TP4 :

Présentation CouchDB :

CouchDB, base de données orientée document. Un système facile à installer, à utiliser et à appréhender. Il expose ses fonctionnalités sous forme d'une API Rest: GET, PUT, POST, DELETE.

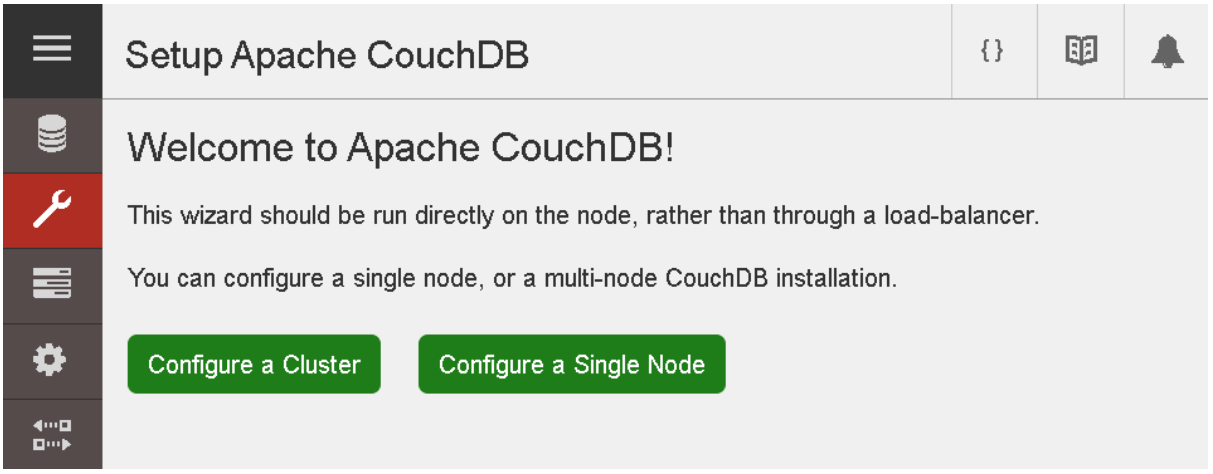
Pour lancer couchDB via Docker :

```
PS C:\Users\Fei3> docker run -d --name couchdbdemo -e COUCHDB_USER=fatimazah
ra -e COUCHDB_PASSWORD=fatimazahra -p 5984:5984 couchdb
Unable to find image 'couchdb:latest' locally
latest: Pulling from library/couchdb
51b27282aaec: Pull complete
ae4ce04d0e1c: Pull complete
f6c9c88e767d: Pull complete
8856ade7c27a: Pull complete
2897b671832f: Pull complete
7f82cbc9af1f: Pull complete
cd726490d7d0: Pull complete
716c80c273f8: Pull complete
4b62383183a1: Pull complete
87d5a41c1790: Pull complete
609e5bdfa7b1: Pull complete
Digest: sha256:a2c8839c86304962ae60df41c9aa51907925b7ca022b69acfcd45663a89da
a77
Status: Downloaded newer image for couchdb:latest
059b8897de85415b183dc6271bec9ac8b099ac1728d412201487502f96f88f98
```

On vérifie que le conteneur est en cours d'exécution :

```
PS C:\Users\Fei3> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS
     PORTS                                       NAMES
059b8897de85   couchdb   "tini -- /docker-ent…"   2 minutes ago   Up 2 minut
es   0.0.0.0:5984->5984/tcp, [::]:5984->5984/tcp   couchdbdemo
c59bc5cfaebc   mongo     "docker-entrypoint.s…"   2 weeks ago     Up 2 hours
     27017/tcp                                    monMongoDB
```

CouchDB propose un client graphique accessible via :
http://localhost:5984/_utils



On utilise un client pour mettre en évidence l'api REST :

```
PS C:\Users\Fei3> curl.exe -X GET http://fatimazahra:fatimazahra@localhost:5
984/
{"couchdb":"Welcome","version":"3.5.1","git_sha":"44f6a43d8","uuid":"3b6d347
559fc8163ede880c700034005","features":["access-ready","partitioned","pluggab
le-storage-engines","reshard","scheduler"],"vendor":{"name":"The Apache Soft
ware Foundation"}}
```

Pour créer une base de donnée films :

```
PS C:\Users\Fei3> curl.exe -X PUT http://fatimazahra:fatimazahra@localhost:5
984/films
{"ok":true}
```

Pour récupérer la base :

```
PS C:\Users\Fei3> curl.exe -X GET http://fatimazahra:fatimazahra@localhost:5
984/films
{"instance_start_time":"1765713489","db_name":"films","purge_seq":"0-g1AAAAB
PeJzLYWBgYMpgTmHgzcvPy09JdcjLz8gvLskBCeexAEmGBiD1HwiyEhlwqEtkSKqHKMgCAIT2GV4
","update_seq":"0-g1AAAACLeJzLYWBgYMpgTmHgzcvPy09JdcjLz8gvLskBCeexAEmGBiD1Hw
iyMpgTGXKBAuymSanJFhZJ6HpwmJLIkFSPot0s2cDYxCQFXXEWABoYKrM","sizes":{"file":1
6692,"external":0,"active":0},"props":{},"doc_del_count":0,"doc_count":0,"di
sk_format_version":8,"compact_running":false,"cluster":{"q":2,"n":1,"w":1,"r
":1}}
```

Pour insérer un nouveau document d dans la base films :

```
PS C:\Users\Fei3> curl.exe -X PUT http://fatimazahra:fatimazahra@localhost:5
984/films -d '{"nom":"malo"}'
```

On peut aussi insérer un document dans un fichier JSON movie:10098.json :

```
PS C:\Users\Fei3> curl.exe -X POST http://fatimazahra:fatimazahra@localhost:
5984/exemple -d @movie:10098.json -H "Content-Type: application/json"
```

Remarque **:** bien que CouchDB puisse générer automatiquement des identifiants, cette pratique est déconseillée, car les identifiants produits sont peu lisibles et rendent l'exploitation des données plus complexe.

Pour insérer une collection de documents :

```
PS C:\Users\Fei3\downloads> curl.exe -X POST http://fatimazahra:fatimazahra@
localhost:5984/films/_bulk_docs `
>>   -H "Content-Type: application/json" `
>>   --data-binary "@films_couchdb.json"
```

Pour récupérer un document sachant son id :

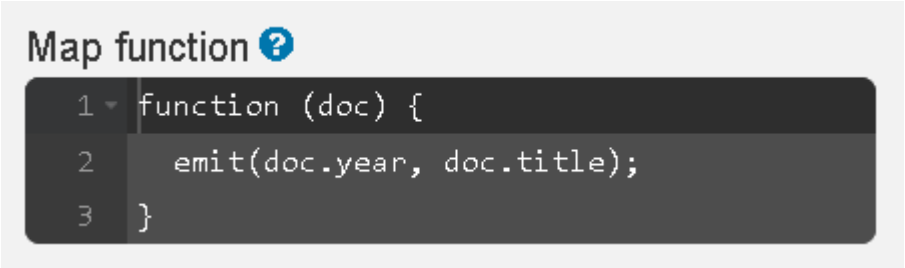
```
PS C:\Users\Fei3\downloads> curl.exe -X GET http://fatimazahra:fatimazahra@l
ocalhost:5984/films/movie:10098
{"_id":"movie:10098","_rev":"1-a9ad1a0a8ec461bac3ce06dee5b3e6a4","title":"Le
 Kid","year":1921,"genre":"Comédie","summary":"Un pauvre vitrier recueille u
n enfant abandonné par sa mère victime d'un séducteur. L'enfant casse des ca
rreaux pour aider son père adoptif, qui l'arrache à des dames patronnesses,
puis le rend à sa mere, devenue riche.","country":"US","director":{"_id":"ar
tist:13848","last_name":"Chaplin","first_name":"Charlie","birth_date":1889},
"actors":[{"last_name":"Chaplin","first_name":"Charlie","birth_date":1889},{
"last_name":"Coogan","first_name":"Jackie","birth_date":1914},{"last_name":"
Purviance","first_name":"Edna","birth_date":1895}]}
```

MapReduce avec CouchDB :

CouchDB est un système libre de droits (sous licence Apache) et qui propose un moteur d'exécution MapReduce. Les fonctions Map et Reduce sont exprimées en JS. Il permet de définir uniquement la fonction Map et puis la fonction Reduce contrairement à d'autres systèmes. Il permet aussi de les sauvegarder donc on peut voir les résultats intermédiaires.

Calculer le nombre de films par année :

La fonction Map :



	id	key	value
	movie:10098	1921	Le Kid
	movie:631	1927	L'Aurore
	movie:832	1931	M le maudit
	movie:3082	1936	Les Temps modernes
	movie:43884	1936	La Charge de la bri…
	movie:777	1937	La Grande Illusion
	movie:223	1940	Rebecca
	movie:596	1940	The Grapes of Wrath
	movie:914	1940	Le Dictateur
	movie:11462	1941	Soupçons
	movie:289	1942	Casablanca
	movie:29084	1943	Le Corbeau
	movie:996	1944	Assurance sur la mort

La phase sort est déjà faite ici.
La fonction Reduce :

Custom Reduce function

```
1 function (keys, values) {
2   return values.length;
3 }
```

key	value
1921	1
1927	1
1931	1
1936	2
1937	1
1940	2
1941	1
1942	1
1943	1
1944	1
1946	1
1947	1
1948	1
1949	1

Nombre de films pour chaque auteur :

Map function

```
function (doc)
{
  for(i=0; i<doc.actors.length; i++){
    emit({"prénom": doc.actors[i].first_n
  }
}
```

Reduce (optional)

CUSTOM

Custom Reduce function

```
function (keys, values) {
  return values.length;
}
```

	key	value
	{ "prénom": "A.J.", "n…	1
	{ "prénom": "Aaron", …	1
	{ "prénom": "Abdelg…	1
	{ "prénom": "Abigail"…	1
	{ "prénom": "Adam", …	2
	{ "prénom": "Adam", …	1
	{ "prénom": "Adel", "…	1
	{ "prénom": "Adèle", …	1
	{ "prénom": "Adolfo",…	1
	{ "prénom": "Adolph…	1
	{ "prénom": "Adrian",…	1
	{ "prénom": "Adrien",…	1
	{ "prénom": "Agnès",…	1
	{ "prénom": "Ahmed"…	1

MapReduce avec MongoDB :

Ce TP s'inscrit dans une démarche de découverte des traitements distribués proposés par MongoDB, à travers l'utilisation du modèle MapReduce. Les données exploitées proviennent de la collection de films étudiée lors d'un précédent TP. Chaque exercice vise à concevoir une chaîne de traitement reposant sur des fonctions de transformation et d'agrégation des données, avec, si nécessaire, une étape supplémentaire de finalisation.

MapReduce :

Compter le nombre total de films :

Map :

```
lesfilms> var mapFunction = function () {
...     emit("nombre_total_films", 1);
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...     return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "nb_films" }
... );
...
... db.nb_films.find();
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation
 instead.
See https://mongodb.com/docs/manual/core/map-reduce for details.
[ { _id: 'nombre_total_films', value: 278 } ]
```

Compter le nombre de films par genre :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.genre) {
...     if (Array.isArray(this.genre)) {
...       this.genre.forEach(function (g) {
...         emit(g, 1);
...       });
...     } else {
...       emit(this.genre, 1);
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.nb_films_par_genre.find()
[
  { _id: 'Guerre', value: 1 },
  { _id: 'Crime', value: 29 },
  { _id: 'Action', value: 36 },
  { _id: 'Drame', value: 96 },
  { _id: 'Fantasy', value: 2 },
  { _id: 'War', value: 1 },
  { _id: 'Comédie', value: 25 },
  { _id: 'Drama', value: 14 },
  { _id: 'Fantastique', value: 4 },
  { _id: 'Histoire', value: 1 },
  { _id: 'Science-Fiction', value: 9 },
  { _id: 'Romance', value: 3 },
  { _id: 'Mystery', value: 1 },
  { _id: 'Aventure', value: 22 },
  { _id: 'Mystère', value: 6 },
  { _id: 'Comedy', value: 1 },
  { _id: 'Adventure', value: 3 },
  { _id: 'Science Fiction', value: 1 },
  { _id: 'Thriller', value: 10 },
  { _id: 'Horreur', value: 8 }
]
Type "it" for more
```

Compter les films par réalisateur :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.director) {
...     emit(this.director, 1);
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "nb_films_par_realisateur" }
... );
...
{ result: 'nb_films_par_realisateur', ok: 1 }
```

Extrait :

```
lesfilms> db.nb_films_par_realisateur.find()
[
  {
    _id: {
      _id: 'artist:5231',
      last_name: 'Petersen',
      first_name: 'Wolfgang',
      birth_date: 1941
    },
    value: 1
  },
  {
    _id: {
      _id: 'artist:138',
      last_name: 'Tarantino',
      first_name: 'Quentin',
      birth_date: 1963
    },
    value: 7
  },
  {
    _id: {
      _id: 'artist:24882',
      last_name: 'Demy',
      first_name: 'Jacques',
      birth_date: 1931
    },
    value: 1
  },
  {
    _id: {
      _id: 'artist:6431',
      last_name: 'Aronofsky',
      first_name: 'Darren',
      birth_date: 1969
    },
    value: 1
  },
```

Compter les acteurs uniques :

Map :

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

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return 1;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "acteurs_uniques" }
... );
...
{ result: 'acteurs_uniques', ok: 1 }
```

```
lesfilms> db.acteurs_uniques.countDocuments()
1210
```

Films par annee de sortie :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.year) {
...     emit(this.year, 1);
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "nb_films_par_annee" }
... );
...
{ result: 'nb_films_par_annee', ok: 1 }
```

```
lesfilms> db.nb_films_par_annee.find().sort({ _id: 1 })
[
  { _id: 1921, value: 1 }, { _id: 1927, value: 1 },
  { _id: 1931, value: 1 }, { _id: 1936, value: 2 },
  { _id: 1937, value: 1 }, { _id: 1940, value: 3 },
  { _id: 1941, value: 1 }, { _id: 1942, value: 1 },
  { _id: 1943, value: 1 }, { _id: 1944, value: 1 },
  { _id: 1946, value: 2 }, { _id: 1947, value: 1 },
  { _id: 1948, value: 1 }, { _id: 1949, value: 1 },
  { _id: 1950, value: 2 }, { _id: 1951, value: 1 },
  { _id: 1952, value: 2 }, { _id: 1954, value: 2 },
  { _id: 1955, value: 1 }, { _id: 1956, value: 2 }
]
Type "it" for more
```

Note moyenne par film à partir des grades :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.grades && this.grades.length > 0) {
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         sum += g.note;
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       emit(this.title, { total: sum, count: count });
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = { total: 0, count: 0 };
...
...   values.forEach(function (v) {
...     result.total += v.total;
...     result.count += v.count;
...   });
...
...   return result;
... };
```

Finalize :

```
lesfilms> var finalizeFunction = function (key, reducedValue) {
...   if (reducedValue.count === 0) {
...     return null;
...   }
...   return reducedValue.total / reducedValue.count;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "note_moyenne_par_film",
...     finalize: finalizeFunction
...   }
... );
...
{ result: 'note_moyenne_par_film', ok: 1 }
```

```
lesfilms> db.note_moyenne_par_film.find()
[
  { _id: 'Le Pianiste', value: 17 },
  { _id: "Claire's Knee", value: 60.25 },
  { _id: 'Minuit à Paris', value: 50.5 },
  { _id: 'Rocco and His Brothers', value: 43 },
  { _id: 'Usual Suspects', value: 61.75 },
  { _id: 'Django Unchained', value: 33.5 },
  { _id: 'Amadeus', value: 54.75 },
  { _id: 'Le Dernier des mohicans', value: 73.25 },
  { _id: 'Eternal Sunshine of the Spotless Mind', value: 53.25 },
  { _id: 'The Social Network', value: 63.75 },
  { _id: 'Stalker', value: 52.25 },
  { _id: 'The Bridges of Madison County', value: 37.75 },
  { _id: 'Harry, un ami qui vous veut du bien', value: 42 },
  { _id: 'Belle de jour', value: 56 },
  { _id: "Les Aventuriers de l'arche perdue", value: 84.75 },
  { _id: 'Le Retour du Jedi', value: 57.5 },
  { _id: 'Parasite', value: 28.75 },
  { _id: 'Witness', value: 35.25 },
  { _id: "L'homme qui en savait trop", value: 47 },
  { _id: 'Collatéral', value: 46.75 }
]
Type "it" for more
```

La fonction map calcule pour chaque film la somme des notes ainsi que le nombre d'évaluations associées. La fonction reduce agrège ces informations, tandis que la fonction finalize permet de déterminer la note moyenne finale pour chaque film.

Note moyenne par genre :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.grades && this.grades.length > 0 && this.genre) {
...
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         sum += g.note;
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       if (Array.isArray(this.genre)) {
...         this.genre.forEach(function (gen) {
...           emit(gen, { total: sum, count: count });
...         });
...       } else {
...         emit(this.genre, { total: sum, count: count });
...       }
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = { total: 0, count: 0 };
...
...   values.forEach(function (v) {
...     result.total += v.total;
...     result.count += v.count;
...   });
...
...   return result;
... };
```

Finalize :

```
lesfilms> var finalizeFunction = function (key, reducedValue) {
...   if (reducedValue.count === 0) {
...     return null;
...   }
...   return reducedValue.total / reducedValue.count;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "note_moyenne_par_genre",
...     finalize: finalizeFunction
...   }
... );
...
{ result: 'note_moyenne_par_genre', ok: 1 }
```

```
lesfilms> db.note_moyenne_par_genre.find().sort({ value: -1 })
[
  { _id: 'Histoire', value: 83.75 },
  { _id: 'War', value: 70 },
  { _id: 'Guerre', value: 69.5 },
  { _id: 'Adventure', value: 62.9166666666666664 },
  { _id: 'Science-Fiction', value: 54.9444444444444 },
  { _id: 'Aventure', value: 54.10227272727273 },
  { _id: 'Crime', value: 53.0603448275862O6 },
  { _id: 'Fantasy', value: 52.5 },
  { _id: 'Mystère', value: 51.2916666666666664 },
  { _id: 'Comedy', value: 50.5 },
  { _id: 'Drame', value: 50.375 },
  { _id: 'Musique', value: 49.75 },
  { _id: 'Romance', value: 48.5 },
  { _id: 'Action', value: 48.22222222222222 },
  { _id: 'Western', value: 47.833333333333336 },
  { _id: 'Fantastique', value: 47 },
  { _id: 'Thriller', value: 46.75 },
  { _id: 'Drama', value: 45.910714285714285 },
  { _id: 'Comédie', value: 45.4 },
  { _id: 'Horreur', value: 45.34375 }
]
Type "it" for more
```

Pour chaque film, la fonction map calcule la somme des notes et le nombre d'évaluations associées, puis associe ces informations à chacun de ses genres. La fonction reduce agrège ces valeurs par genre, tandis que la fonction finalize permet de déterminer la note moyenne finale pour chaque genre.

Note moyenne par réalisateur :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.director && this.grades && this.grades.length > 0) {
...
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         sum += g.note;
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       emit(this.director.last_name + " " + this.director.first_name, {
...         total: sum,
...         count: count
...       });
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = { total: 0, count: 0 };
...
...   values.forEach(function (v) {
...     result.total += v.total;
...     result.count += v.count;
...   });
...
...   return result;
... };
```

Finalize :

```
lesfilms> var finalizeFunction = function (key, reducedValue) {
...   if (reducedValue.count === 0) {
...     return null;
...   }
...   return reducedValue.total / reducedValue.count;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "note_moyenne_par_realisateur",
...     finalize: finalizeFunction
...   }
... );
...
{ result: 'note_moyenne_par_realisateur', ok: 1 }
```

```
lesfilms> db.note_moyenne_par_realisateur.find().sort({ value: -1 })
[
  { _id: 'Wyler William', value: 79.25 },
  { _id: 'Petersen Wolfgang', value: 75.25 },
  { _id: 'Diop Mati', value: 75 },
  { _id: 'Davis Andrew', value: 73.75 },
  { _id: 'Melville Jean-Pierre', value: 72.91666666666667 },
  { _id: 'Lang Fritz', value: 68.25 },
  { _id: 'Laughton Charles', value: 68 },
  { _id: 'Leone Sergio', value: 68 },
  { _id: 'Gondry Michel', value: 67.625 },
  { _id: 'Huston John', value: 66.5 },
  { _id: 'Cassavetes John', value: 66.33333333333333 },
  { _id: 'Besson Luc', value: 65 },
  { _id: 'Lioret Philippe', value: 64.75 },
  { _id: 'de Peretti Thierry', value: 63.25 },
  { _id: 'Lean David', value: 63.25 },
  { _id: 'Jackson Peter', value: 62.916666666666664 },
  { _id: 'Pollack Sydney', value: 62.833333333333336 },
  { _id: 'Brizé Stéphane', value: 62.5 },
  { _id: 'Ozon François', value: 62.5 },
  { _id: 'Pariser Nicolas', value: 62.5 }
]
Type "it" for more
```

La fonction map calcule, pour chaque film, la somme des notes ainsi que le nombre d'évaluations, puis associe ces informations à son réalisateur. La fonction reduce agrège ces valeurs par réalisateur, tandis que la fonction finalize permet de déterminer la note moyenne finale pour chacun d'eux.

Film ayant la note max :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.grades && this.grades.length > 0) {
...
...     var max = null;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         if (max === null || g.note > max) {
...           max = g.note;
...         }
...       }
...     });
...
...     if (max !== null) {
...       emit("max_film", { title: this.title, note: max });
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var best = values[0];
...
...   values.forEach(function (v) {
...     if (v.note > best.note) {
...       best = v;
...     }
...   });
...
...   return best;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "film_note_max" }
... );
...
{ result: 'film_note_max', ok: 1 }
```

```
lesfilms> db.film_note_max.find()
[
  {
    _id: 'max_film',
    value: { title: 'Star Wars, épisode IX', note: 100 }
  }
]
```

Nombre de notes > 70 :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.grades && this.grades.length > 0) {
...
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined && g.note > 70) {
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       emit("notes_sup_70", count);
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...    mapFunction,
...    reduceFunction,
...    { out: "nb_notes_sup_70" }
... );
...
{ result: 'nb_notes_sup_70', ok: 1 }
```

```
lesfilms> db.nb_notes_sup_70.find()
[ { _id: 'notes_sup_70', value: 317 } ]
```

Acteurs pas genre :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.genre && this.actors) {
...
...     var genres = Array.isArray(this.genre) ? this.genre : [this.genre];
...
...     genres.forEach(function (g) {
...       this.actors.forEach(function (a) {
...         var actorName = a.last_name + " " + a.first_name;
...         emit(g, actorName);
...       });
...     }, this);
...   }
... };
```

Reduce (sans doublons) :

```
lesfilms> var reduceFunction = function (key, values) {
...   var unique = [];
...
...   values.forEach(function (v) {
...     if (unique.indexOf(v) === -1) {
...       unique.push(v);
...     }
...   });
...
...   return unique;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...    mapFunction,
...    reduceFunction,
...    { out: "acteurs_par_genre" }
... );
...
{ result: 'acteurs_par_genre', ok: 1 }
```

Extrait :

```
lesfilms> db.acteurs_par_genre.find()
[
  {
    _id: 'Guerre',
    value: [
      'Barbier Christian',
      'Mann Claude',
      'Crauchet Paul',
      'Cassel Jean-Pierre',
      'Ventura Lino',
      'Meurisse Paul',
      'Signoret Simone'
    ]
  },
```

Acteurs apparaissant dans le plus grand nombre de films :

Etape 1 :

Map :

```
lesfilms> var mapFunction = function () {
...    if (this.actors) {
...      this.actors.forEach(function (a) {
...        var actorName = a.last_name + " " + a.first_name;
...        emit(actorName, 1);
...      });
...    }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   return Array.sum(values);
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "nb_films_par_acteur" }
... );
...
{ result: 'nb_films_par_acteur', ok: 1 }
```

Etape 2 :

Map :

```
lesfilms> var mapMaxFunction = function () {
...   emit("max_acteur", {
...     actor: this._id,
...     count: this.value
...   });
... };
```

Reduce :

```
lesfilms> var reduceMaxFunction = function (key, values) {
...   var best = values[0];
...
...   values.forEach(function (v) {
...     if (v.count > best.count) {
...       best = v;
...     }
...   });
...
...   return best;
... };
```

Résultat :

```
lesfilms> db.nb_films_par_acteur.mapReduce(
...   mapMaxFunction,
...   reduceMaxFunction,
...   { out: "acteur_plus_films" }
... );
...
{ result: 'acteur_plus_films', ok: 1 }
```

```
lesfilms> db.acteur_plus_films.find()
[ { _id: 'max_acteur', value: { actor: 'Ford Harrison', count: 13 } } ]
```

Un premier traitement MapReduce permet de calculer le nombre de films par acteur. Un second traitement est ensuite utilisé afin d'identifier l'acteur apparaissant dans le plus grand nombre de films, en comparant l'ensemble des résultats obtenus.

Classement des films selon leur grade majoritaire :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.grades && this.grades.length > 0) {
...
...     var freq = {};
...
...     this.grades.forEach(function (g) {
...       if (g.grade) {
...         if (freq[g.grade] === undefined) {
...           freq[g.grade] = 1;
...         } else {
...           freq[g.grade]++;
...         }
...       }
...     });
...
...     var major = null;
...     var max = 0;
...
...     for (var grade in freq) {
...       if (freq[grade] > max) {
...         max = freq[grade];
...         major = grade;
...       }
...     }
...
...     if (major !== null) {
...       emit(major, this.title);
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = [];
...
...   values.forEach(function (v) {
...     if (Array.isArray(v)) {
...       result = result.concat(v);
...     } else {
...       result.push(v);
...     }
...   });
...
...   return result;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   { out: "films_par_grade_majoritaire" }
... );
...
{ result: 'films_par_grade_majoritaire', ok: 1 }
```

Extrait :

```
lesfilms> db.films_par_grade_majoritaire.find()
[
  {
    _id: 'D',
    value: [
      'Alien : Covenant',
      'Potiche',
      'Minuit à Paris',
      'Star Wars : Le Réveil de la Force',
      'Chambre 212',
      'Atlantique',
      'Parasite',
      'Amanda',
      'The Together Project',
      'Dunkirk',
      'Elle',
      'Manchester by the Sea',
      'La Charge de la brigade légère',
      'Midnight Special',
      'Incendies',
      'Opening Night',
      'Le sacrifice',
      'OSS 117 : Rio ne répond plus',
      'Sang pour sang',
      'Les Bronzés',
      'La Porte du paradis',
      'Cries and Whispers',
      'The King of New York',
      'Nikita',
      'Manhattan',
      'Mulholland Drive',
      "L'Homme qui voulut être roi",
```

Note moyenne par année :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.year && this.grades && this.grades.length > 0) {
...
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         sum += g.note;
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       emit(this.year, { total: sum, count: count });
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = { total: 0, count: 0 };
...
...   values.forEach(function (v) {
...     result.total += v.total;
...     result.count += v.count;
...   });
...
...   return result;
... };
```

Finalize :

```
lesfilms> var finalizeFunction = function (key, reducedValue) {
...   if (reducedValue.count === 0) {
...     return null;
...   }
...   return reducedValue.total / reducedValue.count;
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "note_moyenne_par_annee",
...     finalize: finalizeFunction
...   }
... );
...
{ result: 'note_moyenne_par_annee', ok: 1 }
```

```
lesfilms> db.note_moyenne_par_annee.find().sort({ _id: 1 })
[
  { _id: 1921, value: 51.25 },
  { _id: 1927, value: 33.5 },
  { _id: 1931, value: 68.25 },
  { _id: 1936, value: 58 },
  { _id: 1937, value: 46.5 },
  { _id: 1940, value: 47.083333333333336 },
  { _id: 1941, value: 59 },
  { _id: 1942, value: 24.75 },
  { _id: 1943, value: 47.5 },
  { _id: 1944, value: 55.75 },
  { _id: 1946, value: 51.375 },
  { _id: 1947, value: 47.5 },
  { _id: 1948, value: 30.75 },
  { _id: 1949, value: 23.5 },
  { _id: 1950, value: 55.875 },
  { _id: 1951, value: 57 },
  { _id: 1952, value: 38.875 },
  { _id: 1954, value: 40.25 },
  { _id: 1955, value: 68 },
  { _id: 1956, value: 57.375 }
]
Type "it" for more
```

Réalisateurs avec moyenne > 80 :

Map :

```
lesfilms> var mapFunction = function () {
...   if (this.director && this.grades && this.grades.length > 0) {
...
...     var sum = 0;
...     var count = 0;
...
...     this.grades.forEach(function (g) {
...       if (g.note !== undefined) {
...         sum += g.note;
...         count++;
...       }
...     });
...
...     if (count > 0) {
...       var directorName = this.director.last_name + " " + this.director.f
irst_name;
...       emit(directorName, { total: sum, count: count });
...     }
...   }
... };
```

Reduce :

```
lesfilms> var reduceFunction = function (key, values) {
...   var result = { total: 0, count: 0 };
...
...   values.forEach(function (v) {
...     result.total += v.total;
...     result.count += v.count;
...   });
...
...   return result;
... };
```

Finalize (filtre > 80) :

```
lesfilms> var finalizeFunction = function (key, reducedValue) {
...   if (reducedValue.count === 0) {
...     return null;
...   }
...
...   var avg = reducedValue.total / reducedValue.count;
...
...   if (avg > 80) {
...     return avg;
...   } else {
...     return null;
...   }
... };
```

Résultat :

```
lesfilms> db.films.mapReduce(
...   mapFunction,
...   reduceFunction,
...   {
...     out: "realisateurs_moyenne_sup_80",
...     finalize: finalizeFunction
...   }
... );
...
{ result: 'realisateurs_moyenne_sup_80', ok: 1 }
```