
TP4 MongoDB
Compte rendu du TP

Réalisé par :

Célian WIRTZ (celian.wirtz@edu.univ-paris13.fr)

TP4

Partie 1

1. Introduction

Bien que CouchDB ne soit pas le système NoSQL le plus utilisé comparé à MongoDB ou Cassandra, il présente plusieurs avantages notables :

- simplicité d'installation ;
- facilité d'utilisation ;
- compréhension rapide de son fonctionnement ;
- exposition des fonctionnalités via une **API REST**.

L'objectif de cette première partie est de présenter les concepts de base de CouchDB ainsi que les opérations essentielles telles que la création d'une base de données et l'insertion de documents.

2. Rappels sur l'API REST

CouchDB expose toutes ses fonctionnalités sous forme d'une **API REST**, basée sur le protocole HTTP.

Les principales méthodes HTTP utilisées sont :

- **GET** : permet de récupérer la représentation d'une ressource ;
 - **PUT** : permet de créer une ressource ou de la remplacer si elle existe déjà ;
 - **POST** : permet d'envoyer des données à une ressource (demande de service) ;
 - **DELETE** : permet de supprimer une ressource.
-

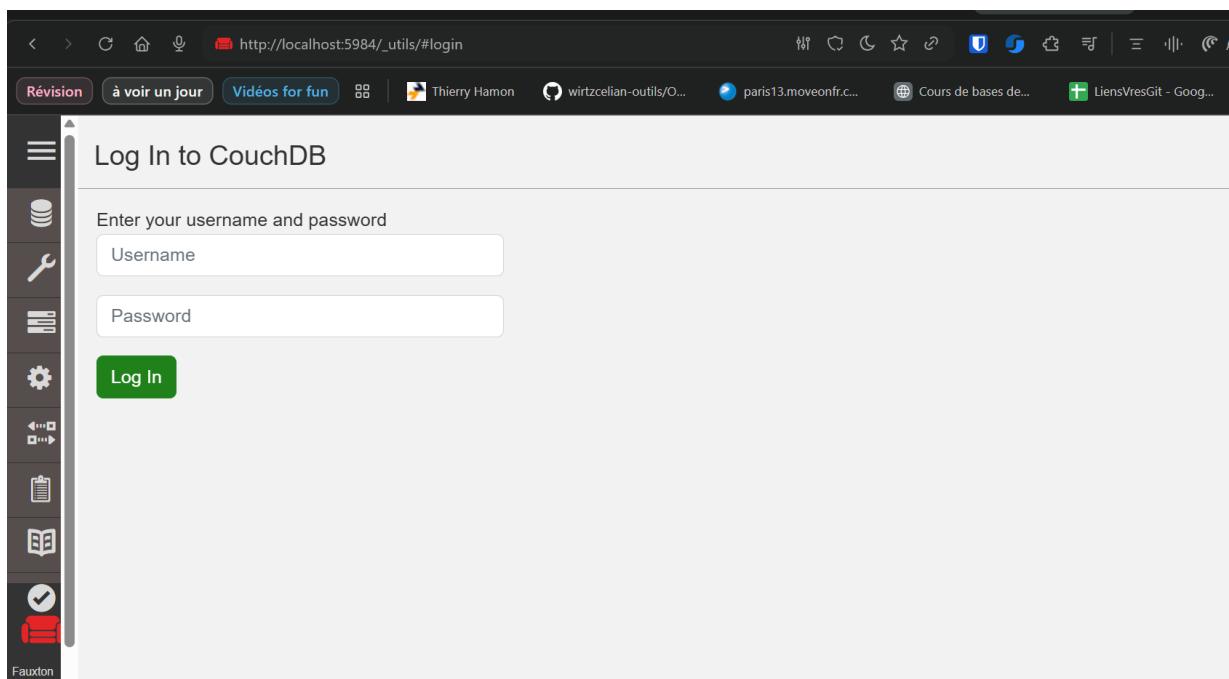
3. Accès à l'interface graphique

CouchDB propose une interface graphique intégrée accessible à l'adresse :

http://localhost:5984/_utils

Après authentification avec le nom d'utilisateur et le mot de passe définis précédemment, on accède à l'interface graphique permettant :

- la gestion des bases de données ;
- la visualisation des documents ;
- l'exploration des données sous forme JSON ou tabulaire.



4 Vérification du serveur CouchDB

`curl -X GET http://a:a@localhost:5984`

Cette requête permet de vérifier que CouchDB est bien lancé et accessible.

```
C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X GET "http://a:a@localhost:5984/"  
{"couchdb":"Welcome", "version":"3.5.1", "git_sha":"44f6a43d8", "uuid":"31a725e16d49376e8e22441268138ba3", "features":["acce  
ss-ready", "partitioned", "pluggable-storage-engines", "reshard", "scheduler"], "vendor":{"name":"The Apache Software Foundat  
ion"}}
```

5. Création d'une base de données

Pour créer une base de données nommée films, on utilise la méthode **PUT** :

```
curl -X PUT http://a:a@localhost:5984/films
```

Si la création réussit, CouchDB retourne une réponse JSON indiquant que l'opération s'est bien déroulée.

Pour consulter les informations de cette base :

```
curl -X GET http://a:a@localhost:5984/films
```

```
C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X PUT "http://a:a@localhost:5984/films"
{"ok":true}

C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X GET "http://a:a@localhost:5984/films"
{"instance_start_time": "1765767679", "db_name": "films", "purge_seq": "0-g1AAAABXeJzLYWBgYMPgTmEQTM4vTc5ISXIwNDLXMwBCwxyQVB4LkGRoAFL_gSArgzmRIRcowG6RZpBimGqJTR8e0xIZkupRjDG3sDBPTE7DpiELAPNtKI8", "update_seq": "0-g1AAAACbeJzLYWBgYMPgTmEQTM4vTc5ISXIwNDLXMwBCwxyQVB4LkGRoAFL_gSArgzmRIRcowG6RZpBimGqJTR8e0xIZkupRjDG3sDBPTE7DpiELAPNtKI8", "sizes": {"file": 16700, "external": 0, "active": 0}, "props": {}, "doc_del_count": 0, "doc_count": 0, "disk_format_version": 8, "compact_running": false, "cluster": {"q": 2, "n": 1, "w": 1, "r": 1}}
```

6. Insertion de documents

6.1 Insertion d'un document simple

CouchDB est une base de données **sans schéma imposé**, ce qui permet d'insérer des documents hétérogènes.

Exemple d'insertion d'un document avec un identifiant explicite :

```
C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X PUT "http://a:a@localhost:5984/films/doc" -H "Content-Type: application/json" -d "{\"cle\":\"valeur\"}"
{"ok":true,"id":"doc","rev":"1-0f3beea1048634b34d8091e22ab3c6fb"}

C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X PUT "http://a:a@localhost:5984/films/doc1" -H "Content-Type: application/json" -d "{\"nom\":\"cel\"}"
{"ok":true,"id":"doc1","rev":"1-5f4521106198983f3feb24896c573121"}
```

```
id "doc"
{
  "id": "doc",
  "key": "doc",
  "value": {
    "rev": "1-0f3beea1048634b34d8091e22ab3c6fb"
  },
  "doc": {
    "_id": "doc",
    "_rev": "1-0f3beea1048634b34d8091e22ab3c6fb",
    "cle": "valeur"
  }
}

id "doc1"
{
  "id": "doc1",
  "key": "doc1",
  "value": {
    "rev": "1-5f4521106198983f3feb24896c573121"
  },
  "doc": {
    "_id": "doc1",
    "_rev": "1-5f4521106198983f3feb24896c573121",
    "nom": "cel"
  }
}
```

Chaque document est stocké sous forme **clé–valeur**, avec un identifiant unique et un numéro de version (`_rev`).

⚠ Si un document avec le même identifiant existe déjà, CouchDB renvoie une **erreur de conflit**.

6.2 Identifiant automatique

Si aucun identifiant n'est fourni, CouchDB génère automatiquement un identifiant unique.

Ce comportement est généralement recommandé dans un contexte distribué, afin d'éviter les conflits liés à la génération manuelle des clés.

7. Insertion en masse (Bulk Insert)

Pour insérer un grand nombre de documents simultanément, CouchDB propose l'API `_bulk_docs`.

Exemple avec un fichier films_couchdb.json contenant une collection de films :

Après l'exécution, on constate que **278 documents** ont été importés dans la base.

```
C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X POST "http://a:a@localhost:5984/films/_bulk_docs" -H "Content-Type: application/json" --data-binary "@films_couchdb.json"
```

Name	Size	# of Docs	Partitioned	Actions
films	261.6 KB	278	No	

_id	actors	country	director	genre
movie:10098	[{"last_name": "Chaplin", "first_name": "Charlie"}]	US	{ "_id": "artist:13848", "last_name": "Chaplin", "first_name": "Charlie", "country": "US", "genre": "Comédie", "director": "Charlie Chaplin", "actors": [{"last_name": "Chaplin", "first_name": "Charlie"}]}	Comédie
movie:1018	[{"last_name": "Watts", "first_name": "Naomi"}]	FR	{ "_id": "artist:5602", "last_name": "Watts", "first_name": "Naomi", "country": "FR", "genre": "Thriller", "director": "Naomi Watts", "actors": [{"last_name": "Watts", "first_name": "Naomi"}]}	Thriller
movie:10238	[{"last_name": "Andersson", "first_name": "Ingmar"}]	SE	{ "_id": "artist:6648", "last_name": "Andersson", "first_name": "Ingmar", "country": "SE", "genre": "Drama", "director": "Ingmar Bergman", "actors": [{"last_name": "Andersson", "first_name": "Ingmar"}]}	Drama
movie:103	[{"last_name": "Brooks", "first_name": "Martin"}]	US	{ "_id": "artist:1032", "last_name": "Brooks", "first_name": "Martin", "country": "US", "genre": "Crime", "director": "Martin Scorsese", "actors": [{"last_name": "Brooks", "first_name": "Martin"}]}	Crime
movie:10362	[{"last_name": "Shaw", "first_name": "Audrey"}]	FR	{ "_id": "artist:20561", "last_name": "Shaw", "first_name": "Audrey", "country": "FR", "genre": "Drama", "director": "Audrey Hepburn", "actors": [{"last_name": "Shaw", "first_name": "Audrey"}]}	Drama
movie:103731	[{"last_name": "Witherspoon", "first_name": "Reese"}]	US	{ "_id": "artist:71872", "last_name": "Witherspoon", "first_name": "Reese", "country": "US", "genre": "Drame", "director": "Reese Witherspoon", "actors": [{"last_name": "Witherspoon", "first_name": "Reese"}]}	Drame
movie:106	[{"last_name": "Schwarzenegger", "first_name": "Arnold"}]	US	{ "_id": "artist:1090", "last_name": "Schwarzenegger", "first_name": "Arnold", "country": "US", "genre": "Science Fiction", "director": "Arnold Schwarzenegger", "actors": [{"last_name": "Schwarzenegger", "first_name": "Arnold"}]}	Science Fiction
movie:10669	[{"last_name": "Cox", "first_name": "Jane"}]	US	{ "_id": "artist:19665", "last_name": "Cox", "first_name": "Jane", "country": "US", "genre": "Drame", "director": "Jane Cox", "actors": [{"last_name": "Cox", "first_name": "Jane"}]}	Drame
movie:10675	[{"last_name": "Ford", "first_name": "John"}]	FR	{ "_id": "artist:3556", "last_name": "Ford", "first_name": "John", "country": "FR", "genre": "Crime", "director": "John Ford", "actors": [{"last_name": "Ford", "first_name": "John"}]}	Crime
movie:10835	[{"last_name": "Yun-Fat", "first_name": "Tony"}]	HK	{ "_id": "artist:11401", "last_name": "Yun-Fat", "first_name": "Tony", "country": "HK", "genre": "Action", "director": "Tony Leung Ka-fai", "actors": [{"last_name": "Yun-Fat", "first_name": "Tony"}]}	Action
movie:10889	[{"last_name": "Rowlands", "first_name": "Linda"}]	US	{ "_id": "artist:11147", "last_name": "Rowlands", "first_name": "Linda", "country": "US", "genre": "Drame", "director": "Linda Rowlands", "actors": [{"last_name": "Rowlands", "first_name": "Linda"}]}	Drame

8. Visualisation et structure des documents

Les documents sont stockés au format **JSON**, pouvant contenir des structures imbriquées (objets et tableaux).

Contrairement aux bases relationnelles, CouchDB permet de **violier la première forme normale**, en stockant des données complexes dans un seul document.

Cependant, cette flexibilité peut entraîner :

- de la **redondance** des données ;

- des **risques d'incohérence**, par exemple lorsqu'un même acteur est dupliqué dans plusieurs documents avec des informations différentes.
-

9. Récupération d'un document par identifiant

Pour récupérer un document précis, il suffit d'utiliser son identifiant :

```
C:\Users\wirtz\OneDrive\Bureau\NoSql\TP4>curl -X GET "http://a:a@localhost:5984/films/movie:1018"
{"_id": "movie:1018", "_rev": "1-d092ae64609792fc8d09e01726f8ff9e", "title": "Mulholland Drive", "year": 2001, "genre": "Thriller", "summary": "A Hollywood, durant la nuit, Rita, une jeune femme, devient amnésique suite à un accident de voiture sur la route de Mulholland Drive. Elle fait la rencontre de Betty Elms, une actrice en devenir qui vient juste de débarquer à Los Angeles. Aidée par celle-ci, Rita tente de retrouver la mémoire ainsi que son identité.", "country": "FR", "director": {"_id": "artist:5602", "last_name": "Lynch", "first_name": "David", "birth_date": 1946}, "actors": [{"last_name": "Watts", "first_name": "Naomi", "birth_date": 1968}, {"last_name": "Hedaya", "first_name": "Dan", "birth_date": 1940}, {"last_name": "Harring", "first_name": "Laura", "birth_date": 1964}, {"last_name": "Miller", "first_name": "Ann", "birth_date": 1923}, {"last_name": "Theroux", "first_name": "Justin", "birth_date": 1971}, {"last_name": "Briscoe", "first_name": "Brent", "birth_date": 1961}]}{}
```

Partie 2

1. Rappel conceptuel (Map / Reduce)

- **Map** : fonction JavaScript s'appliquant à chaque document indépendamment ; elle peut émettre zéro, une ou plusieurs paires (clé, valeur) via emit(key, value).
 - **Shuffle / Sort** : étape intermédiaire qui regroupe les paires par clé et les distribue aux nœuds responsables.
 - **Reduce** : fonction d'agrégation qui combine les valeurs pour une même clé. Dans CouchDB, la réduction est optionnelle et des fonctions intégrées comme _sum, _count sont disponibles.
-

2. Exemple 1 — Nombre de films par année

2.1 Map

Design Document [?](#)

New document [▼](#)

Index name [?](#)
new-view

Map function [?](#)

```
1 function (doc) {  
2     emit(doc.year, doc.title);  
3 }
```

Reduce (optional) [?](#)
NONE [▼](#)

Create Document and then Build Index [Cancel](#)

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 brigade
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
movie:303	1946	Les Enchaînés

2.3 Reduce

Reduce (optional) ?

CUSTOM

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

3. Exemple 2 — Nombre de films par acteur

design/d2

Index name ?

new-view2

Map function ?

```
1 function (doc) {
2   for (i=0; i < doc.actors.length; i++){
3     emit({"prénom":doc.actors[i].first_name, "nom":doc.actors[i].last_name}, doc.title);
4   }
5 }
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

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

key	value
⌚ { "prénom": "A.J.", "nom": "Cook" }	1
⌚ { "prénom": "Aaron", "nom": "Eckhart" }	1
⌚ { "prénom": "Abdelghafour", "nom": "Elaaziz" }	1
⌚ { "prénom": "Abigail", "nom": "Breslin" }	1
⌚ { "prénom": "Adam", "nom": "Driver" }	2
⌚ { "prénom": "Adam", "nom": "Goldberg" }	1
⌚ { "prénom": "Adel", "nom": "Bencherif" }	1

4. Bonnes pratiques et recommandations

- **Émettre des valeurs compactes** dans map (p.ex. 1 pour compter) : réduire la quantité de données transférées lors du shuffle.
- **Utiliser les réduces intégrés** (_sum, _count, _stats) quand c'est possible — plus optimisés.
- **Normaliser les clés** dans la map si le regroupement doit être insensible à la casse ou aux variations d'orthographe.
- **Tester la map sans reduce** pour vérifier les paires intermédiaires avant d'appliquer une agrégation.
- **Vérifier le schéma des documents** : certains documents peuvent manquer de champs (prévoir des conditions dans la map).
- **Titres et métadonnées** : selon le besoin, émettre emit(key, {title: doc.title, year: doc.year}) si vous voulez conserver plus d'information par clé (mais attention au volume).

5. Limites / points d'attention

- CouchDB stocke des vues matérialisées (index) ; lors de gros volumes, la construction et la mise à jour des vues peut prendre du temps.

- Les incohérences possibles liées au choix d'un modèle documentaire (redondance des données) sont le prix du passage à l'échelle ; prévoir une stratégie de normalisation si nécessaire.
- Les noms d'acteurs variables (alias, orthographe) nécessitent un travail de nettoyage/normalisation en amont si l'analyse doit être précise.

Partie 3

1

Design Document ?

New document

Index name ?

new-view

Map function ?

```
1 * function (doc) {
2 *   if (doc.title) {
3 *     emit(null, 1);
4 *   }
5 }
```

Reduce (optional) ?

_sum

Create Document and then Build Index Cancel

key	value
null	278

2

Design Document ?

New document

_design/

nb_film_par_genre

Index name ?

new-view

Map function ?

```
1 v function (doc) {  
2 v   if (doc.genre) {  
3 v     emit(doc.genre, 1);  
4 v   }  
5 }
```

Reduce (optional) ?

_sum

Genre	Count
Action	36
Adventure	3
Aventure	22
Comédie	25
Comedy	1
Crime	29
Drama	14
Drame	96
Fantastique	4

3

Design Document ?

New document ▾

_design/
nb_film_par_réalisateur

Index name ?
new-view

Map function ?

```
1 function (doc) {
2   if (doc.director) {
3     emit(doc.director.last_name + " " + doc.director.first_name, 1);
4   }
5 }
```

Reduce (optional) ?
_sum

key	value
Abrams J.J.	2
Allen Woody	8
Annaud Jean-Jacques	1
Anspach Sólveig	1
Aronofsky Darren	1
Audiard Jacques	2
Bergman Ingmar	5

4

Map function ?

```
1 function (doc) {
2   if (doc.actors && Array.isArray(doc.actors)) {
3     doc.actors.forEach(function(actor){
4       var aid = (actor._id) ? actor._id : (actor.last_name + '|' + actor.first_name + '|' + actor.birth_date);
5       emit([aid, doc._id], null);
6     });
7   }
8 }
```

Reduce (optional) ?
_count

5

Design Document [?](#)

_design/nb_film_par_annee

Index name [?](#)

new-view

Map function [?](#)

```
1 function (doc) {  
2   if (doc.year) {  
3     emit(doc.year, 1);  
4   }  
5 }
```

Reduce (optional) [?](#)

_sum

key	value
1921	1
1927	1
1931	1
1936	2
1937	1
1940	3
1941	1
1942	1

6

```
1 function (doc) {  
2   if (doc.grades && Array.isArray(doc.grades)) {  
3     doc.grades.forEach(function(g){  
4       if (typeof g.note === 'number') {  
5         emit(doc._id, g.note);  
6       }  
7     });  
8   }  
9 }
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 function (keys, values, rereduce) {  
2   if (!rereduce) {  
3     var sum = 0, count = 0;  
4     values.forEach(function(v){ sum += v; count += 1; });  
5     return [sum, count];  
6   } else {  
7     var sum = 0, count = 0;  
8     values.forEach(function(part){  
9       sum += part[0]; count += part[1];  
10    });  
11    return [sum, count];  
12  }
```

7 (même reduce que 6)

Map function ?

```
1 function (doc) {  
2   if (doc.genre && doc.grades && Array.isArray(doc.grades)) {  
3     doc.grades.forEach(function(g){  
4       if (typeof g.note === 'number') emit(doc.genre, g.note);  
5     });  
6   }  
7 }
```

8 (meme reduce)

Map function ?

```
1 v function (doc) {  
2 v   if (doc.director && doc.director._id && doc.grades && Array.isArray(doc.grades)) {  
3 v     var did = doc.director._id;  
4 v     doc.grades.forEach(function(g){  
5 v       if (typeof g.note === 'number') emit(did, g.note);  
6 v     });  
7 v   }  
8 }
```

9

Map function ?

```
1 v function (doc) {  
2 v   if (doc.grades && Array.isArray(doc.grades)) {  
3 v     doc.grades.forEach(function(g){  
4 v       if (typeof g.note === 'number') {  
5 v         emit(null, { movie_id: doc._id, title: doc.title, note: g.note });  
6 v       }  
7 v     });  
8 }  
9 }
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 v function (keys, values, rereduce) {  
2 v   var best = null;  
3 v   values.forEach(function(v){  
4 v     if (!best || (v.note > best.note)) best = v;  
5 v   });  
6 v   return best;  
7 }
```

10

Map function ?

```
1 v function (doc) {  
2 v   if (doc.grades && Array.isArray(doc.grades)) {  
3 v     doc.grades.forEach(function(g){  
4 v       if (typeof g.note === 'number' && g.note > 70) emit(null, 1);  
5 v     });  
6 }  
7 }
```

Reduce (optional) ?

_sum

11

Map function ?

```
1 function (doc) {  
2   if (doc.genre && doc.actors && Array.isArray(doc.actors)) {  
3     doc.actors.forEach(function (a) {  
4       emit([doc.genre, a.last_name || '', a.first_name || '', a.birth_date || ''], {  
5         last_name: a.last_name,  
6         first_name: a.first_name,  
7         birth_date: a.birth_date      });    });  })  
}
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 function (keys, values, rereduce) {  
2   var map = {};  
3   if (!rereduce) {  
4     values.forEach(function(v){  
5       var id = (v.last_name||'') + '|' + (v.first_name||'') + '|' + (v.birth_date||'');  
6       map[id] = v;    });  
7   } else {  
8     values.forEach(function(arr){  
9       arr.forEach(function(v){  
10         var id = (v.last_name||'') + '|' + (v.first_name||'') + '|' + (v.birth_date||'');  
11         map[id] = v;    });  })  
12   var out = [];  
13   for (var k in map) { out.push(map[k]); }  
14   return out;  
15 }
```

key
["Action", "A. Fox", "Vivica", 1964]
["Action", "Allen", "Joan", 1956]
["Action", "Allen", "Nancy", 1950]
["Action", "Atherton", "William", 1947]
["Action", "Bale", "Christian", 1974]
["Action", "Bardem", "Javier", 1969]
["Action", "Bedelia", "Bonnie", 1948]
["Action", "Bell", "Marshall", 1942]
["Action", "Belmondo", "Jean-Paul", 1933]

12

Map function ?

```
1 function (doc) {  
2   if (doc.actors && Array.isArray(doc.actors)) {  
3     doc.actors.forEach(function(a) {  
4       emit([a.last_name, a.first_name, a.birth_date], 1);  
5     });  
6   }  
7 }
```

Reduce (optional) ?

_sum

Les résultats ne peuvent malheureusement pas être triés directement avec Map ou Reduce.

13

Map function ?

```
1 function (doc) {  
2   if (doc.grades && Array.isArray(doc.grades)) {  
3     doc.grades.forEach(function(g){  
4       if (g.grade) emit(doc._id, g.grade);  
5     });  
6   }}
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 function (keys, values, rereduce) {  
2   var counts = {};  
3   if (!rereduce) {  
4     values.forEach(function(g){  
5       counts[g] = (counts[g] || 0) + 1;  
6     });  
7     return counts;  
8   } else {  
9     values.forEach(function(part){  
10       for (var k in part) if (part.hasOwnProperty(k)) counts[k] = (counts[k] || 0) + part[k];  
11     });  
12     return counts;  
13   }}
```

14

Map function ?

```
1 function (doc) {  
2   if (doc.year && doc.grades && Array.isArray(doc.grades)) {  
3     doc.grades.forEach(function(g){  
4       if (typeof g.note === 'number') emit(doc.year, g.note);  
5     });  
6   }  
7 }
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 function (keys, values, rereduce) {  
2   if (!rereduce) {  
3     var sum = 0, count = 0;  
4     values.forEach(function(v){ sum += v; count += 1; });  
5     return [sum, count];  
6   } else {  
7     var sum = 0, count = 0;  
8     values.forEach(function(part){ sum += part[0]; count += part[1]; });  
9     return [sum, count];  
10 }  
11 }
```

Map function ?

```
1 function (doc) {  
2   if (doc.grades && doc.grades.length > 0) {  
3     var sum = doc.grades.reduce((a,b)=>a+b,0);  
4     var avg = sum / doc.grades.length;  
5     var grade = "D";  
6     if (avg >= 90) grade = "A";  
7     else if (avg >= 80) grade = "B";  
8     else if (avg >= 70) grade = "C";  
9     emit(grade, doc.title);  
10 }  
11 }
```

Reduce (optional) ?

CUSTOM

Custom Reduce function

```
1 function (doc) {  
2   if (doc.year && doc.grades) {  
3     doc.grades.forEach(function (g) {  
4       emit(doc.year, g);  
5     });  
6   }  
7 }
```

15

Map function ?

```
1 function (doc) {  
2   if (doc.director && Array.isArray(doc.grades) && doc.grades.length > 0) {  
3     var sum = 0, count = 0;  
4     doc.grades.forEach(function(g){  
5       if (typeof g.note === 'number') {  
6         sum += g.note;  
7         count += 1;  
8       }  
9     });  
10    if (count > 0) {  
11      var key = [doc.director._id || null, doc.director.last_name || '', doc.director.first_name || '];  
12      emit(key, { sum: sum, count: count });  
13    }}}
```

Reduce (optional) ?

CUSTOM



Custom Reduce function

```
1 function (keys, values, rereduce) {  
2   var result = { sum: 0, count: 0, name: null };  
3   values.forEach(function(v){  
4     result.sum += (v.sum || 0);  
5     result.count += (v.count || 0);  
6     if (!result.name && v.name) result.name = v.name;  
7   });  
8   return result;  
9 }
```