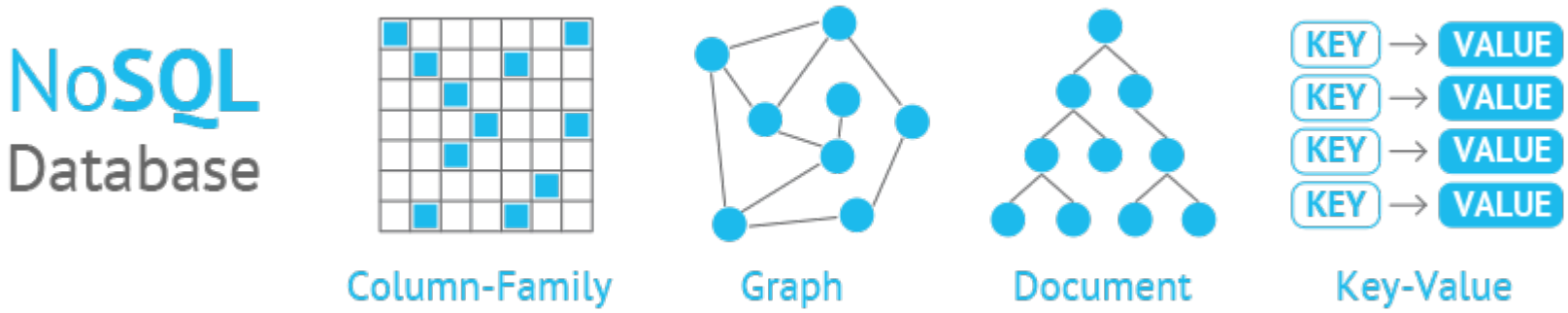


Chapitre 4: Bases de données NoSQL Sous Python

Dr Mohamed Amine Ferrag

Département Informatique

Université de Guelma

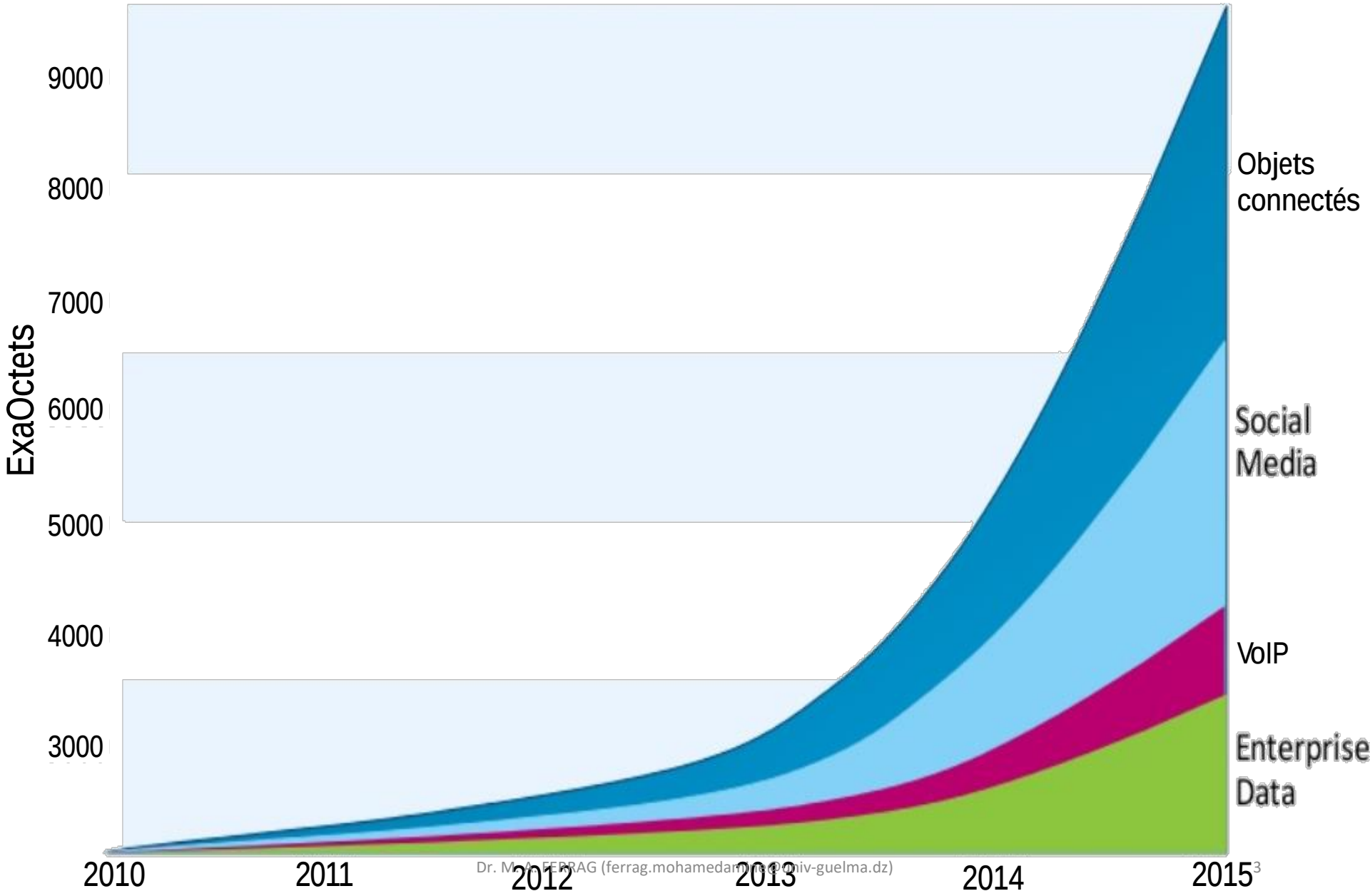
Master 1 STIC

2020-2021

Qu'est-ce qu'une base de données ?

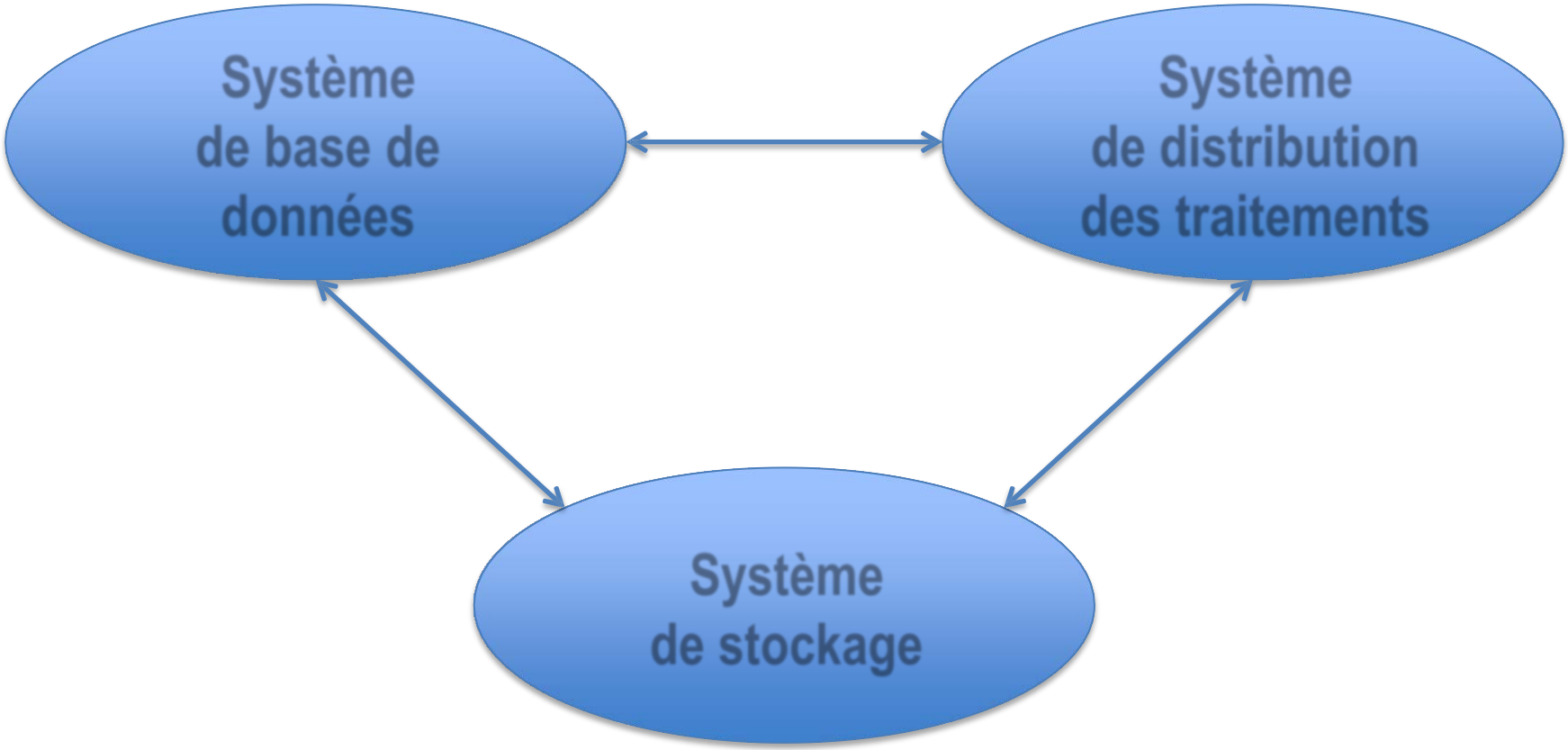
Stocker et retrouver les informations

Le mouvement Big Data

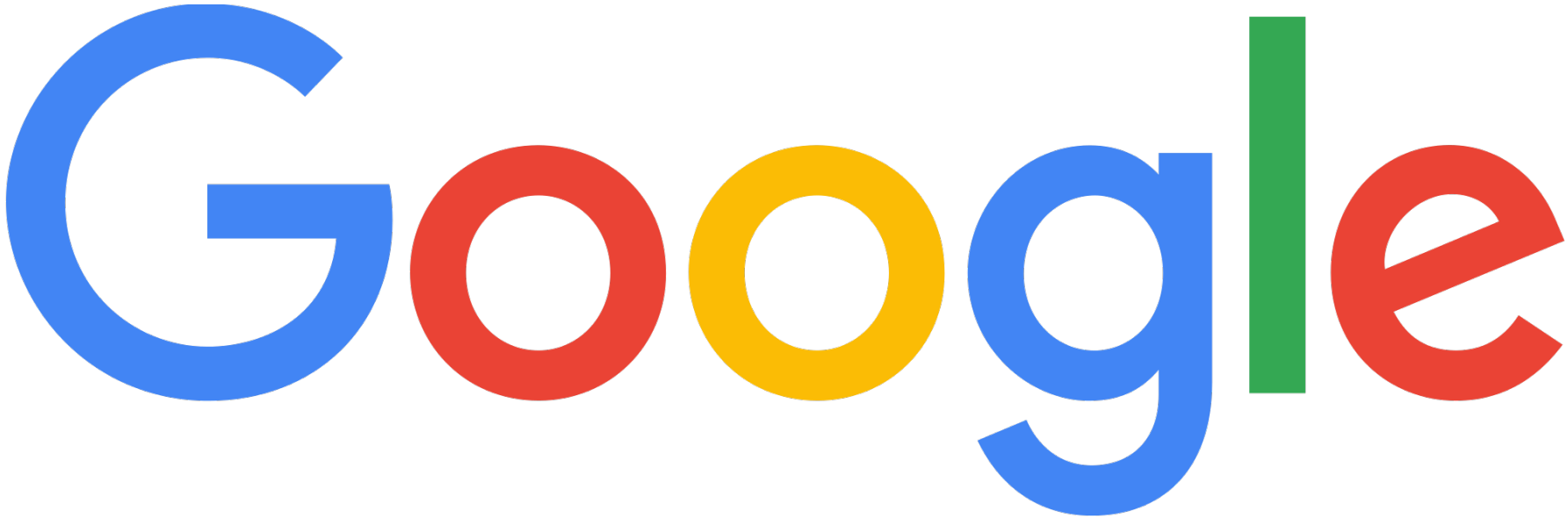


Le mouvement Big Data

- Big data nécessite une architecture distribuée

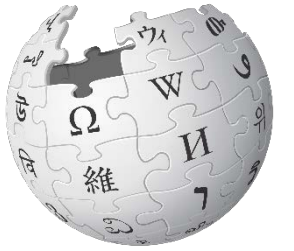


Les débuts de NoSQL



Définition « NoSQL »

- NoSQL (Not only SQL en anglais) désigne une catégorie de systèmes de gestion de base de données (SGBD) qui n'est plus fondée sur l'architecture classique des bases relationnelles. L'unité logique n'y est plus la table, et les données ne sont en général pas manipulées avec SQL



WIKIPEDIA
The Free Encyclopedia

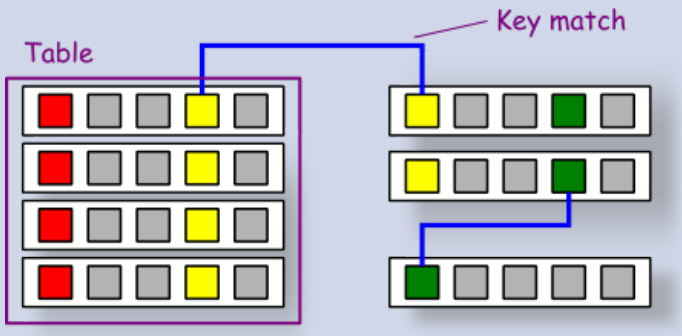
Pourquoi NoSQL ?

- Licence des SGBDR très chère (Oracle, ...).
- Le SQL a un schéma fermé.
- Performances faibles, sur de gros volumes de données, comparées au NoSQL.

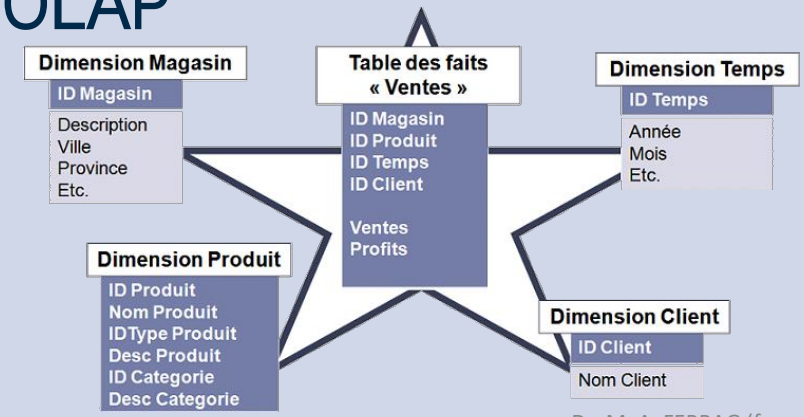
Les grandes familles de bases de données

SQL

OLTP



OLAP

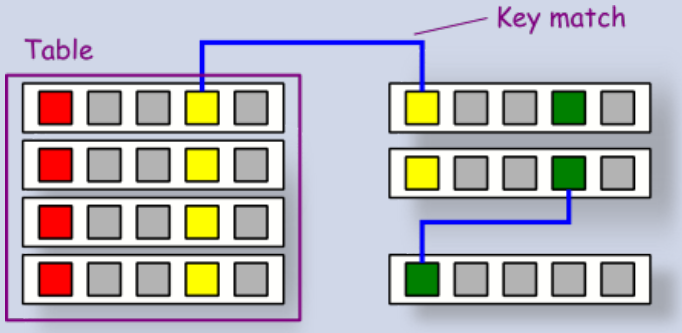
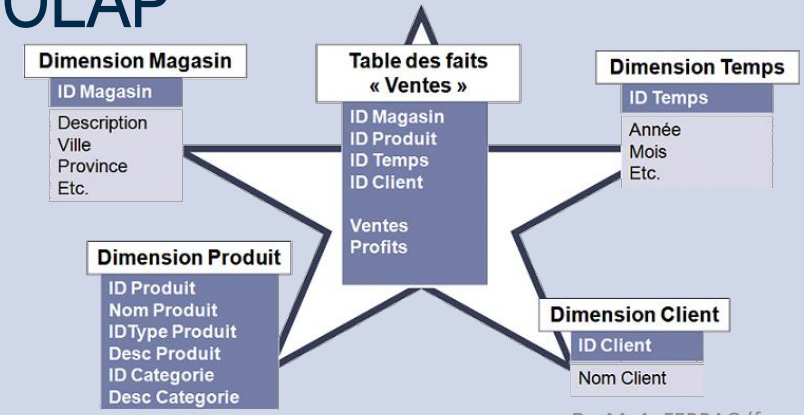


Non SQL

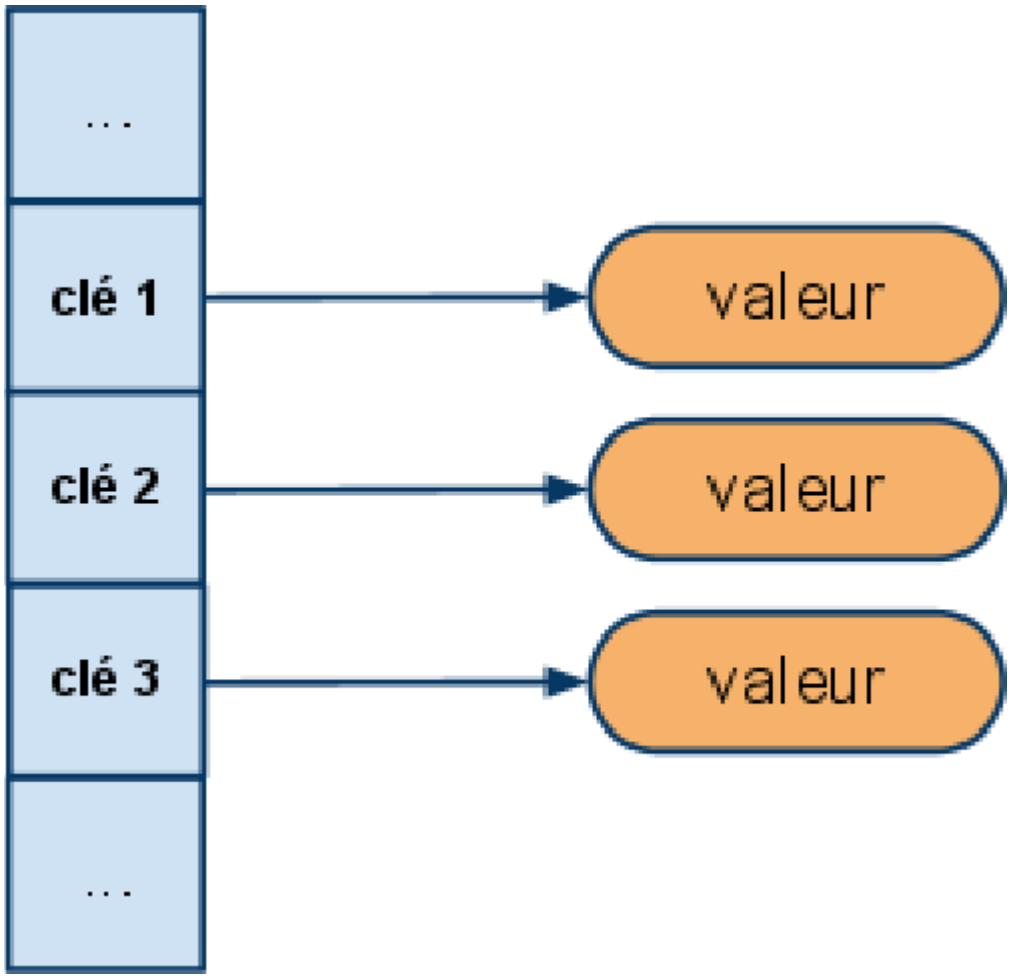
225 SGBD

http://nosql-database.org/

Les grandes familles de bases de données

SQL	Non SQL
OLTP  OLAP 	Clé-valeur Document Colonnes Graphe

Une base de données - **clé-valeur**



- Conçues à l'origine pour répondre aux besoins de performance des géants du Web (Amazon, Google, Facebook)
- La valeur associée à une clé peut être une simple chaine de caractère comme un document, ou encore un objet beaucoup plus complexe pouvant contenir une multitude d'information.

Clés-valeurs

Clés-valeurs

- 1 base = 1 table

Clés primaires	Valeurs
Loto : Samedi 23 avril 2016	2, 15, 17, 22, 26 - 7
La grande question sur la vie, l'univers et le reste	42
logo cnrs	

http://prodev.cnrs.fr/doku.php?id=nosql2016	`<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr" class="no-js">`
	
	`<head>`
	`<meta charset="UTF-8" />`
	`<title>nosql2016 [prodev]</title>`
	…
	`<body>`
	…
	`</body> </html>`

Clés-valeurs

- API simple
  - Put
  - Get
  - Delete
  - *Update (facultatif)*

Clés-valeurs

- Exemple Twitter
  - Utilisateurs

Clés primaires	Valeurs
user:guillaumeharry:nom	HARRY
user:guillaumeharry:prenom	guillaume

  - Messages

Clés primaires	Valeurs
message:post32	"user_id":"guillaumeharry", "time":"26/04/2016", "body":"En pleine présentation"
message:post33	"user_id":"bernardchetrit", "time":"26/04/2016", "body":"En pleine concentration"

  - Mots clés

Clés primaires	Valeurs
pleine	post32, post33
concentration	post33

Clés-valeurs

- Exemple Twitter
  - Contenu de la base de données

Clés primaires	Valeurs
user:guillaumeharry:nom	HARRY
user:guillaumeharry:prenom	guillaume
message:post32	"user_id":"guillaumeharry", "time":"26/04/2016", "body":"En pleine présentation"
message:post33	"user_id":"bernardchetrit", "time":"26/04/2016", "body":"En pleine concentration"
pleine	post32, post33
concentration	post33
tion	présentation, concentration
pre	présentation

Clés-valeurs

- Cible
  - Stockage de gros volume de données
- Avantage
  - Recherche rapide
  - Table à 2 colonnes
- Inconvénient
  - Aucun schéma
  - Pas de garantie d'intégrité

Clés-valeurs

Implémentations

- Dynamo
  - Développée en 2007 et utilisée par Amazon pour gérer le panier d'achat
- Voldemort
  - Développée et utilisée par LinkedIn
- Riak
  - Edité par Basho Technologies
  - Inspirée de Dynamo
  - Base hybride orientée clé/valeur ou document
- Redis
  - Hautement performant
  - Redis n'est pas tolérant aux pannes
- Memcached
  - Cache mémoire distribué
  - Eprouvé depuis de nombreuses années

Clés-valeurs

En production

- The Guardian

  (quotidien d'information britannique)

- GitHub

- Stack Overflow

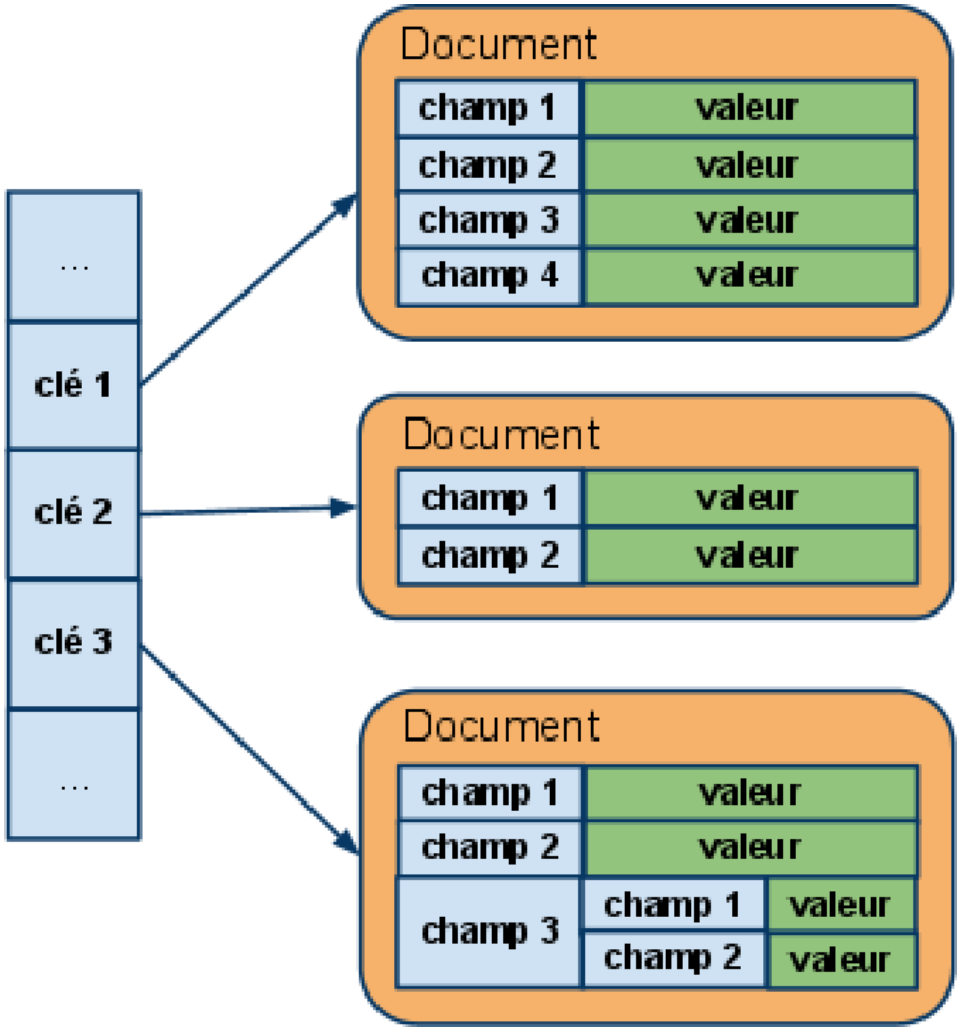
- Craigslist

  (site web américain de petites annonces et de forums de discussion)

Les 4 grandes familles du NoSQL

- Clés-valeurs
- Document
- Orientées colonnes
- Graphes

Document


...
clé 1
clé 2
clé 3
...
Document
champ 1
valeur
champ 2
valeur
champ 3
valeur
champ 4
valeur
Document
champ 1
valeur
champ 2
valeur
Document
champ 1
valeur
champ 2
valeur
champ 3
champ 1
valeur
champ 2
valeur

Document

- Une base de données orientée documents est une base de données destinée aux applications qui gèrent des documents.
- Deux langages sont maintenant principalement utilisés pour représenter les documents structurés : **XML** et **JSON**.
- Il existe deux représentations d'un document :
- Forme sérialisée : c'est la forme courante, où le contenu est marqué par des balises ou par des accolades ouvrante/fermante.
    - Forme arborescente structure du document.
    - La forme des requêtes est dépendante du langage de représentation des documents.
- Il existe deux langages de requêtes XML : **XPath** et **XQuery**.

Document

Exemple


```
{
    "_id" : ObjectId("56a01213f89947e35a4a6d01"),
    "name" : "mycom",
    "description" : {
        "title" : "Conditions générales d'utilisation de My Com",
        "summary" : "Demande d'activation du service My Com après acceptation des conditions générales d'utilisation",
        "content" : "\r\nCGU My COM\r\nOffre de Service\r\nV1.0 Janvier 2016\r\n\r\nI.\tObjet\r\nLes présentes conditions générales d’utilisation (Dénommées ci-après CGU
        "url" : "../docs/My_Com_CGU_v1.0.pdf"
    },
    "version" : "1.0",
    "fromDate" : "2015-12-31T10:00",
    "toDate" : "2015-12-31T10:00",
    "type" : "GTU",
    "dialogMessage" : "Merci d'avoir signé les CGU de My Com. Votre demande de création de compte a bien été enregistrée. Vous allez recevoir un e-mail quand cette créat
    "linkedRequest" : {
        "name" : "mycom",
        "type" : "MAILING_LIST",
        "version" : "1.0"
    },
    "decisionMessage" : "J'accepte les conditions générales d’utilisation du service"
}
{
    "_id" : ObjectId("56a14957f89987311256e59f"),
    "name" : "mycom",
    "description" : {
        "title" : "Liste de diffusion My Com",
        "summary" : "Inscription à la liste de diffusion de My Com",
        "content" : "Votre inscription sur notre liste de diffusion nous permettra de vous \ntenir informé des actualités du service : \n - indisponibilités pour cause d
    },
    "version" : "1.0",
    "type" : "MAILING_LIST",
    "dialogMessage" : "Merci d'avoir signé les CGU de My Com. Votre demande de création de compte a bien été enregistrée. Vous allez recevoir un e-mail quand cette créat
    "linkedToParent" : true,
    "decisionMessage" : "J’accepte d’être inscrit à la liste de diffusion My Com"
}
```

Document


```
{
  "_id": "548855bb3c5f16d690df3be6",
  "infos": {
    "firstName": "John",
    "lastName": "DOE",
    "uid": "john.doe.2",
    "mail": "john.doe@cnrs.fr",
    "idEbooking": "8GEB8AAF",
    "unit": {
      "code": "UMR1234",
      "label": "Unite mixte de travail sur des choses tres interessantes",
      "acronym": "UMTCI",
      "implantationNumber": "123456789"
    },
    "mainDr": {
      "code": "05",
      "label": "Ile de France - Ouest et Nord",
      "supportUnit": "MOY0500"
    },
    "billingCenter": {
      "code": "05",
      "label": "Ile de France - Ouest et Nord",
      "supportUnit": "MOY0500",
      "modifiedByAdmin": false
    },
    "functions": [
      {
        "domain": "CNRS",
        "code": "DU",
        "value": "UMR5678"
      },
      {
        "domain": "INRIA",
        "code": "ABC",
        "value": "ppp_34jjh"
      }
    ],
    "present": true,
    "locked": false,
    "birthDate": "1974-12-10"
  },
  "cards": {
    "4d74fbe0-ab16-11e4-8f72-0002a5d5c51b": {
      "orderID": 1,
      "type": "PASSPORT",
      "number": "34-ERER-DFDFDF",
      "country": "NO",
      "city": "Tromsø",
      "submissionDate": "1998-01-04",
      "expirationDate": "2008-01-04"
    }
  },
  "addresses": {
    "professional": {
      "phone": "33-(1) 23456789",
      "fax": "33-123456780",
      "mail": "john.doe@cnrs.fr",
      "address": "33 bis rue de la mairie",
      "complement": "4eme etage",
      "postalCode": "2A3245",
      "city": "Perpignan",
      "country": "FR"
    },
    "personal": {
      "phone": "33-567766789",
      "mobile": "33-665544333",
      "address": "1221 avenue d'Italie",
      "postalCode": "2A3246",
      "city": "Carcassonne",
      "country": "FR",
      "otherMail": "johnatan.doe@gmail.com"
    }
  },
```

Document

- **Cible**
  - Recherche complexe
- **Avantage**
  - Données semi-structurées
  - Gestion de la version du document
- **Inconvénient**
  - Performances des requêtes

Document

Implémentations

- CouchDB
  - Pas de verrou lors des accès concurrents
- MongoDB
  - Développé par la société 10gen
  - Permet d'indexer les propriétés des documents afin d'optimiser une recherche
  - Support de l'indexation géospatiale
- Riak
  - Base hybride orientée clé/valeur ou document

Document

En production

- Ebay
  - Métadonnées de chaque objet vendu
- Expedia
- McAfee
  - Référentiel central pour la plateforme de détection des menaces
- City of Chicago
  - Données géospatiales pour gérer toutes les urgences (situation des bus, 911, …)
- Telefonica
- Mariott
- PayPal

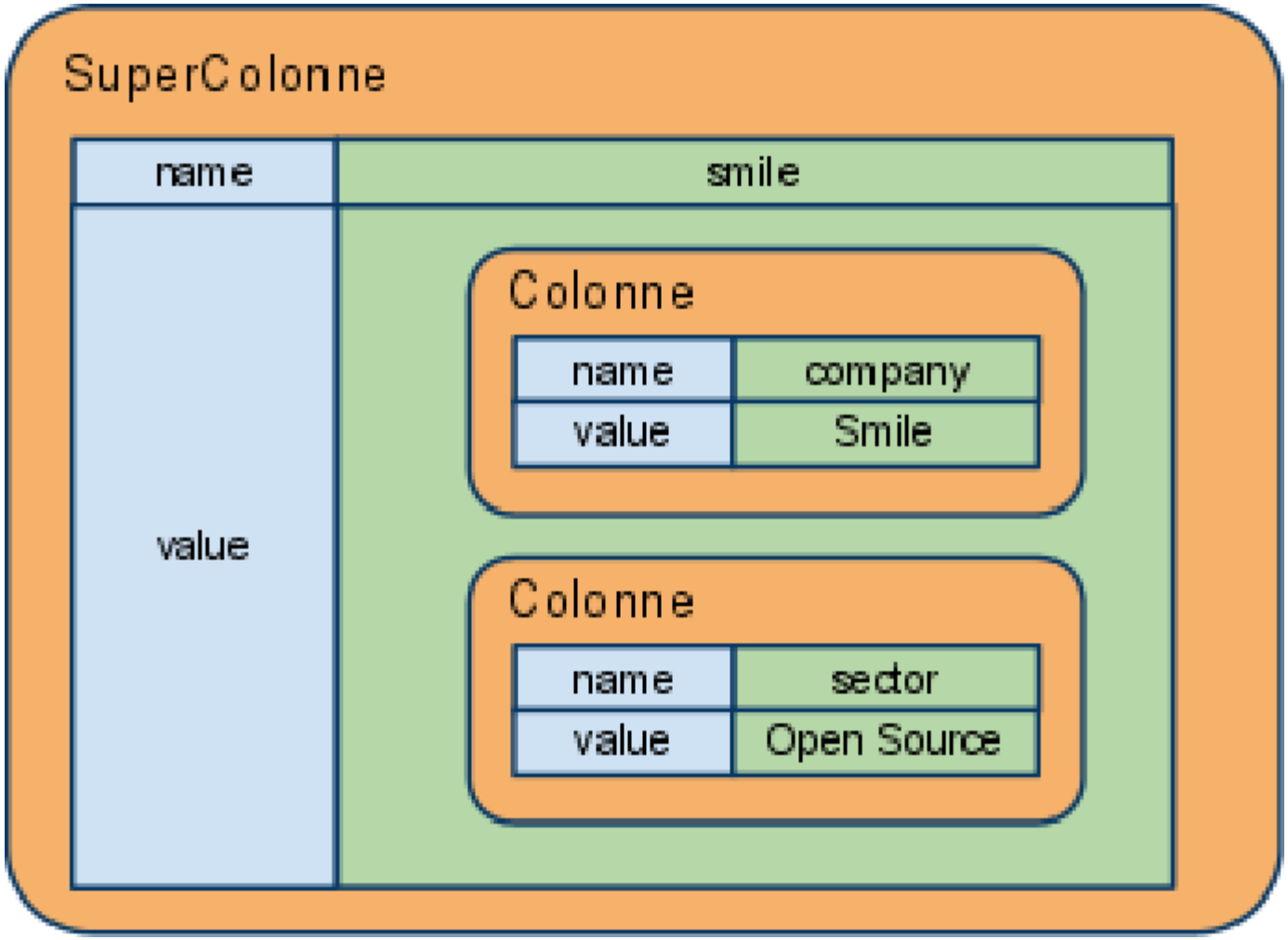
Les 4 grandes familles du NoSQL

- Clés-valeurs
- Document
- Orientées colonnes
- Graphes

Orientées colonnes

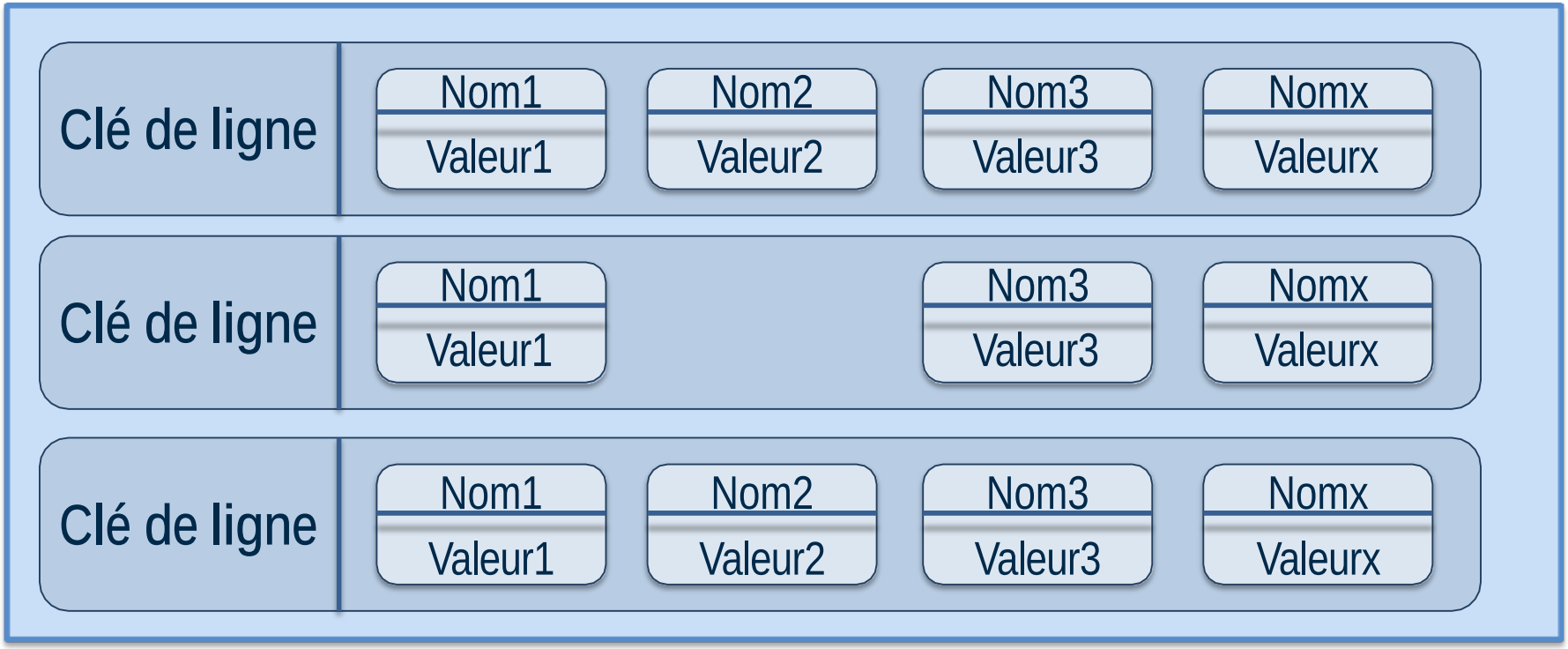
- Ces bases de données NoSQL sont celles se rapprochant le plus des bases de données classiques (SGBDR). En effet, on y retrouve le principe de “table” avec des lignes et des colonnes, mais elles ont deux principales grosses différences :
  - **Les colonnes sont dynamiques**. Au sein d’une même table deux individus peuvent ne pas avoir le même nombre de colonnes car les valeurs nulles ne sont pas stockées (ce qui est le cas dans les SGBDR relationnels).
  - **L’historisation des données** se fait à la valeur et non pas à la ligne comme dans les SGBDR

Orientées colonnes



Orientées colonnes

- **Famille de colonnes**
  - Equivalent à une table dans une base de données relationnelle



Orientées colonnes

- **Structure des données par familles de colonnes**
  - 1 clé pour accéder à un ensemble de colonnes
  - Colonnes sont groupées par famille de colonne
- **Données stockées par colonne**
  - Chaque colonne est définie par un couple clé-valeur
  - Les colonnes sont regroupées par ligne
  - Chaque ligne est identifiée par un identifiant unique.

- **Requête sur**
  - Lignes
  - Familles de colonnes
  - Noms de colonnes

Orientées colonnes

- La colonne est une valeur !
  - Nom de colonne

    Tous les trains qui passent à Valence

  - Tri sur les colonnes (slice)

    Toutes les données de 9h à 11h

- MAIS PAS DE JOINTURE !

Orientées colonnes

- Cible
  - Répond aux problématiques
    - de charge
    - de volume
    - de très haute disponibilité
  - Timeseries
    - Données stockées suivant des timestamps
    - Adapter aux données issues de capteurs
  - Clickstreams
    - Enregistrement des actions d'un utilisateur sur une application
- Avantages
  - Forte tolérance aux pannes

Orientées colonnes

Implémentations

- Google BigTable
- Hbase
  - Clone de **BigTable** développé au sein de l'écosystème Hadoop
  - Très performant en lecture
- Cassandra
  - Initialement créé par Facebook
  - Très performant en écriture
  - Pas de garantie de cohérence stricte

Orientées colonnes

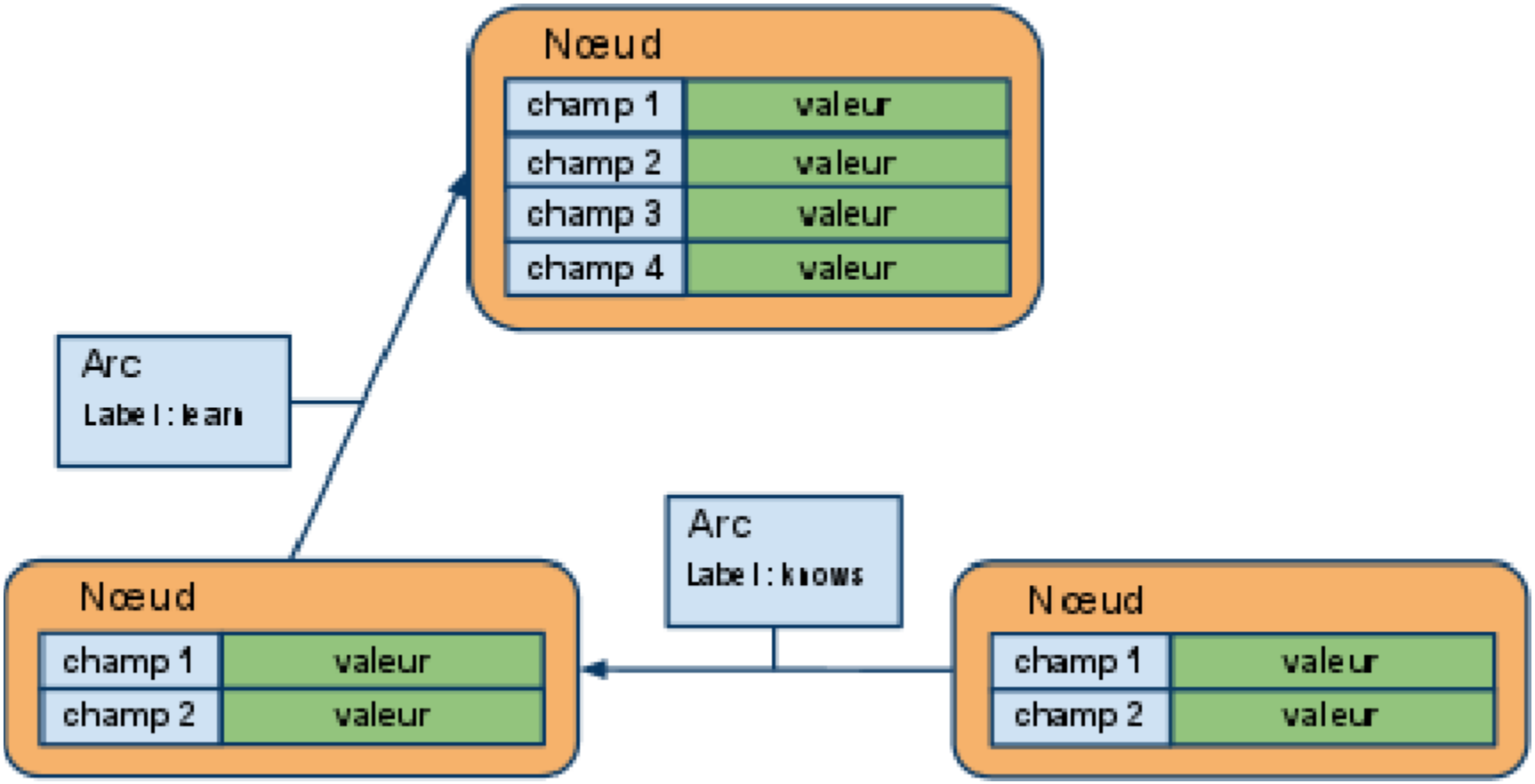
En production

- Google
- Netflix
  - Historique complet des visualisations des 36 millions d'utilisateurs
- Ebay
  - Données sociales (like, own, want)
- Twitter
- Cisco
- Facebook

Les 4 grandes familles du NoSQL

- Clés-valeurs
- Document
- Orientées colonnes
- Graphes

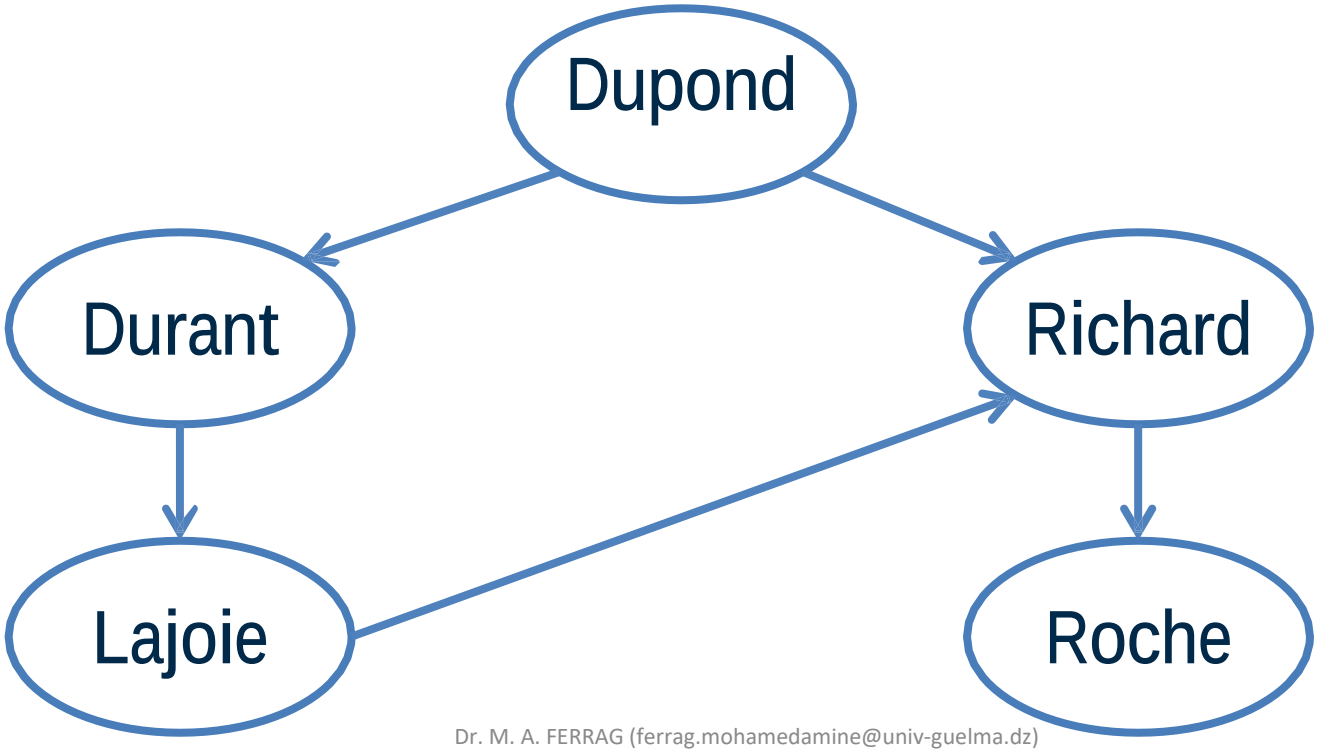
Graphe


Nœud
champ 1
valeur
champ 2
valeur
champ 3
valeur
champ 4
valeur
Arc
Label : learn
Arc
Label : knows
Nœud
champ 1
valeur
champ 2
valeur
Nœud
champ 1
valeur
champ 2
valeur

Graphe

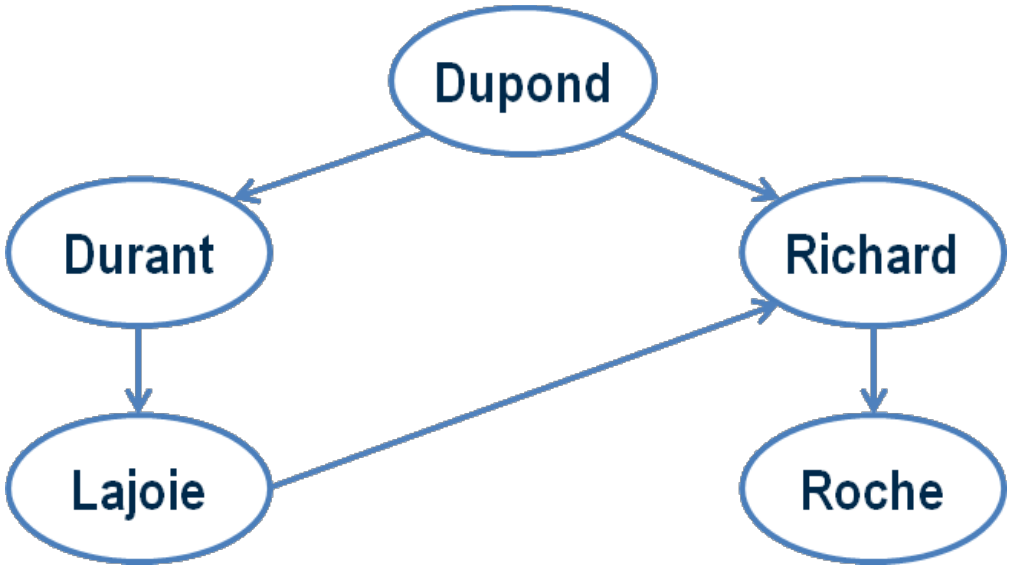
- Permet de décrire les relations entre des entités
  - Modélisation des problématiques spécifiques avec forte connectivité
  - Recherche de liens optimaux

- Exemple : matérialiser un réseau d'amis



Graphe

- Exemple : matérialiser un réseau d'amis en SQL
  - 2 tables USER, USER_FRIENDS

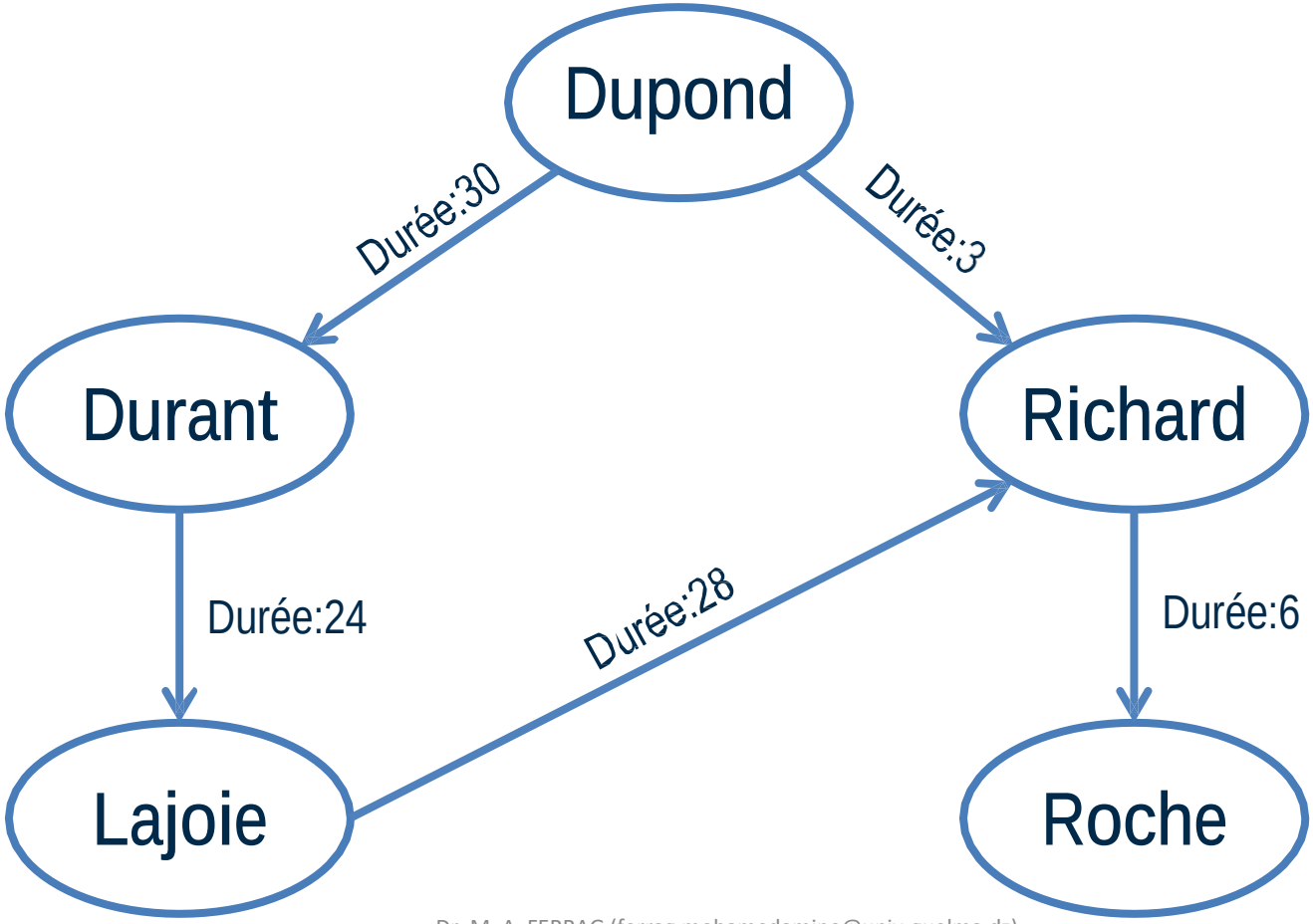


ID	NOM
1	Dupond
2	Durand
3	Richard
4	Ducode
5	Roche

ID	FROM_USER	TO_USER
1	1	2
2	1	3
3	2	5
4	5	3
5	3	4

Graphe

- Exemple : matérialiser un réseau d'amis avec Neo4J
  - Quelles sont les amitiés de plus de 10 ans ?
    - On ajoute des propriétés aux relations



Exemple : matérialiser un réseau d'amis avec Neo4J

- Quelles sont les amitiés de plus de 10 ans ?
  - Et on demande

```
$ match (p1:Personne)-[r:Est_Ami]->(p2:Personne) where r.duree >=10 return r,p1,p2
```

Graph
Rows
Code

r	p1	p2
duree 30	age 35 nom Dupond	nom Durant
duree 24	nom Durant	nom Lajoie
duree 28	nom Lajoie	nom Richard

Returned 3 rows in 1515 ms.

Graphe

- **Cible**
  - Représentation de relations, de réseaux, d'organisations
- **Avantage**
  - Algorithmes de la théorie des graphes (chemin le plus court, degré de relation, …)
- **Inconvénient**
  - Parcours complet de la base obligatoire pour avoir une réponse exhaustive
- **Implémentations**
  - Neo4J

Graphe

Implémentations

- Neo4J
- Titan
- FlockDB
- GraphBase
- InfiniteGraph

Graphe

En production

- Cisco
- Ebay
  - Gestion des livraisons
- SFR
- TomTom
- Lufthansa
- Meetic

Dans ce qui suit, nous allons étudier en detail….

MongoDb - *Historique*

- 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.

- Il est ensuite devenu un des SGBD les plus utilisés, notamment pour les sites web de Craigslist, eBay, Foursquare, SourceForge.net, Viacom, et le New York Times

MongoDB - *Documents*

- MongoDB est orienté document. Qu'est ce qu'un document ?
- Un document est la représentation d'une donnée en BSON.
- BSON = *Binary JSON.*
- **JSON** (*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.

MongoDB - *Documents*

- **Exemple :**

```
{
"name" : "MongoDB",
"type" : "database",
"count" : 1,
"info" : {
        x : 203,
        y : 102
}
}
```

MongoDB - *Organisation*

- Un serveur MongoDB est composé de bases de données.
- Une base de données contient des collections.
- Les collections possèdent les documents.
- Chaque document possède un identifiant unique généré par MongoDB, le champ _id.

MongoDB - *Démarrage*

- MongoDB vient avec un shell : **bin/mongo**
- Démarrage avec : **bin/mongod**
- Afficher la base de données courante : **db**
- Afficher la liste des bases de données : **show dbs**
- Sélectionner une base de données : **use <name>**
- Afficher les collections : **show collections**

MongoDB - *Démarrage*

Installer MongoDb

- Vous pouvez trouver MongoDb en téléchargement ici: http://www.mongodb.org/downloads
- Une fois que vous avez récupéré l'archive de MongoDb qui correspond à votre OS, vous devez trouver 2 fichiers:

  mongod(.exe) mongo(.exe)

- *mongod* sert à démarrer le moteur de base de données tandis que *mongo* est un interpréteur de commandes.

MongoDB - *Démarrage*

- Ouvrez un premier invite de commande pour lancer le moteur de base de données:

```
mongod
```

- Si le démarrage est réussi, vous devez voir que le processus se lance sur le port 27017 qui est le port par défaut utilisé par MongoDb.
- Un message comme celui-ci vous l'indique: *waiting for connections on port 27017*
- Puis ouvrez un second invite de commande pour lancer l'intrepréteur de commandes :

```
mongo
```

MongoDB – *Creation d'une base de donnée*

- Vous pouvez utiliser la commande ***USE*** pour créer et sélectionner une base de donnée.
- **Exemple :**

use *ETUDIANTSTIC*

- Si vous voulez afficher la base de donnée courante, tapez la commande ***db***
- Si vous voulez supprimer la base de donnée courante, tapez la commande

db.dropDataBase()

MongoDB – *Insertion des données*

- Si vous voulez inserer des données, utilisez cette commande :

db.{nom_de_collections}.insert(les données)

- Exemple :

db.players.insert()

- Vous pouvez également insérer plusieurs documents

db.players.insert([{ les données du document 1 } , { les données du document 2},….])


```
db.players.insert(
{
        "position":"Goalie",
        "id":8476434,
        "twitterURL":"https://twitter.com/JohnGibson35",
        "weight":216,
        "imageUrl":"http://3.cdn.nhle.com/photos/mugs/8476434.jpg",
        "birthplace":"Pittsburgh, PA, USA",
        "twitterHandle":"JohnGibson35",
        "age":21,
        "name":"John Gibson",
        "birthdate":"July 14, 1993",
       "number":36
    })
```


https://docs.mongodb.org/manual/reference/bson-types/


```
db.players.insert(
{
      "position":"Goalie",
       "id":8475883,
       "twitterURL":"https://twitter.com/f_andersen30",
       "weight":236,
       "imageUrl":"http://3.cdn.nhle.com/photos/mugs/8475883.jpg",
       "birthplace":"Herning, DNK",
       "twitterHandle":"f_andersen30",
       "age":25,
       "name":"Frederik Andersen",
       "birthdate":"October 02, 1989",
       "number":31
      })
```

MongoDB – *Afficher les collections*

- Pour afficher les collections, utiliser la commande

show.collections()

- Pour afficher les documents d'une collections, utiliser la commande

db.nom_collection.find(…)

Exemple : db.players.find()

- Pour sélectionner un document

db.nom_collection.findOne(…) ← **Requête**

- Pour sélectionner et modifier un document, utilisez la commande

db.nom_collection.findAndModify(…)

MongoDB – *Supprimer collections et documents*

- Pour supprimer les collections, utilisez cette commande :

db.nom_collection.remove()

- Pour supprimer les documents, utilisez cette commande :

db.nom_collection.remove({préciser le ID du document})

- Pour faire la mise à jour des documents, utilisez cette commande :

db.nom_collection.update({préciser le ID du document}, {les données à jour})

MongoDB – *Python*

- Python a une bibliothèque pour MongoDB. Le nom de la bibliothèque disponible est «PyMongo». Pour importer cela, exécutez la commande suivante:

from pymongo import MongoClient

MongoDB – *Python*

- Créer une connexion: après l'importation du module, il faut créer un MongoClient.

from pymongo import MongoClient

client = MongoClient()

MongoDB – *Python*

- Ensuite, connectez-vous à l'hôte et au port par défaut. La commande suivante permet de connecter MongoClient sur l'hôte local qui s'exécute sur le numéro de port 27017.

client = MongoClient ('hôte', numéro de port)

exemple: - client = MongoClient ('localhost', 27017)

- ***Cela peut aussi être fait en utilisant la commande suivante:***

client = MongoClient("mongodb://localhost:27017/")

MongoDB – *Python*

- Accéder aux objets de la base de données: Pour créer une base de données ou basculer vers une base de données existante, nous utilisons:

mydatabase = client['name_of_the_database']

- ***ou***

mydatabase = client.name_of_the_database

MongoDB – *Python*

- Accessing the Collection : Collections are equivalent to Tables in RDBMS. We access a collection in PyMongo in the same way as we access the Tables in the RDBMS. To access the table, say table name “myTable” of the database, say “mydatabase”:

mycollection = mydatabase[‘myTable’]

- ***ou***

mycollection = mydatabase.myTable

MongoDB – *Python*

- Accessing the Collection : Collections are equivalent to Tables in RDBMS. We access a collection in PyMongo in the same way as we access the Tables in the RDBMS. To access the table, say table name “myTable” of the database, say “mydatabase”:

mycollection = mydatabase[‘myTable’]

- ***ou***

mycollection = mydatabase.myTable

MongoDB – *Python*

- Insérer les données dans une collection:

insert_one()

ou

insert_many()

PyMongo crée une collection

jupyter Untitled42 Last Checkpoint: 8 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [9]:

```
#!/usr/bin/python3

from pymongo import MongoClient

cars = [ {'name': 'Audi', 'price': 52642},
    {'name': 'Mercedes', 'price': 57127},
    {'name': 'Skoda', 'price': 9000},
    {'name': 'Volvo', 'price': 29000},
    {'name': 'Bentley', 'price': 350000},
    {'name': 'Citroen', 'price': 21000},
    {'name': 'Hummer', 'price': 41400},
    {'name': 'Volkswagen', 'price': 21600} ]

client = MongoClient('mongodb://localhost:27017/')

with client:

    db = client.testdb

    db.cars.insert_many(cars)
```

L'exemple crée une nouvelle collection de voitures. Il contient huit documents.

MongoClient est utilisé pour communiquer avec MongoDB. Nous transmettons à MongoClient un nom d'hôte et un numéro de port.

Nous obtenons une référence à la base de données testdb.

Avec la méthode insert_many (), nous insérons huit documents dans la collection de voitures, qui est également créée automatiquement.

PyMongo list collections

In [11]:
```
#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:
    db = client.testdb
    print(db.collection_names())
```

```
[u'cars', u'system.indexes']
```

Avec collection_names (), nous obtenons la liste des collections disponibles dans la base de données.

PyMongo supprimer les collections

La méthode drop () supprime une collection de la base de données.

In [12]:
```
#!/usr/bin/python3

from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')

with client:

    db = client.testdb
    db.cars.drop()
```

db.collection.find()

Sélectionne des documents dans une collection ou une vue et renvoie un curseur sur les documents sélectionnés.

- db.collection.find(query…)

La requête suivante utilise $ gt pour renvoyer des documents dont la valeur de quantité est supérieure à 4.

- db.collection.find({ qty: { $gt: 4 } })

db.collection.distinct()

Recherche les valeurs distinctes pour un champ spécifié dans une collection ou une vue unique et renvoie les résultats dans un tableau.

- db.collection.distinct(...)

L'exemple suivant renvoie les valeurs distinctes pour le champ dept de tous les documents de la collection inventory

- db.inventory.distinct("dept")

db.collection.aggregate()

Calcule les valeurs globales pour les données d'une collection ou d'une vue.

- db.collection.aggregate()

L'opération d'agrégation suivante sélectionne les documents dont le statut est égal à "A", regroupe les documents correspondants dans le champ cust_id, calcule le total de chaque champ cust_id à partir de la somme du champ du montant et trie les résultats par ordre décroissant.

- db.orders.aggregate([

```
{ $match: { status: "A" } },
{ $group: { _id: "$cust_id", total: { $sum: "$amount" } } },
{ $sort: { total: -1 } }
])
```

$match

Filtre les documents pour ne transmettre que les documents correspondant à la ou aux conditions spécifiées.

L'étape $ match a la forme de prototype suivante:

- {$ match: {<query>}}

- Pour d’autres requêtes, veuillez lire la documentation du MongoDB

https://docs.mongodb.com/

REFERENCES

- Guillaume HARRY et Olivier PORTE , Formation NoSQL, CNRS http://rbdd.cnrs.fr/