Introduction à MapeReduce avec MongoDB

Prenez la collection de films utilisée lors du TP n°1. L'objectif de cet exercice est de vous familiariser avec l'écriture de fonctions MapReduce dans MongoDB. N'oubliez pas de publier votre travail dans votre dépôt Git.

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

```
sample_mflix> var map1 = function () {
...   emit("total", 1);
... };
...
... var reduce1 = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   map1,
...   reduce1,
...   { out: "r1_nb_films" }
... );
...
... db.r1_nb_films.find();
...
[ { _id: 'total', value: 278 } ]
sample_mflix>
```

   ⇨ la fonction map émet la paire clé-valeur ("total", 1) pour chaque film, et la fonction reduce applique Array.sum sur toutes les valeurs de la clé "total" pour obtenir le nombre total de films.

2. Compter le **nombre de films par genre**.

```
sample_mflix> var mapGenre = function () {
...   emit(this.genre, 1);
... };
...
... var reduceGenre = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapGenre,
...   reduceGenre,
...   { out: "r_genre_count" }
... );
...
... db.r_genre_count.find();
...
[
  { _id: 'Mystère', value: 6 },
  { _id: 'Western', value: 3 },
  { _id: 'Crime', value: 29 },
  { _id: 'Drama', value: 14 },
  { _id: 'Thriller', value: 10 },
  { _id: 'Mystery', value: 1 },
  { _id: 'Action', value: 36 },
  { _id: 'Science Fiction', value: 1 },
  { _id: 'Science-Fiction', value: 9 },
  { _id: 'Guerre', value: 1 },
  { _id: 'Musique', value: 2 },
  { _id: 'War', value: 1 },
  { _id: 'Fantasy', value: 2 },
  { _id: 'Adventure', value: 3 },
  { _id: 'Histoire', value: 1 },
  { _id: 'Fantastique', value: 4 },
  { _id: 'Horreur', value: 8 },
  { _id: 'Comédie', value: 25 },
  { _id: 'Aventure', value: 22 },
  { _id: 'Comedy', value: 1 }
]
Type "it" for more
```

⇨ Explication : la fonction map émet 1 avec comme clé le genre du film, la reduce applique Array.sum pour obtenir le nombre de films par genre.

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

```
sample_mflix> var mapReal = function () {
...   if (this.director) {
...     emit(this.director.first_name + " " + this.director.last_name, 1);
...   }
... };
...
... var reduceReal = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapReal,
...   reduceReal,
...   { out: "r_realisateur_count" }
... );
...
... db.r_realisateur_count.find();
...
[
  { _id: 'Sam Raimi', value: 3 },
  { _id: 'André Téchiné', value: 1 },
  { _id: 'Nicolas Silhol', value: 1 },
  { _id: 'Renny Harlin', value: 1 },
  { _id: 'Bryan Singer', value: 1 },
  { _id: 'Henri-Georges Clouzot', value: 1 },
  { _id: 'Jonathan Demme', value: 1 },
  { _id: 'Jean-Luc Godard', value: 3 },
  { _id: 'John Cassavetes', value: 3 },
  { _id: 'François Ozon', value: 1 },
  { _id: 'John Huston', value: 1 },
  { _id: 'Andrey Zvyagintsev', value: 1 },
  { _id: 'Akira Kurosawa', value: 3 },
  { _id: 'Thomas Lilti', value: 2 },
  { _id: 'Georges Lautner', value: 1 },
  { _id: 'Len Wiseman', value: 1 },
  { _id: 'David Cronenberg', value: 4 },
  { _id: 'Mikhaël Hers', value: 1 },
  { _id: 'Alain Corneau', value: 1 },
  { _id: 'Charlie Chaplin', value: 4 }
]
Type "it" for more
```

⇨ Explication : la map utilise le nom complet du réalisateur comme clé et émet 1 pour chaque film, la reduce additionne pour donner le nombre de films réalisés par chaque personne.

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

```
sample_mflix> var mapActUniq = function () {
...   if (this.actors) {
...     this.actors.forEach(function (a) {
...       emit(a.first_name + " " + a.last_name, 1);
...     });
...   }
... };
...
... var reduceActUniq = function (key, values) {
...   return 1;
... };
...
... db.movies.mapReduce(
...   mapActUniq,
...   reduceActUniq,
...   { out: "r_acteurs_uniques" }
... );
...
... db.r_acteurs_uniques.count();
...
1210
sample_mflix>
```

⇨ Explication : chaque acteur est utilisé comme clé, la reduce renvoie toujours 1 pour éliminer les doublons, et le count() sur la collection résultat donne le nombre d'acteurs différents

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

```
sample_mflix> var mapYear = function () {
...   emit(this.year, 1);
... };
...
... var reduceYear = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapYear,
...   reduceYear,
...   { out: "r_films_par_annee" }
... );
...
... db.r_films_par_annee.find();
...
[
  { _id: 1966, value: 3 }, { _id: 1999, value: 6 },
  { _id: 1980, value: 8 }, { _id: 1988, value: 3 },
  { _id: 1998, value: 2 }, { _id: 1969, value: 1 },
  { _id: 1995, value: 7 }, { _id: 1971, value: 1 },
  { _id: 1941, value: 1 }, { _id: 1964, value: 1 },
  { _id: 1989, value: 2 }, { _id: 1963, value: 4 },
  { _id: 1993, value: 4 }, { _id: 1985, value: 2 },
  { _id: 1937, value: 1 }, { _id: 1973, value: 3 },
  { _id: 1972, value: 4 }, { _id: 2010, value: 6 },
  { _id: 2014, value: 2 }, { _id: 1960, value: 5 }
]
Type "it" for more
```

⇨ Explication : la clé est l'année (year) et la reduce calcule la somme des 1 pour obtenir le nombre de films sortis chaque année.

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

```
sample_mflix> var mapFilmAvg = function () {
...   if (this.grades && this.grades.length > 0) {
...     var sum = 0;
...     for (var i = 0; i < this.grades.length; i++) {
...       sum += this.grades[i].note;
...     }
...     emit(this.title, { sum: sum, count: this.grades.length });
...   }
... };
...
... var reduceFilmAvg = function (key, values) {
...   var res = { sum: 0, count: 0 };
...   values.forEach(function (v) {
...     res.sum += v.sum;
...     res.count += v.count;
...   });
...   return res;
... };
...
... var finalizeFilmAvg = function (key, reduced) {
...   return reduced.sum / reduced.count;
... };
...
... db.movies.mapReduce(
...   mapFilmAvg,
...   reduceFilmAvg,
...   { out: "r_note_moyenne_par_film", finalize: finalizeFilmAvg }
... );
...
... db.r_note_moyenne_par_film.find();
...
[
  { _id: 'Minority Report', value: 40 },
  { _id: 'The King of New York', value: 47.75 },
  { _id: 'Corporate', value: 61 },
  { _id: 'Batman Begins', value: 28.75 },
  { _id: 'Black book', value: 51.5 },
  { _id: 'Memento', value: 31.75 },
  { _id: 'Shining', value: 32.5 },
  { _id: 'Amadeus', value: 54.75 },
  { _id: 'Vice', value: 41 },
  { _id: 'Usual Suspects', value: 61.75 },
  { _id: "L'Homme de Rio", value: 24 },
  { _id: 'Atlantique', value: 75 },
  { _id: 'Eternal Sunshine of the Spotless Mind', value: 53.25 },
  { _id: 'Les Temps modernes', value: 55.75 },
  { _id: 'Interstellar', value: 50 },
  { _id: "Kagemusha, l'ombre du guerrier", value: 31.5 },
  { _id: 'On connaît la chanson', value: 58 },
```

⇨ Explication : la map calcule pour chaque film la somme des notes et le nombre de notes, la reduce additionne ces sommes si besoin, et la fonction finalize retourne la moyenne sum/count

7. Calculer la **note moyenne par genre**.

```
sample_mflix> var mapGenreAvg = function () {
...   if (this.grades && this.grades.length > 0 && this.genre) {
...     var sum = 0;
...     for (var i = 0; i < this.grades.length; i++) {
...       sum += this.grades[i].note;
...     }
...     emit(this.genre, { sum: sum, count: this.grades.length });
...   }
... };
...
... var reduceGenreAvg = function (key, values) {
...   var res = { sum: 0, count: 0 };
...   values.forEach(function (v) {
...     res.sum += v.sum;
...     res.count += v.count;
...   });
...   return res;
... };
...
... var finalizeGenreAvg = function (key, reduced) {
...   return reduced.sum / reduced.count;
... };
...
... db.movies.mapReduce(
...   mapGenreAvg,
...   reduceGenreAvg,
...   { out: "r_note_moyenne_par_genre", finalize: finalizeGenreAvg }
... );
...
... db.r_note_moyenne_par_genre.find();
...
[
  { _id: 'Crime', value: 53.060344827586206 },
  { _id: 'Mystère', value: 51.291666666666664 },
  { _id: 'Western', value: 47.833333333333336 },
  { _id: 'Drama', value: 45.910714285714285 },
  { _id: 'Drame', value: 50.375 },
  { _id: 'Thriller', value: 46.75 },
  { _id: 'Science Fiction', value: 42.75 },
  { _id: 'Action', value: 48.22222222222222 },
  { _id: 'Science-Fiction', value: 54.94444444444444 },
  { _id: 'Guerre', value: 69.5 },
  { _id: 'Mystery', value: 15.5 },
  { _id: 'Musique', value: 49.75 },
  { _id: 'Fantasy', value: 52.5 },
  { _id: 'War', value: 70 },
```

⇨ Explication : chaque film contribue à son genre avec la somme de ses notes et leur nombre, la reduce fusionne ces valeurs, puis finalize calcule la moyenne globale par genre.

8. Calculer la **note moyenne par réalisateur**.

```
sample_mflix> var mapRealAvg = function () {
...   if (this.grades && this.grades.length > 0 && this.director) {
...     var sum = 0;
...     for (var i = 0; i < this.grades.length; i++) {
...       sum += this.grades[i].note;
...     }
...     var key = this.director.first_name + " " + this.director.last_name;
...     emit(key, { sum: sum, count: this.grades.length });
...   }
... };
...
... var reduceRealAvg = function (key, values) {
...   var res = { sum: 0, count: 0 };
...   values.forEach(function (v) {
...     res.sum += v.sum;
...     res.count += v.count;
...   });
...   return res;
... };
...
... var finalizeRealAvg = function (key, reduced) {
...   return reduced.sum / reduced.count;
... };
...
... db.movies.mapReduce(
...   mapRealAvg,
...   reduceRealAvg,
...   { out: "r_note_moyenne_par_realisateur", finalize: finalizeRealAvg }
... );
...
... db.r_note_moyenne_par_realisateur.find();
...
[
  { _id: 'Sam Raimi', value: 51.833333333333336 },
  { _id: 'André Téchiné', value: 37.5 },
  { _id: 'Nicolas Silhol', value: 61 },
  { _id: 'Renny Harlin', value: 31.75 },
  { _id: 'Bryan Singer', value: 61.75 },
  { _id: 'Henri-Georges Clouzot', value: 47.5 },
  { _id: 'Jonathan Demme', value: 45.75 },
  { _id: 'Jean-Luc Godard', value: 41.25 },
  { _id: 'John Cassavetes', value: 66.33333333333333 },
  { _id: 'François Ozon', value: 62.5 },
  { _id: 'John Huston', value: 66.5 },
  { _id: 'Andrey Zvyagintsev', value: 59 },
  { _id: 'Akira Kurosawa', value: 35.916666666666664 },
  { _id: 'Thomas Lilti', value: 36 },
```

⇨ Explication : on regroupe les films par réalisateur avec sum et count, la reduce cumule sur tous ses films, et finalize donne la moyenne des notes par réalisateur.

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

```
sample_mflix> var mapMaxFilm = function () {
...   if (this.grades && this.grades.length > 0) {
...     var maxNote = this.grades[0].note;
...     for (var i = 1; i < this.grades.length; i++) {
...       if (this.grades[i].note > maxNote) {
...         maxNote = this.grades[i].note;
...       }
...     }
...     emit(1, { title: this.title, max: maxNote });
...   }
... };
...
... var reduceMaxFilm = function (key, values) {
...   var best = values[0];
...   values.forEach(function (v) {
...     if (v.max > best.max) best = v;
...   });
...   return best;
... };
...
... db.movies.mapReduce(
...   mapMaxFilm,
...   reduceMaxFilm,
...   { out: "r_film_max_note" }
... );
...
... db.r_film_max_note.find();
...
[ { _id: 1, value: { title: 'Star Wars, épisode IX', max: 100 } } ]
```

⇨ Explication : chaque film envoie sa meilleure note avec son titre, regroupé sous la même clé 1, puis la reduce ne garde que l'objet ayant la note maximale

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

```
sample_mflix> var mapGt70 = function () {
...   if (this.grades) {
...     this.grades.forEach(function (g) {
...       if (g.note > 70) {
...         emit("gt70", 1);
...       }
...     });
...   }
... };
...
... var reduceGt70 = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapGt70,
...   reduceGt70,
...   { out: "r_notes_gt70" }
... );
...
... db.r_notes_gt70.find();
...
[ { _id: 'gt70', value: 317 } ]
sample_mflix>
```

⇨ Explication : la map émet 1 pour chaque note strictement supérieure à 70, la reduce additionne tous ces 1 pour donner le nombre total

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

```
sample_mflix> var mapActGenre = function () {
...   if (this.genre && this.actors) {
...     this.actors.forEach(function (a) {
...       var name = a.first_name + " " + a.last_name;
...       emit(this.genre, name);
...     }, this);
...   }
... };
...
... var reduceActGenre = function (key, values) {
...   var uniq = {};
...   values.forEach(function (v) { uniq[v] = true; });
...   return Object.keys(uniq);
... };
...
... var finalizeActGenre = function (key, reduced) {
...   return reduced;
... };
...
... db.movies.mapReduce(
...   mapActGenre,
...   reduceActGenre,
...   { out: "r_acteurs_par_genre", finalize: finalizeActGenre }
... );
...
... db.r_acteurs_par_genre.find();
...
[
  {
    _id: 'Drame',
    value: [
      'Jonas Bloquet',
      'Virginie Efira',
      'Laurent Lafitte',
      'Charles Berling',
      'Judith Magre',
      'Isabelle Huppert',
      'Christian Berkel',
      'Anne Consigny',
      'Park So-dam',
      'Choi Woo-shik',
      'Cho Yeo-jeong',
      'Lee Sun-kyun',
      'Song Kang-ho',
      'Luàna Bajrami',
      'Noémie Merlant'
```

⇨ Explication : la map associe chaque acteur aux genres de ses films, la reduce construit un ensemble pour supprimer les doublons, et la valeur finale est la liste unique d'acteurs pour chaque genre.

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

B C

```
sample_mflix> var mapActCount = function () {
...   if (this.actors) {
...     this.actors.forEach(function (a) {
...       var name = a.first_name + " " + a.last_name;
...       emit(name, 1);
...     });
...   }
... };
...
... var reduceActCount = function (key, values) {
...   return Array.sum(values);
... };
...
... db.movies.mapReduce(
...   mapActCount,
...   reduceActCount,
...   { out: "r_films_par_acteur" }
... );
...
... // top acteurs
... db.r_films_par_acteur.find().sort({ value: -1 }).limit(10);
...
[
  { _id: 'Harrison Ford', value: 13 },
  { _id: 'Robert De Niro', value: 8 },
  { _id: 'Kirsten Dunst', value: 7 },
  { _id: 'Al Pacino', value: 7 },
  { _id: 'Viggo Mortensen', value: 6 },
  { _id: 'Samuel L. Jackson', value: 6 },
  { _id: 'Carrie Fisher', value: 6 },
  { _id: 'Brad Pitt', value: 5 },
  { _id: 'Vincent Lacoste', value: 5 },
  { _id: 'Catherine Deneuve', value: 5 }
]
sample_mflix>
```

⇨ Explication : la map émet 1 par film pour chaque acteur, la reduce somme ces valeurs pour obtenir le nombre de films par acteur, puis un tri décroissant permet de trouver ceux qui apparaissent le plus.

13. Classer les films par **lettre de grade majoritaire** (`A`)

14.

```
sample_mflix> var mapGradeMaj = function () {
...   if (this.grades && this.grades.length > 0) {
...     var counts = {};
...     this.grades.forEach(function (g) {
...       counts[g.grade] = (counts[g.grade] || 0) + 1;
...     });
...     var best = null;
...     var bestCount = -1;
...     for (var gr in counts) {
...       if (counts[gr] > bestCount) {
...         best = gr;
...         bestCount = counts[gr];
...       }
...     }
...     emit(this.title, best);
...   }
... };
...
... var reduceGradeMaj = function (key, values) {
...   return values[0];
... };
...
... db.movies.mapReduce(
...   mapGradeMaj,
...   reduceGradeMaj,
...   { out: "r_grade_majoritaire" }
... );
...
... db.r_grade_majoritaire.find();
...
[
  { _id: 'Minority Report', value: 'F' },
  { _id: 'The King of New York', value: 'D' },
  { _id: 'Corporate', value: 'B' },
  { _id: 'Batman Begins', value: 'A' },
  { _id: 'Black book', value: 'A' },
  { _id: 'Memento', value: 'A' },
  { _id: 'Shining', value: 'F' },
  { _id: 'Amadeus', value: 'C' },
  { _id: 'Vice', value: 'E' },
  { _id: 'Usual Suspects', value: 'B' },
  { _id: "L'Homme de Rio", value: 'E' },
  { _id: 'Atlantique', value: 'D' },
  { _id: 'Eternal Sunshine of the Spotless Mind', value: 'A' },
  { _id: 'Les Temps modernes', value: 'D' },
  { _id: 'Interstellar', value: 'A' },
  { _id: "Kagemusha, l'ombre du guerrier", value: 'B' },
```

15.

16. Calculer la **note moyenne par année** de sortie des films.

```
sample_mflix> var mapYearAvg = function () {
...   if (this.year && this.grades && this.grades.length > 0) {
...     var sum = 0;
...     for (var i = 0; i < this.grades.length; i++) {
...       sum += this.grades[i].note;
...     }
...     emit(this.year, { sum: sum, count: this.grades.length });
...   }
... };
...
... var reduceYearAvg = function (key, values) {
...   var res = { sum: 0, count: 0 };
...   values.forEach(function (v) {
...     res.sum += v.sum;
...     res.count += v.count;
...   });
...   return res;
... };
...
... var finalizeYearAvg = function (key, reduced) {
...   return reduced.sum / reduced.count;
... };
...
... db.movies.mapReduce(
...   mapYearAvg,
...   reduceYearAvg,
...   { out: "r_note_moyenne_par_annee", finalize: finalizeYearAvg }
... );
...
... db.r_note_moyenne_par_annee.find();
...
[
  { _id: 1996, value: 62.1875 },
  { _id: 1956, value: 57.375 },
  { _id: 1981, value: 65.83333333333333 },
  { _id: 1952, value: 38.875 },
  { _id: 1974, value: 45.3125 },
  { _id: 1958, value: 51.125 },
  { _id: 1990, value: 48.92857142857143 },
  { _id: 2012, value: 50.25 },
  { _id: 2013, value: 47.916666666666664 },
  { _id: 2017, value: 49.861111111111114 },
  { _id: 1970, value: 66.25 },
```

Explication : on regroupe les films par year avec somme et nombre de notes, la reduce aggrège les contributions, puis finalize divise la somme totale par le nombre total de notes pour chaque année

17. Identifier les **réalisateurs dont la note moyenne de tous leurs films est supérieure à 80**.

```
sample_mflix> var mapReal80 = function () {
...   if (this.director && this.grades && this.grades.length > 0) {
...     var sum = 0;
...     for (var i = 0; i < this.grades.length; i++) {
...       sum += this.grades[i].note;
...     }
...     var key = this.director.first_name + " " + this.director.last_name;
...     emit(key, { sum: sum, count: this.grades.length });
...   }
... };
...
... var reduceReal80 = function (key, values) {
...   var res = { sum: 0, count: 0 };
...   values.forEach(function (v) {
...     res.sum += v.sum;
...     res.count += v.count;
...   });
...   return res;
... };
...
... var finalizeReal80 = function (key, reduced) {
...   return reduced.sum / reduced.count;
... };
...
... db.movies.mapReduce(
...   mapReal80,
...   reduceReal80,
...   { out: "r_note_real_moyenne", finalize: finalizeReal80 }
... );
...
... db.r_note_real_moyenne.find({ value: { $gt: 80 } });
...

sample_mflix>
```

Explication : cette map/reduce calcule d'abord la moyenne des notes pour chaque réalisateur, puis une requête find sur la collection résultat conserve uniquement ceux dont la moyenne est strictement supérieure à 80