Rapport du TP : Introduction à MapReduce avec MongoDB

Nom/Prénom : GNOUG Omayma
Date : 1/12/2025
Sujet :Prise en main de MongoDB

Introduction:

Ce travail pratique s'inscrit dans le cadre de l'étude des bases de données **NoSQL** et se concentre sur l'utilisation de **MongoDB**, un système de gestion de base de données orienté documents. L'objectif principal est de se familiariser avec la manipulation de données semi-structurées.

1. Compte total de films

```
db.movies.mapReduce(
  function() { emit("total", 1); },
  function(key, values) { return Array.sum(values); },
  { out: "total_count" }
);
```

2. Nombre de films par genre

```
db.movies.mapReduce(
  function() { this.genres.forEach(g => emit(g, 1)); },
  function(key, values) { return Array.sum(values); },
  { out: "count_by_genre" }
);
```

3. Nombre de films par réalisateur

```
db.movies.mapReduce(
  function() { emit(this.director, 1); },
  function(key, values) { return Array.sum(values); },
  { out: "count_by_director" }
);
```

4. Nombre d'acteurs uniques

```
db.movies.mapReduce(
  function() { this.actors.forEach(a => emit(a, 1)); },
  function(key, values) { return 1; },
  { out: "unique_actors_list" }
);
```

5. Nombre de films par année

```
db.movies.mapReduce(
  function() { emit(this.year, 1); },
  function(key, values) { return Array.sum(values); },
  { out: "count_by_year" }
);
```

6. Note moyenne par film

```
db.movies.mapReduce(
  function() {
    var avg = Array.avg(this.grades.map(g => g.score));
    emit(this.title, avg);
  },
  function(key, values) { return values[0]; },
  { out: "avg_score_per_movie" }
);
```

7. Note moyenne par genre

```
db.movies.mapReduce(
  function() {
    this.genres.forEach(g => {
      this.grades.forEach(grade => emit(g, { sum: grade.score, count: 1 }));
    });
  },
  function(key, values) {
    var res = { sum: 0, count: 0 };
    values.forEach(v => { res.sum += v.sum; res.count += v.count; });
    return res;
  },
  {
    out: "avg_score_by_genre",
    finalize: function(key, val) { return val.sum / val.count; }
  }
);
```

8. Note moyenne par réalisateur

```
db.movies.mapReduce(
  function() {
    this.grades.forEach(g => emit(this.director, { sum: g.score, count: 1 }));
  },
  function(key, values) {
    var res = { sum: 0, count: 0 };
    values.forEach(v => { res.sum += v.sum; res.count += v.count; });
    return res;
  },
  {
    out: "avg_score_by_director",
    finalize: function(key, val) { return val.sum / val.count; }
  }
);
```

9. Film avec la note maximale la plus élevée

```
db.movies.mapReduce(
  function() {
    var maxS = Math.max(...this.grades.map(g => g.score));
    emit("max_score_finder", { title: this.title, score: maxS });
  },
  function(key, values) {
    return values.reduce((prev, curr) => (curr.score > prev.score) ? curr : prev);
  },
  { out: "highest_score_movie" }
);
```

10. Compter les notes > 70

```
db.movies.mapReduce(
  function() {
    this.grades.forEach(g => { if(g.score > 70) emit("scores_gt_70", 1); });
  },
  function(key, values) { return Array.sum(values); },
  { out: "count_high_grades" }
);
```

11. Acteurs par genre (sans doublons)

```
db.movies.mapReduce(
  function() {
    this.genres.forEach(g => { this.actors.forEach(a => emit(g, [a])); });
  },
  function(key, values) {
    var unique = new Set();
    values.forEach(list => list.forEach(a => unique.add(a)));
    return Array.from(unique);
  },
  { out: "unique_actors_by_genre" }
);
```

12. Acteurs les plus fréquents

```
db.movies.mapReduce(
  function() { this.actors.forEach(a => emit(a, 1)); },
  function(key, values) { return Array.sum(values); },
  { out: "actor_appearances" }
);
```

13. Classement par lettre de grade majoritaire

```
db.movies.mapReduce(
  function() {
    var freq = {};
    this.grades.forEach(g => { freq[g.grade] = (freq[g.grade] || 0) + 1; });
    var major = Object.keys(freq).reduce((a, b) => freq[a] > freq[b] ? a : b);
    emit(major, this.title);
  },
  function(key, values) { return values.join(", "); },
  { out: "movies_by_major_letter" }
);
```

14. Note moyenne par année

```
db.movies.mapReduce(
  function() {
    this.grades.forEach(g => emit(this.year, { sum: g.score, count: 1 }));
  },
  function(key, values) {
    var res = { sum: 0, count: 0 };
    values.forEach(v => { res.sum += v.sum; res.count += v.count; });
    return res;
  },
  {
    out: "avg_score_by_year",
    finalize: function(key, val) { return val.sum / val.count; }
  }
);
```

15. Réalisateurs avec moyenne > 80

```
db.movies.mapReduce(
  function() {
    var sum = Array.sum(this.grades.map(g => g.score));
    emit(this.director, { sum: sum, count: this.grades.length });
  },
  function(key, values) {
    var res = { sum: 0, count: 0 };
    values.forEach(v => { res.sum += v.sum; res.count += v.count; });
    return res;
  },
  {
    out: "top_directors_80",
    finalize: function(key, val) {
      var avg = val.sum / val.count;
      return avg > 80 ? avg : null;
    }
  }
);
```

Conclusion

En conclusion, ce TP a permis de valider les compétences fondamentales nécessaires à l'administration et à l'exploitation d'une base MongoDB. La manipulation des fonctions et des outils d'agrégation a mis en évidence la flexibilité du modèle de données par rapport aux systèmes relationnels classiques.

L'analyse des performances via la méthode `explain()` a démontré l'importance cruciale des index (simples ou composés) pour garantir la rapidité des requêtes sur des collections volumineuses. Finalement, MongoDB s'affirme comme une solution robuste et particulièrement adaptée aux besoins modernes de traitement de données hétérogènes et aux analyses analytiques flexibles.