Utilisation des graphes d'entités

Quelques liens :

- API JPA 2.0 (JEE 8) [1]
- Tutorial JPA (JEE 7) [2]
- Une documentation sur JPQL [3]
- La page de Wikipedia [4]
- De très bons exemples [5]

1 Préalables

Nous utilisons depuis plusieurs séances les entités et leurs relations. Par commodité et pour éviter un chargement de données excessif, nous utilisons en général le chargement retardé des relations.

Ce mécanisme permet de limiter la quantité de données transmise mais il implique que la connexion vers la base soit toujours ouverte afin de récupérer, à la demande, les données manquantes. cela pose un problème dans une approche transactionnelle. Hormis cette difficulté, le chargement retardé implique le lancement de plusieurs requêtes de récupération des données annexes. Cela peut avoir un impact très fort sur l'efficacité de nos traitements.

Nous allons essayer de régler ces problèmes avec les graphes d'entités.

2 Mise en oeuvre

A savoir avant le tutoriel :

- Vous placerez les entités dans le package prévu (`myapp.jpa.model`).
- Vous ajouterez à vos entités les constructeurs basés sur les propriétés qui permettent de créer facilement des instances (annotations `@Data` , `@NoArgsConstructor` et `@AllArgsConstructor`).
- Vous construirez les **repositories** pour chaque entité.
- Vous construirez une classe de test unitaire pour mettre en oeuvre les exemples en demandant à Spring l'injection d'un `EntityManager` .

Travail à faire : Suivez ce tutoriel [6] mais avec la procédure suivante :

- Prévoyez un test `populate` exécuté en premier (voir comment faire [7]) qui va créer en base une grande quantité d'utilisateurs, de publications et de commentaires. Il est évidemment très important que ces instances soient en liaison les unes avec les autres (plusieurs dizaines de publications par utilisateur et plusieurs dizaines de commentaires par publication). **Attention** : vous devrez annoter la classe de test par `@Rollback(false)` afin que les modifications effectuées par `populate` ne soient pas oubliées (car c'est la politique par défaut).
- Utilisez ces données pour tester chaque fonction proposée par le tutoriel en vérifiant les données récupérées et celles qui nécessitent une opération supplémentaire (il faut bien déterminer la frontière du graphe des données).

Après le tutoriel :

1. https://javaee.github.io/javaee-spec/javadocs/javax/persistence/package-summary.html
2. https://docs.oracle.com/javaee/7/tutorial/partpersist.htm
3. http://download.oracle.com/docs/cd/E13189_01/kodo/docs40/full/html/ejb3_overview_query.html
4. http://en.wikipedia.org/wiki/Java_Persistence_API
5. http://www.java2s.com/Tutorial/Java/0355__JPA/Catalog0355__JPA.htm
6. https://www.baeldung.com/jpa-entity-graph
7. https://www.baeldung.com/junit-5-test-order

- Ajoutez à vos entités les relations inverses (`User->Post` et `User->Comment`) et testez le bon fonctionnement des EG. Vous pouvez notamment faire deux EG : `user-with-posts` et `user-with-comments` .
- Essayez le graphe `user-with-posts-and-comments` . Vous allez découvrir une limitation des graphes d'entités (plus d'information [8]).

3 Graphe d'entités et Spring Data

Vous pouvez demander à Spring Data d'utiliser les graphes d'entités en

- ajoutant une méthode annotée `@Query` dans vos dépôts ;
- ajoutant à cette méthode l'annotation ci-dessous.

```
@EntityGraph("nom-du-graphe-d-entités")
```

Travail à faire : Vérifiez par un test unitaire la bonne prise en compte du graphe d'entités (les données sont-elles chargées ?).

4 Chargement retardé avec JPAQL

Pour compléter le problème du chargement retardé, il vous est possible de demander explicitement le chargement d'une donnée au travers d'une requête JPA (clause `FETCH`).

En vous aidant de la requête ci-dessous, ajoutez une méthode au dépôt de gestion des publications et testez le bon chargement des commentaires (mais pas du propriétaire de la publication).

```
select p from Post p join fetch p.comments where (id = :id)
```

5 Performance

Vérifiez par une mesure des temps d'exécution qu'il existe une différence importante entre un chargement retardé géré de manière manuelle (appel à la méthode `size()` par exemple) et la solution des graphes d'entités.

Vérifiez cette différence notamment si la BD est stockée sur disque.

8. https://www.baeldung.com/java-hibernate-multiplebagfetchexception