# Applications orientées données (NSY135)

## 11 – Le langage HQL

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

# Sous-requêtes

- Comme en SQL, HQL permet l'expression de sous-requêtes, ce qui revient souvent à une nouvelle manière d'exprimer des jointures
- Exemple : une nouvelle manière d'obtenir les films dirigés par Clint Eastwood :

```
from Film as film
where  film.realisateur in (from Artiste where nom='Eastwood')
```

- Par rapport à l'expression standard avec un **join**, il existe une différence : comme la classe **Artiste** n'apparaît que dans la sous-requête, les instances de **Artiste** ne figurent pas dans le résultat.
- Il n'est donc pas nécessaire d'utiliser une clause **select** pour se restreindre aux films.

# Sous-requêtes

- Voici une autre version de la requête qui ramène les films dans lesquels figure un rôle 'McClane' :

```
from Film as film
where  film in (select role.pk.film from Role as role
where role.nom='McClane')
```

- Enfin, voici une autre version de la requête qui recherche les films dirigés par Clint Eastwood, cette fois avec une sous-requête **exists**.

```
select film.titre
from Film as film
where  exists (from Artiste as a where a = film.realisateur and a.nom='Eastwood')
```

- Cette expression est très compliquée par rapport à celle basée sur la jointure par navigation (et en particulier la jointure implicite, voir ci-dessus).

# Sous-requêtes (suite)

- Les sous-requêtes en HQL sont tout à fait similaires à celles de SQL.
- Pour rappel, les quantificateurs suivants sont composables avec les opérateurs de comparaison, et s'appliquent à des sous-requêtes qui ramènent plusieurs lignes :
  - **any**: vrai si **au moins** une ligne ramenée par la sous-requête satisfait la comparaison;
  - **all**: vrai si **toutes** les lignes ramenées par la sous-requête satisfontt la comparaison.
- Des synonymes pour **any** sont **in** et **some**.
- Exemples :
  - recherche les films dont au moins un acteur est né avant 1940 :

```
from Film as film
where 1940 > any (select  role.pk.acteur.anneeNaissance
from Role as role
where role.pk.film=film)
```

  - Il est beaucoup plus simple d'exprimer ce genre de requête avec un **join**.

# Sous-requêtes (suite)

- Les sous-requêtes en HQL sont tout à fait similaires à celles de SQL.
- Pour rappel, les quantificateurs suivants sont composables avec les opérateurs de comparaison, et s'appliquent à des sous-requêtes qui ramènent plusieurs lignes :
  - **any**: vrai si **au moins** une ligne ramenée par la sous-requête satisfait la comparaison;
  - **all**: vrai si **toutes** les lignes ramenées par la sous-requête satisfontt la comparaison.
- Des synonymes pour **any** sont **in** et **some**.
- Exemples :
  - Utilisation de **all** (les films dont **tous** les acteurs sont nés avant 1940) :

```
from Film as film
where 1940 > all (select  role.pk.acteur.anneeNaissance
from Role as role
where role.pk.film=film)
```

  - Comme en SQL, il existe plusieurs syntaxes possibles pour une même requête. La précédente s'exprime aussi avec **not exists**.

# Créations d'objets spécifiques

- Il existe des requêtes **ad hoc** qui créent des résultats constitués de **projections** sur certains attributs

```
select film.titre, film.annee, film.realisateur.prenom, film.realisateur.nom
from Film as film
```

- Il semble que la requête implique un travail de décodage fastidieux, avec une liste de tableaux à parcourir, chaque tableau contenant 4 objets correspondant aux 4 attributs projetés
- **Cependant**, il y a un avantage : comme la requête ne nécessite pas la création d'objets persistants, on évite le placement de ces objets dans le cache de la session
- On peut gagner (légèrement) en performance, ce qui sera utile dans certains cas et vaudra le surcroît de programmation nécessaire

# Regroupement et agrégation

Le **group by**, le **having** et les fonctions d'agrégation sont similaires à SQL

- La note moyenne des films

```
select avg(note)
from Notation
```

# Regroupement et agrégation

Le **group by**, le **having** et les fonctions d'agrégation sont similaires à SQL

- Les films et leur nombre d'acteurs

```
select film, count(*)
from Film as film join film.roles as role
group by film
```

- On aimerait faire "select film, count(film.roles) from Film as film", mais HQL n'accepte pas cette syntaxe.

# Regroupement et agrégation

Le **group by**, le **having** et les fonctions d'agrégation sont similaires à SQL

- Les metteurs en scène ayant réalisé au moins trois films

```
select artiste.nom, count(*)
from Artiste as artiste join artiste.filmsRealises as film
group by artiste
having count(*)  > 3
```