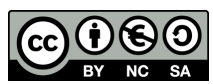
BDBIO - SGBD orientés document (MongoDB)

Fabien Duchateau

fabien.duchateau [at] univ-lyon1.fr

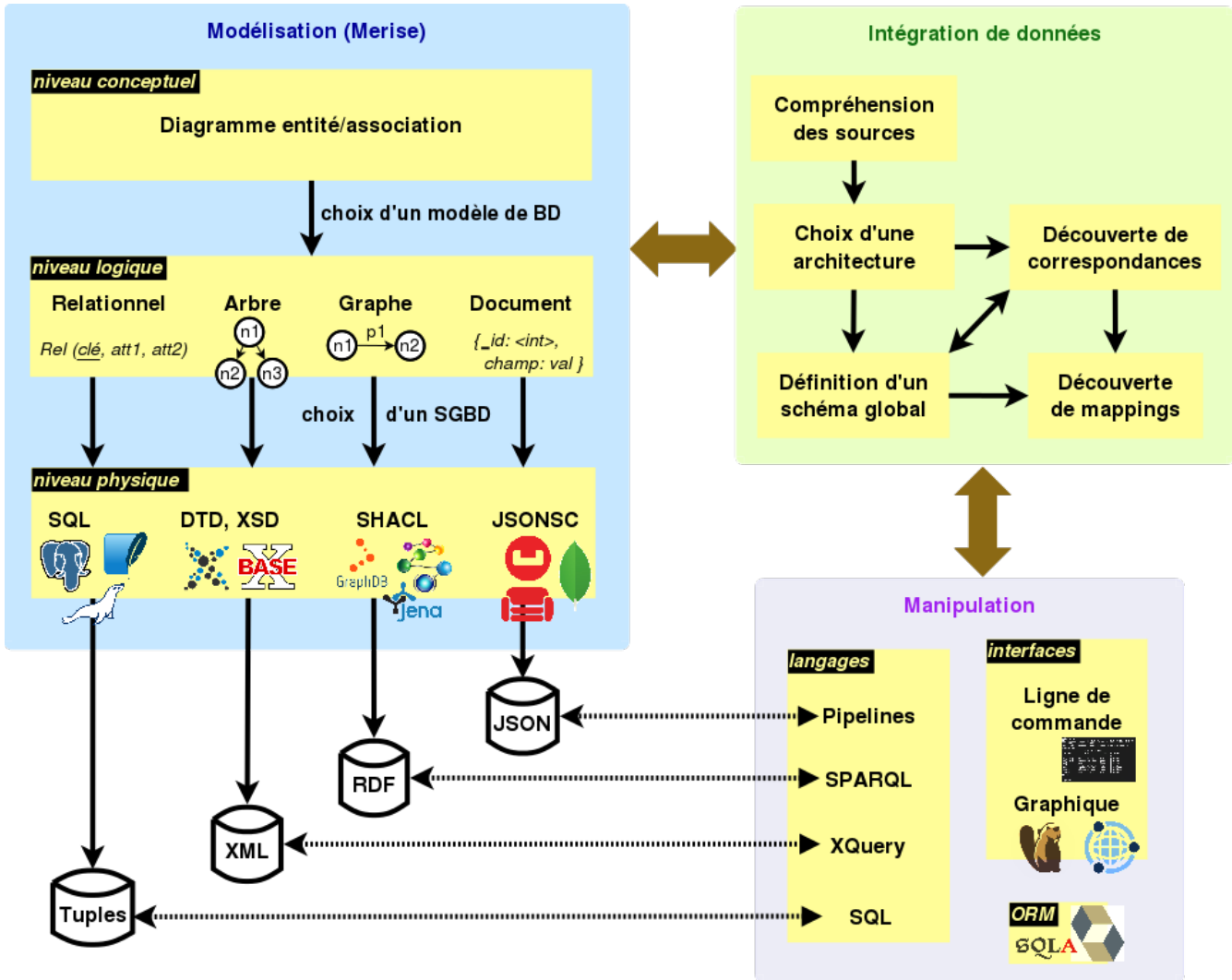
Université Claude Bernard Lyon 1

2024 - 2025



https://perso.liris.cnrs.fr/fabien.duchateau/BDBIO/

Aperçu des notions de BDBIO



Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

Rappels

Les données d'application sont principalement gérées par des systèmes de gestion de bases de données (SGBD), qui suivent un modèle de données

Modèle	**Sérialisation**	**LDD**	**LMD**	**Exemples SGBD**
Relationnel	n-uplets	SQL	SQL	SQLite, MariaDB, PostgreSQL
Arbre	XML	DTD, XML Schema	XPath, XQuery	XBase, existDB

Comment apporter davantage de flexibilité au niveau du schéma, avec un format peu contraignant ?

BD orientée documents

BD orientée documents = collection de documents (clé-document)

- Notion abstraite de "document"
  - structure arborescente sous forme d'une liste de champ/valeur
- Applications : gestion de contenu (bibliothèque numérique), événements, catalogues, analytique temps réel, etc.
- Mais pas de standards pour les langages LDD et LMD

Exemples de SGBD : **MongoDB**, CouchBase, CouchDB, ...

---

https://en.wikipedia.org/wiki/Document-oriented_database
https://www.mongodb.org/
https://www.couchbase.com/
https://couchdb.apache.org/

Plan

Concepts de MongoDB

Modélisation en MongoDB

Manipulation avec MongoDB

Caractéristiques de MongoDB

- SGBD orienté documents
- Open-source
- Populaire (5ème SGBD le plus utilisé)
- Passage à l'échelle horizontal (sharding) et réplication
- Système CP (cohérent et résistant au morcellement)
  - "strong consistency" (lectures sur serveur primaire)
  - "eventual consistency" (lectures sur différents serveurs)
- Traitements distribués (Map Reduce, *aggregation pipeline*)
- GUI (Compass, Robot 3T) et nombreux drivers

---

http://www.mongodb.com/
http://robomongo.org/download

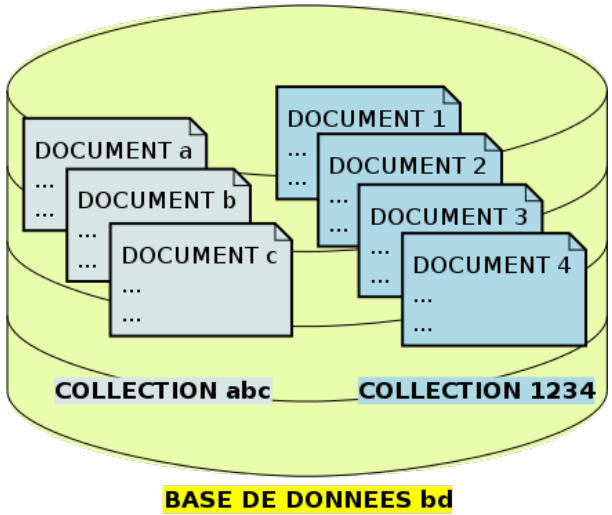
Concepts principaux - BD et collection

Base de données (∼ *base de données* en modèle relationnel) :

- Ensemble de collections
- Espace de stockage

Collection (∼ *table* en modèle relationnel) :

- Ensemble de documents qui partagent un objectif ou des similarités
- Pas de schéma prédéfini (mais possibilité de valider un schéma)



http://docs.mongodb.com/manual/core/databases-and-collections/

https://www.mongodb.com/docs/manual/core/schema-validation/

Concepts principaux - document

Document ($\sim$ *ligne, tuple* en modèle relationnel) :

- Un enregistrement dans une collection
- Syntaxe et stockage au format BSON
- Identifiant d'un document (clé **"_id"**)

BSON = "Binary JSON" (JavaScript Object) avec améliorations :

- Ensemble de paires clé/valeur
- Une valeur peut être un objet complexe (liste, document, ensemble de valeurs, etc.)
- Représentation de nouveaux types (e.g., dates)
- Facilité de parsing (e.g., entiers stockés sur 32/64 bits)

---

http://docs.mongodb.com/manual/core/document/
http://bsonspec.org/

Concepts principaux - document (2)

Syntaxe d'un document en MongoDB (format BSON) :

- **_id** est un identifiant (généré ou manuel)
- **att-1** est une clé dont la valeur est une chaîne de caractères
- **att-2** est une clé dont la valeur est un entier
- **att-3** est une clé dont la valeur est une liste de valeurs
- **att-k** est une clé dont la valeur est un document inclus

```
{
_id : <identifiant>,
"att-1" : "val-1",
"att-2" : val-2,
"att-3" : ["val-31", "val-32", ...]
...
"att-k" : {
            "att-k1" : "val-k1",
            ...
          }
}
```

Jeu de données

```
{ _id: "Ana",
  annee: 2020,
  groupes: ["A", "A1"],
  notes: [{ue: "BD", note: 17},
          {ue: "WEB", note: 18}]
}
{ _id: "Bob",
  annee: 2022,
  groupes: ["A", "A2"],
  notes: [{ue: "BD", note: 19},
          {ue: "WEB", note: 14}]
}
{ _id: "Cya",
  annee: 2021,
  groupes: ["A", "A1"],
  notes: [{ue: "BD", note: 14}]
}
{ _id: "Dan",
  annee: 2020,
  groupes: ["B", "B1"],
  notes: [{ue: "BD", note: 9},
          {ue: "WEB", note: 16}]
}
```

Ana

annee : 2020
groupes : [A, A1]
notes : [
{ue : 'BD', note : 17},
{ue : 'WEB', note : 18}]

Bob

annee : 2022
groupes : [A, A2]
notes : [
{{ue : 'BD', note : 19},
{ue : 'WEB', note : 14}]

Cya

annee : 2021
groupes : [A, A1]
notes : [
{{ue : 'BD', note : 14}]

Dan

annee : 2020
groupes : [B, B1]
notes : [
{{ue : 'BD', note : 9},
{ue : 'WEB', note : 16}]

*Collection **etu** contenant 4 documents représentant chacun un·e étudiant·e, son année d'inscription, ses groupes et ses notes*

https://pipoya.itch.io/pipoya-free-rpg-character-sprites-32x32

Plan

Modélisation d'une BD

Solutions diverses pour modélisation logique (dont typage)

Modélisation guidée par les besoins applicatifs (types de requêtes, ratio lecture/écriture, croissance du nombre de documents, etc.) :

- Accès par une clé (identifiant d'un document)
- Schéma optionnel ⇒ ajout d'un champ à tout moment
- Pas de jointure ⇒ redondances possibles
- Inclusion ⇒ moins de lectures
- Opérations atomiques sur un seul document, mais pas sur plusieurs ⇒ état incohérent temporaire (souvent tolérable)

Pas de niveau physique (schéma flexible) mais un standard json-schema défini
Vera et al. Data modeling for NoSQL document-oriented databases, CEUR (2015)
Atzeni et al. Data modeling in the NoSQL world, CSI (2020)
http://docs.mongodb.com/manual/core/data-model-design/

Relations inter-documents

Relation entre les documents de différentes collections :

- Par référence : l'identifiant d'un document (son "_id") est utilisé comme valeur attributaire dans un autre document
  - se rapproche du modèle de données normalisées
  - nécessite des requêtes supplémentaires côté applicatif
  - exemple : entités fortement connectées (livres/auteurs)
- Par inclusion ("embedded") : un "sous-document" est utilisé comme valeur
  - philosophie "non-relationnelle" (pas de jointure)
  - meilleures performances en lecture
  - redondance possible
  - exemple : entités fréquemment lues ensemble (article/commentaires)

---

http://docs.mongodb.com/manual/core/data-modeling-introduction/

Exemple de modélisation - 2 collections

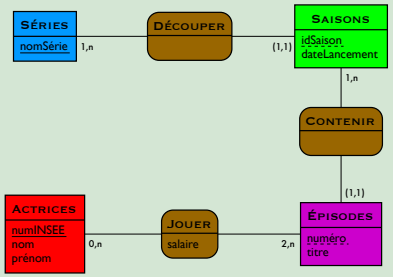


Diagramme E/A (niveau conceptuel)

```
[{# collection Actrices
  _id: <string>,
  nom: <string>,
  prénom: <string>,
}]
```

```
[{# collection Series
  _id: <string>,
  nomSérie: <string>,
  saisons: [{ # inclusion saison
    _id: <number>,
    dateLancement: <string>,
    episodes: [ # inclusion épisode
      {
        _id: <string>,
        titre: <string>,
        roles: [
          { # référence actrice
            _id: <string>,
            salaire: <number>
          },
          ...
        ]
      },
      ...
    ]},
    ...
  ]
}]
```

Typage des 2 collections (niveau logique) : liens série-saison et saison-épisode par inclusion, liens épisode-actrice par référence

Exemple de modélisation - 3 collections

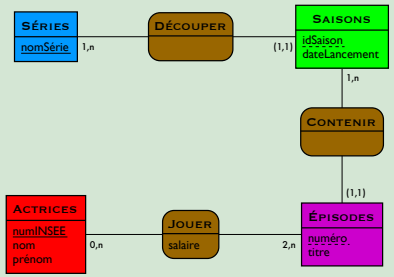


Diagramme E/A (niveau conceptuel)

```
[{ # collection Saisons
  _id: <number>,
  dateLancement: <string>,
  episodes: [ # inclusion épisodes
    {
      _id: <string>,
      titre: <string>,
    },
    ...
  ]}
}]
```

```
[{# collection Series
  _id: <string>,
  nomSérie: <string>,
  saisons: [_id, _id, ...] #
        référence saison
}]
```

```
[{# collection Actrices
  _id: <string>,
  nom: <string>,
  prénom: <string>,
  salaires: [ # liste des épisodes
    {
      _id: <string>, # référence épisode
      salaire: <number>
    },
    ...
  ]
}]
```

Typage des 3 collections (niveau logique) : liens saison-épisode par inclusion, liens série-saison et actrice-épisode par référence

Plan

Requêtage avec MongoDB

Pas de standard pour le langage de manipulation des documents

Toute opération sur un seul document est atomique :

- Insert
- Find
- Update
- Delete

Lors d'insertion et mises à jour, la base de données et la collection sont automatiquement créées si elles n'existent pas

Langage de requête de MongoDB basé sur des motifs (patterns)

---

http://docs.mongodb.org/manual/crud/

Interrogation

db.collection.find(<condition>, <projection>**)**

- Le document <*condition*> spécifie les critères pour sélectionner les documents pertinents
- Le document facultatif <*projection*> indique les champs à inclure dans les documents résultats :
  - si non spécifié, retourne tous les champs
  - identifiant *_id* inclus par défaut
  - `<clé> : 1 | true` (inclusion) ou `<clé> : 0 | false` (exclusion)
  - pour les champs de documents imbriqués, `<clé>.<sous-clé> : 1` ou `<clé>: {<sous-clé> : 1}`

---

https://www.mongodb.com/docs/manual/reference/method/db.collection.find/

https://www.mongodb.com/docs/manual/reference/operator/query/

Interrogation - exemples de projection

```
db.etu.find()
```

Ana
annee : 2020
groupes : [A, A1]
notes : [
{ue : 'BD', note : 17},
{ue : 'WEB', note : 18}]

Bob
annee : 2022
groupes : [A, A2]
notes : [
{{ue : 'BD', note : 19},
{ue : 'WEB', note : 14}]

Cya
annee : 2021
groupes : [A, A1]
notes : [
{{ue : 'BD', note : 14}]

Dan
annee : 2020
groupes : [B, B1]
notes : [
{{ue : 'BD', note : 9},
{ue : 'WEB', note : 16}]

```
db.etu.find({},
  {
    annee: 1,
    groupes: true
  }
)
```

Ana
annee : 2020
groupes : [A, A1]

Bob
annee : 2022
groupes : [A, A2]

Cya
annee : 2021
groupes : [A, A1]

Dan
annee : 2020
groupes : [B, B1]

Exemples d'interrogation qui retourne tous les documents avec tous les champs (gauche) et seulement l'année et les groupes (droite)

Interrogation - exemples de sélection

```
db.etu.find({
    annee: 2022
  }
)
```

```
db.etu.find({
    annee: {$gt: 2020}
  },
  {
    notes: 0,
  }
)
```

Bob

annee : 2022
groupes : [A, A2]
notes : [
{{ue : 'BD', note : 19},
{ue : 'WEB', note : 14}]

Cya

annee : 2021
groupes : [A, A1]

Bob

annee : 2022
groupes : [A, A2]

Exemples qui retournent les documents de 2022 (gauche), et ceux avec une année supérieure à 2020, sans les notes (droite)

Insertion, mise à jour, suppression

```
db.etu.insertOne({
    _id: "Zoé",
    annee: 2022,
    annee_premiere_insc: 2022
        ,
    cursus_precedent: {
      etablissement: 'XYZ',
      diplome: true
    }
})
```

Zoé

annee : 2022
annee_premiere_insc : 2022,
cursus_precedent : { etablissement : 'XYZ', diplome : true}

```
db.etu.updateOne(
  { _id: "Cya" },
  { $push: { notes : {
        ue: "WEB", note:
         11 }}}
)
```

```
{
  acknowledged: true,
  matchedCount: 1,
  modifiedCount: 1
}
```

```
db.etu.deleteMany(
  { $not: {annee: 2022
        }}
)
```

```
{
  acknowledged: true,
  deletedCount: 3
}
```

Exemples d'insertion d'une nouvelle étudiante (gauche), d'une mise à jour de la note WEB de Cya (milieu) et d'une suppression des documents dont la date n'est pas 2022 (droite)

---

https://www.mongodb.com/docs/manual/reference/method/db.collection.insertOne/

https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/

https://www.mongodb.com/docs/manual/tutorial/remove-documents/

Python et MongoDB

Des API comme Pymongo, des Object-Document Model, etc.

```python
from pymongo import MongoClient

# connecting and getting collection
client = MongoClient('localhost', 27017)
coll_users = client.your_db.users

# inserting a document
coll_users.insert_one({"name" : "Alice"})

# querying and printing documents
for user in coll_users.find():
  print(user)

# count number of documents
nb = coll_users.count_documents({})
```

---

http://api.mongodb.com/python/current/
http://api.mongodb.com/python/current/tools.html

Bilan

- **Mouvement NoSQL / NoRel / NewSQL :**
  - paradigmes clé-valeur, colonne, document et graphe

- **MongoDB** :
  - modélisation guidée par les besoins applicatifs, mais un standard pour valider un schéma (json-schema)
  - langage de manipulation dépendant du SGBD, mais des propositions indépendantes (JSONiq, jq, jsonq ou json-query)
  - traitements distribués sur une collection (aggregation pipeline)

Notions approfondies dans l'UE MIF24 - BD NoSQL