Université Cote d'Azur/FDS(UEH)

MongoDB : Gestion d'un cabinet médical

Etudiants :
Douilly RODELY
Djimy SURLIN
Pierre Rubens MILORME
Bob Charlemagne PIERRE

Professeur :
Gabriel MOPOLO-MOKE

23 septembre 2024

Table des matières

1 Choix du sujet

SUJET : Gestion d'un cabinet médical

2 MCD MERISE

3 Conversion du MCD MERISE en des objets MONGODB et classes java

3.1 Spécification des Modèles de Documents

MongoDB est un système de base de données NoSQL orienté document, qui stocke les données sous forme de documents JSON/BSON. Les objets sont donc organisés en collections et chaque document peut être indépendant en termes de structure.
Dans notre cas de figure nous avons identifié deux collections (Medecin et Patient) suite à la conversion du MCD en objets MONGODB en respectant les préconisations et règles d'organisation des bases de données NOSQL et MONGODB en particulier, notamment en dénormalisant les données autant que possible.

3.1.1 Collection "Medecin" (document JSON/BSON)

```
{
  "_id": "ObjectId",
  "nom": "String",
  "sexe": "String",
  "date_naissance": "Date",
  "specialite": "String",
  "email": "String",
  "cv": "String",
  "list_telephones": ["String"],
  "list_prenoms": ["String"],
  "adresse": {
    "numero": "Number",
    "rue": "String",
    "code_postal": "Number",
    "ville": "String"
  },
  "list_rendez_vous": [
    {
      "patient": "ObjectId",
      "date_rendez_vous": "Date",
      "motif": "String"
    }
  ],
  "list_consultations": [
    {
      "patient": "ObjectId",
```

```
            "raison": "String",
            "diagnostic": "String",
            "date_consultation": "Date",
            "list_examens": [
              {
                "details_examen": "String",
                "date_examen": "Date"
              }
            ],
            "list_prescriptions": [
              {
                "details_prescription": "String",
                "date_prescription": "Date"
              }
            ]
          }
        ]
      }
```

Exemple de document de médecin

```
  {
    "_id": "66f0820f580fe26567ae4f8d",
    "nom": "Kertzmann",
    "sexe": "Other",
    "date_naissance": "1997-08-08",
    "specialite": "Gynecologist",
    "email": "susy_mohr@metz.test",
    "cv": "Qui doloremque debitis et autem placeat natus qui quia
       voluptatibus.",
    "list_telephones": [
      "409.780.8066",
      "740-880-4984"
    ],
    "list_prenoms": [
      "Beau",
      "Jesica"
    ],
    "adresse": {
      "numero": "7777",
      "rue": "Francesco Lights",
      "code_postal": "32176-8834",
      "ville": "Isabellashire"
    },
    "list_rendez_vous": [
      {
        "patient": "66f0820f580fe26567ae4f8e",
        "date_rendez_vous": "2024-10-04",
        "motif": "Quaerat doloremque nobis."
      }
    ],
```

```
    "list_consultations": [
      {
        "patient": "66f0820f580fe26567ae4f8f",
        "raison": "Magnam dolorum suscipit.",
        "diagnostic": "Et nihil pariatur minima quam.",
        "date_consultation": "2024-08-24",
        "list_examens": [
          {
            "details_examen": "Rerum in et similique voluptates.",
            "date_examen": "2024-09-07"
          }
        ],
        "list_prescriptions": [
          {
            "details_prescription": "Et et non facere non.",
            "date_prescription": "2024-09-14"
          }
        ]
      }
    ]
  }
```

NB : Le modèle n'est pas un schéma strict. Les documents ne doivent pas nécessairement respecter en tout point le modèle de document. Le document ci-dessous est présenté à titre indicatif.

3.1.2 Collection "Patient" (document JSON)

```
{
  "_id": "ObjectId",
  "numero_securite_sociale": "String",
  "nom": "String",
  "sexe": "String",
  "date_naissance": "Date",
  "email": "String",
  "poids": "Number",
  "hauteur": "Number",
  "list_telephones": ["String"],
  "list_prenoms": ["String"],
  "adresse": {
    "numero": "Number",
    "rue": "String",
    "code_postal": "Number",
    "ville": "String"
  },
  "list_rendez_vous": [
    {
      "medecin": "ObjectId",
      "date_rendez_vous": "Date",
      "motif": "String"
```

```
    }
  ],
  "list_consultations": [
    {
      "medecin": "ObjectId",
      "raison": "String",
      "diagnostic": "String",
      "date_consultation": "Date",
      "list_examens": [
        {
          "details_examen": "String",
          "date_examen": "Date"
        }
      ],
      "list_prescriptions": [
        {
          "details_prescription": "String",
          "date_prescription": "Date"
        }
      ],
      "facture": {
        "Montant_Total": "Number",
        "date_facture": "Date"
      }
    }
  ],
  "antecedents_medicaux": [
    {
      "type": "String",
      "description": "String",
      "date": "Date"
    }
  ],
  "allergies": ["String"]
}
```

Exemple de document de patient

```
  {
    "_id": "66f08628c97d6838ff4b3b48",
    "numero_securite_sociale": "847-37-8324",
    "nom": "Abernathy",
    "sexe": "O",
    "date_naissance": "2009-08-12",
    "email": "matthew_hand@donnelly.test",
    "poids": 77.64,
    "hauteur": 1.66,
    "list_telephones": [
      "357-824-3744",
      "(365) 603-4456"
    ],
```

```
    "list_prenoms": [
      "Tom",
      "Jamar"
    ],
    "adresse": {
      "numero": "2945",
      "rue": "Walter Mills",
      "code_postal": "28820",
      "ville": "Friesenport"
    },
    "list_rendez_vous": [
      {
        "medecin": "66f08628c97d6838ff4b3b49",
        "date_rendez_vous": "2024-10-02",
        "motif": "Qui culpa delectus."
      }
    ],
    "list_consultations": [
      {
        "medecin": "66f08628c97d6838ff4b3b4a",
        "raison": "Et qui labore.",
        "diagnostic": "Porro totam ut ut possimus.",
        "date_consultation": "2024-09-21",
        "list_examens": [
          {
            "details_examen": "Laboriosam quo vitae quam quasi.",
            "date_examen": "2024-09-11"
          }
        ],
        "list_prescriptions": [
          {
            "details_prescription": "Tenetur sunt quae ex iusto.",
            "date_prescription": "2024-09-21"
          }
        ],
        "facture": {
          "Montant_Total": 329.15,
          "date_facture": "2024-09-16"
        }
      },
      {
        "medecin": "66f08628c97d6838ff4b3b4b",
        "raison": "Exercitationem animi ex.",
        "diagnostic": "Perspiciatis sequi placeat dolor amet.",
        "date_consultation": "2024-09-20",
        "list_examens": [
          {
            "details_examen": "Est eius excepturi fuga tenetur.",
            "date_examen": "2024-09-11"
          }
```

```
        ],
        "list_prescriptions": [
          {
            "details_prescription": "Quis aut dolores quia rerum.",
            "date_prescription": "2024-09-15"
          }
        ],
        "facture": {
          "Montant_Total": 480.4,
          "date_facture": "2024-09-19"
        }
      }
    ],
    "antecedents_medicaux": [
      {
        "type": "Chronic",
        "description": "Ducimus repellat eos labore quis.",
        "date": "2023-08-05"
      },
      {
        "type": "Acute",
        "description": "Eum dolorem labore et occaecati.",
        "date": "2024-06-21"
      }
    ],
    "allergies": [
      "eligendi",
      "ab"
    ]
  }
```


NB : Le modèle n'est pas un schéma strict. Les documents ne doivent pas nécessairement respecter en tout point le modèle de document. Le document ci-dessous est présenté à titre indicatif.

3.2 Spécification des classes et des méthodes JAVA

3.2.1 Méthodes CRUD

```java
// Classe Medecin
import java.util.Date;

public class Medecin {
    private String nom;
    private String sexe;
    private Date dateNaissance;
    private String specialite;
    private String email;

```

```
11      public Medecin(String nom, String sexe, Date
           dateNaissance, String specialite, String email) {
12          this.nom = nom;
13          this.sexe = sexe;
14          this.dateNaissance = dateNaissance;
15          this.specialite = specialite;
16          this.email = email;
17      }
18
19      public String getNom() {
20          return nom;
21      }
22
23      public void setNom(String nom) {
24          this.nom = nom;
25      }
26
27      public String getSexe() {
28          return sexe;
29      }
30
31      public void setSexe(String sexe) {
32          this.sexe = sexe;
33      }
34
35      public Date getDateNaissance() {
36          return dateNaissance;
37      }
38
39      public void setDateNaissance(Date dateNaissance) {
40          this.dateNaissance = dateNaissance;
41      }
42
43      public String getSpecialite() {
44          return specialite;
45      }
46
47      public void setSpecialite(String specialite) {
48          this.specialite = specialite;
49      }
50
51      public String getEmail() {
52          return email;
53      }
54
55      public void setEmail(String email) {
56          this.email = email;
57      }
58  }
59
```

```java
// Classe Patient
import java.util.Date;
import java.util.List;

public class Patient {
    private String id;
    private String numeroSecuriteSociale;
    private String nom;
    private String sexe;
    private Date dateNaissance;
    private String email;
    private double poids;
    private double hauteur;
    private List<String> listTelephones;
    private List<String> listPrenoms;
    private Adresse adresse;
    private List<RendezVous> listRendezVous;
    private List<Consultation> listConsultations;
    private List<AntecedentMedical> antecedentsMedicaux;
    private List<String> allergies;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getNumeroSecuriteSociale() {
        return numeroSecuriteSociale;
    }

    public void setNumeroSecuriteSociale(String
       numeroSecuriteSociale) {
        this.numeroSecuriteSociale = numeroSecuriteSociale;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getSexe() {
        return sexe;
    }

```

```
109     public void setSexe(String sexe) {
110         this.sexe = sexe;
111     }
112 
113     public Date getDateNaissance() {
114         return dateNaissance;
115     }
116 
117     public void setDateNaissance(Date dateNaissance) {
118         this.dateNaissance = dateNaissance;
119     }
120 
121     public String getEmail() {
122         return email;
123     }
124 
125     public void setEmail(String email) {
126         this.email = email;
127     }
128 
129     public double getPoids() {
130         return poids;
131     }
132 
133     public void setPoids(double poids) {
134         this.poids = poids;
135     }
136 
137     public double getHauteur() {
138         return hauteur;
139     }
140 
141     public void setHauteur(double hauteur) {
142         this.hauteur = hauteur;
143     }
144 
145     public List<String> getListTelephones() {
146         return listTelephones;
147     }
148 
149     public void setListTelephones(List<String> listTelephones
        ) {
150         this.listTelephones = listTelephones;
151     }
152 
153     public List<String> getListPrenoms() {
154         return listPrenoms;
155     }
156 
157     public void setListPrenoms(List<String> listPrenoms) {
```

```
158             this.listPrenoms = listPrenoms;
159         }
160
161         public Adresse getAdresse() {
162             return adresse;
163         }
164
165         public void setAdresse(Adresse adresse) {
166             this.adresse = adresse;
167         }
168
169         public List<RendezVous> getListRendezVous() {
170             return listRendezVous;
171         }
172
173         public void setListRendezVous(List<RendezVous>
                listRendezVous) {
174             this.listRendezVous = listRendezVous;
175         }
176
177         public List<Consultation> getListConsultations() {
178             return listConsultations;
179         }
180
181         public void setListConsultations(List<Consultation>
                listConsultations) {
182             this.listConsultations = listConsultations;
183         }
184
185         public List<AntecedentMedical> getAntecedentsMedicaux() {
186             return antecedentsMedicaux;
187         }
188
189         public void setAntecedentsMedicaux(List<AntecedentMedical
                > antecedentsMedicaux) {
190             this.antecedentsMedicaux = antecedentsMedicaux;
191         }
192
193         public List<String> getAllergies() {
194             return allergies;
195         }
196
197         public void setAllergies(List<String> allergies) {
198             this.allergies = allergies;
199         }
200 }
201
202 // Classe Examen
203 public class Examen {
204     private String detailsExamen;
```

```
205     private Date dateExamen;
206 
207     public String getDetailsExamen() {
208         return detailsExamen;
209     }
210 
211     public void setDetailsExamen(String detailsExamen) {
212         this.detailsExamen = detailsExamen;
213     }
214 
215     public Date getDateExamen() {
216         return dateExamen;
217     }
218 
219     public void setDateExamen(Date dateExamen) {
220         this.dateExamen = dateExamen;
221     }
222 }
223 
224 // Classe Prescription
225 public class Prescription {
226     private String detailsPrescription;
227     private Date datePrescription;
228 
229     public String getDetailsPrescription() {
230         return detailsPrescription;
231     }
232 
233     public void setDetailsPrescription(String
        detailsPrescription) {
234         this.detailsPrescription = detailsPrescription;
235     }
236 
237     public Date getDatePrescription() {
238         return datePrescription;
239     }
240 
241     public void setDatePrescription(Date datePrescription) {
242         this.datePrescription = datePrescription;
243     }
244 }
245 
246 // Classe Facture
247 public class Facture {
248     private double montantTotal;
249     private Date dateFacture;
250 
251     public double getMontantTotal() {
252         return montantTotal;
253     }
```

```
254
255     public void setMontantTotal(double montantTotal) {
256         this.montantTotal = montantTotal;
257     }
258
259     public Date getDateFacture() {
260         return dateFacture;
261     }
262
263     public void setDateFacture(Date dateFacture) {
264         this.dateFacture = dateFacture;
265     }
266 }
267
268 // Classe AntecedentMedical
269 public class AntecedentMedical {
270     private String type;
271     private String description;
272     private Date date;
273
274     public String getType() {
275         return type;
276     }
277
278     public void setType(String type) {
279         this.type = type;
280     }
281
282     public String getDescription() {
283         return description;
284     }
285
286     public void setDescription(String description) {
287         this.description = description;
288     }
289
290     public Date getDate() {
291         return date;
292     }
293
294     public void setDate(Date date) {
295         this.date = date;
296     }
297 }
298
299 // Classe Adresse
300 public class Adresse {
301     private int numero;
302     private String rue;
303     private int codePostal;
```

```
304     private String ville;
305
306     public int getNumero() {
307         return numero;
308     }
309
310     public void setNumero(int numero) {
311         this.numero = numero;
312     }
313
314     public String getRue() {
315         return rue;
316     }
317
318     public void setRue(String rue) {
319         this.rue = rue;
320     }
321
322     public int getCodePostal() {
323         return codePostal;
324     }
325
326     public void setCodePostal(int codePostal) {
327         this.codePostal = codePostal;
328     }
329
330     public String getVille() {
331         return ville;
332     }
333
334     public void setVille(String ville) {
335         this.ville = ville;
336     }
337 }
338
339 // Classe RendezVous
340 public class RendezVous {
341     private Medecin medecin;
342     private Date dateRendezVous;
343     private String motif;
344
345     public Medecin getMedecin() {
346         return medecin;
347     }
348
349     public void setMedecin(Medecin medecin) {
350         this.medecin = medecin;
351     }
352
353     public Date getDateRendezVous() {
```

```
354             return dateRendezVous;
355         }
356
357         public void setDateRendezVous(Date dateRendezVous) {
358             this.dateRendezVous = dateRendezVous;
359         }
360
361         public String getMotif() {
362             return motif;
363         }
364
365         public void setMotif(String motif) {
366             this.motif = motif;
367         }
368     }
369
370     // Classe Consultation
371     public class Consultation {
372         private Medecin medecin;
373         private String raison;
374         private String diagnostic;
375         private Date dateConsultation;
376         private List<Examen> listExamens;
377         private List<Prescription> listPrescriptions;
378         private Facture facture;
379
380         public Medecin getMedecin() {
381             return medecin;
382         }
383
384         public void setMedecin(Medecin medecin) {
385             this.medecin = medecin;
386         }
387
388         public String getRaison() {
389             return raison;
390         }
391
392         public void setRaison(String raison) {
393             this.raison = raison;
394         }
395
396         public String getDiagnostic() {
397             return diagnostic;
398         }
399
400         public void setDiagnostic(String diagnostic) {
401             this.diagnostic = diagnostic;
402         }
403
```

```
404     public Date getDateConsultation() {
405         return dateConsultation;
406     }
407 
408     public void setDateConsultation(Date dateConsultation) {
409         this.dateConsultation = dateConsultation;
410     }
411 
412     public List<Examen> getListExamens() {
413         return listExamens;
414     }
415 
416     public void setListExamens(List<Examen> listExamens) {
417         this.listExamens = listExamens;
418     }
419 
420     public List<Prescription> getListPrescriptions() {
421         return listPrescriptions;
422     }
423 
424     public void setListPrescriptions(List<Prescription>
        listPrescriptions) {
425         this.listPrescriptions = listPrescriptions;
426     }
427 
428     public Facture getFacture() {
429         return facture;
430     }
431 
432     public void setFacture(Facture facture) {
433         this.facture = facture;
434     }
435 }
```

2. Méthode pour insérer un médecin :

```
1 public void insertMedecin(Medecin medecin) {
2     MongoCollection<Document> medecinCollection = database.
         getCollection("Medecin");
3     Document doc = new Document("nom", medecin.getNom())
4                         .append("sexe", medecin.getSexe())
5                         .append("date_naissance", medecin.
                             getDateNaissance())
6                         .append("specialite", medecin.
                             getSpecialite())
7                         .append("email", medecin.getEmail());
8     medecinCollection.insertOne(doc);
9 }
```

3. `public void insertPatient(Patient patient) {`

```
        MongoCollection<Document> patientCollection =
            database.getCollection("Patient");

        Document addressDoc = new Document("numero", patient.
            getAdresse().getNumero())
                .append("rue", patient.getAdresse().getRue())
                .append("code_postal", patient.getAdresse().
                    getCodePostal())
                .append("ville", patient.getAdresse().
                    getVille());

        Document doc = new Document("numero_securite_sociale"
            , patient.getNumeroSecuriteSociale())
                .append("nom", patient.getNom())
                .append("sexe", patient.getSexe())
                .append("date_naissance", patient.
                    getDateNaissance())
                .append("email", patient.getEmail())
                .append("poids", patient.getPoids())
                .append("hauteur", patient.getHauteur())
                .append("list_telephones", patient.
                    getListTelephones())
                .append("list_prenoms", patient.
                    getListPrenoms())
                .append("adresse", addressDoc)
                .append("list_rendez_vous", patient.
                    getListRendezVous())
                .append("list_consultations", patient.
                    getListConsultations())
                .append("antecedents_medicaux", patient.
                    getAntecedentsMedicaux())
                .append("allergies", patient.getAllergies());

        patientCollection.insertOne(doc);
    }
```

4. Méthode pour lire un medecin par son id

```
public Medecin getMedecinById(String id) {
    MongoCollection<Document> medecinCollection = database.
        getCollection("Medecin");
    Document query = new Document("_id", new ObjectId(id));
    Document result = medecinCollection.find(query).first();
    if (result != null) {
        return new Medecin(result.getString("nom"), result.
            getString("specialite"));
    }
    return null;
}
```

5. Méthode pour mettre à jour un medecin

```
public void updateMedecin(String id, String email) {
    MongoCollection<Document> medecinCollection = database.
       getCollection("Medecin");
    medecinCollection.updateOne(Filters.eq("_id", new
       ObjectId(id)), Updates.set("email", email));
}
```

6. Méthode pour supprimer un medecin

```
public void deleteMedecin(String id) {
    MongoCollection<Document> medecinCollection = database.
       getCollection("Medecin");
    medecinCollection.deleteOne(Filters.eq("_id", new
       ObjectId(id)));
}
```

3.2.2 Indexation secondaire

1. Création d'index secondaire sur le champ "nom"

```
medecinCollection.createIndex(Indexes.ascending("nom"));
```

3.2.3 Méthode de Consultation (Jointure, Groupement)

1. Groupement des medecins par specialité

```
public void groupMedecinsBySpecialite() {
    MongoCollection<Document> medecinCollection = database.
       getCollection("Medecin");
    AggregateIterable<Document> groupResult =
       medecinCollection.aggregate(Arrays.asList(
        Aggregates.group("$specialite", Accumulators.sum("
           total", 1))
    ));

    for (Document doc : groupResult) {
        System.out.println(doc.toJson());
    }
}
```

4 Compléments sur le moteur NoSql MONGODB

4.1 Modèles de données supportés

4.1.1 Modèle de données : Document (JSON/BSON)

MongoDB stocke les données sous forme de documents JSON (ou BSON, une version binaire de JSON). Chaque document est un ensemble de paires clé-valeur. Cela permet de gérer des structures complexes, des tableaux et des sous-documents imbriqués. Le modèle est flexible, sans schéma strict (schema-less).

4.1.2 Usage

Convient aux cas où la structure des données peut varier ou être modifiée fréquemment.

4.2 Procédure d'installation du moteur et des utilitaires

4.2.1 Installation

Relativement simple. MongoDB offre des packages pour les systèmes d'exploitation populaires (Linux, Windows, macOS). Il suffit d'installer MongoDB et de démarrer le service MongoDB ("mongod").

4.2.2 Installation (Windows)

Voici un guide étape par étape pour installer MongoDB sur un système Windows :

1. Téléchargement de `MongoDB` :
   (a) Accédez à la page de téléchargement de MongoDB Community Edition.
   (b) Choisissez la version de MongoDB compatible avec votre système (en général, Windows 64-bit).
   (c) Téléchargez l'installeur `.msi` pour Windows.
2. Installation de `MongoDB` :
   (a) Ouvrez le fichier `.msi` téléchargé pour lancer l'installation.
   (b) Suivez les étapes de l'assistant d'installation :
      i. Sélectionnez l'option `Complete` pour installer `MongoDB` avec les options par défaut.
      ii. Cochez l'option `Install MongoDB as a Service` afin que `MongoDB` démarre automatiquement au démarrage de Windows.
      iii. Vous pouvez également installer `MongoDB Compass`, l'interface graphique de `MongoDB` (facultatif mais recommandé pour gérer vos bases de données).
3. Configurer MongoDB
   (a) Après l'installation, vous devez configurer `MongoDB` :
      i. Créer le répertoire de données : MongoDB utilise le dossier `C:\data\db`

   Par défaut pour stocker ses bases de données. Vous devez créer ce répertoire manuellement.

i. Ouvrez l'Explorateur de fichiers.
ii. Allez dans le disque `C:\`.
iii. Créez un dossier nommé `data`.
iv. À l'intérieur de `data`, créez un sous-dossier nommé `db` : `C:\data\db`

4. Vérifier que MongoDB est bien installé : Ouvrez un terminal (invité de commandes) et tapez la commande suivante pour vérifier que MongoDB est correctement installé :

   ```
   mongod --version
   ```

5. Lancer MongoDB : MongoDB devrait s'exécuter en tant que service, mais vous pouvez aussi le démarrer manuellement.
   — Via l'invite de commandes : Tapez cette commande `mongod` pour démarrer MongoDB. MongoDB sera lancé et écoutera par défaut sur le port `27017`.
   — Accéder à MongoDB avec le shell. Ouvrez un nouvel onglet dans l'invite de commandes et tapez : `mongo`. Cela lancera le shell MongoDB, où vous pourrez interagir avec votre base de données.
   — Utiliser MongoDB Compass (facultatif) Si vous avez installé MongoDB Compass, vous pouvez l'utiliser pour interagir avec MongoDB via une interface graphique en lançant l'application.

   Si `MongoDB` a été installé en tant que service, vous pouvez démarrer, arrêter ou redémarrer `MongoDB` directement depuis le gestionnaire de services de Windows :

   (a) Ouvrez "Services" (tapez `services.msc` dans l'Invite de commandes).
   (b) Recherchez le service "MongoDB".
   (c) Cliquez avec le bouton droit pour démarrer, arrêter ou redémarrer le service.

4.2.3 Installation (Linux)

```
bash > sudo apt-get install -y mongodb & systemctl start mongodb
```

4.2.4 Installation via Docker

1. Installer *Docker Desktop*.
2. Dans un terminal ou invité de commandes : `docker run -d -p 1521:1521 -e ORACLE_PASSWORD=<your password> gvenzl/oracle-xe`

4.2.5 Utilitaires

1. Outils de ligne de commande comme "mongo", "mongodump", et "mongoimport".
   — Mongo fait référence au client interactif utilisé pour interagir avec une base de données `MongoDB`.
      — Vous pouvez l'utiliser pour exécuter des commandes MongoDB, écrire des requêtes, gérer des collections, etc.
      — Commande typique pour se connecter à une base MongoDB : `mongo` Si vous voulez spécifier une base de données particulière : `mongo <nom_de_la_base>`

— Mongodump est un outil utilisé pour sauvegarder une base de données MongoDB. Il crée un *snapshot* des collections sous forme de fichiers BSON (Binary JSON).
  — Cela est utile pour les sauvegardes ou pour déplacer des données d'un environnement à un autre. Commande de base pour sauvegarder une base de données : `mongodump -db <nom_de_la_base>` Vous pouvez aussi spécifier un répertoire de sortie : `mongodump -db <nom_de_la_base> -out /chemin/vers/sauvegarde`
— Mongoimport permet de restaurer ou importer des données dans MongoDB depuis des fichiers au format JSON, CSV ou TSV. Commande de base pour importer un fichier JSON dans une collection MongoDB :
`mongoimport --db <nom\_de\_la\_base> --collection <nom\_collection> --file <chemin\_vers\_le\_fichier.json>`
. Pour des fichiers CSV, vous pouvez ajouter l'option `-type csv` :

```
mongoimport --db <nom\_de\_la\_base> --collection
<nom\_de\_la\_collection> --type csv --file
<chemin\_vers\_le\_fichier.csv> --headerline
```

Ces trois outils sont essentiels pour la gestion, la sauvegarde et la restauration des données dans MongoDB.

2. MongoDB Compass offre une interface utilisateur graphique pour interagir avec les données.

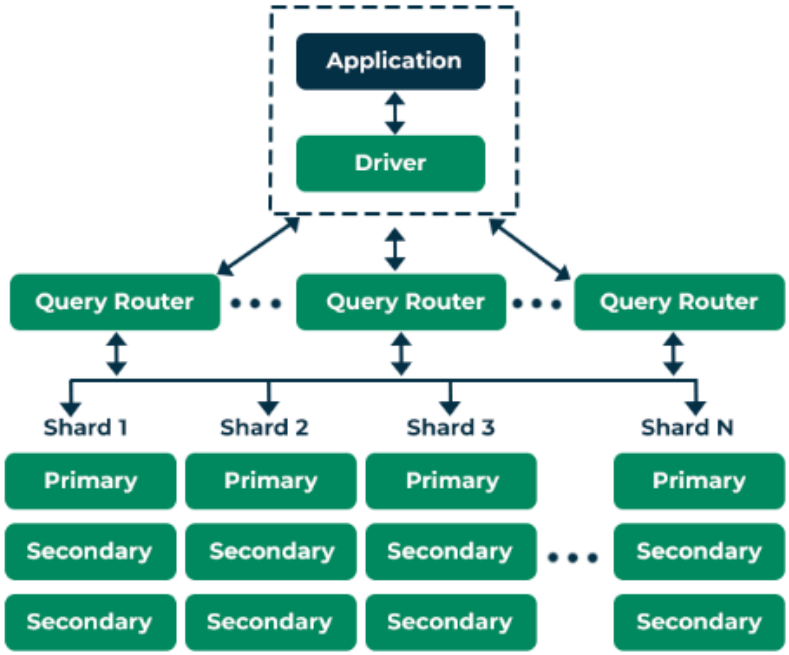
4.3 Architecture du moteur NoSQL (schéma)

4.3.1 Architecture

MongoDB suit une architecture maître-esclave (ou primaire-secondaire) dans un ensemble de *replicas*. Les données sont distribuées en partitions (sharding) pour faciliter la scalabilité horizontale. Les partitions sont réparties entre plusieurs serveurs.

4.3.2 Schéma

— Sharding pour le partitionnement.
— Réplication pour la haute disponibilité.
— Chaque ensemble de replicas a un replica *primaire* ou maître qui gère les écritures.



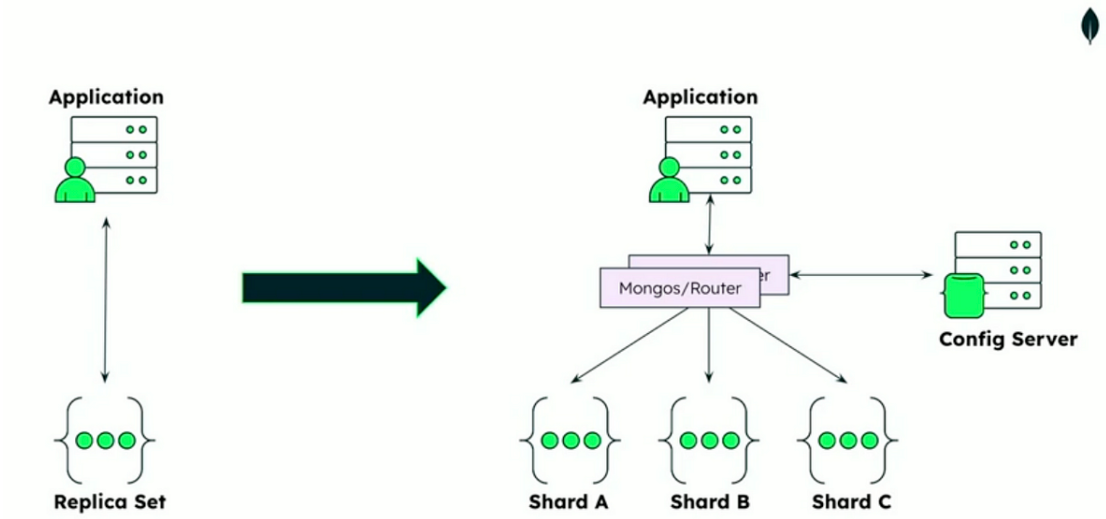
4.4 Méthode de partitionnement

4.4.1 Sharding

MongoDB utilise un mécanisme de *sharding* pour partitionner horizontalement les données. Les documents sont distribués entre différents partitions en fonction de la clé de sharding (clé de partitionnement).

4.4.2 Schéma

— Clé de partitionnement choisie pour diviser les documents.
— Chaque partition stocke une portion des données.



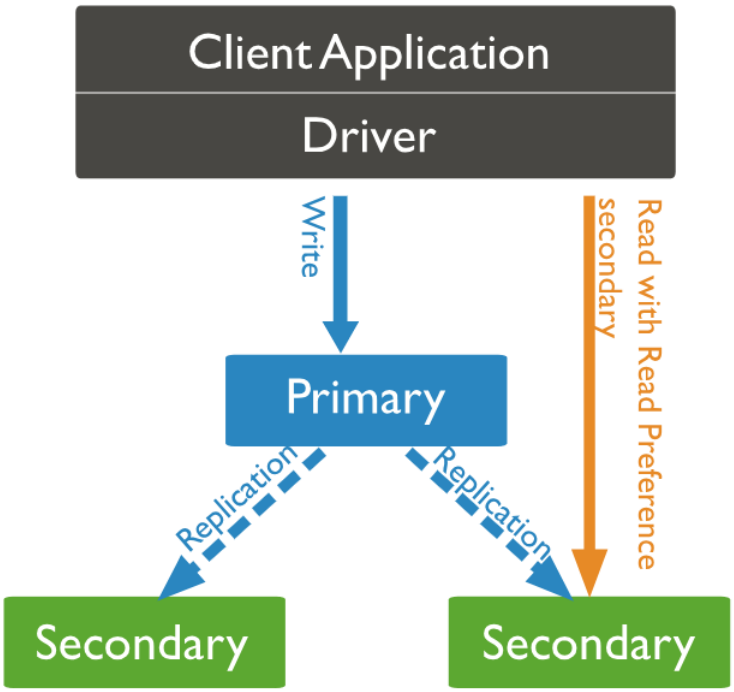
4.5 Méthode de réplication

4.5.1 Réplication

MongoDB utilise des *replicas* pour la haute disponibilité. Un ensemble de replicas contient un replica primaire et plusieurs secondaires. Le primaire gère les écritures, et les réplicas (secondaires) répliquent les données.

4.5.2 Schéma

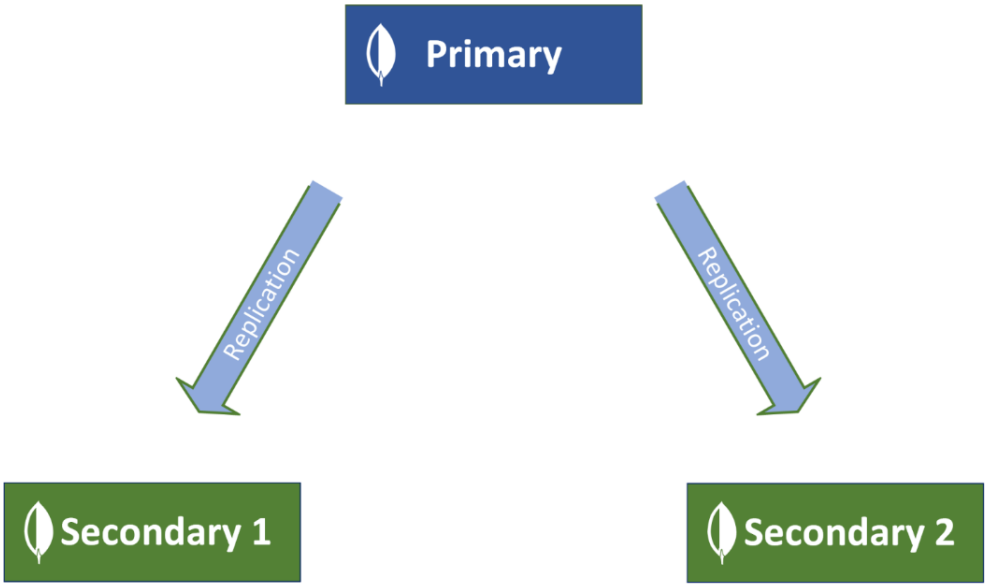
— Primaire pour les écritures.
— Secondaires pour la redondance et les lectures.



4.6 Montée en charge

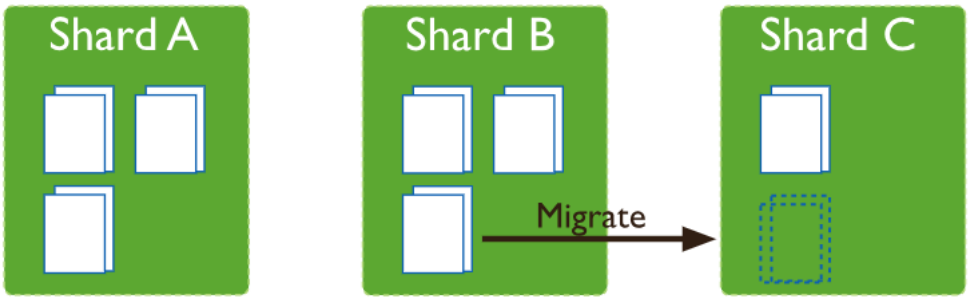
4.6.1 Montée en charge verticale

MongoDB permet une montée en charge verticale grâce au mécanisme de réplication sur plusieurs serveurs dans une même partition. En ajoutant de nouveaux serveurs ou replicas, les données peuvent être répliquées de manière synchrone par défaut pour mieux répartir la charge. L'ajout de nouveaux esclaves ou replicas dans une partition pour étendre la base de données verticalement permet de faciliter les opérations de lecture garantir et améliorer la disponibilité.



4.6.2 Montée en charge horizontale

MongoDB permet une montée en charge horizontale grâce au partitionnement sur plusieurs serveurs. En ajoutant de nouveaux serveurs, les partitions peuvent être redistribuées pour mieux répartir la charge. L'ajout de partitions pour étendre la base de données horizontalement permet de faciliter les opérations d'écriture.

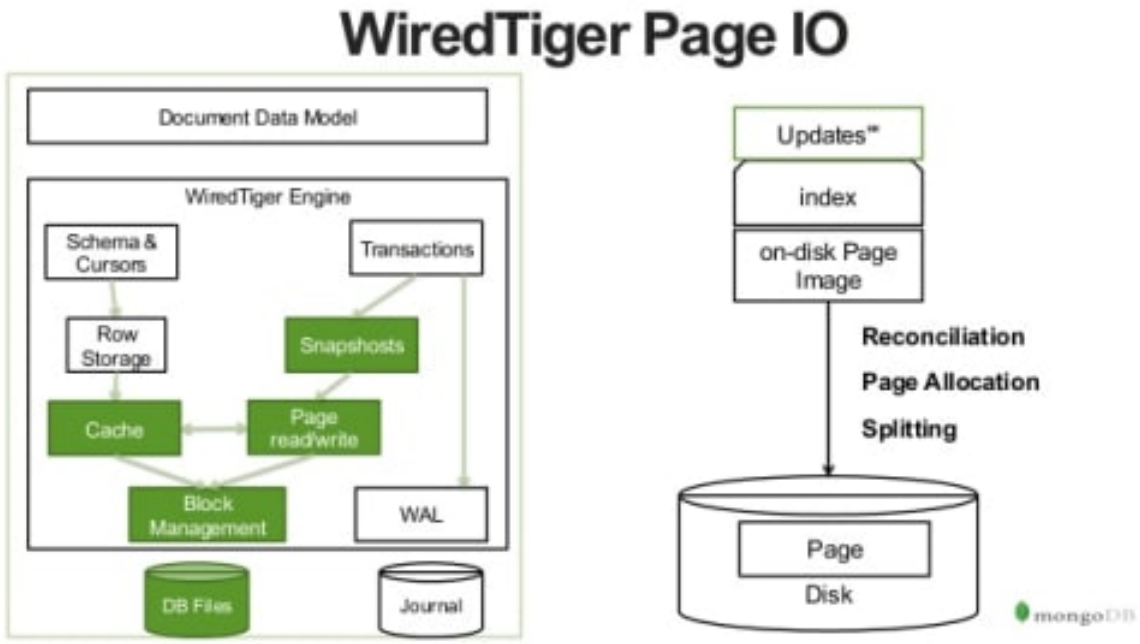


4.7 Gestion du cache mémoire

MongoDB utilise le cache de *mappage mémoire*. Le moteur *WiredTiger* utilise la mémoire disponible pour les lectures, mais laisse au système d'exploitation la gestion précise de l'utilisation de la mémoire.

4.7.1 Schéma

WiredTiger, mémoire tampon pour les écritures et mappage des pages de disque dans la mémoire.



5 Génération automatique des données

Dans cette section nous allons générer des données pour nos deux collections *Medecin* et *Patient*. Pour cela nous utilisons le package `gem` du langage `Ruby`. Ci-dessous la procédure d'installation de `Ruby` et `Gem` :

1. Installation de Ruby
   - macOS : Ruby est généralement déjà pré-installé.
   - Windows : Ruby est téléchargeable a l'adresse `https://rubyinstaller.org/`.
   - Linux (Ubuntu) : `sudo apt install ruby`.
2. Ouvrir un Terminal/Command Prompt :
   - macOS/Linux : Open a terminal.
   - Windows : Open Command Prompt or PowerShell.
3. `gem install faker`
4. `gem install mongo`
5. `gem list` (vérification de l'installation)

5.1 Script de génération de 1000 médecins

```
require 'faker'
require 'json'
require 'date'
require 'mongo'

def generate_random_document
  {
    "_id" => BSON::ObjectId.new,
    "nom" => Faker::Name.last_name,
```

```
10          "sexe" => ["Male", "Female", "Other"].sample,
11          "date_naissance" => Faker::Date.birthday(min_age: 25,
               max_age: 65),
12          "specialite" => ["Cardiologist", "Dermatologist", "
               Pediatrician", "Orthopedic Surgeon", "Neurologist", "
               Gynecologist"].sample,
13          "email" => Faker::Internet.email,
14          "cv" => Faker::Lorem.sentence(word_count: 10),
15          "list_telephones" => [Faker::PhoneNumber.phone_number,
               Faker::PhoneNumber.phone_number],
16          "list_prenoms" => [Faker::Name.first_name, Faker::Name.
               first_name],
17          "adresse" => {
18            "numero" => Faker::Address.building_number,
19            "rue" => Faker::Address.street_name,
20            "code_postal" => Faker::Address.zip_code,
21            "ville" => Faker::Address.city
22          },
23          "list_rendez_vous" => [
24            {
25              "patient" => BSON::ObjectId.new,
26              "date_rendez_vous" => Faker::Date.forward(days: 23),
27              "motif" => Faker::Lorem.sentence(word_count: 3)
28            }
29          ],
30          "list_consultations" => [
31            {
32              "patient" => BSON::ObjectId.new,
33              "raison" => Faker::Lorem.sentence(word_count: 3),
34              "diagnostic" => Faker::Lorem.sentence(word_count: 5),
35              "date_consultation" => Faker::Date.backward(days: 30),
36              "list_examens" => [
37                {
38                  "details_examen" => Faker::Lorem.sentence(
                       word_count: 5),
39                  "date_examen" => Faker::Date.backward(days: 15)
40                }
41              ],
42              "list_prescriptions" => [
43                {
44                  "details_prescription" => Faker::Lorem.sentence(
                       word_count: 5),
45                  "date_prescription" => Faker::Date.backward(days:
                       10)
46                }
47              ]
48            }
49          ]
50        }
51      end
```

```
52
53      documents = Array.new(1000) { generate_random_document }
54
55      # Optionally, you can print the documents or save them to a
            file
56      File.open("medecins_data.json", "w") do |f|
57        f.write(JSON.pretty_generate(documents))
58      end
59
60      puts "Generated 1000 MongoDB documents!"
```

5.2 Script de génération de 1000 patients

```
1       require 'mongo'
2       require 'faker'
3       require 'json'
4       require 'date'
5
6       # Function to generate a random MongoDB document for a patient
7       def generate_random_patient
8         {
9           "_id" => BSON::ObjectId.new,
10          "numero_securite_sociale" => Faker::IdNumber.valid,
11          "nom" => Faker::Name.last_name,
12          "sexe" => ["M", "F", "O"].sample,
13          "date_naissance" => Faker::Date.birthday(min_age: 0,
                max_age: 100),
14          "email" => Faker::Internet.email,
15          "poids" => Faker::Number.between(from: 45.0, to: 120.0).
                round(2),
16          "hauteur" => Faker::Number.between(from: 1.5, to: 2.0).
                round(2),
17          "list_telephones" => [Faker::PhoneNumber.phone_number,
                Faker::PhoneNumber.phone_number],
18          "list_prenoms" => [Faker::Name.first_name, Faker::Name.
                first_name],
19          "adresse" => {
20            "numero" => Faker::Address.building_number,
21            "rue" => Faker::Address.street_name,
22            "code_postal" => Faker::Address.zip_code,
23            "ville" => Faker::Address.city
24          },
25          "list_rendez_vous" => [
26            {
27              "medecin" => BSON::ObjectId.new,
28              "date_rendez_vous" => Faker::Date.forward(days: 23),
29              "motif" => Faker::Lorem.sentence(word_count: 3)
30            }
31          ],
```

```
32          "list_consultations" => generate_consultations(Faker),
33          "antecedents_medicaux" => generate_antecedents_medicaux(
               Faker),
34          "allergies" => [Faker::Lorem.word, Faker::Lorem.word]
35        }
36      end
37
38      # Function to generate consultations for the patient
39      def generate_consultations(faker)
40        consultations_list = []
41
42        2.times do
43          examens_list = [
44            {
45              "details_examen" => Faker::Lorem.sentence(word_count:
                   5),
46              "date_examen" => Faker::Date.backward(days: 15)
47            }
48          ]
49
50          prescriptions_list = [
51            {
52              "details_prescription" => Faker::Lorem.sentence(
                   word_count: 5),
53              "date_prescription" => Faker::Date.backward(days: 10)
54            }
55          ]
56
57          facture = {
58            "Montant_Total" => faker::Number.between(from: 50.0, to:
                 500.0).round(2),
59            "date_facture" => Faker::Date.backward(days: 7)
60          }
61
62          consultations_list << {
63            "medecin" => BSON::ObjectId.new,
64            "raison" => Faker::Lorem.sentence(word_count: 3),
65            "diagnostic" => Faker::Lorem.sentence(word_count: 5),
66            "date_consultation" => Faker::Date.backward(days: 30),
67            "list_examens" => examens_list,
68            "list_prescriptions" => prescriptions_list,
69            "facture" => facture
70          }
71        end
72
73        consultations_list
74      end
75
76      # Function to generate medical history for the patient
77      def generate_antecedents_medicaux(faker)
```

```ruby
    [
      {
        "type" => "Chronic",
        "description" => Faker::Lorem.sentence(word_count: 5),
        "date" => Faker::Date.backward(days: 500)
      },
      {
        "type" => "Acute",
        "description" => Faker::Lorem.sentence(word_count: 5),
        "date" => Faker::Date.backward(days: 300)
      }
    ]
  end

  # Generate 1000 patient documents
  patients = Array.new(1000) { generate_random_patient }

  # Optionally, save to a JSON file
  File.open("patients_data.json", "w") do |f|
    f.write(JSON.pretty_generate(patients))
  end

  puts "Generated 1000 MongoDB patient documents!"
```

5.3 Génération des données dans des fichiers JSON

5.3.1 Données des médecins

1. Mettre le script pour la génération des données pour les médecins dans un fichier, par exemple `medecins_script.rb`
2. `ruby medecins_script.rb`
3. le fichier `medecins_data.json` a été généré et contient les données des médecins au format *JSON*.

5.3.2 Données des patients

1. Mettre le script pour la génération des données pour les patients dans un fichier, par exemple `patients_script.rb`
2. `ruby patients_script.rb`
3. le fichier `patients_data.json` a été généré et contient les données des patients au format *JSON*.