MIF04 GDW – TP
Map/Reduce sur MongoDB – partie 2

Résumé

Cette seconde partie du TP va prolonger le précédent en approfondissant Map/Reduce en MongoDB. L'environnement de travail est le même.

1 Modalités de rendu

Ce TP est à réaliser seul ou en binôme sous Linux. Ce TP sera rendu au plus tard le *vendredi 21 décembre 2018 à 23h59* sur `https://tomuss.univ-lyon1.fr/` dans la cellule *Rendu_TP_Mongo*. Le fichier `javascript` de départ à compléter est disponible à l'adresse suivante. Vous devrez rendre ce fichier complété avec vos réponses en ne faisant *qu'un seul dépôt par binôme*.

`https://perso.liris.cnrs.fr/romuald.thion/files/Enseignement/MIF04/MIF04-GDW-TPMongoDB-2-A-COMPLETER.js`

Commencez par remplir les deux objets `json` `etu_1` et `etu_2`. Si vous êtes seul, mettez la chaîne vide dans les champs de `etu_2`.

La correction étant *automatisée*, vous devrez *impérativement* respecter les noms, les prototypes des fonctions et surtout *la structure des résultats attendus* tels que précisés dans le fichier `javascript` de départ. Si les champs des objets `json` du résultat ne sont pas ceux demandés, la réponse sera considérée comme *fausse*. Notez que si les identifiants comptent, l'ordre des résultats n'importe pas, les valeurs seront triés lors de l'évaluation automatique. Vous remplirez les parties indiquées `// TODO` . Pour que le fichier `javascript` soit valide, une implémentation qui ne fait pas le résultat attendu est fournie, vous devez la remplacer.

Vous *n'ajouterez aucun traitement supplémentaire aux fichiers que vous rendrez*. Si vous avez un message à transmettre au correcteur, faites le avec des commentaires `javascript`. Tout manquement sera sévèrement sanctionné.

Dans ce TP, vous devrez, sauf mention explicite, répondre aux exercices avec des requêtes Map/Reduce. vous devrez résoudre chaque exercice *avec un seul job* Map/Reduce. Les fonctions `map` et `reduce` qui constituent le job de la question *Y* de l'exercice *X* sont nommées dans le fichier de départ `eXqY_map` et `eXqY_red`.

2 Jobs Map/Reduce sur la collection `grades`

Exercice 1 : Analyse de la collection `grades`

Toutes les questions de cet exercice portent sur la collection `grades`.

1. Calculer pour chaque étudiant·e (`student_id`) son nombre d'enregistrements.
2. Calculer le nombre de paires (`student_id`, `class_id`).
3. Conclure sur les clefs (au sens relationnel) de la collection. Justifiez dans `ex1q3_str`.
4. Calculer pour chaque type de note (*score* en anglais) son nombre d'occurrences dans la collection. Parcourez [1] pour cela le tableau *scores*.

Exercice 2 : Des statistiques sur la collection `grades`

Toutes les questions de cet exercice portent également sur la collection `grades`. On sait désormais qu'il n'y a que trois type d'épreuves, à savoir `exam`, `homework` et `quiz`. Pour chaque UE, on considère que l'ensemble des épreuves de type *exam* comptent pour 40% de la moyenne, celles de type `quizz` pour 20% et celles de type devoirs maison (*homework*) pour les 40% restant. Les notes sont toutes sur 100.

1. Calculer, pour chaque étudiant·e et chaque type d'épreuve, le nombre de notes de ce type qu'elle ou il a eu.
2. Calculer, pour chaque étudiant·e, la liste (le tableau en `javascript`) des types d'épreuves qu'elle ou il a eu. Il n'est pas demandé d'enlever les doublons dans la liste : la longueur de cette liste doit donc être le nombre d'épreuves passées. *Attention*, il faut *impérativement* que le type des valeurs émises par le `map` (retournées par `emit`) *soient du même type que la valeur de sortie du* reduce (retournée par `return`) [2]. Regardez le résultat attendu pour comprendre comment la liste est encapsulée en `javascript`.
3. Calculer la moyenne de chaque étudiant·e aux UEs où elle ou il est inscrit·e.
4. Pour chaque UE, donner la note maximale et la note minimale d'examen obtenue à cette UE.
5. Pour chaque UE, donner moyenne des notes d'examens. Vous aurez besoin pour cela d'utiliser une fonction [3] `finalize(key, val)` nommée en l'espèce `ex2q5_fin`.

Exercice 3 : De la performance de Map/Reduce

Les questions de cet exercice portent sur la collection `zips`.

1. Calculer la même chose que le job MAP/REDUCE fourni (fonctions `ex3_map` et `ex3_red`) en utilisant l'*aggregation pipeline*. [4] Donnez les *stages* dans la variable `ex3_stages`.
2. Vous pouvez activer les statistiques d'exécution de l'*aggregation pipeline* en exécutant avec la commande suivante `db.zips.explain("executionStats").aggregate(ex3_stages);`. Pour MAP/REDUCE, le temps d'exécution est disponible dans le champ `timeMillis` du résultat. Comparez la vitesse d'exécution des deux approches sur ce cas en répétant l'exécution 20 fois. Compléter pour cela la fonction `ex3_eval_perf()`. Donnez le ratio de performance « temps d'exécution MAP/REDUCE/ temps d'exécution *aggregation pipeline* » obtenu dans la variable `ex3_performance`.

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...of
2. *the type of the return object must be identical to the type of the value emitted by the map function.*, voir https://docs.mongodb.com/manual/reference/command/mapReduce/#mapreduce-reduce-cmd
3. https://docs.mongodb.com/manual/reference/command/mapReduce/#requirements-for-the-finalize-function
4. https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/