Nom : Loubna MANGOUCHI
N°Étudiant : 124099931
Classe : INFO3

Découverte de MapReduce avec MongoDB

Introduction

Ce TP s'inscrit dans le cadre de l'apprentissage des bases de données NoSQL. Il a pour objectif de découvrir MongoDB, une base de données orientée documents, ainsi que le paradigme MapReduce. À travers différentes requêtes appliquées à une collection de films, ce TP permet de manipuler des données semi-structurées et d'explorer les possibilités d'analyse offertes par MongoDB.

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 ayant 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. Nombre de notes strictement supérieures à 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 répétition)

```
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 apparaissant le plus fréquemment

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

13. Classement des films par lettre de note dominante

```
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 ayant une note moyenne supérieure à 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

Ce TP m'a permis de comprendre les principes fondamentaux de MongoDB et du modèle MapReduce. Les différentes requêtes réalisées ont mis en évidence la souplesse du modèle orienté documents, particulièrement adapté à la gestion de données complexes et hétérogènes.

L'utilisation des fonctions d'agrégation et l'étude des performances à l'aide de `explain()` ont également montré l'impact déterminant des index sur l'efficacité des requêtes. En conclusion, MongoDB apparaît comme une solution performante et flexible pour les applications modernes nécessitant des analyses avancées sur de grands volumes de données.