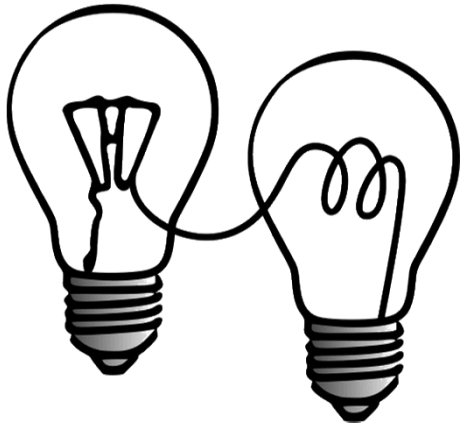
# LES BASES NOSQL ET LES OUTILS DE TYPE HADOOP

Philippe Lacomme

placomme@isima.fr
Tel. 04 73 40 75 85

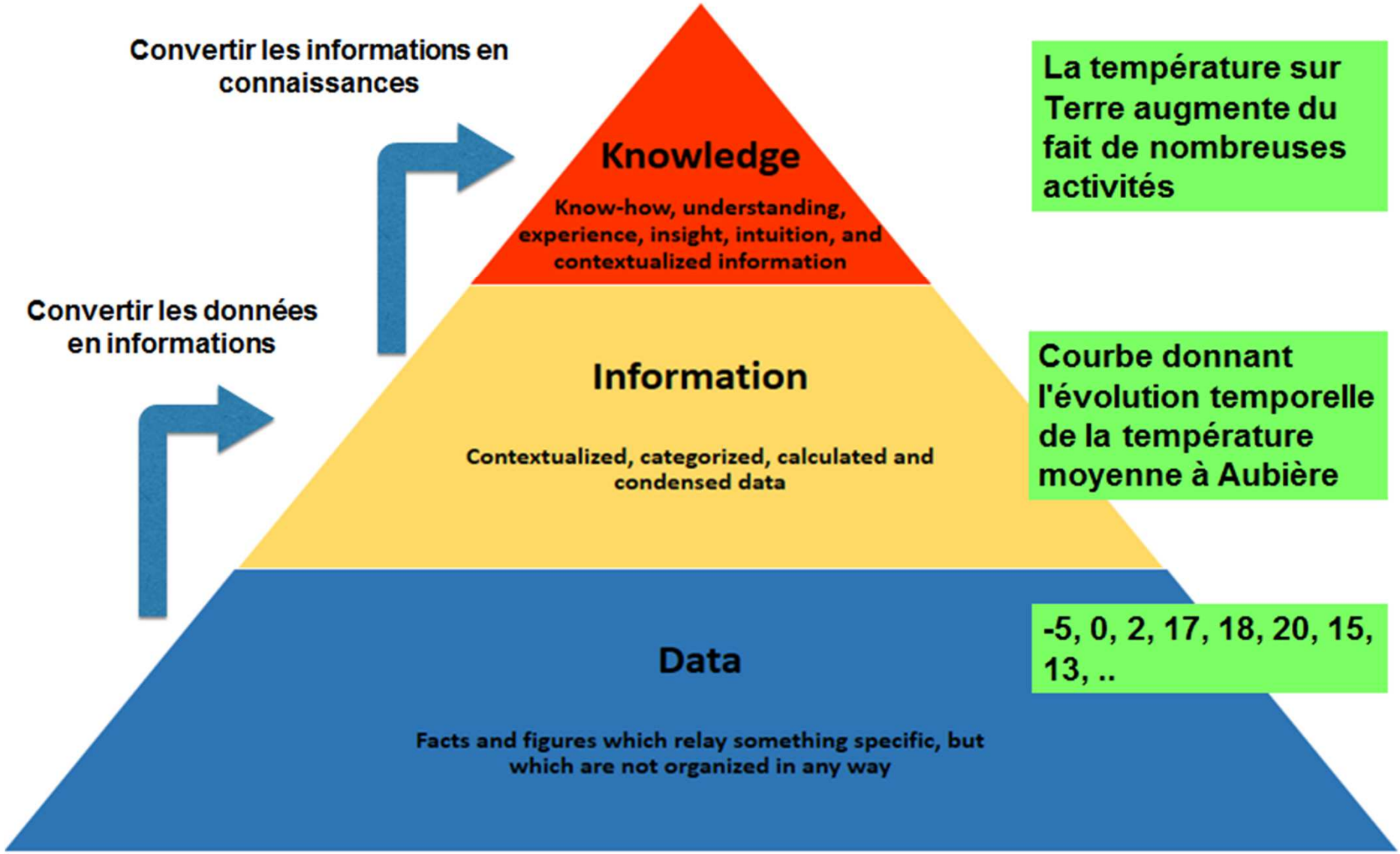

http://www.isima.fr/~lacomme/NoSQL/

# 4 enjeux



- Stocker (Comment ?)
- Extraire (fouille de données)
- Restituer
- Exploiter les données (nouvelle valorisation)

# L’aspect connaissances

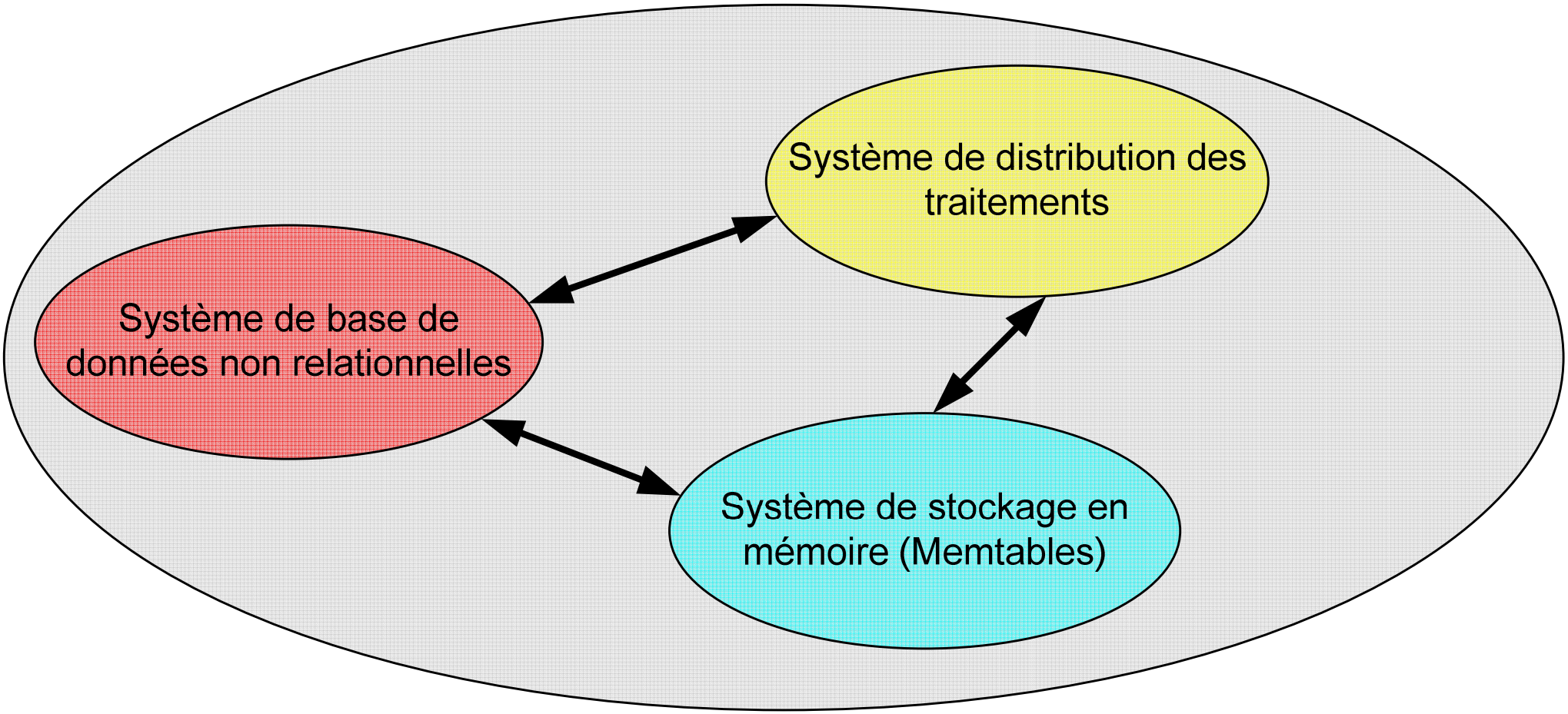



http://www.knowledge-management-tools.net/knowledge-information-data.html

# Trois éléments

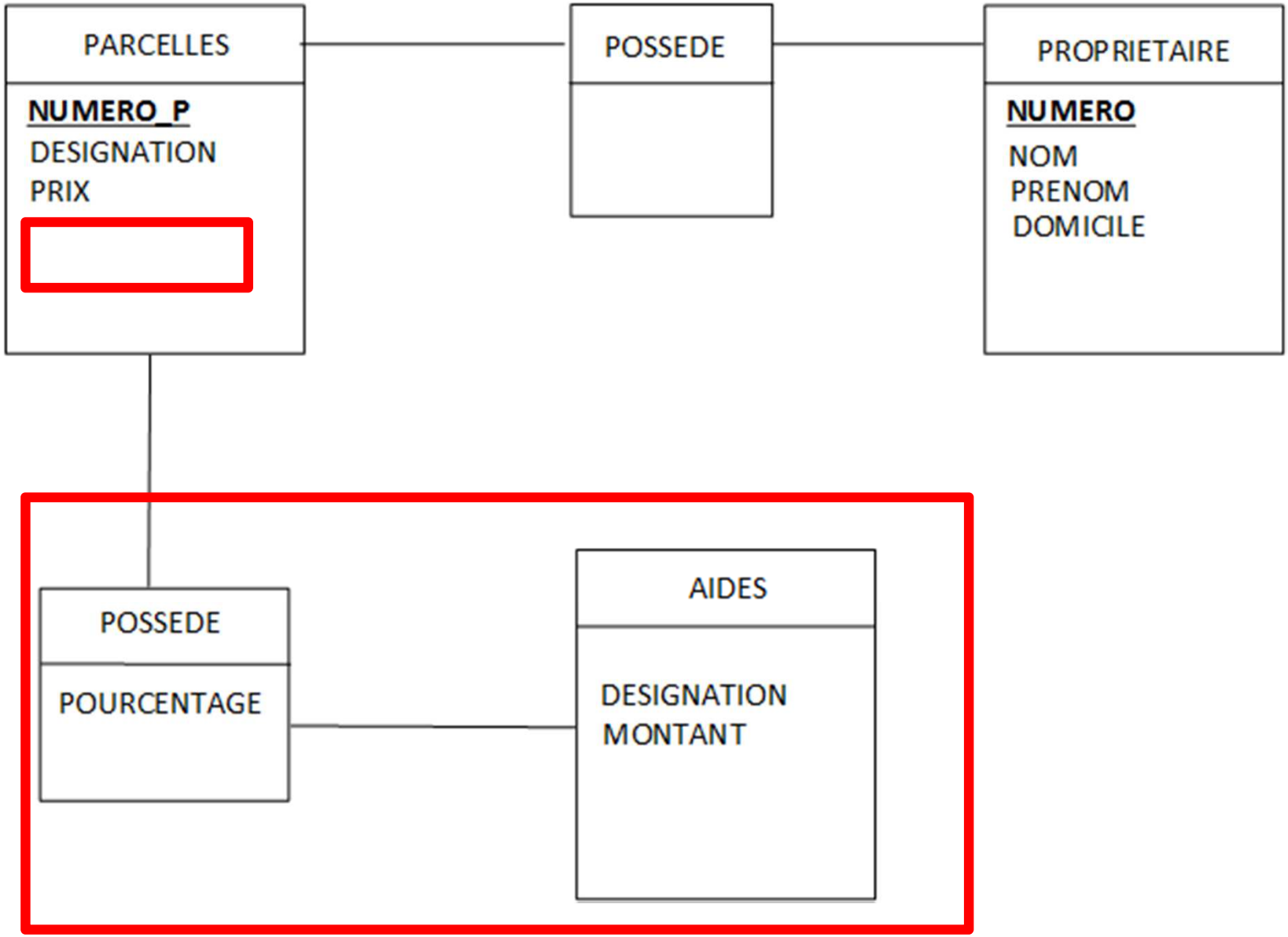
- bases de données non relationnelles, appelées à tort NoSQL ;
- un système de distribution des traitements ;
- un système de stockages de données en mémoire pour accélérer les traitements.

# Trois éléments

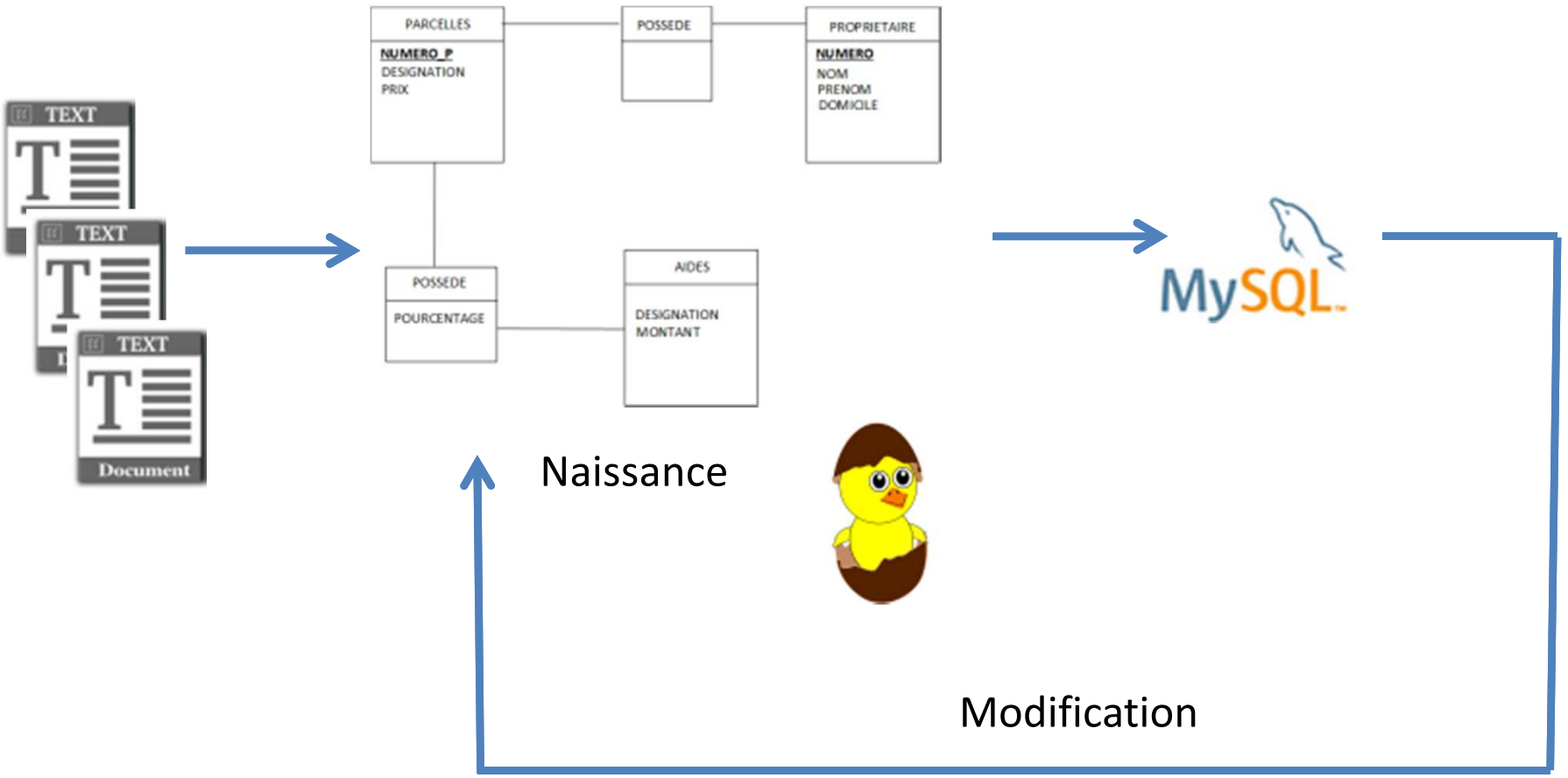


# Autrefois….
# Dans notre vieux temps

# Problème 2


PARCELLES
NUMERO_P
DESIGNATION
PRIX
POSSEDE
PROPRIETAIRE
NUMERO
NOM
PRENOM
DOMICILE
POSSEDE
POURCENTAGE
AIDES
DESIGNATION
MONTANT

# La vraie vie...


TEXT
TEXT
TEXT
Document
PARCELLES
NUMERO_P
DESIGNATION
PRIX
POSSEDE
PROPRIETAIRE
NUMERO
NOM
PRENOM
DOMICILE
POSSEDE
POURCENTAGE
AIDES
DESIGNATION
MONTANT
MySQL
Naissance
Modification

# Bilan

- MCD/MLD → base relationnelle

- Adaptée au cas où les données ne changent que rarement de structure

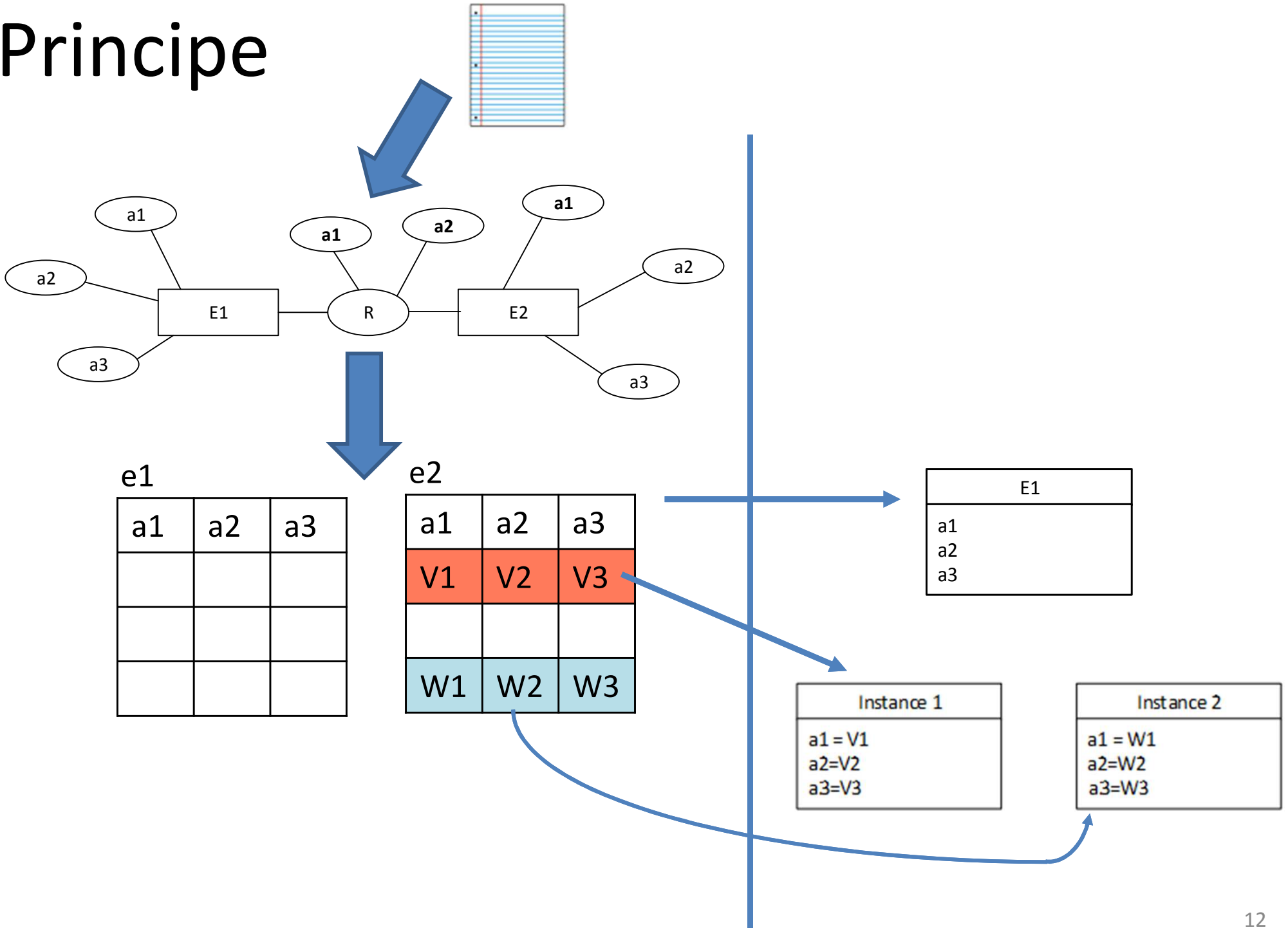
# Une solution : base NoSQL

- **Clé-valeur :** il s'agit de la représentation la plus simple. Cette structure est très adaptée à la gestion de caches ou pour fournir un accès rapide aux informations. Elle fonctionne comme un grand tableau associatif et retourne une valeur dont elle ne connaît pas la structure.
- **Clé-valeur avec contraintes d'intégrités :** une représentation évoluée qui autorise en plus la définition de contraintes d'intégrité.
- **Document :** une représentation qui ajoute au modèle **clé-valeur**, l'association d'une valeur à structure non plane, c'est-à-dire qui nécessiterait un ensemble de jointures en logique relationnelle.
- **Colonne :** une autre évolution du modèle **clé-valeur** qui permet de disposer d'un très grand nombre de valeurs sur une même ligne, permettant ainsi de stocker les relations de type **one-to-many**. Contrairement au système **clé-valeur**, celui-ci permet d'effectuer des requêtes par clé.
- **Graphe :** une représentation qui permet la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables.

# Existant

| Type | Nom | Langage | Date création | Langage d'interrogation natif | Web Services | Bibliothèque (API) |
|---|---|---|---|---|---|---|
| Colonne | Hbase | Java | 2007 | Non | / | Java |
| | Hypertable | C++ | 2007 | HQL | / | / |
| | Cloudata | Java | 2011 | CQL | REST | JAVA |
| Clé-Valeur | Membase | C et C++ | 2009 | / | / | / |
| | Kyoto | C++ | / | / | / | C, C++, Java, C#, Python, Ruby, Perl… |
| | Redis | C | 2009 | / | / | C, C++, Java, Python, Ruby, … |
| | Oracle | / | 2012 | / | / | Java, C |
| Clé-Valeur/CI | Cassandra | Java | 2008 | / | / | Java, Python, PHP, Ruby… |
| | Voldemort | Java | 2008 | / | / | / |
| | Riak | Erlang, C, Javascript | 2008 | / | / | Java, Ruby, Python, PHP, Javascript |
| Document | CouchDB | C, Javascript | 2005 | / | REST | / |
| | MongoDB | C++ | / | OUI | / | C, C++, Java, C#, Ruby, Javascript... |
| Graphe | Neo4j | Java | 2003 | SPARQL | REST | Java, Python, Ruby, PHP |
| | FlockDB | Scala | 2010 | / | / | / |

# Principe


a1
a2
a3
E1
a1
a2
R
a1
a2
a3
E2
e1
a1
a2
a3
e2
a1
a2
a3
V1
V2
V3
W1
W2
W3
E1
a1
a2
a3
Instance 1
a1 = V1
a2=V2
a3=V3
Instance 2
a1 = W1
a2=W2
a3=W3

# Principe

mongoDB

| Instance 1 |
| --- |
| a1=V1<br>a2=V2<br>a3=V3 |

| Instance 2 |
| --- |
| a1=W1<br>a2=W2<br>a3=W3<br>a4=W4 |

| Instance 3 |
| --- |
| a2=t2<br>a3=t3<br>a4=t4<br>a5=t5 |

| OBJECT |
| --- |
| |

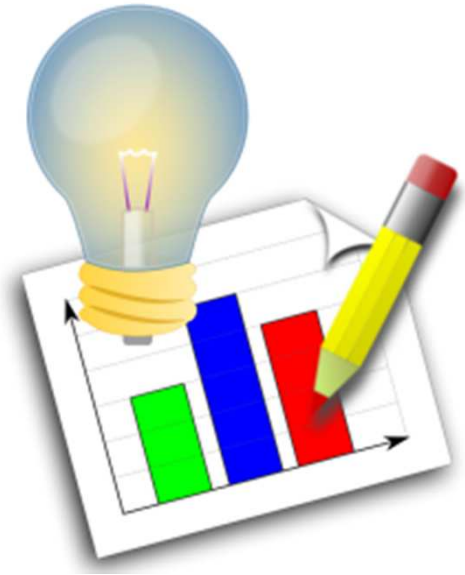
| Instance 1 |
| --- |
| a1=V1<br>a2=V2<br>a3=V3 |

```
dbObject = new BasicDBObject();
dbObject.put("Numero", "3547");
dbObject.put("Nom", "Chambord");
dbObject.put("Prenom", "Emilie");
coll.insert(dbObject);
```

| Instance 2 |
| --- |
| a1=W1<br>a2=W2<br>a3=W3<br>a4=W4 |

```
dbObject = new BasicDBObject();
dbObject.put("Numero", "85478");
dbObject.put("Nom", "Castafiore");
dbObject.put("Prenom", "Emilie");
dbObject.put("Domicile", "Paris");
coll.insert(dbObject);
```

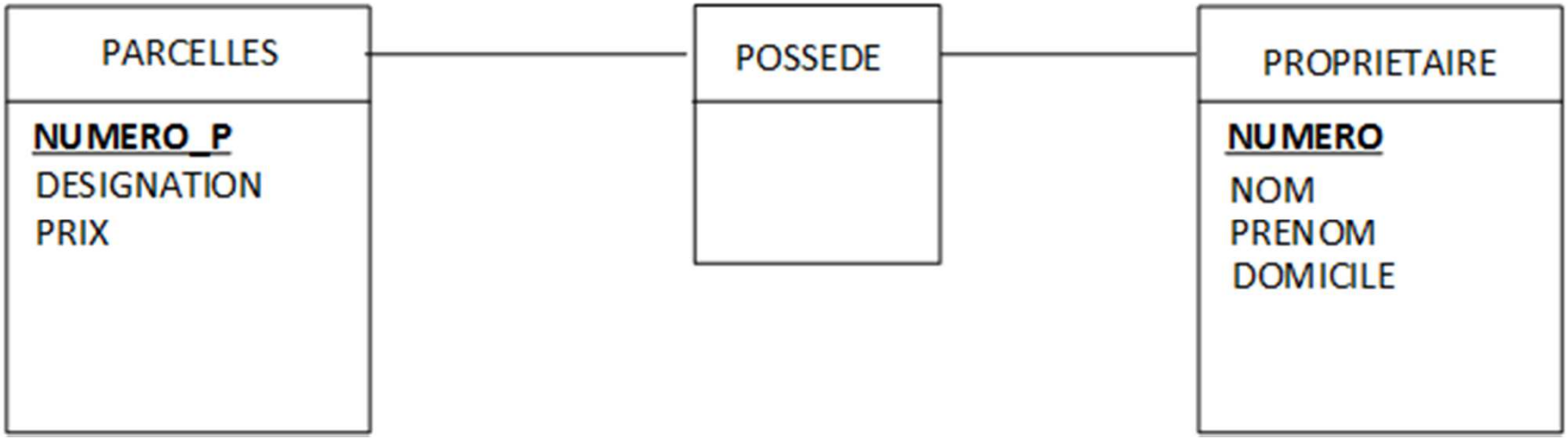
# MongoDB



# Connexion


```
try
{
  MongoClient mongoClient = new MongoClient();
  DB db = mongoClient.getDB("bibliotheque");

  // Code


}
catch(Exception E)
{
   System.out.println(E.getMessage());
}
```

# Le modèle classique




Parcelles

| Numero | Designation | Prix/ha | Proprietaire |
|---|---|---|---|
| 10101 | Terrain à bâtir | 500 | Emilie Castafiore |
| 11111 | Friches | 54 | Emilie Chambord |
| 80808 | Landes | 10 | Emilie Castafiore |
| 90909 | Culture | 25 | Roland Momo |
| 202022 | Elevage | 30 | Sylvie Fabière |

Propriétaires

| Nom | Prénom | Domicile | Numero |
|---|---|---|---|
| Castafiore | Emilie | Paris | 85478 |
| Chambord | Emilie | Nice | 3547 |
| Dupont | Pierre | Avignon | 542563 |
| Fabiere | Sylvie | Bordeaux | 52136 |
| Momo | Roland | Toulouse | 8547585 |
| Tintin | Thierry | Clermont | 78545 |

# Insertion d'information


```
MongoClient mongoClient = new MongoClient();
DB db = mongoClient.getDB( "bibliotheque" );

DBCollection coll = db.getCollection("Auteur");

// insertion
BasicDBObject dbObject = new BasicDBObject();
dbObject.put("Numero", "85478");
dbObject.put("Nom", "Castafiore");
dbObject.put("Prenom", "Emilie");
dbObject.put("Domicile", "Paris");
coll.insert(dbObject);

dbObject = new BasicDBObject();
dbObject.put("Numero", "3547");
dbObject.put("Nom", "Cha
dbObject.put("Prenom", "E
dbObject.put("Domicile", "
coll.insert(dbObject);
```

```
DBObject myDoc = coll.findOne();
System.out.println(myDoc);

// Affichage
System.out.println("affichage de tous les elements");
DBCursor cursor = coll.find();

while(cursor.hasNext())
{
  System.out.println(cursor.next());
}
```

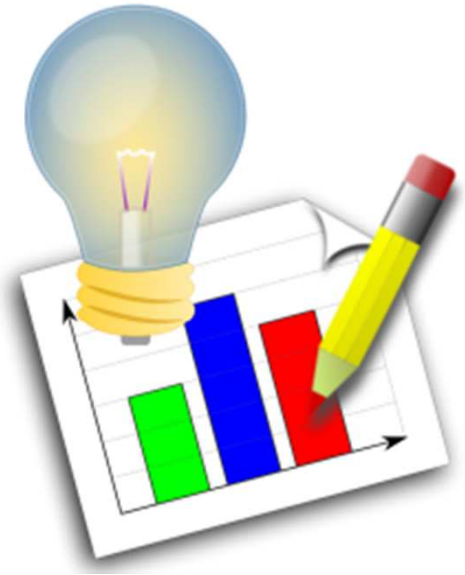
```
affichage element 1
{ "_id" : { "$oid" : "52d7aec259b83266723cc082"} , "Numero" : "85478" , "Nom" : "Castafiore" , "Prenom" : "Emilie" , "Domicile" : "Paris"}
affichage de tous les elements
{ "_id" : { "$oid" : "52d7aec259b83266723cc082"} , "Numero" : "85478" , "Nom" : "Castafiore" , "Prenom" : "Emilie" , "Domicile" : "Paris"}
{ "_id" : { "$oid" : "52d7aec259b83266723cc083"} , "Numero" : "3547" , "Nom" : "Chambord" , "Prenom" : "Emilie" , "Domicile" : "Nice"}
"_id" : {
"_id" : {
"_id" : {
"_id" : {
"v" : 1 ,
```

```
"Numero" : "85478" , "Nom" : "Castafiore" , "Prenom" : "Emilie"

"Numero" : "85478" , "Nom" : "Castafiore" , "Prenom" : "Emilie"
"Numero" : "3547" , "Nom" : "Chambord" , "Prenom" : "Emilie" ,
"Numero" : "542563" , "Nom" : "Dupont" , "Prenom" : "Pierre" ,
"Numero" : "52136" , "Nom" : "Fabière" , "Prenom" : "Sylvie" ,
"Numero" : "8547585" , "Nom" : "Momo" , "Prenom" : "Roland" , "
"Numero" : "78545" , "Nom" : "Tintin" , "Prenom" : "Thierry" ,
```

# Liens table/collection


| NUMERO_L | Titre | Prix |
|---|---|---|
| 10101 | aaaaa | 10 |
| 11111 | ee | 54 |
| 80808 | cccc | 45 |
| 90909 | ddddd | 35 |
| 202022 | bb | 25 |

```
DBCollection coll_livre = db.getCollection("Parcelles");

BasicDBObject dbObject_Livre = new BasicDBObject();
dbObject_Livre.put("Numero", "10101");
dbObject_Livre.put("Titre", "aaaaa");
dbObject_Livre.put("Prix", 10);
coll_livre.insert(dbObject_Livre);
```



# ORACLE NoSQL

# Ajouter

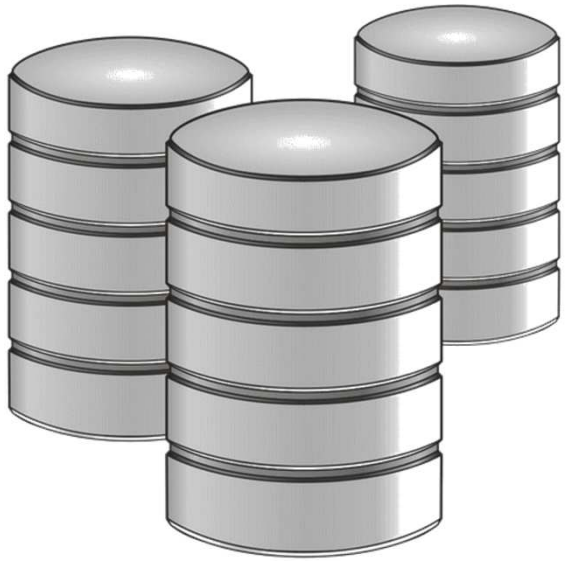

| Nom | Numéro |
|---|---|
| Castafiore | 85478 |
| Chambord | 3547 |
| Dupont | 542563 |
| Fabière | 52136 |
| Momo | 8547585 |
| Tintin | 78545 |

```
String keyString1 = "85478";
String valueString1 = "Castafiore";

Key une_cle1 = Key.createKey(keyString1);
Value une_valeur1 = Value.createValue(valueString1.getBytes());
store.put(une_cle1, une_valeur1);
```

# Ajouter


| Nom | Prénom | Domicile | Numero |
|---|---|---|---|
| Castafiore | Emilie | Paris | 85478 |
| Chambord | Emilie | Nice | 3547 |
| Dupont | Pierre | Avignon | 542563 |
| Fabiere | Sylvie | Bordeaux | 52136 |
| Momo | Roland | Toulouse | 8547585 |
| Tintin | Thierry | Clermont | 78545 |

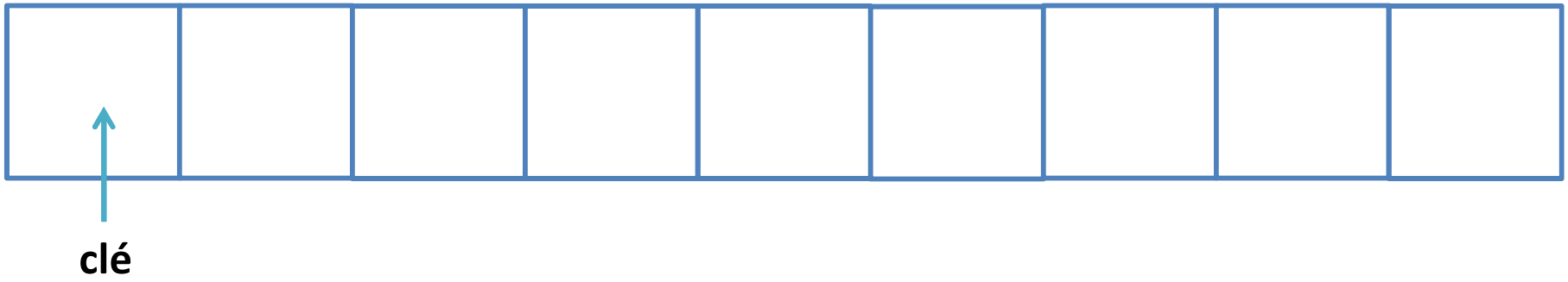
```
t_proprietaire un_proprietaire = new t_proprietaire ();
un_proprietaire.nom = "Dupont";
un_proprietaire.prenom = "Pierre";
un_proprietaire.domicile = "Paris";

tableau_des_cles_de_recherche.clear();
tableau_des_cles_de_recherche.add("Dupont");
tableau_des_cles_de_recherche.add("Pierre");

Key myKey = Key.createKey(tableau_des_cles_de_recherche);
Value myValue = Value.createValue(un_auteur.toString().getBytes());
store.put(myKey, myValue);
```



# Principes communs

# Principe : accès direct

T : Tableaux associatifs




T[i] ➔ accès en O(1)

# Magique ou non ?

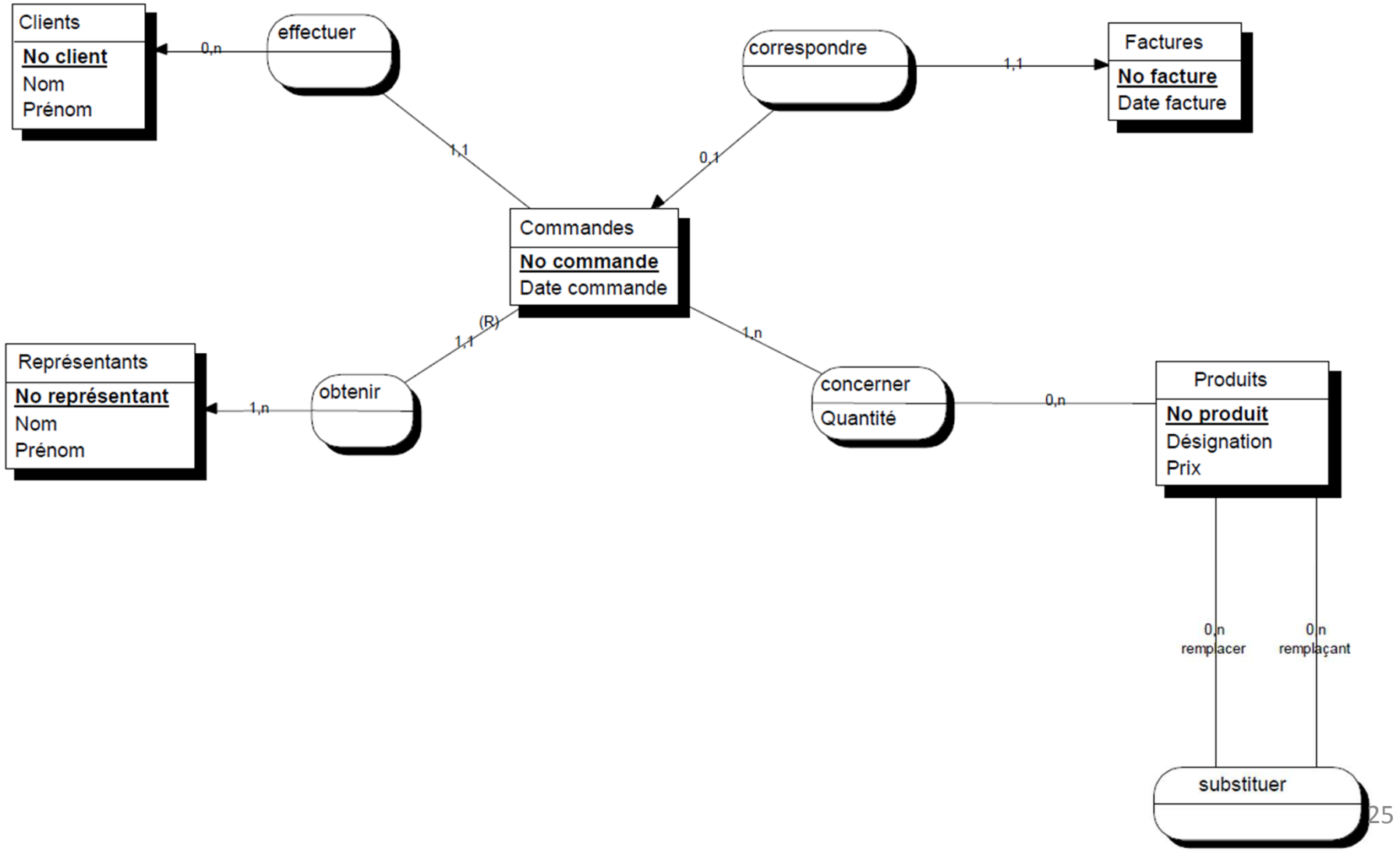
ISIMA

**TP de Big Data**

**Michaël PLAN & Ana Alice BARROS NETO**

Comparaison des performances de bases relationnelles et non relationnelles

# Un énoncé « classique »



# Magique ou non ?

| Action | Nombre d'objets | MySQL | mongoDB |
|---|---|---|---|
| CREATE | 100 | 5,54 | 1,172 |
| | 10000 | 470,931 | 1,092 |
| | 1000000 | 12641,552 | 57,096 |
| SELECT | 100 | 0,014 | 0 |
| | 10000 | 0,078 | 0,063 |
| | 1000000 | 3,759 | 4,244 |

# Magique ou non ?

http://www.datastax.com/wp-content/themes/datastax-2014-08/files/NoSQL_Benchmarks_EndPoint.pdf?2

END POINT

## Benchmarking Top NoSQL Databases

Apache Cassandra, Couchbase, HBase, and MongoDB

Originally Published: April 13, 2015

Revised: May 27, 2015

http://www.endpoint.com/

# Magique ou non ?

http://www.datastax.com/wp-content/themes/datastax-2014-08/files/NoSQL_Benchmarks_EndPoint.pdf?2

END POINT

## Benchmarking Top NoSQL Databases

Apache Cassandra, Couchbase, HBase, and MongoDB

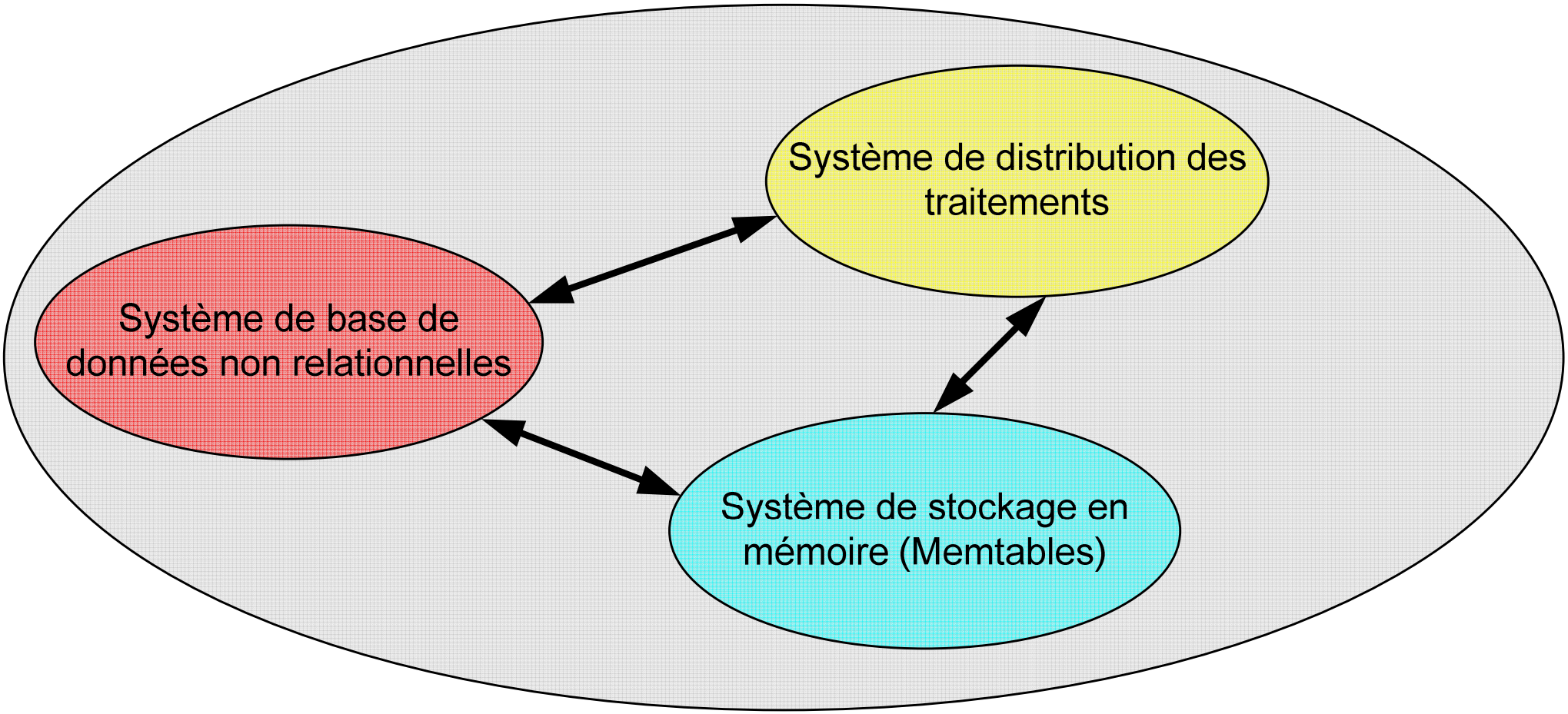
Originally Published: April 13, 2015

Revised: May 27, 2015

http://www.endpoint.com/

# CALCULS DISTRIBUES
# Hadoop and co…..

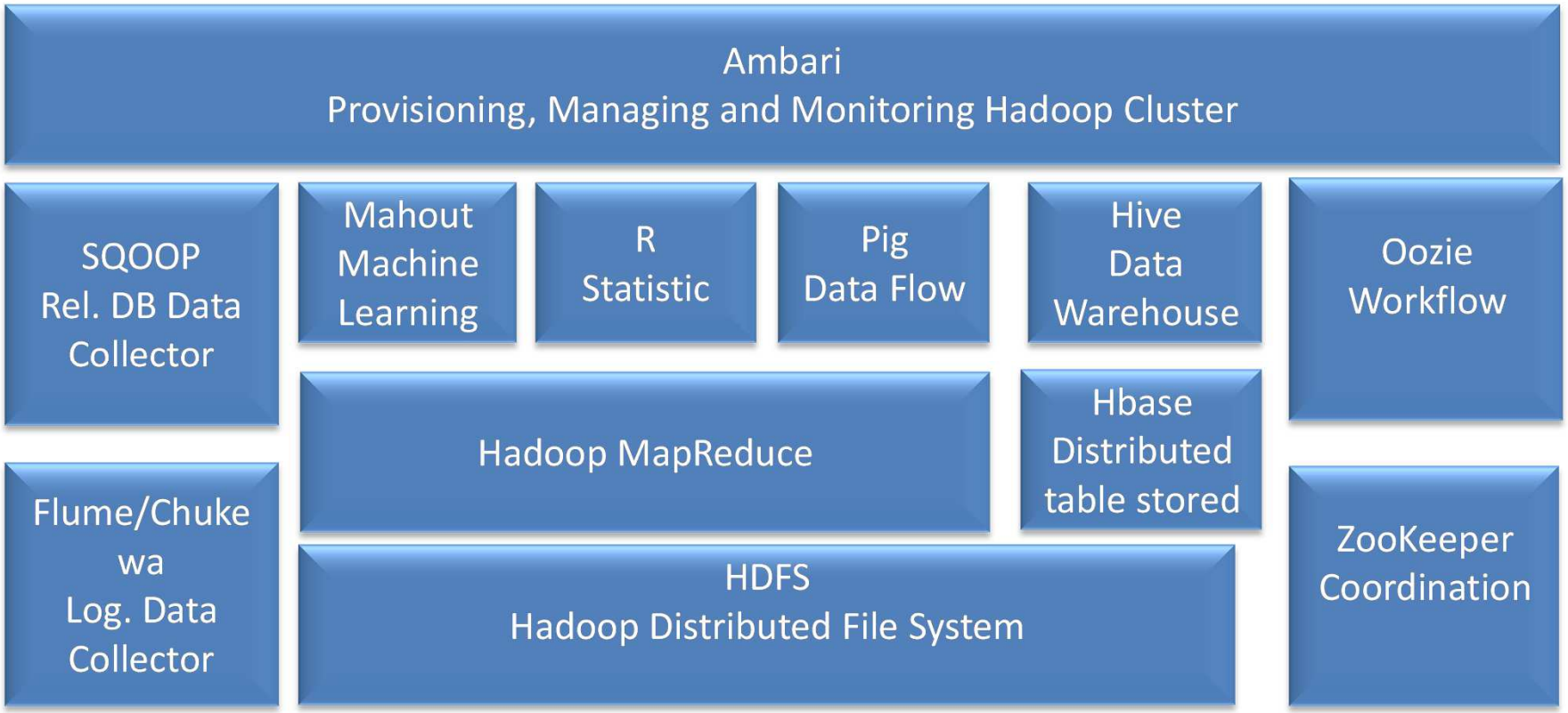
# Trois éléments



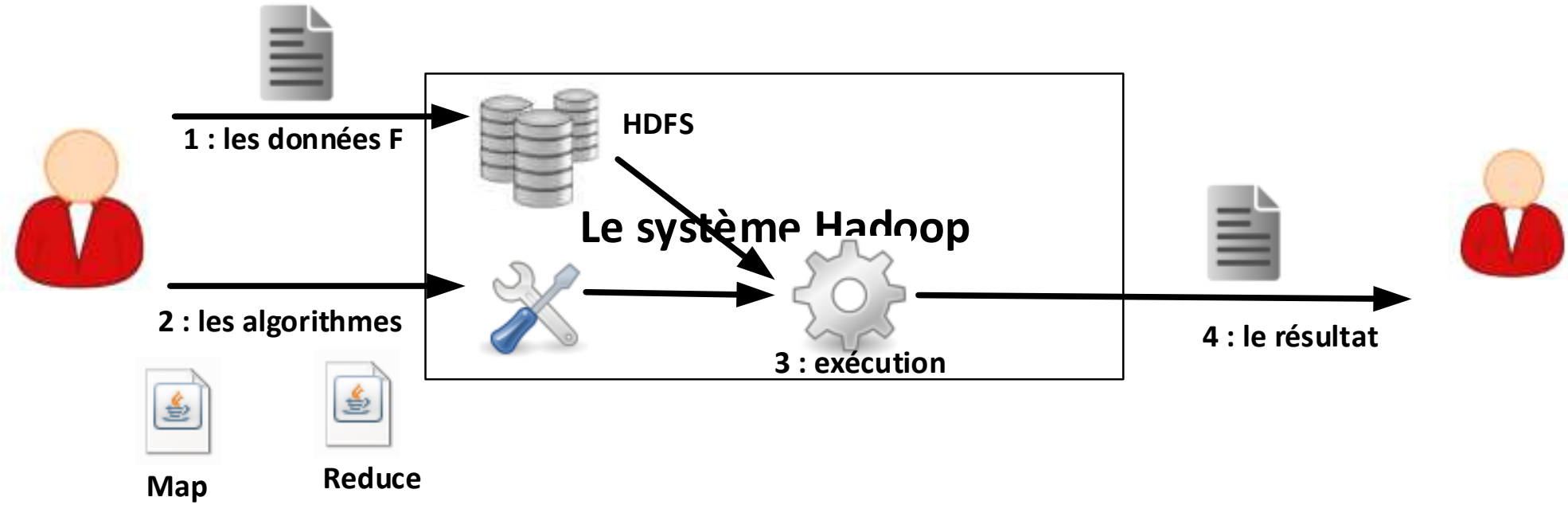

hadoop


APACHE
STORM™

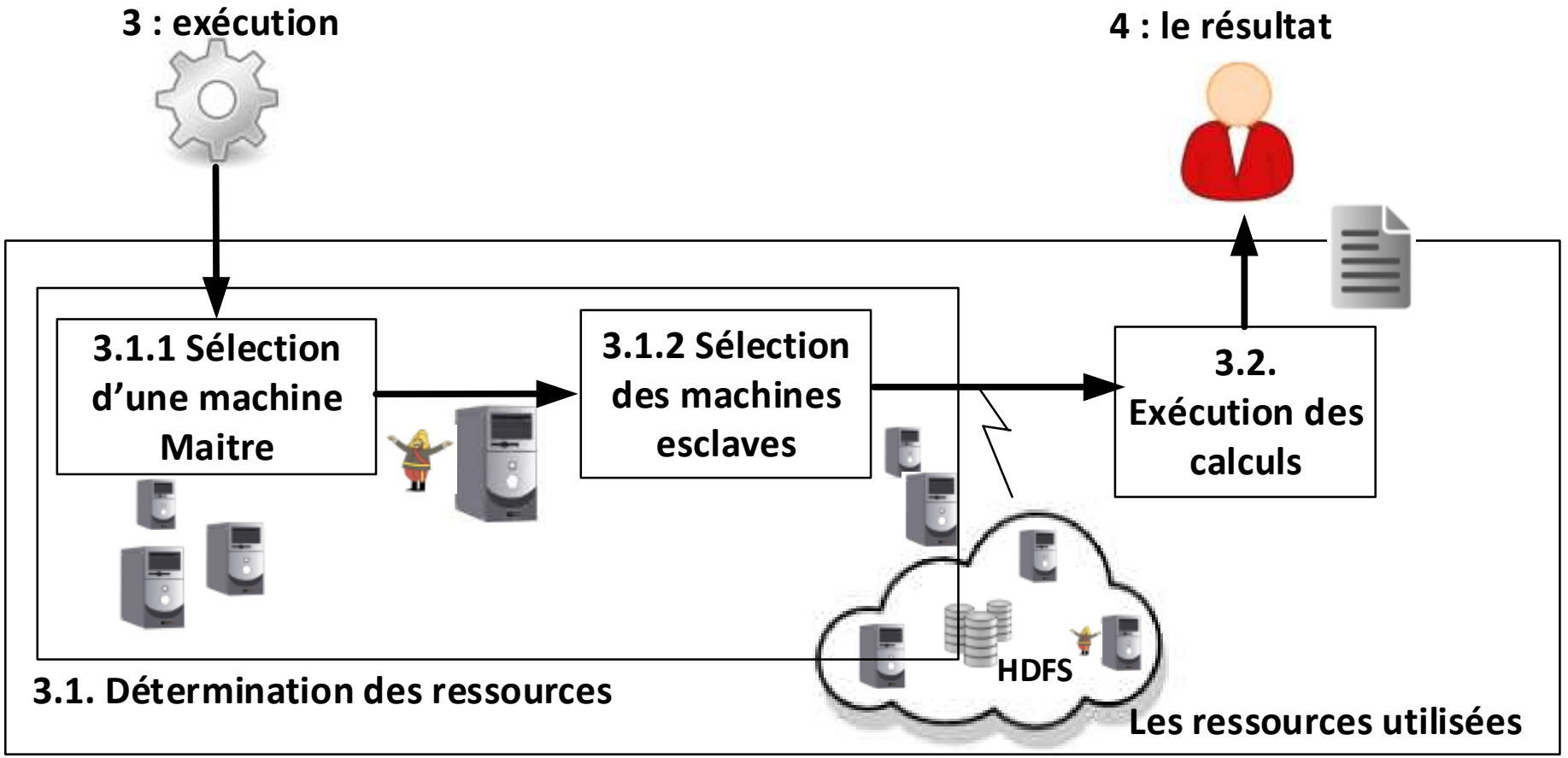
# Ecosystème Hadoop



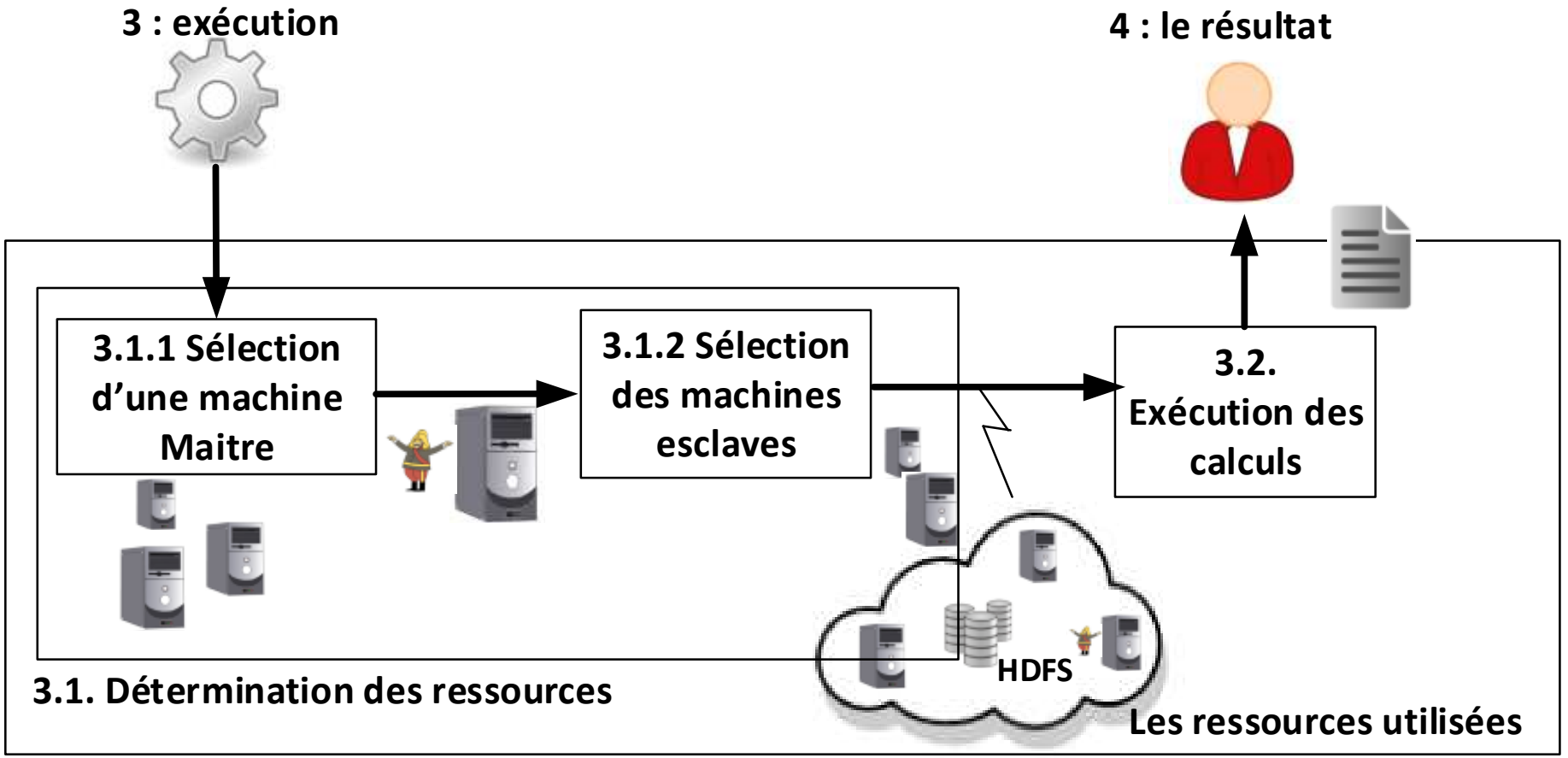
# Utilisation



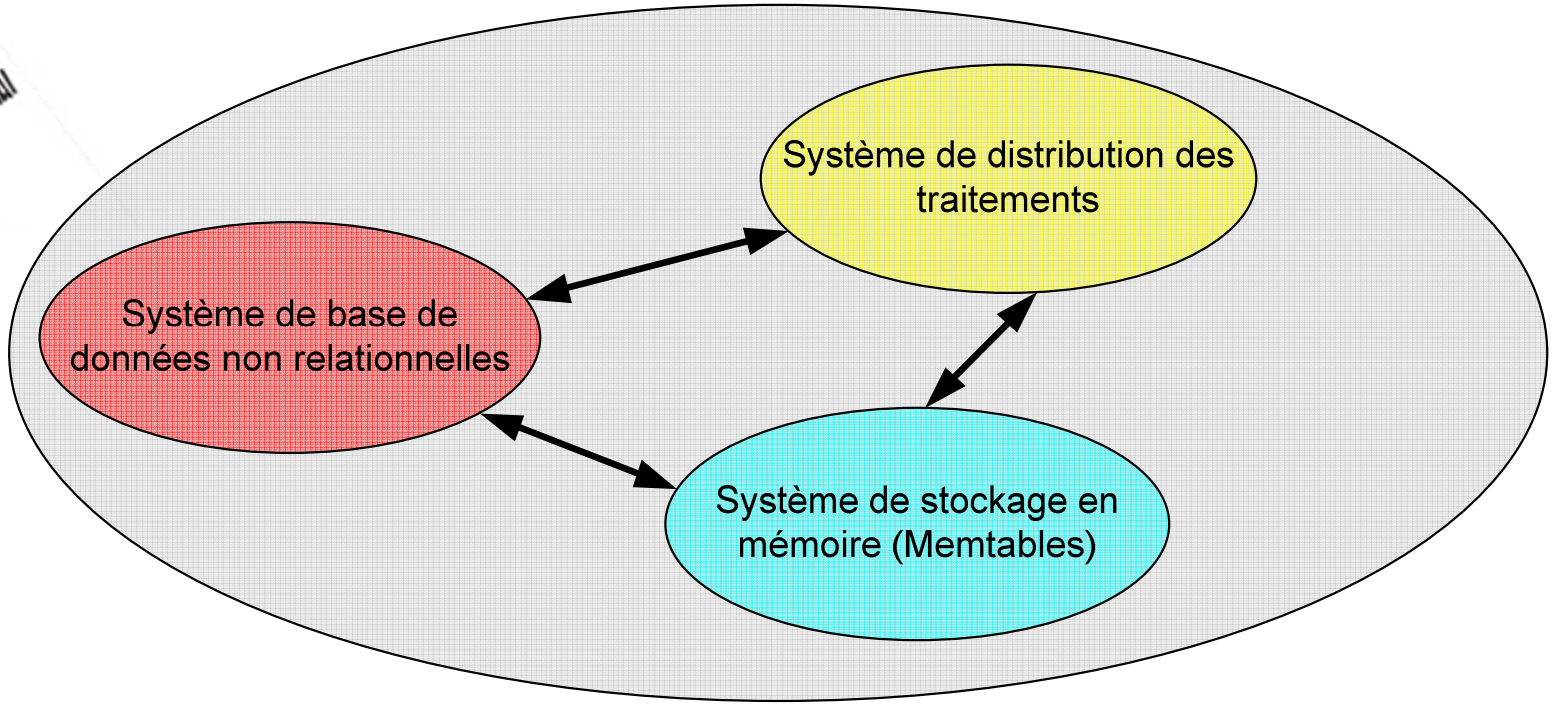
# Etape 3


3 : exécution
4 : le résultat
3.1.1 Sélection d'une machine Maitre
3.1.2 Sélection des machines esclaves
3.2. Exécution des calculs
HDFS
3.1. Détermination des ressources
Les ressources utilisées

# Etape 3


3 : exécution
4 : le résultat
3.1.1 Sélection d'une machine Maitre
3.1.2 Sélection des machines esclaves
3.2. Exécution des calculs
HDFS
3.1. Détermination des ressources
Les ressources utilisées

# Conclusion

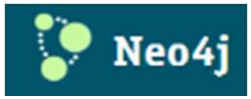

Système de distribution des traitements
Système de base de données non relationnelles
Système de stockage en mémoire (Memtables)

# Formation

**2-3 Juin 2016**


Installation, configuration, utilisation

**13-14-15 Juin 2016**




Contact : Loic Yon (resp. Formation Continue)
Tel. 04 73 40 50 42
Email : loic.yon@isima.fr