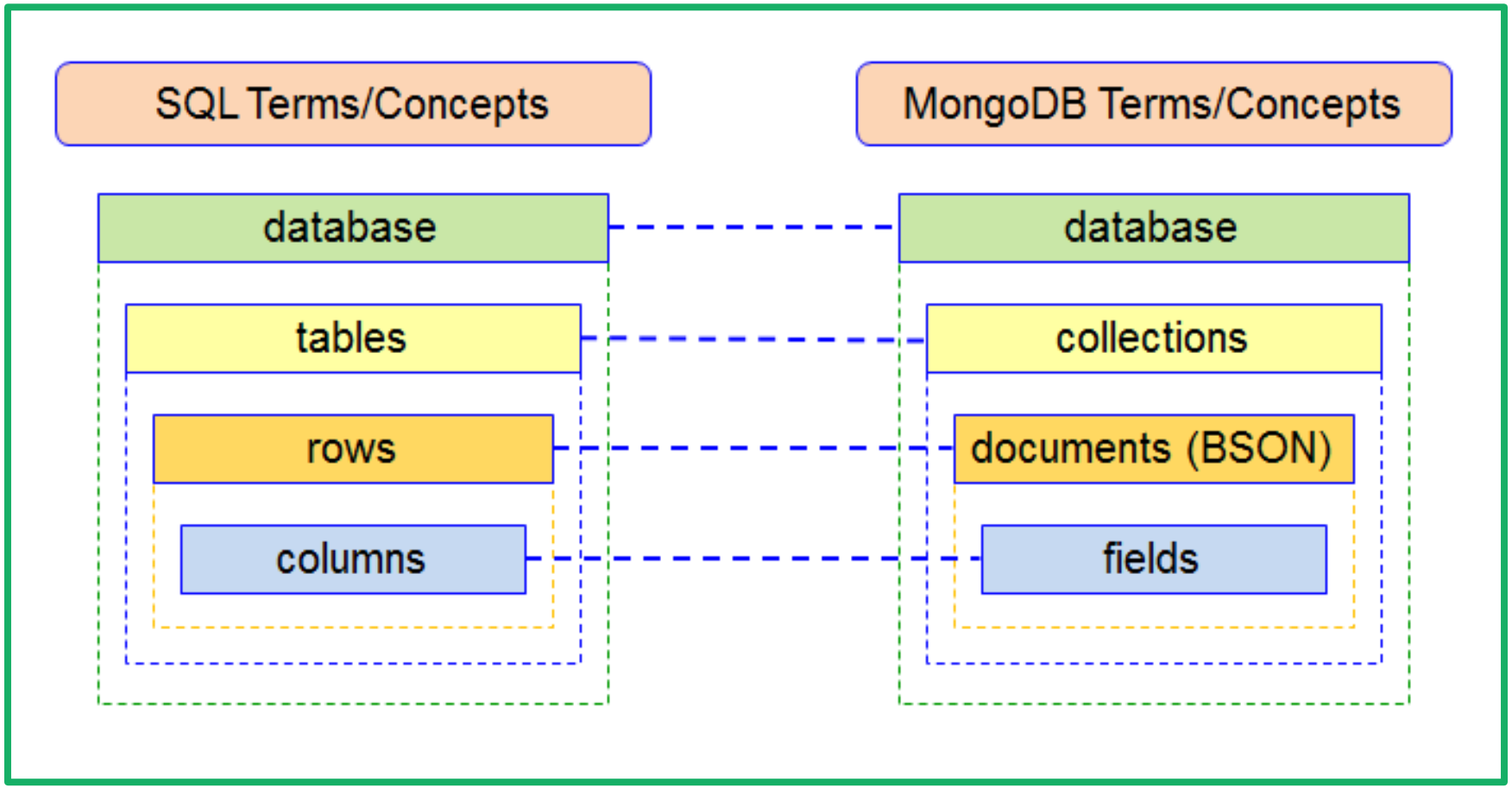
name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York
...	...	...	...

```
sql
SELECT s.name, m.mark, d.name as "school name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

Définir le concept de bases de données NoSQL

Formateur : Said GAHI

SQL vs NoSQL



Résumé de Cours

Installation des Outils du Travail

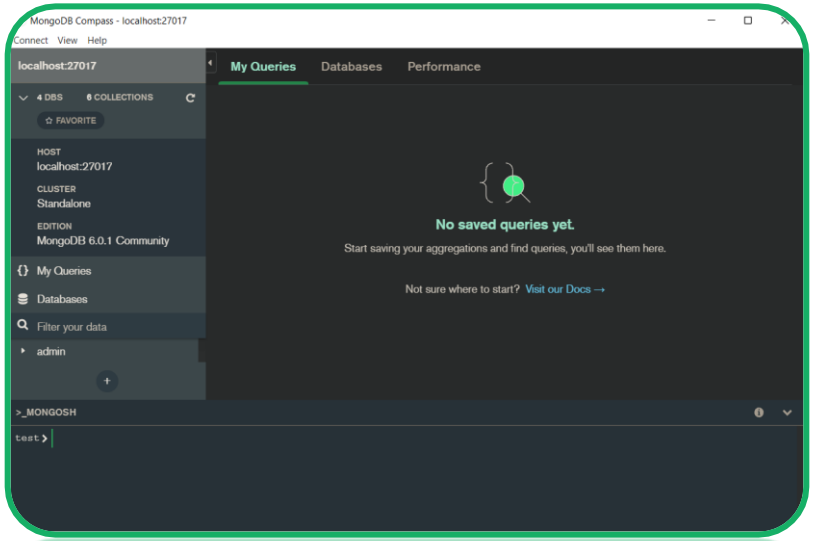
Formateur : Said GAHI

Outils

DOWNLOAD

❶ MongoDB

https://www.mongodb.com/download-center/community/releases/archive



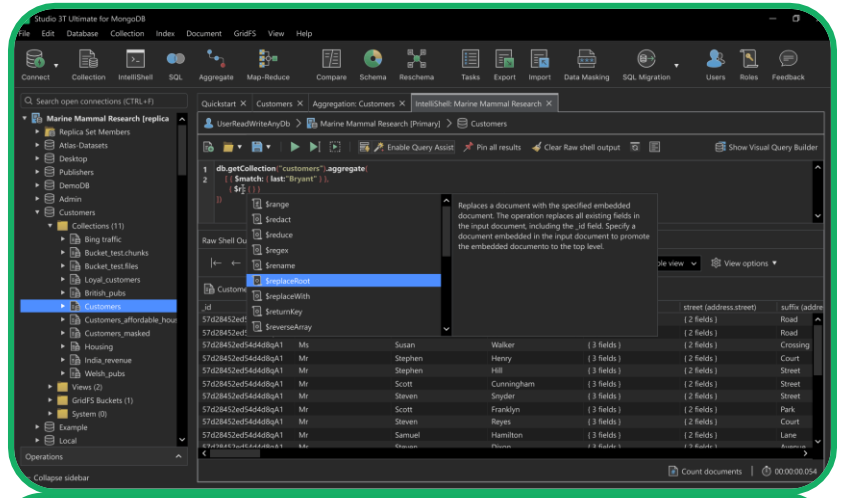
Mongosh

❷

mongoDB Compass

https://www.mongodb.com/try/download/compass

❸ Studio 3T



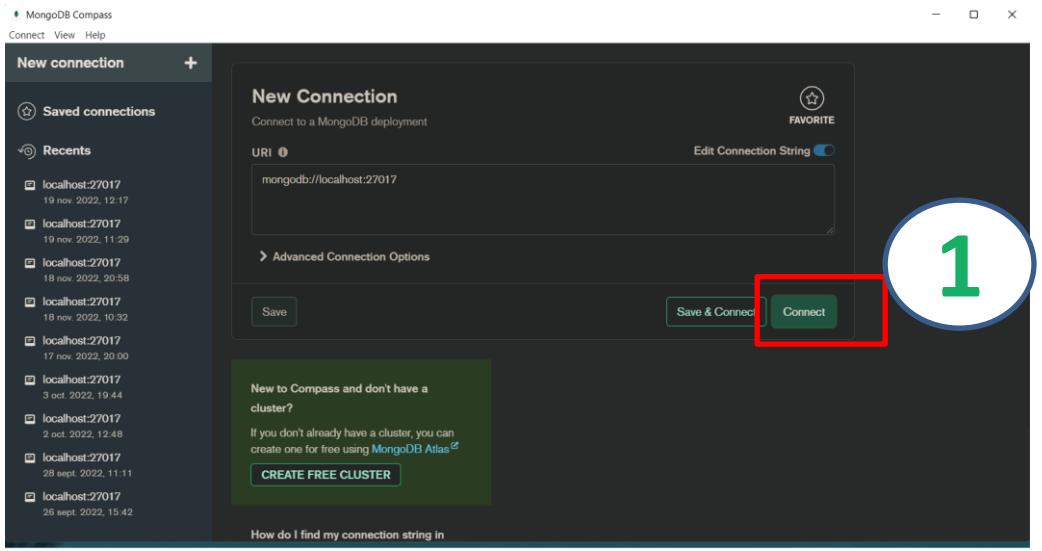
https://studio3t.com/download/

MongoDB CRUD Opérations

Formateur : Said GAHI

Opérations

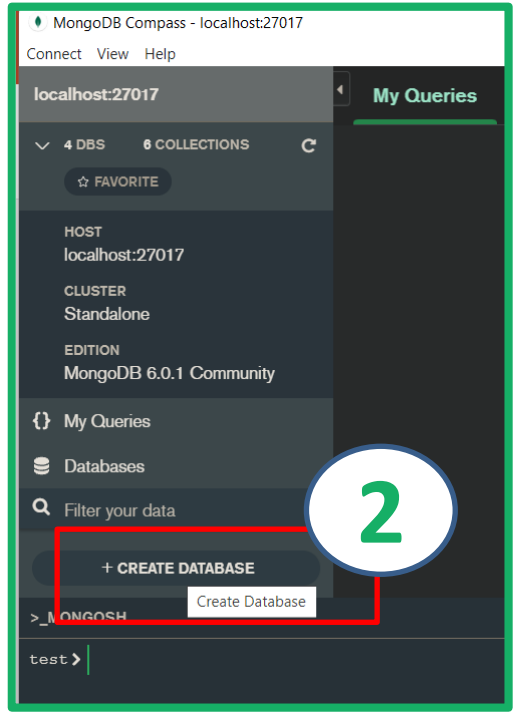
Création de la base de données



use nom_database

```
>_MONGOSH
> use magasin
< 'switched to db magasin'
magasin>
```

Ou bien



MongoDB CRUD Opérations

Formateur : Said GAHI

Opérations

Création des Collections

Permet de créer et ouvrir une base de données

```
use magasin
db.createCollection("produits")
```

```
db.dropDatabase()
```

```
use ecole
db.createCollection("stagiaires")
```

```
use magasin
show dbs
db
show collections
```

```
> use magasin
< 'switched to db magasin'
> show dbs
< admin    48.00 KiB
  config  108.00 KiB
  ecole   112.00 KiB
  local    96.00 KiB
magasin>
```

Formateur : Said GAHI

Supprimer et modifier des Collections

```
use magasin
db.produits.drop()
```

```
> db.produits.renameCollection("products")
< { ok: 1 }
> show collections
< products
test >
```

```
db.produits.renameCollection("products")
```

MongoDB CRUD Opérations

Formateur : Said GAHI

Opérations

Insertion des documents dans la collection

```
db.etudiants.insert()
db.etudiants.insertOne()
db.etudiants.insertMany()
```

Exemple 1

```
db.etudiants.insertOne(
 {
      Numero : "111",
      Nom : "Alami",
      Prenom : "ahmed",
      Age : 22,
      Adresse : { rue: 28, ville: "agadir" , tel: "068782215" } ,
      Notes :[12, 5.5, 13.75, 14]
 }
)
```

Exemple 2

```
db.etudiants.insertOne(
 {
      Numero : "111",
      Nom : "Alami",
      Prenom : "ahmed",
      Age : 22
 }
)
```

MongoDB CRUD Opérations

Formateur : Said GAHI

Opérations

Insertion des documents dans la collection

Exemple 1

```
var x=[
                {
        Numero : "112",
        Nom : "kabiri",
        Prenom : "ahmed",
        Age : 20,
        Adresse : { rue: 27, ville: "tanger" , tel: "068792255" } ,
        Notes :[9, 5.5, 11.75, 10.5]
                },
                {
        Numero : "113",
        Nom : "rami",
        Prenom : "amina",
        Age : 23,
        Adresse : { rue: 30, ville: "tanger" , tel: "067895554" } ,
        Notes :[6.5, 8.5, 13.25, 11.5]
                }
]


db.etudiants.insert(x)
```

Objet Etudiant

Objet Adresse

Formateur : Said GAHI

Insertion des documents dans la collection

Exemple 2

```
db.inventory.insertMany(
[
{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },


{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },


{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }


]

)
```

Formateur : Said GAHI

Opérations

Insertion des documents dans la collection

Autres méthodes

db.collection.updateOne()

db.collection.updateMany()

db.collection.findAndModify()

db.collection.findOneAndUpdate()

db.collection.findOneAndReplace()

db.collection.bulkWrite()

Formateur : Said GAHI

Mettre à jour des documents dans MongoDB

MongoDB met à disposition la fonction `update` avec différents opérateurs en fonction du type de mise à jour souhaité. La fonction update prend deux arguments obligatoires :

- un document représentant la condition de recherche des documents de la collection
- un document représentant la mise à jour souhaitée

Ajouter ou remplacer un champ existant avec $set

```
db.etudiants.update({Numero:'113'},{$set:{genre:'F'}})
```

db.etudiants.update({},{})

Formateur : Said GAHI

Opérations

Exemple 1

Mettre à jour des documents dans MongoDB

Dans cet exemple, on a simplement rajouté un champ **genre** dans le document de **numero '113'**.

```
db.etudiants.update({Numero:'113'},{$set:{genre:'F'}})
```

```
> db.etudiants.update({Numero:'113'},{$set:{genre:'F'}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.etudiants.find()
```

```
> db.etudiants.find()
< { _id: ObjectId("63821518b634e717d0c83cac"),
    Numero: '112',
    Nom: 'kabiri',
    Prenom: 'ahmed',
    Age: 20,
    Adresse: { rue: 27, ville: 'tanger', tel: '068792255' },
    Notes: [ 9, 5.5, 11.75, 10.5 ] }
  { _id: ObjectId("63821518b634e717d0c83cad"),
    Numero: '113',
    Nom: 'rami',
    Prenom: 'amina',
    Age: 23,
    Adresse: { rue: 30, ville: 'tanger', tel: '067895554' },
    Notes: [ 6.5, 8.5, 13.25, 11.5 ] }
```

Formateur : Said GAHI

Opérations

Exemple 2

Mettre à jour des documents dans MongoDB

1. Ajouter le champ **niveau** pour tous les documents

```
db.etudiants.update({}, {$set:{niveau:'bac+5'} })
```

```
> db.etudiants.update({},{$set:{niveau:'bac+5'}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.etudiants.find()
```

```
< { _id: ObjectId("63821518b634e717d0c83cac"),
    Numero: '112',
    Nom: 'kabiri',
    Prenom: 'ahmed',
    Age: 20,
    Adresse: { rue: 27, ville: 'tanger', tel: '068792255' },
    Notes: [ 9, 5.5, 11.75, 10.5 ],
    niveau: 'bac+5' }
  { _id: ObjectId("63821518b634e717d0c83cad"),
    Numero: '113',
    Nom: 'rami',
    Prenom: 'amina',
    Age: 23,
    Adresse: { rue: 30, ville: 'tanger', tel: '067895554' },
    Notes: [ 6.5, 8.5, 13.25, 11.5 ],
    genre: 'F' }
```

Formateur : Said GAHI

Opérations

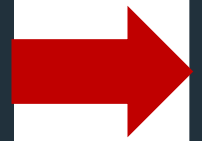
Exemple 2

Mettre à jour des documents dans MongoDB

1. Ajouter le champ **niveau** pour tous les documents

```
db.etudiants.update({}, {$set:{niveau:'bac+5'} },{multi:true} )
```

```
> db.etudiants.update({},{$set:{niveau:'bac+5'}},{multi:true})
{ acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 1,
  upsertedCount: 0 }
> db.etudiants.find()
```



```
{ _id: ObjectId("63821518b634e717d0c83cac"),
  Numero: '112',
  Nom: 'kabiri',
  Prenom: 'ahmed',
  Age: 20,
  Adresse: { rue: 27, ville: 'tanger', tel: '068792255' },
  Notes: [ 9, 5.5, 11.75, 10.5 ],
  niveau: 'bac+5' }
{ _id: ObjectId("63821518b634e717d0c83cad"),
  Numero: '113',
  Nom: 'rami',
  Prenom: 'amina',
  Age: 23,
  Adresse: { rue: 30, ville: 'tanger', tel: '067895554' },
  Notes: [ 6.5, 8.5, 13.25, 11.5 ],
  genre: 'F',
  niveau: 'bac+5' }
```

Formateur : Said GAHI

Opérations

Exemple 3

Mettre à jour des documents dans MongoDB

1- Ajouter le champ **frais** pour tous les documents avec valeur par défaut 900

```
db.etudiants.update({}, {$set:{frais:'bac+5’} },{multi:true} )
```

2- Incrémenter un champ numérique existant avec $inc

```
db.etudiants.update({},{$inc:{frais:50}},{multi:true})
```

```
> db.etudiants.update({},{$inc:{frais:50}},{multi:true})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0 }
```

```
> db.etudiants.find()
< { _id: ObjectId("63821518b634e717d0c83cac"),
    Numero: '112',
    Nom: 'kabiri',
    Prenom: 'ahmed',
    Age: 20,
    Adresse: { rue: 27, ville: 'tanger', tel: '068792255' },
    Notes: [ 9, 5.5, 11.75, 10.5 ],
    niveau: 'bac+5',
    frais: 950 }
```

Formateur : Said GAHI

Opérations

Exemple 4

Mettre à jour des documents dans MongoDB

1- Mettre à jour un tableau avec $push ou $pull

Si j'utilise **$set** sur un tableau, je vais remplacer le tableau existant par un nouvelle élément. Comment mettre à jour le tableau sans écraser les données existantes ?

L'opérateur **$push** permet de rajouter un nouvel élément à un tableau.

```
db.etudiants.update({Numero:'113'},{$push:{Notes:15.75}})
```

```
> db.etudiants.update({Numero:'113'},{$push:{Notes:15.75}})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
```

```
{ _id: ObjectId("63821518b634e717d0c83cad"),
  Numero: '113',
  Nom: 'rami',
  Prenom: 'amina',
  Age: 23,
  Adresse: { rue: 30, ville: 'tanger', tel: '067895554' },
  Notes: [ 6.5, 8.5, 13.25, 11.5, 15.75 ],
  genre: 'F',
  niveau: 'bac+5',
  frais: 950 }
```

Formateur : Said GAHI

Opérations

Exemple 4

Mettre à jour des documents dans MongoDB

1- Mettre à jour un tableau avec $push ou $pull

Pour supprimer un élément, on peut utiliser **$pull**. Ainsi, si je souhaite supprimer la note 15.75 de « 113 », je lance la commande suivante :

```
db.etudiants.update({Numero:'113'},{$pull:{Notes:15.75}})
```

```
{ _id: ObjectId("63821518b634e717d0c83cad"),
  Numero: '113',
  Nom: 'rami',
  Prenom: 'amina',
  Age: 23,
  Adresse: { rue: 30, ville: 'tanger', tel: '067895554' },
  Notes: [ 6.5, 8.5, 13.25, 11.5 ],
  genre: 'F',
  niveau: 'bac+5',
  frais: 950 }
```

Formateur : Said GAHI

Supprimer des documents dans MongoDB

La méthode **remove()** de MongoDB est utilisée pour supprimer un document de la collection. La méthode **remove()** accepte deux paramètres. Le premier concerne les critères de suppression et le deuxième concerne le drapeau « justOne ».

```
db.etudiants.remove(
<query>,
<justOne>
)
```

`db.etudiants.remove({})` : Pour supprimer tous les documents

`db.etudiants.remove( { age : 30 } )` : supprimer tous les étudiants don't l'age est égale à 30

Supprimer un seul document correspondant à une condition

```
db.etudiants.remove( { nom : "alami" }, 1 )
```

Formateur : Said GAHI

Opérations

Exemple 1

Interroger les données dans MongoDB

```
db.etudiants.find()
```

```
db.etudiants.find().pretty()
```

```
db.etudiants.find().count()
```

Retourner toutes les occurences d'une collection avec find()

```
db.collectionName.find()
db.collectionName.find({})
```

Retourner uniquement la première occurence d'une collection avec findOne()

```
db.collectionName.findOne()
db.collectionName.findOne({})
```

Résumé de Cours

MongoDB CRUD Opérations

Formateur : Said GAHI

SQL vs NoSQL

Interroger les données dans MongoDB

Opérations

NoSQL	SQL
db.etudiants.find( )	SELECT * FROM etudiants
db.etudiants.find({numero:'112' })	SELECT * FROM etudiants Where **numero='112'**
db.etudiants.find( ).count()	SELECT count(*) FROM etudiants
db.etudiants.find({"nom": "alami", "age": 20})	SELECT * FROM etudiants WHERE nom = 'alami' AND age = 20

Résumé de Cours

MongoDB CRUD Opérations

Formateur : Said GAHI

SQL vs NoSQL

Interroger les données dans MongoDB

Opérations

NoSQL	SQL
db.etudiants.find({}, {nom: true})	SELECT nom FROM etudiants
db.etudiants.find({numero:'112' }, {nom:true, prenom:true})	SELECT nom,prenom FROM etudiants Where **numero='112'**
db.etudiants.find({"Adresse.ville" : 'tanger'})	SELECT * FROM etudiants where ville= **"tanger"**
	SELECT nom, age,ville FROM etudiants WHERE genre='F' and ville='agadir'

Formateur : Said GAHI

Opérations

Exemple 2

Interroger les données dans MongoDB

d'exploitation	format	exemple	SGBDR déclarations similaires
égal	{<key>:<value> }	db.etudiants.find({"nom":"alami"}).pretty()	where nom = 'alami'
moins que	{<key>:{$lt:<value>}}	db.etudiants.find({"age":{$lt:20}}).pretty()	where age < 20
Inférieur ou égal à	{<key>:{$lte:<value>}}	db.etudiants.find({"age":{$lte:20}}).pretty()	where age <= 20
plus de	{<key>:{$gt:<value>}}	db.etudiants.find({"age":{$gt:20}}).pretty()	where age > 20
Supérieur ou égal à	{<key>:{$gte:<value>}}	db.etudiant.find({"age":{$gte:20}}).pretty()	where age >= 20
N'est pas égal	{<key>:{$ne:<value>}}	db.etudiants.find({"age":{$ne:20}}).pretty()	where age != 20

MongoDB CRUD Opérations

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs de comparaison

La syntaxe des requêtes avec des opérateurs de comparaison est la suivante :

db.etudiants.find({"x": {operateur: val

MongoDB

```
db.notes.find({"notes": {$gte: 13}})
```

SQL

```
SELECT *
FROM notes
WHERE notes >= 13
```

MongoDB

```
db.collectionName.find(
    {"a": { $in: ["chaine1", "chaine2"] }
    }
)
```

SQL

```
SELECT *
FROM t
WHERE a IN ("chaine1", "chaine2")
```

Opérateur logique	Mot clé en MongoDB
=	$eq
≠	$ne
<	$lt
>	$gt
≤	$lte
≥	$gte
∈	$in
∉	$nin
clé existante	$exists
|.|	$size

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs de comparaison

L'opérateur **$exists** vérifie l'existence d'une clé dans un document.
Sa syntaxe en MongoDB est :

```
db.etudiants.find (
{"niveau": { $exists: true} }
)
```

Cette requête renvera donc les documents ayant une clé **a**.

L'opérateur **$size** permet de récupérer les documents pour lesquels l'attribut considéré est une liste d'une certaine taille. Sa syntaxe en MongoDB est la suivante :

```
db.etudiants.find(
    {"Notes": { $size: 4} }
)
```

Opérateur logique	Mot clé en MongoDB
=	$eq
≠	$ne
<	$lt
>	$gt
≤	$lte
≥	$gte
∈	$in
∉	$nin
clé existante	$exists
|.|	$size

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs de comparaison

Les opérateurs de comparaison de listes

Operateur	Description	Exemple
$in	Appartient à la liste	Sélectionner les stagiaires dont l'option est **FullStack** ou **Cyber Security:** selection = **{"option":{"$in":["FullStack","Cyber Security"]}}** db.Stagiaires.find**(selection,{"nom":1,"prenom":1,"_id":0})**
$nin	N'appartient pas à la liste	

```
{
    "nom" : "Ennaim",
    "prenom" : "Nidal"
}
{
    "nom" : "Alami",
    "prenom" : "Salim"
}
```

MongoDB CRUD Opérations

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs Logiques

Les différents opérateurs logiques en MongoDB sont : **$or, $not et $nor**. Ces opérateurs permettent de tester plusieurs conditions simultanément.

ET logique

MongoDB

```
db.collectionName.find(
    {"a": 1, "b": 5}
)
```

SQL

```
SELECT *
FROM t
WHERE a = 1 and b = 5
```

Ecrire une requête qui affiche la liste des étudiants qui habite à agadir et ayant l'âge supérieur à 20

Ecrire une requête qui affiche le nombre des étudiants qui habite à agadir et ayant l'âge supérieur à 20

Ecrire une requête qui affiche le nom, age, et niveau des étudiants qui habite à agadir et ayant l'âge supérieur à 20

MongoDB CRUD Opérations

Formateur : Said GAHI

SQL vs NoSQL

Requêtes plus complexes en utilisant des opérateurs

Opérateurs Logiques

OU logique

MongoDB

```
db.collectionName.find(
    {$or :
      [
        {"a": 1},
        {"b": 5}
      ]
    }
)
```

SQL

```
SELECT *
FROM t
WHERE a = 1 or b = 5
```

Ecrire une requête qui affiche la liste des étudiants qui habite à agadir ou tanger

Ecrire une requête qui affiche le nombre des étudiants qui habite à agadir ou ayant l'âge supérieur à 20

Ecrire une requête qui affiche le nom, age, et niveau des étudiants qui habite à agadir oniveau bac+5

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs Logiques

NON OU logique

```
db.collectionName.find(
{$nor :
        [ {"a": 1},
          {"b": "blue"}
        ]
} )
```

Ecrire une requête qui affiche la liste des étudiants qui n'habite à agadir ou tanger

Ecrire une requête qui affiche le nombre des étudiants qui n'habite à agadir ou ayant l'âge supérieur à 20

Ecrire une requête qui affiche le nom, age, et niveau des étudiants qui n'habite à agadir oniveau bac+5

Formateur : Said GAHI

Requêtes plus complexes en utilisant des opérateurs

Opérateurs Logiques

NON logique

```
db.collectionName.find(

 {$not : {"a": 1} }

)
```

- Ecrire une requête qui affiche la liste des étudiants qui n'habite à agadir
- Ecrire une requête qui affiche le nombre des étudiants qui n'habite à agadir
- Ecrire une requête qui affiche le nom, age, et niveau des étudiants qui n'habite à agadir

Formateur : Said GAHI

Méthodes utiles

Quelques méthodes utiles

Pour connaître la **liste des collections** contenues dans une base de données, on utilise la commande suivante sur la base de données considérée :

db.getCollectionInfos()

La méthode **distinct** permet de renvoyer toutes les valeurs distinctes du champ spécifié

Syntaxe et exemple en MongoDB

```
db.collectionName.distinct(
    champ,
    {...} // Ici, une condition
)
```

Équivalent de la syntaxe en SQL

```
SELECT DISTINCT(champ)
FROM collectionName
WHERE ... -- Ici, une condition
```

Formateur : Said GAHI

Méthodes utiles

Quelques méthodes utiles

Trier les documents résultat : la méthode sort

Trier en ordre croissante

```
db.collectionName.find({}).sort( {"key1" : 1} )
```

Trier en ordre décroissant

```
db.collectionName.find({}).sort( {"key1" : -1} )
```

Limiter le nombre de documents retournés : la méthode **limit**

```
db.collectionName.find({...}).limit(2)
```

Le concept d'aggrégation en MongoDB

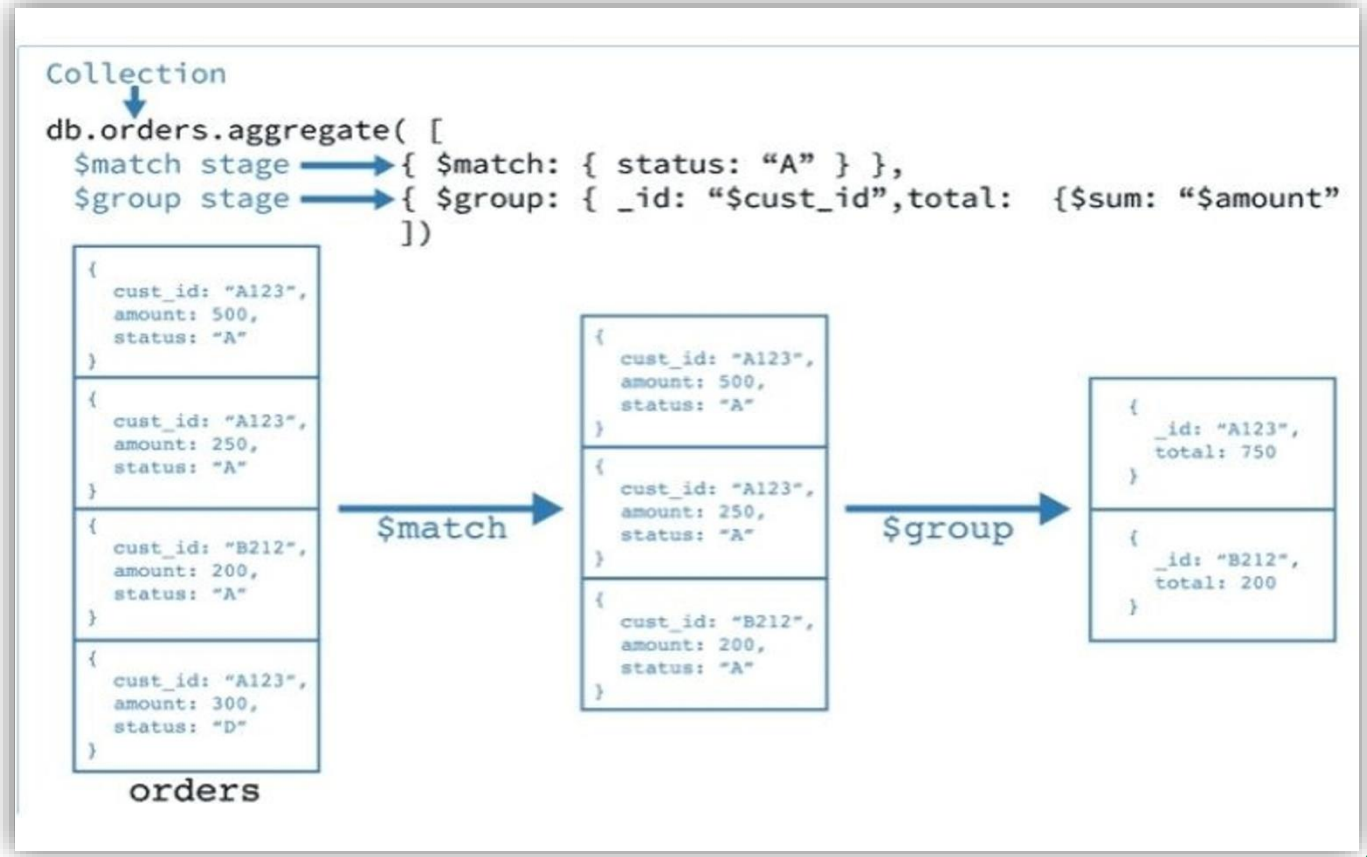
Formateur : Said GAHI

Définition

Syntaxe

L'aggregation Mongo est une série d'opérations visant à prendre une collection de documents en entrée, procésser les documents et rendre une vision analytique de cette collection

db.collection.aggregate()



QUIZ – Chapitre I

Qu'est-ce qui caractérise MongoDB ?

1. C'est un modèle orienté graphique
2. C'est un modèle orienté document
3. C'est un modèle structuré

Quelle méthode ci-dessous ne fait pas partie du langage MongoDB ?

1. `db.collectionName.find({}).orderby({"key" : 1})`
2. `db.collectionName.find({}).sort({"key" : 1})`
3. `db.collectionName.find({}).limit(3)`

Que nous renvoie cette requête sur la collection `notes` de la base `etudiants` ?

```
db.notes.find({}, {"nom": true, "_id": false})
```

1. Affiche **tout** le contenu de la collection `notes`
2. Les noms des étudiants de la base et la clé `_id` qui identifie chaque étudiant
3. Tous les noms des étudiants de la base, mais pas les autres clés

Comment récupérer la liste des étudiants ayant obtenu exactement deux notes ?

1. `db.notes.find({"notes": {$size: 2}})`
2. `db.notes.find({"notes": {$exists: true}})`
3. `db.notes.find({$or: [{"notes": {$size: 1}},{"notes": {$size: 2}}]})`

Quel opérateur permet de ne renvoyer que les documents qui ne vérifient aucune condition de la liste ?

1. L'opérateur `$eq`
2. L'opérateur `$nor`
3. L'opérateur `$gt`

Résumé de Cours

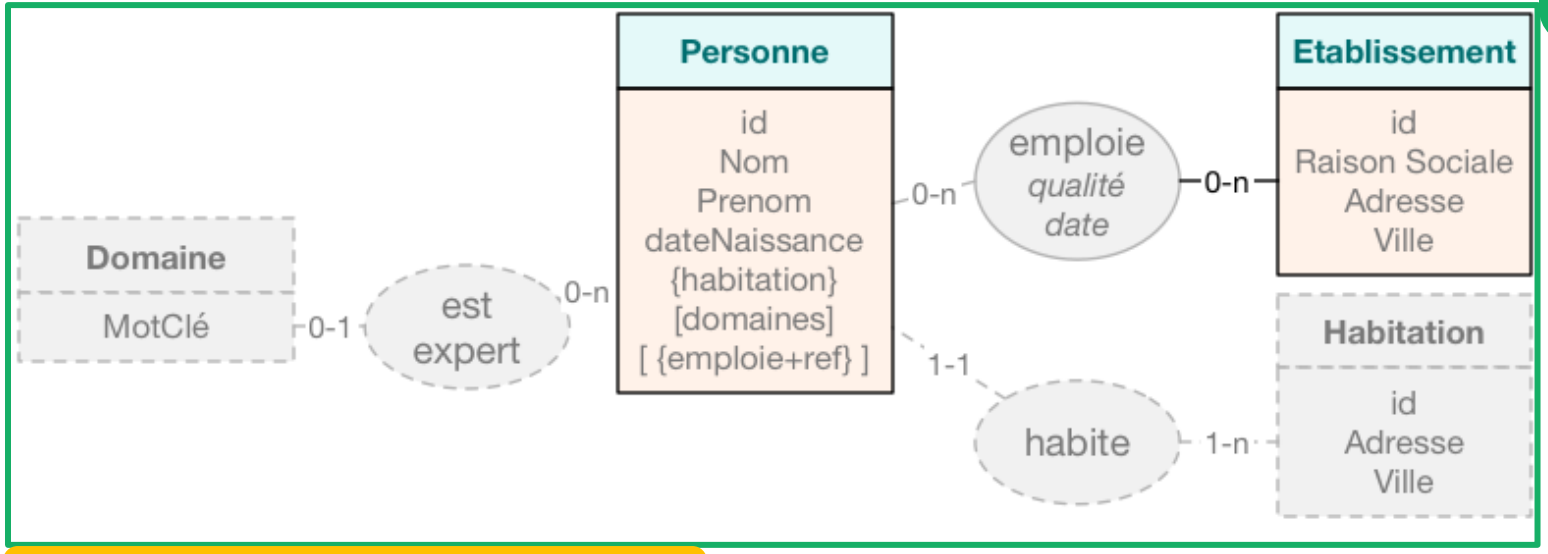
MongoDB CRUD Opérations avec Python

Formateur : Said GAHI

Modélisation

Présentation et installation

Exemples



MCD

```bson
{
  "_id" : 1,    "nom" : "Travers",   "prenom" : "Nicolas",
  "domaines" : ["SGBD", "NoSQL", "RI", "XML"],
  "emplois" : [
    {"id_etablissement" : "100", "qualité" : "Maître de Conférences",
        "date" : "01/09/2007"},
    {"id_etablissement" : "101", "qualité" : "Vacataire",
        "date" : "01/09/2012"}
  ],
  "Habite" : {"adresse" : "292 rue Saint Martin", "ville" : "Paris"}
}
```

Représentation en format BSON

Résumé de Cours

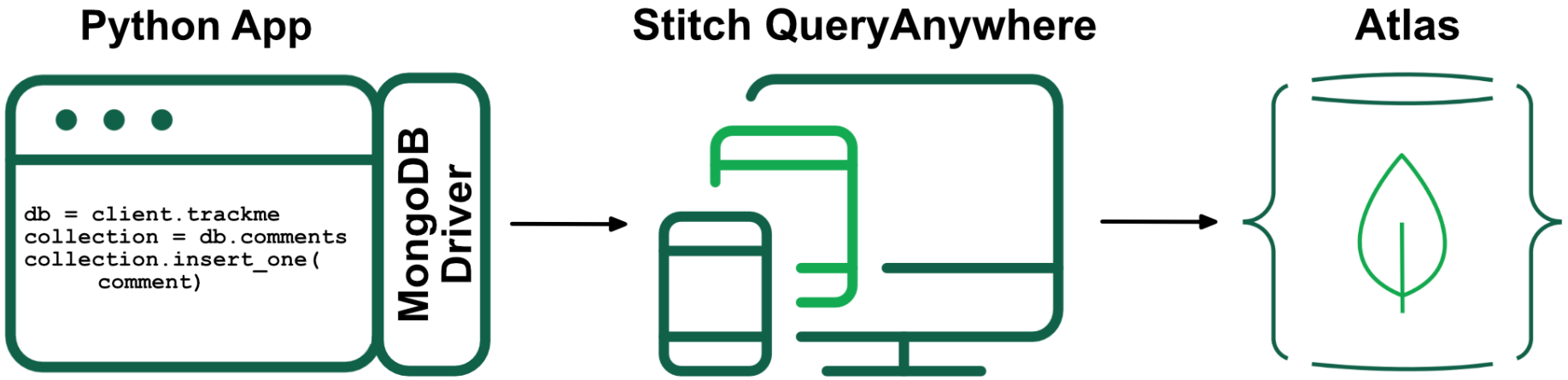
MongoDB CRUD Opérations avec Python

Formateur : Said GAHI

MongoDB et Python

Requêtes depuis Python

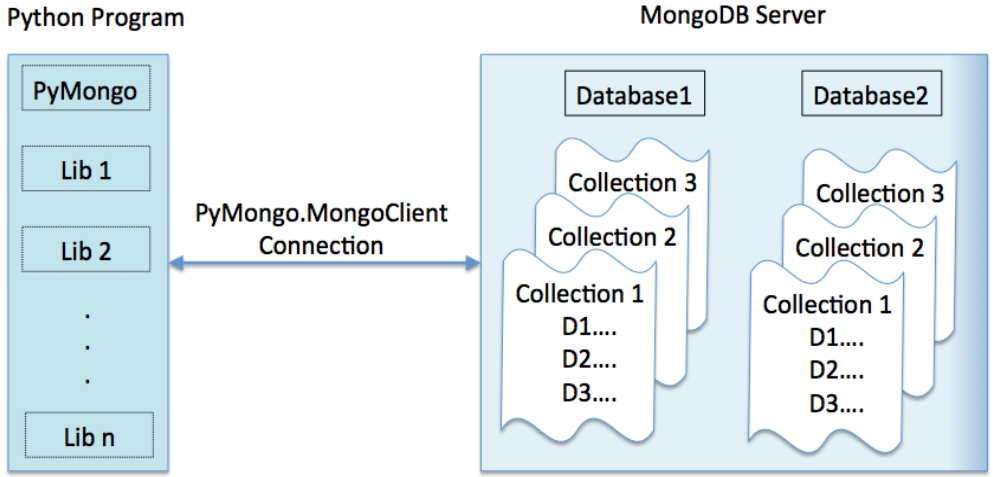
Présentation et installation



PyMongo est une librairie Python contenant des outils pour travailler avec MongoDB et MongodbAtlas

Pour installer la librairie, nous pouvons le faire avec la commande **pip install pymongo**, dans un terminal.

```
from pymongo import MongoClient, InsertOne
client = MongoClient("mongodb://localhost:27017")
db = client.mydb
collection = db.pays
```



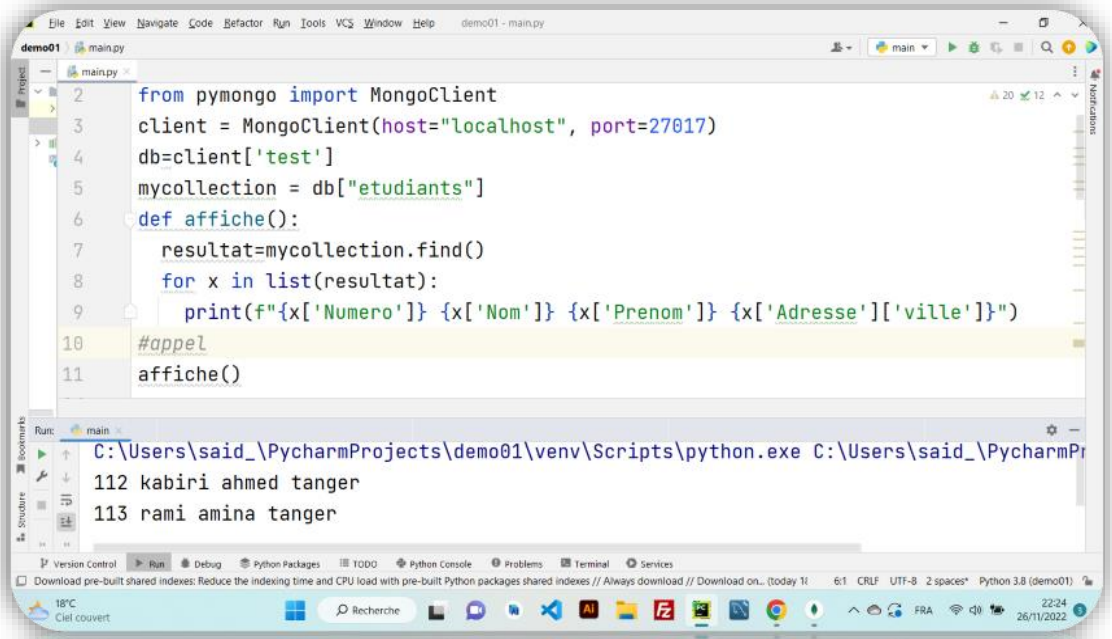
Formateur : Said GAHI

MongoDB et Python

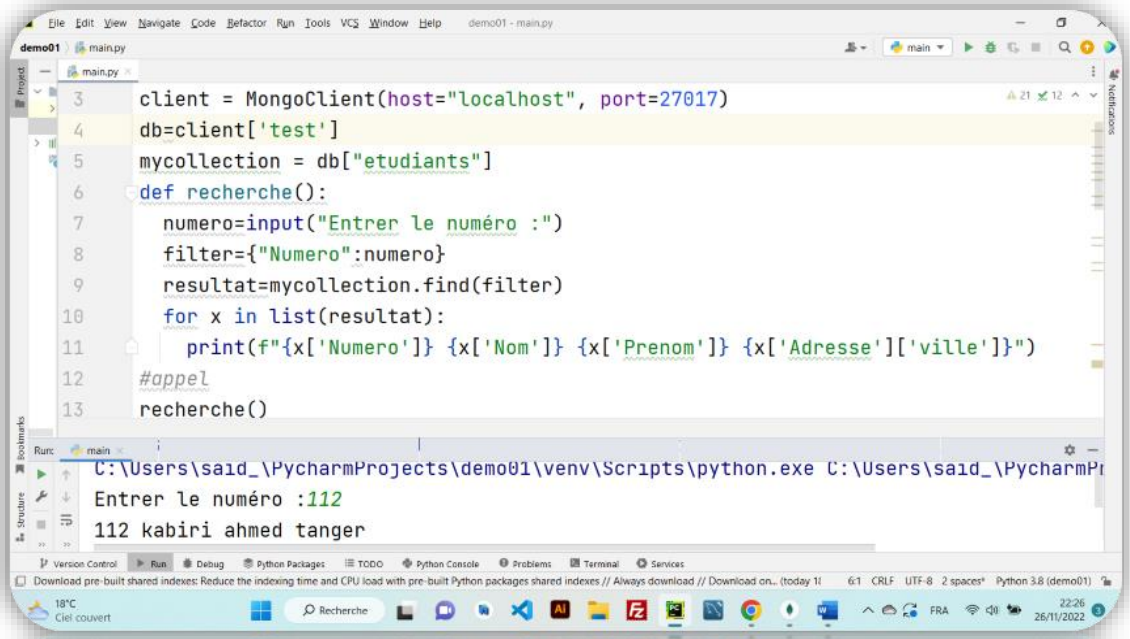
Requêtes depuis Python

Présentation et installation

Pour installer la librairie, nous pouvons le faire avec la commande **pip install pymongo**, dans un terminal.



Afficher tous les étudiants



Recherche un étudiant

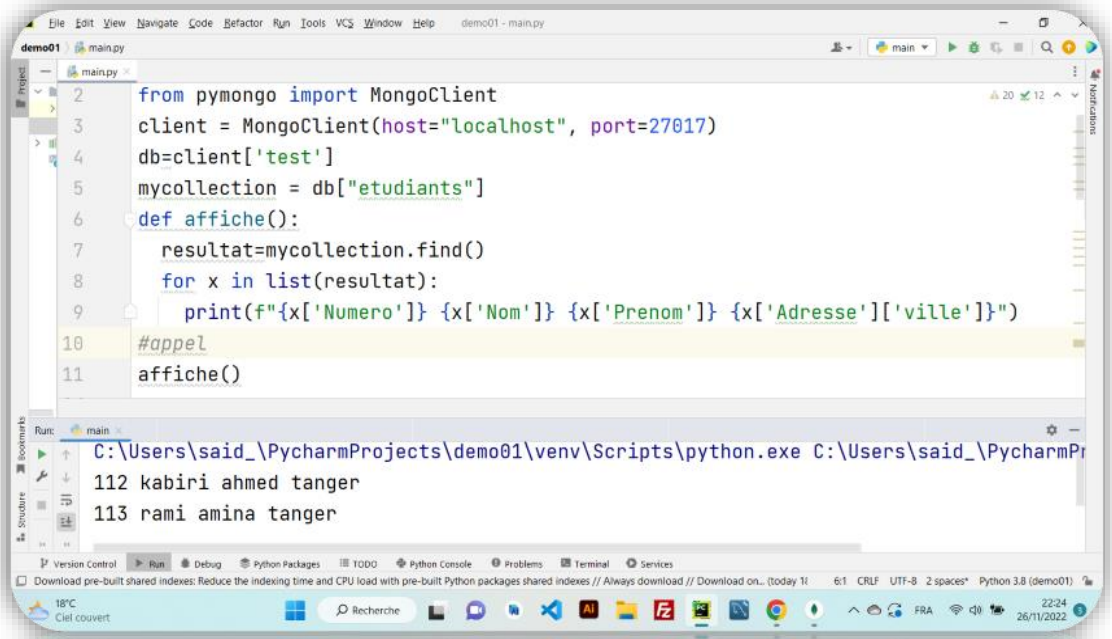
Formateur : Said GAHI

MongoDB et Python

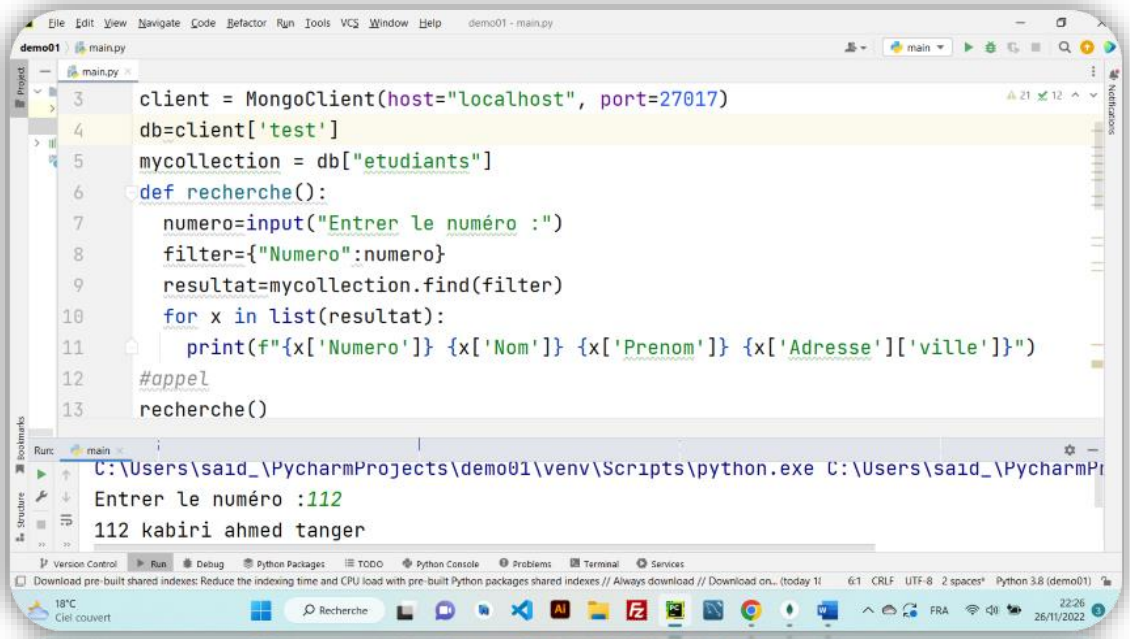
Requêtes depuis Python

Présentation et installation

Pour installer la librairie, nous pouvons le faire avec la commande **pip install pymongo**, dans un terminal.



Afficher tous les étudiants



Recherche un étudiant

Résumé de Cours

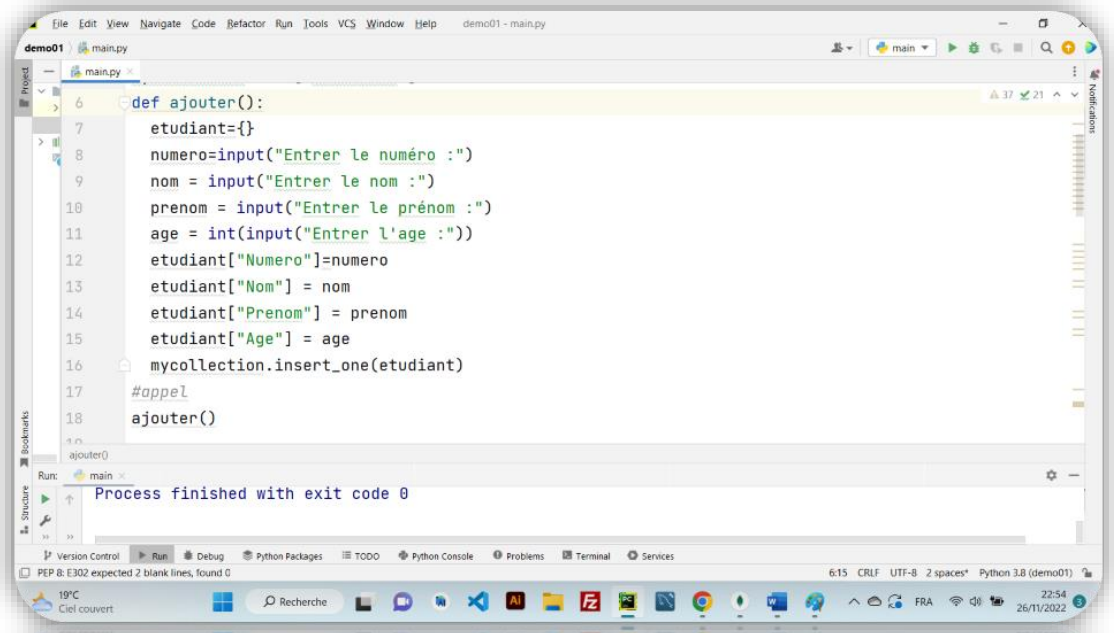
MongoDB CRUD Opérations avec Python

Formateur : Said GAHI

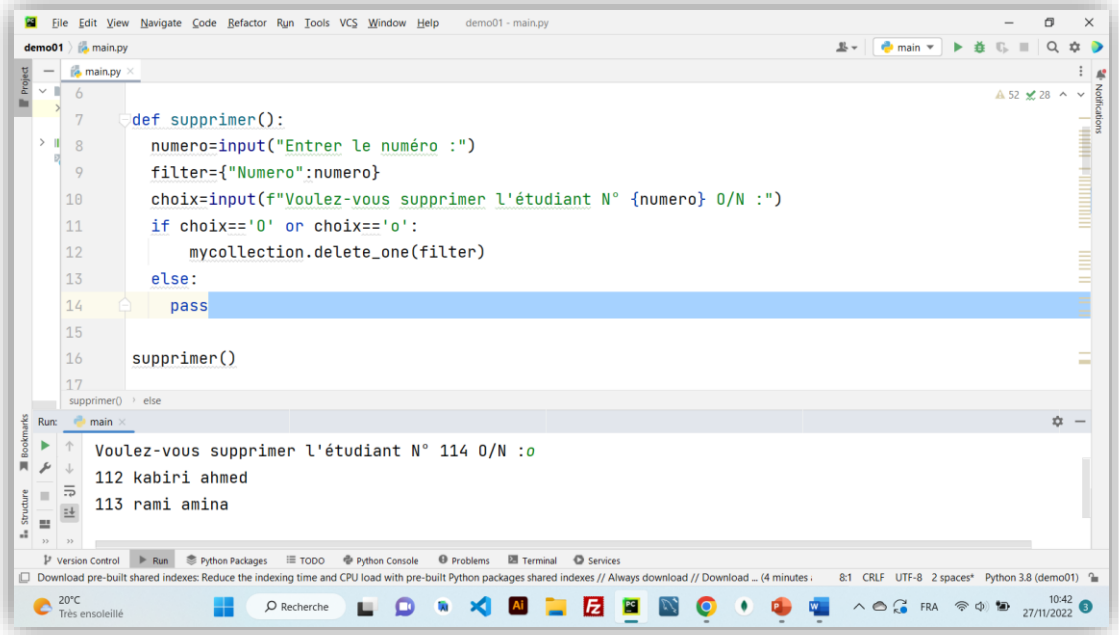
MongoDB et Python

Requêtes depuis Python

Présentation et installation



Ajouter un étudiant



Supprimer un étudiant