- Des valeurs génériques de type tableau, objet, booléen, nombre, chaîne ou null.

2 > AVANTAGES

Ces types de données sont suffisamment génériques et abstraits pour, d'une part, pouvoir être représentés dans n'importe quel langage de programmation, d'autre part, pouvoir représenter n'importe quelle donnée concrète.
Le principal avantage de JSON est qu'il est simple à mettre en œuvre par un développeur tout en étant complet.
Au rang des avantages, on peut également citer :

- peu verbeux, ce qui le rend lisible aussi bien par un humain que par une machine ;
- facile à apprendre, car sa syntaxe est réduite et non extensible (bien qu'il souffre de quelques limitations) ;
- ses types de données sont connus et simples à décrire.

3 MONGO DB

D'après Wikipédia, MongoDB est développé depuis 2007 par MongoDB. Cette entreprise travaillait alors sur un système de Cloud computing, informatique à données largement réparties, similaire au service Google App Engine de Google. Sa première version considérée comme industriellement viable a été la 1.4, en 2010.
MongoDB est une base de données open source orientée documents qui fournit de hautes performances, une haute disponibilité, et mise à l'échelle automatique.

3.1 LES COMMANDES DE LA BASE DE DONNÉES

1 > CONNEXION ET CRÉATION D'UNE BASE DE DONNÉES

Pour choisir la base sur laquelle vous voulez travailler, il faut la sélectionner à l'aide de la commande use db (db étant à remplacé par le nom de la base de données choisie - ici slam3).

```
use <base de données>
```

2 > SUPPRESSION D'UNE BASE DE DONNÉES

```
db.dropDatabase()
```

3 > LISTER LES BASES DE DONNÉES EXISTANTES

```
Show dbs
```


4 > LISTER LES COMMANDES POSSIBLES SUR LES BASES DE DONNÉES

```
db.help()
```

3.2 LES COMMANDES POUR LES COLLECTIONS

1 > LISTER LES COLLECTIONS

```
Show collections
```

2 > CRÉATION D'UNE COLLECTION

```
db.createCollection('[Nom de collection]')
```

3 > INSÉRER UN DOCUMENT

```
db.<Nom de la collection>.insert({cle : valeur})
```

4 > VÉRIFIER L'AJOUT

```
db.<Nom de la collection>.find()
```

5 > SUPPRESSION D'UNE COLLECTION

```
db.<Nom de la collection>.drop()
```

3.3 LES MODIFICATIONS D'UNE COLLECTION

```
db.<Nom de la collection>.update(1)
```

Remarque	1: prend au moins 2 paramètres éléments concernés par la modification et les types de modifications

1 > LES MODIFICATIONS POSSIBLES

COMMANDE	DÉSIGNATION
$set	Modifier la valeur d'un champ
$unset	Supprimer un champ
$inc	Incrémenter la valeur d'un champ
$mul	Multipliser l'ancienne valeur d'un champ par la valeur spécifiée
$rename	Renommer un champ

Exemple :

```
db.personne.update({nom:"bob"},{$set:{ville:'Marseille'}})
```

Supprimer un champ d'un document :

```
db.personne.update({nom: "bob"},{$unset:{prenom:1}})
```

L'incrémentation de 20 pour les valeurs de age :

```
db.personne.update({nom: "bob"},{$inc:{age:20}})
```

2 > LA SUPPRESSION D'UN DOCUMENT

La suppression d'un document se fait grâce à la méthode remove.

```
db.<nom de la collection>.remove ({<paramètre>:<valeur>})
```

3.4 LE FILTRAGE D'UNE COLLECTION

1 > RÉCUPÉRER TOUS LES DOCUMENTS D'UNE COLLECTION

```
db.<nom de la collection>.find()
```

2 > RÉCUPÉRER TOUS LES DOCUMENTS SELON DES CRITÈRES

```
db.<nom de la collection>.find({paramètre : valeur...})
```

3 > COMPTER LE NOMBRE DE DOCUMENTS

```
db.<nom de la collection>.find().count()
```

4 > TRIER LE RÉSULTAT DANS UN ORDRE CROISSANT

```
db.<nom de la collection>.find().sort({parametre:1})
```

5 > TRIER LE RÉSULTAT DANS UN ORDRE DÉCROISSANT

```
db.<nom de la collection>.find().sort({parametre:-1})
```


6 >LIMITER LE NOMBRE DE DOCUMENT À AFFICHER

```
db.<nom de la collection>.find().limit(<nbre>)
```

7 > NE PAS AFFICHER UN CHAMP

```
db.<nom de la collection>.find({},{parametre : 0})
```

3.5 QUELQUES EXPRESSIONS RÉGULIÈRES

1 > COMMENCER PAR UNE LETTRE

```
db.<nom de la collection>.find({parametre : /^<lettre>/})
```

2 > TERMINER PAR UNE LETTRE

```
db.<nom de la collection>.find({parametre : /<lettre>$/})
```

Exemples :

les personnes dont le nom commence par W :

```
db.personnes.find({nom : /^w/})
```

Les personnes dont le nom commence par e ou h :

```
db.personnes.find({nom : /^[eh]/})
```

Les personnes dont le nom commence par une lettre comprise entre e et h :

```
db.personnes.find({nom: /^[e-h]/})
```

Les personnes dont le nom se termine par h :

```
db.personnes.find({nom : /h$/})
```

3.6 LES OPÉRATEURS DE COMPARAISON

OPÉRATION	DÉSIGNATION
$gt :	Supérieur à
$gte :	Supérieur ou égal
$lt :	Inférieur à
$lte :	Inférieur ou égal à
$eq :	Égal à
$ne :	Différent de
$in :	dans
$nin :	Not in

Exemple :

Les personnes de plus de 20 ans :

```
db.personnes.find({"age":{$gt:20}})
```

3.7 LES OPÉRATEURS LOGIQUES

OPÉRATION	DÉSIGNATION
$and :	et
$or :	ou

Exemple :

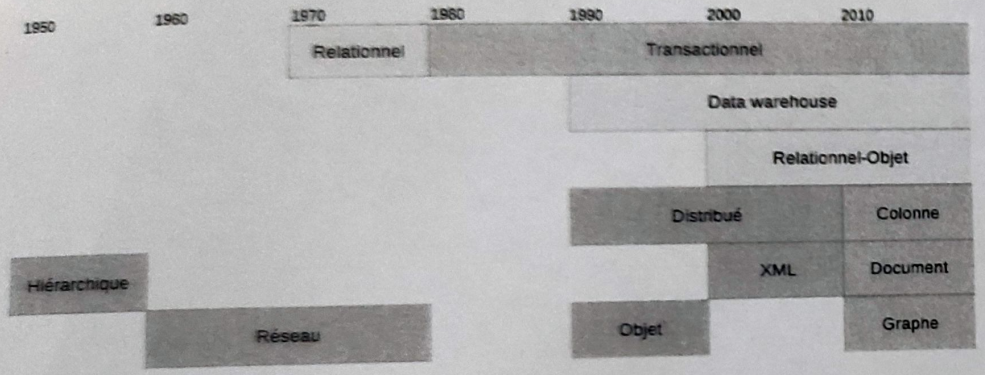
Les personnes dont l'age est compris entre 30 et 40 ans :

```
db.personne.find(
    {$and:
        [
            { age:{$gte:20}},
            { age:{$lte:30}}
        ]
    })
```

No_SQL

1 HISTORIQUE

1.1 RELATIONNEL VS NON-RELATIONNEL



Les BD NoSQL remettent en cause l'hégémonie des SGBDR telle qu'elle s'est constitué dans les années 1980.

1.2 LA DOMINATION DU MODÈLE RELATIONNEL AU BIG DATA

La première fonction d'une base de données est de permettre de stocker et retrouver l'information.

1 > RELATIONNEL ET CONTRÔLE DE L'INTÉGRITÉ DES DONNÉES

À la naissance de l'informatique, plusieurs modèles de stockage de l'information sont explorés, comme les modèles hiérarchique ou réseau.
Mais c'est finalement le modèle relationnel qui l'emporte dans les années 1970 car c'est lui qui permet de mieux assurer le contrôle de l'intégrité des données, grâce à un modèle théorique puissant et simple.

2 > L'ÉMERGENCE DU BIG DATA

Les évolutions logicielles suivent assez naturellement les évolutions matérielles. Les premiers SGBD étaient construits autour de mainframes et dépendaient des capacités de stockage de l'époque. Le succès du modèle relationnel est dû non seulement aux qualités du modèle lui-même mais aussi aux optimisations de stockage que permet la réduction de la redondance des données. Avec la généralisation des interconnexions de réseaux, l'augmentation de la bande passante sur Internet et la diminution du coût de machines moyennement puissantes, de nouvelles possibilités ont vu le jour, dans le domaine de

l'informatique distribuée et de la virtualisation, par exemple. Le passage au XXIe siècle a vu les volumes de données manipulées par certaines entreprises ou organismes, notamment ceux en rapport avec Internet, augmenter considérablement. Données scientifiques, réseaux sociaux, opérateurs téléphoniques, bases de données médicales, agences nationales de défense du territoire, indicateurs économiques et sociaux, etc., l'informatisation croissante des traitements en tout genre implique une multiplication exponentielle de ce volume de données qui se compte maintenant en pétaoctets (100 000 téraoctets). C'est ce que les AngloSaxons ont appelé le Big Data. La gestion et le traitement de ces volumes de données sont considérés comme un nouveau défi de l'informatique, et les moteurs de bases de données relationnelles traditionnels, hautement transactionnels, semblent totalement dépassés.

3 > LES PRINCIPALES BASES NOSQL OPENSOURCE

- CouchDB, un produit du projet Apache couchdb.apache.org. C'est une base de type "documentaire".
- Cassandra développée chez Facebook, et maintenant en Open Source, dans la couveuse de projets de la fondation cassandra.apache.org. C'est une base de type "colonnes".
- MongoDB, une base NoSQL orientée document, particulièrement bien diffusée et professionnalisée mongodb.org. Un enregistrement dans MondoDB est un document, qui est une structure de données champ-valeur. Les documents dans MongoDB sont similaires à des objets JSON. Les valeurs d'un champ peuvent inclure d'autres documents, des tableaux, ou même des tableaux de documents.
- Neo4j

2 LES FICHIERS JSON

2.1 JSON ET SON INTÉRÊT

1 > DÉFINITION

JSON ou JavaScript Object Notation, est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple. Créé par Douglas Crockford entre 2002 et 2005, il est décrit par la RFC 7159 de l'IETF.
Un document JSON a pour fonction de représenter de l'information accompagnée d'étiquettes permettant d'en interpréter les divers éléments, sans aucune restriction sur le nombre de celles-ci.
Un document JSON ne comprend que deux types d'éléments structurels :

- Des ensembles de paires nom / valeur ;
- Des listes ordonnées de valeurs.

Ces mêmes éléments représentent trois types de données :

- Des objets ;
- Des tableaux ;