Correction complète – NoSQL et MongoDB

Partie 1 : Théorie (10 points)

1. Définition de NoSQL

NoSQL (Not Only SQL) désigne une catégorie de systèmes de gestion de base de données qui ne suivent pas le modèle relationnel traditionnel. Ils sont conçus pour gérer des volumes importants de données non structurées ou semi-structurées, avec une grande scalabilité horizontale et une flexibilité de schéma.

2. Pourquoi choisir NoSQL ?

• Scalabilité horizontale
• Flexibilité du schéma
• Performances élevées en lecture/écriture
• Gestion des données non structurées
• Adapté aux applications modernes

3. SQL vs NoSQL

SQL : tables, schéma rigide, SQL standard, scalabilité verticale, ACID fort.
NoSQL : documents/clé-valeur/graphes, schéma flexible, scalabilité horizontale, BASE.

4. ACID et CAP

ACID : Atomicité, Cohérence, Isolation, Durabilité.
CAP : Cohérence, Disponibilité, Tolérance au partitionnement.
ACID concerne les transactions, CAP concerne les systèmes distribués.

5. Architectures NoSQL

Document (MongoDB), Clé-valeur (Redis), Colonnes larges (Cassandra), Graphes (Neo4j).

6. Exemples d'utilisation

Documents : catalogue produits. Clé-valeur : cache sessions. Colonnes larges : logs. Graphes : réseaux sociaux.

7. Concepts MongoDB

Base de données, Collection, Document, Field.

8. find() vs aggregate()

find() : requêtes simples. aggregate() : pipeline d’agrégation pour analyses complexes.

9. Opérateurs MongoDB

$eq, $gt, $lt, $in, $and, $or.

10. Requête étudiants 20–25 ans Informatique

```
db.etudiants.find({
  age: { $gte: 20, $lte: 25 },
  filiere: "Informatique"
});
```

11. Cours du professeur Kan

```
db.cours.aggregate([
  { $match: { professeur: "Kan" } }
]);
```

Partie 2 : Modélisation (2 points)

1. Embarqué vs Référencé

Embarqué : rapide, atomique mais redondance possible. Référencé : évite la redondance mais nécessite des jointures.

2. Structure des notes

```
{
  "_id": ObjectId("..."),
  "nom": "Dupont",
  "cours": [
    { "nomCours": "BDD", "note": 16 },
    { "nomCours": "Algo", "note": 14 }
  ]
}
```

Partie 3 : Indexation (2 points)

1. Importance des index

Améliorent performances, réduisent le scan, supportent l’unicité.

2. Création d'index

```
db.etudiants.createIndex({ nom: 1 });
```

Partie 4 : Agrégation (2 points)

Pipeline d’agrégation

Suite d’étapes $match, $group, $sort, $project.

Moyenne des notes

```
db.notes.aggregate([
  { $group: { _id: "$etudiantId", moyenne: { $avg: "$note" } } },
  { $sort: { moyenne: -1 } }
]);
```

Partie 5 : Pratique (4 points)

Ajout étudiant

```
db.etudiants.insertOne({
  nom: "Fatima",
  age: 21,
  filiere: "Maths"
});
```

Mise à jour filière

```
db.etudiants.updateOne(
  { nom: "Ali" },
  { $set: { filiere: "Mathématiques appliquées" } }
);
```

Suppression âge > 30

```
db.etudiants.deleteMany({ age: { $gt: 30 } });
```

Recherche Informatique

```
db.etudiants.find({ filiere: /Informatique/i });
```

Moyenne > 15

```
db.cours.aggregate([
  { $group: { _id: "$etudiant_id", moyenne: { $avg: "$note" } } },
  { $match: { moyenne: { $gt: 15 } } }
]);
```