Applications orientées données (NSY135)

12 – Optimisation des lectures

Auteurs: Raphaël Fournier-S'niehotta et Philippe Rigaux
(philippe.rigaux@cnam.fr,fournier@cnam.fr)

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

Plan du cours

Introduction

- Nous avons étudié HQL dans une optique de **sélection** de données à matérialiser.
- Dans cette section nous abordons l'option **fetch** qui indique que l'on veut charger les objets associés dans le cache.
- Si, par exemple, on sait à l'avance que l'on va traiter un film, son metteur en scène et ses acteurs, on peut décider de charger par une seule requête tout ce sous-graphe.

L'option fetch

- On ne peut pas utiliser **fetch** sans **left join**.
- Le **fetch** indique que les objets de la table associée doivent être placés dans le cache.

```
select film
from Film as film
left join fetch film.roles as role
```

- Pour chaque film placé dans le cache, on initialise donc la collection des rôles.

Combiner requêtes de chargement et de sélection

- Attention à ne pas combiner sans précaution une requête de chargement (avec **fetch**) et de sélection (avec **where**).
- Prenons la requête suivante : on veut les films dont un des rôles est "McClane".
- On l'exprime de la manière suivante :

```
select film
from Film as film
left join fetch film.roles as role
where role.nom= 'McClane'
```

- Le **fetch** indique que l'on veut charger dans le cache le graphe **Film-Role pour les lignes sélectionnées dans la base**.
- Mais ici nous avons un problème : comme nous avons exprimé une **restriction** avec la clause **where** sur les rôles, seuls les rôles dont le nom est McClane seront chargés.
- La mauvaise surprise est que quand nous naviguerons du film vers ses rôles, nous n'en trouverons qu'un, ce qui n'était pas l'objectif initial.

Combiner requêtes de chargement et de sélection

- La solution est d'utiliser deux fois la table **Role**.
- La première occurence, sans restriction, apparaît dans le **left join fetch**, la seconde dans le **where**.
- Voici une expression qui me semble claire (d'autres sont possibles) :

```
from Film as film
left join fetch film.roles as role
where film in (select r2.pk.film from Role as r2 where nom= 'McClane')
```

- Ce n'est pas très élégant, mais cela correspond à la satisfaction de deux besoins différents : **sélectionner** des données et les **charger** sous forme d'objet.

1+n requêtes et HQL

- Reprenons le problème des 1+n requêtes.
- Il serait préférable d'effectuer la jointure suivante en lieu et place de la requête SQL qui sélectionne simplement les films :

```
select * from Film as film, Notation as note
where film.id=note.id_film
```

- Nous savons maintenant obtenir cette jointure **et le chargement du sous-graphe Film-Notation** avec la requête HQL suivante :

```
select distinct film from Film as film left join fetch film.notations
```

1+n requêtes et HQL

- En utilisant cette expression HQL, les notations seront placées dans le cache, et la navigation sur les notations pour chaque film ne nécessitera plus de l'exécution de la seconde requête SQL :

```
// On recherche les films
Set<Film> films = session.createQuery ("select distinct film from Film as film "
                                       + "left join fetch film.notations");

for (Film film: films) {
  // Pour chaque film on traite les notes
  for (Notation notation: films.getNotation()) {
    // Faire qq chose avec la notation
  }
}
```

Plan du cours

Résumé

- La question du chargement des objets par des requêtes SQL est importante et malheureusement assez complexe pour des raisons déjà évoquées :
  - changements d'une version à une autre,
  - confusion de la documentation,
  - différences JPA/Hibernate,
  - complexité des options disponibles.
- On a essayé de dissimuler certains aspects (les **proxy**, les **wrappers**) dont le rôle est peu clair et ne sont sans doute pas indispensables à la bonne gestion du problème
- Le bon réglage des requêtes SQL est un art qui implique une très bonne compréhension de l'exécution des requêtes dans un système relationnel, l'utilisation des index, des algorithmes de jointure, de tri, des ressources mémoire, etc.
- Si vous n'êtes pas spécialiste, l'aide d'un DBA pour optimiser votre application peut s'avérer nécessaire.

Recommandations

- Dans la configuration, déclarez toutes vos associations en **Lazy**.
- Configurez la session pour produire l'affichage des requêtes SQL.
- Explorez votre application systématiquement pour analyzer les requêtes SQL qui sont produites; dans une application MVC c'est assez simple : exécutez chaque action et regardez quelle(s) requête(s) SQL est/sont produite(s).
- Si vous constatez trop de requêtes : remplacez les opérations de navigation par une requête HQL initiale qui charge les objets auxquels l'action va accéder; si possible réduisez à une seule requête SQL produite par action.
- Si vous constatez qu'une ou plusieurs requêtes SQL paraîssent trop complexes, produisez son plan d'exécution, et modifiez éventuellement vos expressions HQL pour la décomposer.