

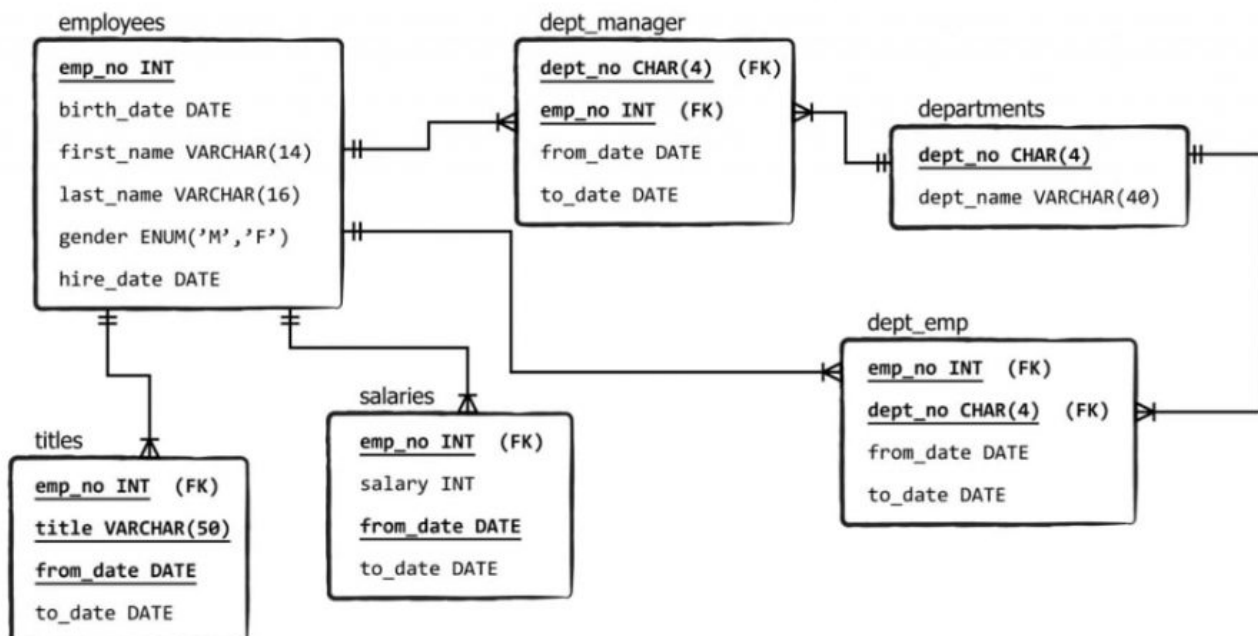
Migration de SQL vers NoSQL

Déroulement du projet

Dans ce projet, nous allons migrer une base de données PostgreSQL contenant 6 tables vers une base de données orientée documents, en utilisant MongoDB. La base relationnelle de départ est structurée de la manière suivante :

- Une table **employees**.
- Une table **departments**.
- Une table de jonction **dept_emp** pour prendre en charge la relation de plusieurs à plusieurs entre les employés et les départements. Un département a plusieurs employés. Un employé peut appartenir à différents départements à différentes dates, et éventuellement simultanément.
- Une table de liaison **dept_manager** pour prendre en charge la relation de plusieurs à plusieurs entre les employés et les départements. Même structure que "dept_emp".
- Une table **titles**. Il y a une relation de un à plusieurs entre les employés et les titres. Un employé a plusieurs titres (simultanément ou à différentes dates). Un enregistrement de titre se réfère à un employé (via emp_no).
- Une table **salaries** dont la structure est similaire à la table "titles". Relation de un à plusieurs entre les employés et les salaires.

L'objectif est d'effectuer la migration en restructurant la base de sorte à améliorer les performances d'accès. Il s'agit de dénormaliser la structure, quitte à ajouter de la redondance.



1 Création et alimentation des tables avec PostgreSQL

1. Avec le terminal PostgreSQL (**psql**), créez une base de données "employees", s'y connecter, et exécutez les scripts de création et d'alimentation des tables de la base "employees".
2. Proposez des commandes pour lister les enregistrements des tables.

2 Coût des jointures en SQL

1. Proposez une commande SQL pour effectuer une jointure entre les tables **employees**, **salaries** et **titles**.
2. Proposez une ou plusieurs commande(s) pour mesurer le temps d'exécution de cette jointure. Quel est le temps d'exécution de cette requête sur votre machine ?

*Consultez la documentation de la commande **EXPLAIN**.*

Pour rappel, une vue est une table virtuelle qui est le résultat d'une requête. Ses enregistrements n'existent qu'au moment de l'accès à la vue par l'exécution de la requête sous-jacente.

3. Proposez une commande pour créer une vue "matérialisée", dont le contenu est le résultat de cette jointure. Une vue matérialisée n'est rien d'autre qu'une vraie table. Une fois créée et alimentée, par exemple par déclenchements de trigger, son contenu existe une fois pour toute.
4. Quel est le temps d'exécution sur votre machine, d'une requête SQL listant les enregistrement de cette vue ?

3 Export de données PostgreSQL vers fichiers JSON

Nous exportons maintenant les données des différentes tables dans des fichiers **JSON**. Heureusement, **PostgreSQL** dispose de nombreuses fonctions pour gérer les fichiers **JSON**. Pour les tâches ci-après, consultez la documentation des fonctions [row_to_json](#), [json_agg](#), et [COPY](#).

1. Exportez le contenu de chacune des tables dans des collections distinctes pour MongoDB. Pour ce faire, proposez des commandes qui convertissent les enregistrements vers des fichiers **JSON** en utilisant la fonction **row_to_json**.
2. Modifiez vos commandes précédentes pour agrégez ces lignes dans un tableau **JSON**, en utilisant la fonction **json_agg**.
3. Enfin, à l'aide de la commande **COPY**, convertissez ces données en texte et enregistrez-les dans des fichiers **JSON**.

4 Import dans MongoDB des données des fichiers JSON

Avec la commande **mongoimport** et les options appropriées, importez chacune des collections issues de vos 6 fichiers **JSON** dans une base de données du même nom, **employees**.

5 Coût des jointures avec MongoDB et dénormalisation des données

Vous avez vu dans la section 2 qu'avec le langage **SQL**, lier les données des trois tables **employees**, **salaries**, et **titles** via le champ **emp_no** se fait simplement avec une commande de quelques lignes.

Avec MongoDB, vous pouvez effectuer une jointure en utilisant la fonction **aggregate()** que vous connaissez, et la commande **\$lookup**.

1. Consultez la documentation de **\$lookup** et proposez une commande MongoDB pour effectuer une jointure entre les collections **employees** et **titles**.
2. Modifiez votre commande avec un pipeline à 2 étapes, de sorte à effectuer une jointure entre les 3 collections **employees**, **titles** et **salaries**.
3. Quel est le temps d'exécution de cette dernière jointure ? Proposez une commande pour mesurer ce temps d'exécution.

Un processus correct de dénormalisation consiste à fusionner un côté de la relation de un à plusieurs dans l'autre afin de fournir un document unique avec toutes les informations nécessaires pour la requête. Un

exemple de ce type de situation est la relation entre les employés et les salaires, ainsi que la relation entre les employés et les titres. Si l'utilisation des données fournies par ces deux collections est toujours liée à la collection des employés, une bonne solution est de fusionner le côté "many" de la relation (par exemple, un titre est détenu par un employé, un employé peut avoir plusieurs titres différents).

Une méthode possible pour dénormaliser est d'utiliser comme vous venez de le faire la fonction `aggregate()`, en partant du côté "un" de la relation, c'est-à-dire `employees`, et d'incorporer l'enregistrement des collections du côté "plusieurs" de la relation en utilisant l'opérateur `$lookup`.

4. Complétez votre pipeline avec un opérateur de projection (`$project`) afin d'éviter la création d'informations inutiles telles qu'un doublon du `emp_no` (utilisé pour relier les documents à la collection des employés) ou un doublon de l'ID d'objet (`_id`, utilisé pour rendre chaque document unique dans la collection).

La sortie obtenue doit être un document `employee` avec en plus deux champs tableaux (`array`) "titles" et "salaries" contenant les documents précédemment contenus dans les collections "titles" et "salaries".

5. En utilisant l'opérateur `$out` ou `$merge`, ajoutez une étape à votre pipeline pour sauvegarder les documents résultants de vos opérations dans une nouvelle collection.
6. Quel est le délai pour accéder à toutes les informations des documents de la nouvelle collection ?

6 Conclusions

Dans quel(s) cas de figure cette dénormalisation, ainsi proposée, est-elle opportune ? Et dans quel(s) cas ne l'est-elle pas ?