TP4 — Introduction au MapReduce avec MongoDB

Bases de Données NoSQL

Nom : Haiballa El Varougou

Année : 2025–2026

Table des matières

1 Introduction

Ce TP a pour objectif de manipuler le paradigme **MapReduce** proposé par MongoDB, en s'appuyant sur la collection de films utilisée lors des TPs précédents. MongoDB permet de réaliser des traitements distribués à l'aide de trois fonctions :

- une fonction `map` ;
- une fonction `reduce` ;
- éventuellement une fonction `finalize`.

Chaque exercice correspond à une transformation MapReduce appliquée à la collection `films`.

La syntaxe générale utilisée est la suivante :

```
db.films.mapReduce(mapFunction, reduceFunction, {
  out: "<nom_collection_sortie>"
})
```

2 Exercices MapReduce

2.1 Compter le nombre total de films

Map

```
function () {
  emit("total_films", 1);
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.2 Compter le nombre de films par genre

Map

```
function () {
  if (this.genre) {
    this.genre.forEach(g => emit(g, 1));
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.3 Nombre de films par réalisateur

Map

```
function () {
  if (this.director) {
    emit(this.director, 1);
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.4 Compter les acteurs distincts

Map

```
function () {
  if (this.actors) {
    this.actors.forEach(a => emit(a, 1));
  }
}
```

Reduce

```
function (key, values) {
  return 1;
}
```

2.5 Films par année de sortie

Map

```
function () {
  emit(this.year, 1);
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.6 Note moyenne par film

Map

```
function () {
  if (this.grades) {
    const avg =
      this.grades.reduce((a, b) => a + b.score, 0) /
      this.grades.length;
    emit(this.title, avg);
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values) / values.length;
}
```

2.7 Note moyenne par genre

Map

```
function () {
  if (this.grades && this.genre) {
    const avg =
      this.grades.reduce((a, b) => a + b.score, 0) /
      this.grades.length;
    this.genre.forEach(g => emit(g, avg));
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values) / values.length;
}
```

2.8 Note moyenne par réalisateur

Map

```
function () {
  if (this.director && this.grades) {
    const avg =
      this.grades.reduce((a, b) => a + b.score, 0) /
      this.grades.length;
    emit(this.director, avg);
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values) / values.length;
}
```

2.9 Film avec la note maximale

Map

```
function () {
  if (this.grades) {
    const max = Math.max(...this.grades.map(g => g.score));
    emit(this.title, max);
  }
}
```

Reduce

```
function (key, values) {
  return Math.max(...values);
}
```

2.10 Nombre de notes supérieures à 70

Map

```
function () {
  if (this.grades) {
    const count = this.grades.filter(g => g.score > 70).length;
    emit("notes_sup_70", count);
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.11 Acteurs par genre (sans doublons)

Map

```
function () {
  if (this.genre && this.actors) {
    this.genre.forEach(g =>
      this.actors.forEach(a => emit(g, a))
    );
  }
}
```

Reduce

```
function (key, values) {
  return Array.from(new Set(values));
}
```

2.12 Acteurs présents dans le plus grand nombre de films

Map

```
function () {
  if (this.actors) {
    this.actors.forEach(a => emit(a, 1));
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values);
}
```

2.13 Classement des films par grade majoritaire

Map

```
function () {
  if (this.grades) {
    const freq = {};
    this.grades.forEach(g => {
      freq[g.grade] = (freq[g.grade] || 0) + 1;
    });
    const major =
      Object.keys(freq).reduce((a, b) =>
        freq[a] > freq[b] ? a : b
      );
    emit(major, this.title);
  }
}
```

Reduce

```
function (key, values) {
  return values;
}
```

2.14 Note moyenne par année

Map

```
function () {
  if (this.grades && this.year) {
    const avg =
      this.grades.reduce((a, b) => a + b.score, 0) /
      this.grades.length;
    emit(this.year, avg);
  }
}
```

Reduce

```
function (key, values) {
  return Array.sum(values) / values.length;
}
```

2.15 Réalisateurs avec une moyenne supérieure à 80

Map

```
function () {
  if (this.director && this.grades) {
    const avg =
      this.grades.reduce((a, b) => a + b.score, 0) /
      this.grades.length;
    emit(this.director, { sum: avg, count: 1 });
  }
}
```

Reduce

```
function (key, values) {
  let s = 0, c = 0;
  values.forEach(v => {
    s += v.sum;
    c += v.count;
  });
  return { sum: s, count: c };
}
```

Finalize

```
function (key, value) {
  const avg = value.sum / value.count;
  return avg > 80 ? avg : null;
}
```

Fin du document