Laville Martin
Magnin Antoine
Salaün Nathan

Compte rendu projet base de données

UNIVERSITÉ DE NANTES

Partie 1 : Choix du jeu de données et de la technologie

Nous avons décidé de prendre comme données les statistiques des dépôts GitHub (commits, pull request, contributeurs). Nos données ont été obtenues via l'API GitHub. Cette API nous fournit les données sous format JSON, nous avons donc choisi d'utiliser MongoDB car le traitement des données était facilité.

Nous avons donc la possibilité d'obtenir les données historisées de plusieurs dépôt GitHub (granularité d'une semaine), contenant :

- Le nombre de commits et le détails des additions et suppressions
- Le nombre d'issues et leur statut
- Le nombre de pull requests et leur statut
- Les informations sur les releases
- Les contributeurs pour chaque dépôt, avec pour chacun :
    - Les issues qu'il a ouvertes
    - Les pull requests qu'il a soumises
    - Les statistiques sur les commits et le détails des additions et suppressions

Vous trouverez en annexe les schémas des données de chaque collection (trois par dépôt plus une qui contient tous les dépôts).

Voici le lien du GitHub du projet, depuis lequel vous pourrez trouver les scripts d'intégration ainsi que nos quelques requêtes : https://github.com/natinusala/projetbddmaster

Partie 2 : Intégration des données

Nous avons décidé d'utiliser un script Python pour intégrer les différentes données de l'API dans notre base de données. En effet, l'API est découpée en plusieurs méthodes qui renvoient chacune des informations différentes. Le script trie et recoupe ces informations par semaine, et génère un fichier JSON qui contient tout.

Les données retournées par l'API ne sont pas historisées, on a donc dû les historiser nous même - le traitement est donc long, car pour chaque issue on doit parcourir chaque semaine pour l'y ajouter, en prenant soin de la rouvrir si elle a été fermée plus tard que la semaine courante.

Le second script prend en entrée le fichier JSON généré par le premier, et génère (ou complète) via `mongoimport` quatre collections :

- `owner_repo_weeks` : les données concernant chaque semaine
- `owner_repo_collaborators :` les données concernant les collaborateurs
- `owner_repo_infos` : les informations diverses sur le dépôt lui-même
- `repos_list` : liste des dépôts présents dans la base de données

Où `owner` est le compte possédant le dépôt, et `repo` le nom du dépôt.

Partie 3 : Requêtes

Comment essayer ces requêtes ?

1. Installez les dépendances
    a. Python 3
    b. Les modules pip3
        i. pytz
        ii. python-dateutil
    c. MongoDB 3.4
2. Clonez le dépôt
3. Depuis votre compte GitHub, créez une clé OAuth et mettez la dans le fichier `oauth_token.txt` (à créer)
4. Exécutez les scripts suivants avant d'aller vous chercher un café
    a. `./create_json.py libretro RetroArch`
    b. `./create_json.py libretro Lakka-LibreELEC`
    c. `./create_json.py libretro libretro-common`
    d. `./create_json.py libretro libretro-samples`
    e. `./populate_mongodb.py libretro RetroArch`
    f. `./populate_mongodb.py libretro Lakka-LibreELEC`
    g. `./populate_mongodb.py libretro libretro-common`
    h. `./populate_mongodb.py libretro libretro-samples`
5. Pour exécuter une requête
    a. `./queries/la_requete.js`

1. Première semaine de contribution d'un utilisateur

first_week_of_contrib.js

Requête faite sur le dépôt `libretro_RetroArch_weeks`. On récupère d'abord l'id de l'utilisateur (ici "natinusala") grâce à la commande `find()`, puis on récupère le premier résultat de la recherche des semaines, triées par ordre croissant. Ceci correspond alors à la première semaine de contribution. Ici le résultat :

```
{
    "_id" : "1431216000"
}
```

Ici, on obtient le timestamp de la semaine, qui, une fois traduit, donne la semaine du dimanche 10/05/2015. En utilisant un tri décroissant lors de la requête, on obtient alors la dernière semaine de contribution.

2. Le contributeur le plus actif du dépôt

most_active_contributor.js

Requête faite sur le dépôt `libretro_Lakka-LibreELEC_contributors`. Afin de juger de qui est le contributeur le plus actif, on décide de prendre comme critère le nombre

de commits. Pour ce faire, on récupère le login et le nombre de commit du premier résultat de la recherche des nombres totaux de commits, triés par ordre décroissant. Dans MongoDB, l'id est également donné par défaut, il n'est donc pas nécessaire de le demander lors de la requête. On obtient le résultat suivant :

```
{
    "_id" : 306166,
    "total_commits" : 12630,
    "login" : "sraue"
}
```

On constate ici qu'avec 12 630 commits, l'utilisateur sraue est le plus actif.

3. Liste des dépôts d'un utilisateur

repos_of_user.js

On récupère d'abord la liste des dépôts à partir de la collection `repos_list`. Puis à l'aide d'un `forEach`, on itère sur les dépôts pour vérifier si l'utilisateur existe dans la liste des contributeurs. Ainsi, on obtient soit la liste des dépôts du contributeur au format JSON s'il a contribué dans au moins un des dépôts, soit undefined dans le cas contraire.

4. Liste triée des dépôts les plus actifs

most_active_repos.js

Pour cette requête, nous traitons les dépôts en fonction de leur moyenne de commit par semaine. Cette requête est un script qui effectue un map d'un MapReduce, puis un sort :

- Pour chaque dépôt, on map (donne une liste de couples {dépôt, moyenne}) :
    - Pour chaque semaine, on map le nombre de commits (donne une liste de nombres de commits)
    - Puis on reduce la liste du nombre de commits en effectuant une addition sur chaque élément (donne un nombre, la somme de tous les commits)
    - Puis on divise par la taille de la liste (donne un nombre, une moyenne)
    - Enfin on renvoie un couple {dépôt, moyenne}
- Puis on trie cette liste selon la moyenne

On a utilisé la fonction map de mongo, seulement une fois le map terminé, on se retrouve avec un tableau classique, donc le reduce et le sort sont ceux de JavaScript. Le fonctionnement est le même qu'un MapReduce classique.

5. Classement des contributeurs par dépôt

contributors_ranking_per_repo.js

Pour obtenir le classement des contributeurs par dépôt, on effectue simplement un map de chaque dépôt vers la liste de ses contributeurs, triée par nombre total de commits. On récupère donc une liste des dépôts, avec pour chaque dépôt une liste de contributeurs triée.

6. Moyenne d'activité par mois et par dépôt

average_activity_per_month.js

Cette requête nous permet d'obtenir l'activité moyenne d'un dépôt sur un mois. Nous avons estimé qu'un mois équivalait à 4 semaines. Le résultat obtenu nous donne l'information sur le nombre moyen de commits, de délétions et d'additions sur un dépôt lors d'un mois.

[22]On prend donc chaque dépôt puis on effectue une réduction sur les commits des semaines en les prenant 4 par 4 sur les additions, suppressions et commits totaux. On effectue ensuite une moyenne et on l'associe au dépôt.

7. La semaine record d'un utilisateur tous dépôts confondus

best_week_of_user.js

Cette requête nous renvoie pour un utilisateur donné la semaine où il aura effectué le plus de commit. Pour cela, on itère sur les dépôts en recherchant le maximum.

8. L'utilisateur ayant le plus contribué le plus en une semaine sur un dépôt

best_week_of_repo.js

A l'inverse, cette requête nous donne l'utilisateur qui a le plus contribué sur une semaine sur un dépôt. On itère sur toutes les semaines d'un dépôt (par contributeur) et on affiche celle ayant eu le maximum de commits. C'est le record de commits pour ce dépôt, pour un seul utilisateur.

9. Classement des utilisateurs ayant contribué au plus de dépôts

ranking_of_most_repos_contributed_to.js

On itère sur chaque contributeur de chaque dépôt, et on incrémente le compte pour chaque contributeur à chaque fois qu'on tombe sur lui. On obtient donc le total des dépôts sur lequel il a contribué.

10. Classement des dépôts par nombre de pull request de la dernière semaine.

repos_ranking_by_merged_pr.js

Ici on map à chaque dépôt le nombre de pull requests fusionnées à la dernière semaine, puis on trie le tableau obtenu par ordre décroissant. On obtient ainsi l'id, le nom et le possesseur de chaque dépôt, et le nombre de pull requests associées.

Annexes

Schéma des collections / exemples de données

Pour plus de lisibilité, nous avons décidé de ne pas mettre tous les résultats ici (le premier par exemple fait 14000 lignes). Utilisez l'outil en ligne http://jsonviewer.stack.hu/ et copiez/collez les fichiers JSON des schémas que vous voulez visualiser.

- `libretro_RetroArch_weeks` (semaine `1509840000`): https://pastebin.com/y8ZXWsTD
- `libretro_RetroArch_contributors` : https://pastebin.com/DE1WdUFq
- `libretro_RetroArch_infos` :


```
[
	{
		"_id" : "repo_infos",
		"repo" : "RetroArch",
		"owner" : "libretro",
		"homepage" : "http://www.libretro.com",
		"language" : "C",
		"fullname" : "libretro/RetroArch",
		"description" : "Cross-platform, sophisticated frontend for the libretro API. Licensed GPLv3."
	}
]
```


- repos_list :


```
[
	{
		"_id" : "libretro_libretro-samples",
		"owner" : "libretro",
		"repo" : "libretro-samples"
	},
	{
		"_id" : "libretro_libretro-common",
		"repo" : "libretro-common",
		"owner" : "libretro"
	},
	{
		"_id" : "libretro_RetroArch",
		"repo" : "RetroArch",
		"owner" : "libretro"
	},
	{
		"_id" : "libretro_Lakka-LibreELEC",
		"owner" : "libretro",
		"repo" : "Lakka-LibreELEC"
	}
]
```