TP MapReduce avec MongoDB

---

1. Compter le nombre total de films dans la collection.

```
filmsdb> var mapTotal = function () {
...   emit("total", 1);
... };
...
... var reduceTotal = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapTotal,
...   reduceTotal,
...   { out: "total_movies" }
... );
...
{ result: 'total_movies', ok: 1 }
filmsdb> db.total_movies.find().pretty();
...
[ { _id: 'total', value: 278 } ]
filmsdb>
```

Explication :
L'idée consiste à émettre la valeur 1 pour chaque film à l'aide de la fonction *Map*, puis à additionner toutes ces valeurs dans la fonction *Reduce* afin d'obtenir le nombre total de films.

2. Compter le nombre de films par genre.

```
filmsdb> var mapGenre = function () {
...   if (this.genre) {
...     emit(this.genre, 1);
...   }
... };
...
... var reduceGenre = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapGenre,
...   reduceGenre,
...   { out: "movies_by_genre" }
... );
...
{ result: 'movies_by_genre', ok: 1 }
```

Résultat :

```
filmsdb> db.movies_by_genre.find().pretty(
...
[
  { _id: 'Horreur', value: 8 },
  { _id: 'Musique', value: 2 },
  { _id: 'Science-Fiction', value: 9 },
  { _id: 'Romance', value: 3 },
  { _id: 'War', value: 1 },
  { _id: 'Science Fiction', value: 1 },
  { _id: 'Drama', value: 14 },
  { _id: 'Mystère', value: 6 },
  { _id: 'Thriller', value: 10 },
  { _id: 'Fantasy', value: 2 },
  { _id: 'Histoire', value: 1 },
  { _id: 'Mystery', value: 1 },
  { _id: 'Crime', value: 29 },
  { _id: 'Adventure', value: 3 },
  { _id: 'Guerre', value: 1 },
  { _id: 'Drame', value: 96 },
  { _id: 'Comedy', value: 1 },
  { _id: 'Action', value: 36 },
  { _id: 'Comédie', value: 25 },
```

Explication :

La phase *Map* utilise le genre comme clé et émet la valeur 1 pour chaque film.
La phase *Reduce* additionne ces valeurs afin d'obtenir le nombre total de films par genre.

3. Compter le nombre de films réalisés par chaque réalisateur.

```
filmsdb> var mapDirector = function () {
...   if (this.director && this.director.first_name && this.director.last_name) {
...     var directorName = this.director.first_name + " " + this.director.last_name;
...     emit(directorName, 1);
...   }
... };
...
... var reduceDirector = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapDirector,
...   reduceDirector,
...   { out: "movies_by_director" }
... );
...
{ result: 'movies_by_director', ok: 1 }
filmsdb>
```

Résultat :

```
filmsdb> db.movies_by_director.find().pretty();
...
[
  { _id: 'Andrew Davis', value: 1 },
  { _id: 'Ladj Ly', value: 1 },
  { _id: 'Thomas Lilti', value: 2 },
  { _id: 'John Carpenter', value: 1 },
  { _id: 'Jacques Tati', value: 1 },
  { _id: 'Peter Weir', value: 1 },
  { _id: 'Bruno Podalydès', value: 1 },
  { _id: 'Florian Henckel von Donnersmarck', value: 1 },
  { _id: 'Charles Vidor', value: 1 },
  { _id: 'Martin Scorsese', value: 4 },
  { _id: 'Gene Kelly', value: 1 },
  { _id: 'Terrence Malick', value: 2 },
  { _id: 'David Fincher', value: 4 },
  { _id: 'David Lean', value: 1 },
  { _id: 'Martin McDonagh', value: 1 },
  { _id: 'John Boorman', value: 1 },
  { _id: 'Paul Verhoeven', value: 5 },
  { _id: 'Tim Burton', value: 2 },
  { _id: 'Clint Eastwood', value: 2 },
  { _id: 'Michel Hazanavicius', value: 2 }
]
```

Explication :

La phase *Map* émet le nom du réalisateur comme clé et la valeur 1 pour chaque film. La phase *Reduce* additionne ces valeurs afin d'obtenir le nombre de films réalisés par chaque réalisateur.

4. Compter le nombre d'acteurs uniques apparaissant dans tous les films.

```
filmsdb> var mapActors = function () {
...   if (this.actors && Array.isArray(this.actors)) {
...     this.actors.forEach(function (a) {
...       // clé unique (on inclut birth_date si dispo pour éviter les homonymes)
...       var key = a.first_name + " " + a.last_name + "|" + (a.birth_date ?? "NA");
...       emit(key, 1);
...     });
...   }
... };
...
... var reduceActors = function (key, values) {
...   // on s'en fiche du nombre d'occurrences, juste de l'unicité
...   return 1;
... };
...
... db.movies.mapReduce(
...   mapActors,
...   reduceActors,
...   { out: "unique_actors" }
... );
...
{ result: 'unique_actors', ok: 1 }
filmsdb>
```

Résultat :

```
filmsdb> db.unique_actors.find().limit(10).pretty();
...
[
  { _id: 'James Stewart|1908', value: 1 },
  { _id: 'Daniel Craig|1968', value: 1 },
  { _id: 'Blythe Danner|1943', value: 1 },
  { _id: 'John Gavin|1931', value: 1 },
  { _id: 'Maxim Gaudette|1974', value: 1 },
  { _id: 'Jacques Tati|1907', value: 1 },
  { _id: 'Renato Salvatori|1933', value: 1 },
  { _id: 'Edward G. Robinson|1893', value: 1 },
  { _id: 'Richard Bright|1937', value: 1 },
  { _id: 'Michael Gough|1916', value: 1 }
]
filmsdb>
```

Explication :

La phase *Map* parcourt la liste `actors` de chaque film et émet une clé représentant l'acteur (nom + prénom + date de naissance).
La phase *Reduce* regroupe les occurrences par clé, ce qui permet d'obtenir un document par acteur unique. On compte ensuite ces documents avec **`countDocuments()`.**

5. Lister le nombre de films par année de sortie.

```
filmsdb> var mapYear = function () {
...   if (this.year) {
...     emit(this.year, 1);
...   }
... };
...
... var reduceYear = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapYear,
...   reduceYear,
...   { out: "movies_by_year" }
... );
...
```

Résultats :

```
... db.movies_by_year.find().pretty();
...
[
  { _id: 1965, value: 1 }, { _id: 1948, value: 1 },
  { _id: 2018, value: 8 }, { _id: 1990, value: 7 },
  { _id: 1927, value: 1 }, { _id: 1950, value: 2 },
  { _id: 1944, value: 1 }, { _id: 1997, value: 6 },
  { _id: 1946, value: 2 }, { _id: 2000, value: 5 },
  { _id: 1955, value: 1 }, { _id: 1974, value: 4 },
  { _id: 1967, value: 5 }, { _id: 1957, value: 3 },
  { _id: 2012, value: 4 }, { _id: 1931, value: 1 },
  { _id: 1940, value: 3 }, { _id: 1952, value: 2 },
  { _id: 1970, value: 2 }, { _id: 1942, value: 1 }
]
Type "it" for more
filmsdb>
```

Explication :
La phase Map utilise l'année de sortie comme clé et émet la valeur 1 pour chaque film.
La phase Reduce additionne les valeurs associées à chaque année afin d'obtenir le nombre de films par année de sortie.

6. Calculer la note moyenne par film à partir du tableau grades

```
filmsdb> var mapGrades = function () {
...   if (this.grades && Array.isArray(this.grades)) {
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       sum += g;
...       count += 1;
...     });
...
...     emit(this.title, { total: sum, count: count });
...   }
... };
...
... var reduceGrades = function (key, values) {
...   var total = 0;
...   var count = 0;
...
...   values.forEach(function (v) {
...     total += v.total;
...     count += v.count;
...   });
...
...   return { total: total, count: count };
... };
...
... var finalizeGrades = function (key, reducedValue) {
...   return reducedValue.total / reducedValue.count;
... };
...
... db.movies.mapReduce(
...   mapGrades,
...   reduceGrades,
...   {
...     out: "average_grade_by_movie",
...     finalize: finalizeGrades
...   }
... );
...
{ result: 'average_grade_by_movie', ok: 1 }
filmsdb>
```

Résultat :

```
filmsdb> db.average_grade_by_movie.find().pretty();
...
[
  { _id: 'Amadeus', value: NaN },
  { _id: 'Potiche', value: NaN },
  { _id: "Faute d'amour", value: NaN },
  { _id: 'La Grande vadrouille', value: NaN },
  { _id: 'La Favorite', value: NaN },
  { _id: 'Annie Hall', value: NaN },
  { _id: 'Cries and Whispers', value: NaN },
  { _id: 'The Lord of the Rings: The Return of the King', value: NaN },
  { _id: 'Bicycle Thieves', value: NaN },
  { _id: 'Django Unchained', value: NaN },
  { _id: 'Kill Bill : Volume 2', value: NaN },
  { _id: 'Le Dernier Métro', value: NaN },
  { _id: 'La Nuit américaine', value: NaN },
  { _id: 'Les Bronzés', value: NaN },
  { _id: 'Dracula', value: NaN },
  { _id: "L'Homme qui voulut être roi", value: NaN },
  { _id: 'Les Vikings', value: NaN },
  { _id: 'Spider-Man 3', value: NaN },
  { _id: 'Total Recall', value: NaN },
  { _id: "Je vais bien, ne t'en fais pas", value: NaN }
```

Explication :

La phase *Map* calcule la somme et le nombre de notes du tableau `grades` pour chaque film.
La phase *Reduce* additionne ces valeurs, puis la fonction *Finalize* calcule la moyenne des notes pour chaque film.

7. Calculer la note moyenne par genre

```
filmsdb> var mapAvgGenre = function () {
...   if (this.genre && this.grades && Array.isArray(this.grades) &&
this.grades.length > 0) {
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       sum += g;
...       count += 1;
...     });
...
...     emit(this.genre, { total: sum, count: count });
...   }
... };
...
... var reduceAvgGenre = function (key, values) {
...   var total = 0;
...   var count = 0;
...
...   values.forEach(function (v) {
        total += v.total;

...
...   values.forEach(function (v) {
...     total += v.total;
...     count += v.count;
...   });
...
...   return { total: total, count: count };
... };
...
... var finalizeAvgGenre = function (key, reducedValue) {
...   return reducedValue.total / reducedValue.count;
... };
...
... db.movies.mapReduce(
...   mapAvgGenre,
...   reduceAvgGenre,
...   {
...     out: "average_grade_by_genre",
...     finalize: finalizeAvgGenre
...   }
... );
...
{ result: 'average_grade_by_genre', ok: 1 }
filmsdb>
```

Résultats :

```
filmsdb> db.average_grade_by_genre.find().pretty();
...
[
  { _id: 'Adventure', value: NaN },
  { _id: 'Action', value: NaN },
  { _id: 'Comédie', value: NaN },
  { _id: 'Drame', value: NaN },
  { _id: 'Comedy', value: NaN },
  { _id: 'Fantastique', value: NaN },
  { _id: 'Guerre', value: NaN },
  { _id: 'Aventure', value: NaN },
  { _id: 'Western', value: NaN },
  { _id: 'Mystery', value: NaN },
  { _id: 'Science-Fiction', value: NaN },
  { _id: 'Musique', value: NaN },
  { _id: 'Horreur', value: NaN },
  { _id: 'Thriller', value: NaN },
  { _id: 'Drama', value: NaN },
  { _id: 'Mystère', value: NaN },
  { _id: 'Science Fiction', value: NaN },
  { _id: 'Histoire', value: NaN },
  { _id: 'Romance', value: NaN },
  { _id: 'War', value: NaN }
```

Explication :

La phase *Map* calcule la somme et le nombre de notes du tableau `grades` pour chaque film, puis regroupe ces informations par genre.
La phase *Reduce* additionne ces sommes et effectifs par genre, et la fonction *Finalize* calcule la moyenne des notes pour chaque genre.

8. Calculer la note moyenne par réalisateur.

```
filmsdb> var mapAvgDirector = function () {
...   if (
...     this.director &&
...     this.director.first_name &&
...     this.director.last_name &&
...     this.grades &&
...     Array.isArray(this.grades) &&
...     this.grades.length > 0
...   ) {
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       sum += g;
...       count += 1;
...     });
...
...     var directorName = this.director.first_name + " " + this.dire
ctor.last_name;
...     emit(directorName, { total: sum, count: count });
...   }
... };
...
... var reduceAvgDirector = function (key, values) {
```

Résultat :

```
filmsdb> db.average_grade_by_director.find().pretty();
...
[
  { _id: 'Andrew Davis', value: NaN },
  { _id: 'Ladj Ly', value: NaN },
  { _id: 'Thomas Lilti', value: NaN },
  { _id: 'John Carpenter', value: NaN },
  { _id: 'Jacques Tati', value: NaN },
  { _id: 'Peter Weir', value: NaN },
  { _id: 'Bruno Podalydès', value: NaN },
  { _id: 'Florian Henckel von Donnersmarck', value: NaN },
  { _id: 'Charles Vidor', value: NaN },
  { _id: 'Martin Scorsese', value: NaN },
  { _id: 'Gene Kelly', value: NaN },
  { _id: 'Terrence Malick', value: NaN },
  { _id: 'David Fincher', value: NaN },
  { _id: 'David Lean', value: NaN },
  { _id: 'Martin McDonagh', value: NaN },
  { _id: 'John Boorman', value: NaN },
  { _id: 'Paul Verhoeven', value: NaN },
  { _id: 'Tim Burton', value: NaN },
  { _id: 'Clint Eastwood', value: NaN },
```

Explication :

La phase *Map* calcule la somme et le nombre de notes pour chaque film et émet ces informations en utilisant le réalisateur comme clé.
La phase *Reduce* regroupe les valeurs par réalisateur, et la fonction *Finalize* calcule la note moyenne pour chaque réalisateur.

9. Trouver le film avec la note maximale la plus élevée

```
filmsdb> var mapMaxGrade = function () {
...   if (this.grades && Array.isArray(this.grades) && this.grades.le
ngth > 0) {
...     var maxGrade = Math.max.apply(null, this.grades);
...     emit("max", { title: this.title, grade: maxGrade });
...   }
... };
...
... var reduceMaxGrade = function (key, values) {
...   var max = values[0];
...
...   values.forEach(function (v) {
...     if (v.grade > max.grade) {
...       max = v;
...     }
...   });
...
...   return max;
... };
...
... db.movies.mapReduce(
...   mapMaxGrade,
...   reduceMaxGrade,
...   { out: "best_rated_movie" }
... );
```

Explication :

La phase *Map* calcule la note maximale de chaque film et l'émet avec une clé unique.
La phase *Reduce* compare ces valeurs et conserve le film ayant la note maximale la plus élevée.

10. Compter le nombre de notes supérieures à 70 dans tous les films.

```
filmsdb> var mapHighGrades = function () {
...   if (this.grades && Array.isArray(this.grades)) {
...     this.grades.forEach(function (g) {
...       if (g > 70) {
...         emit("count", 1);
...       }
...     });
...   }
... };
...
... var reduceHighGrades = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapHighGrades,
...   reduceHighGrades,
...   { out: "grades_above_70" }
... );
...
{ result: 'grades_above_70', ok: 1 }
filmsdb>
```

Explication :

La phase *Map* parcourt toutes les notes du tableau `grades` et émet la valeur `1` pour chaque note strictement supérieure à 70.
La phase *Reduce* additionne ces valeurs afin d'obtenir le nombre total de notes supérieures à 70 dans l'ensemble des films.

11. Lister tous les acteurs par genre, sans doublons.

```
filmsdb> var mapActorsByGenre = function () {
...   if (this.genre && this.actors && Array.isArray(this.actors)) {
...     this.actors.forEach(function (a) {
...       var actorKey = a.first_name + " " + a.last_name + "|" + (a.
birth_date ?? "NA");
...       emit(this.genre, actorKey);
...     }, this);
...   }
... };
...
... var reduceActorsByGenre = function (key, values) {
...   // supprimer les doublons
...   return Array.from(new Set(values));
... };
...
... db.movies.mapReduce(
...   mapActorsByGenre,
...   reduceActorsByGenre,
...   { out: "actors_by_genre" }
... );
...
{ result: 'actors_by_genre', ok: 1 }
filmsdb>
```

Explication :

La phase *Map* associe chaque acteur au genre du film dans lequel il apparaît.
La phase *Reduce* supprime les doublons afin d'obtenir, pour chaque genre, la liste des acteurs uniques.

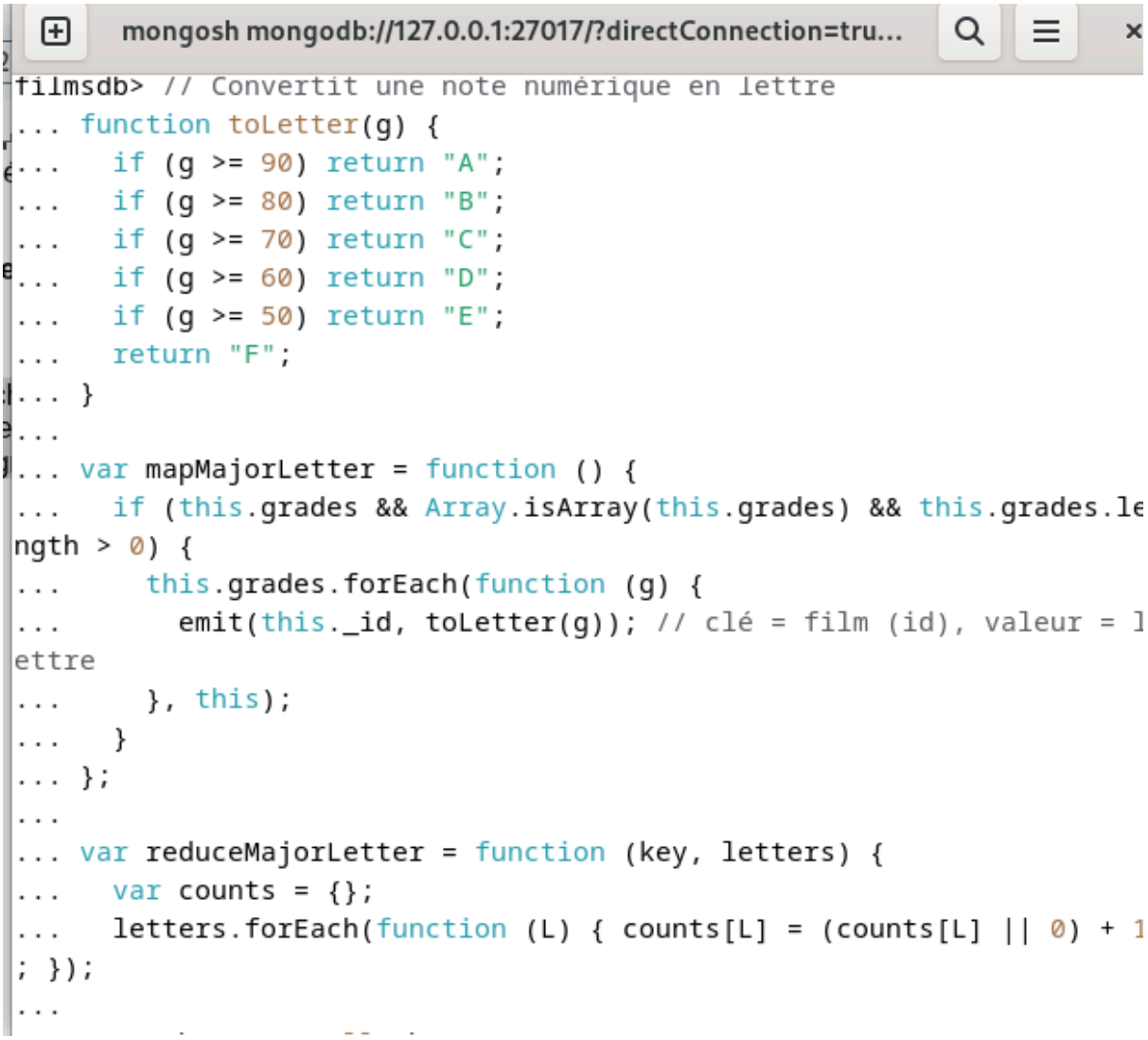
12. Trouver les acteurs apparaissant dans le plus grand nombre de films.

```
filmsdb> var mapActorCount = function () {
...   if (this.actors && Array.isArray(this.actors)) {
...     this.actors.forEach(function (a) {
...       var actorKey = a.first_name + " " + a.last_name + "|" + (a.
birth_date ?? "NA");
...       emit(actorKey, 1);
...     });
...   }
... };
...
... var reduceActorCount = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapActorCount,
...   reduceActorCount,
...   { out: "movies_by_actor" }
... );
...
{ result: 'movies_by_actor', ok: 1 }
filmsdb>
```

Explication :

La première opération MapReduce compte le nombre de films pour chaque acteur.
La seconde compare ces résultats afin d'identifier l'acteur apparaissant dans le plus grand nombre de films.

13. Classer les films par lettre de grade majoritaire (A , B , C , etc.)



Explication :

La phase *Map* transforme chaque note d'un film en lettre (A–F) et l'émet. La phase *Reduce* compte les lettres et conserve celle qui apparaît le plus souvent (grade majoritaire). Ensuite, on regroupe les films par cette lettre pour obtenir le classement global.

14. Calculer la note moyenne par année de sortie des films

Explication :

La phase *Map* calcule la somme et le nombre de notes de chaque film et les associe à son année de sortie.
La phase *Reduce* regroupe ces valeurs par année, puis la fonction *Finalize* calcule la moyenne des notes pour chaque année.