# NHibernate

Matthias COLIN
Juillet 2020

# Problématique

## Base de données relationnelle

- Modèle relationnel :
  - table, colonnes avec type, clé primaire, contraintes
  - association avec clé étrangère : colonne (1,n) ou table (n,n)
- Donnée : ligne d'1 table identifiée par un id (clé primaire)

## Programmation Orientée Objet

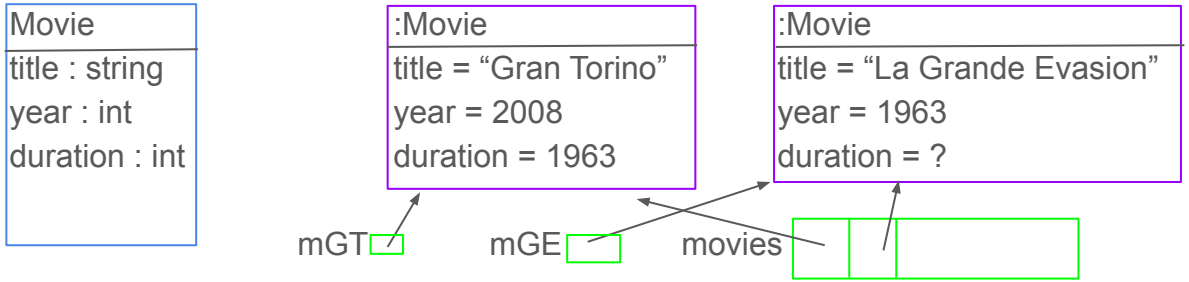
- Modèle objet :
  - classe, champs/attributs/propriétés, méthodes
  - Association : référence objet, collection, unidirectionnelle ou bidirectionnelle
  - héritage
- Donnée : objet/instance géré par référence

# Problématique (2)

| Movies | | | |
|---|---|---|---|
| id<br>int pk | title<br>varchar(250) | year<br>smallint | duration<br>smallint |
| 1 | Gran Torino | 2008 | 116 |
| 2 | La Grande Evasion | 1963 | NULL |

select

insert
update
delete



# Problématique

- Tâches répétitives
  - Adapter résultat d'un select en objets : new Movie{title=row[1], year=row[2],...}
  - Générer les insert/update/delete à partir du contenu des objets:
    - Insert into Movies (title, year, …) values (m.title, m.year, …)
    - Suivre chaque objet par son Id en base de donnée pour update/delete
  - Jongler entre 2 modèles pour écrire les requêtes SQL
  - S'adapter à chaque SGBDR pour les petites différences de SQL
- Solution : Object Relational Mapping
  - Ecrire la correspondance de modèle 1 fois
  - Faire correspondre automatiquement ligne d'une table et objet de la classe correspondante
  - Gérer un cache d'objets persistants
  - Ecrire les requêtes SQL avec le point de vue objet, indépendamment des SGBDR

# Historique

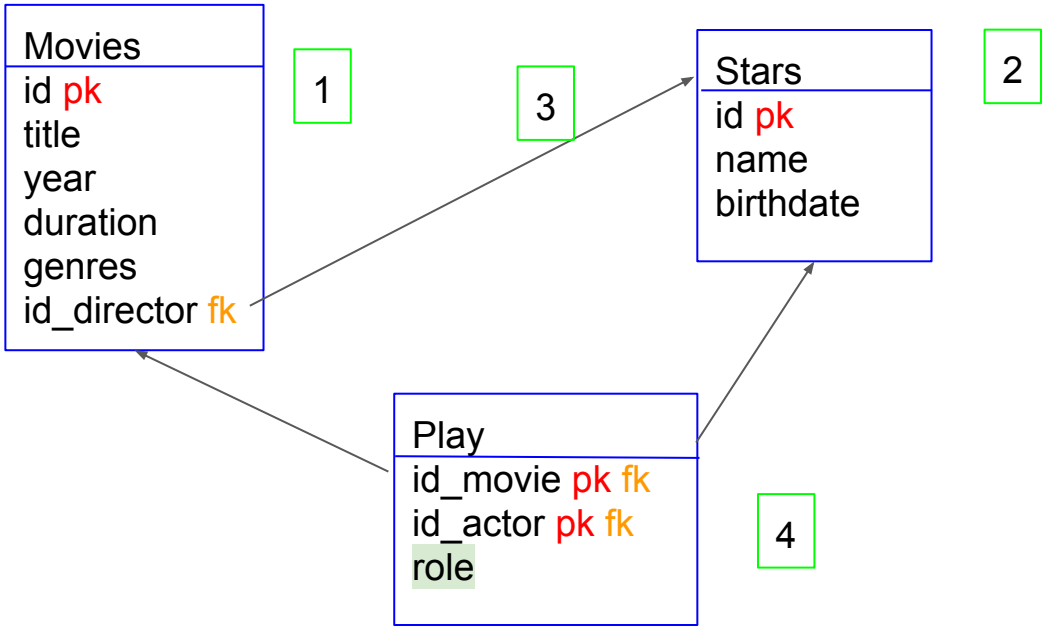
- 2001 - Hibernate (Red Hat) : ORM Java
- 2005 - NHibernate 1.0 (Red Hat) : portage sur framework .NET
- 2006 - NHibernate devient un projet de la communauté Open Source
- 2010 - v3.0 avec support .NET 3.5, LinQ, typed criteria API
- 2011 - v3.1 : fluent configuration
- 2014 - v4.0 : support .NET 4+
- 2017 - v5.0 : support .NET 4.6.1+
- 2018 - v5.1 : support .NET Standard 2.0 et .NET Core 2.0

# Sites Web

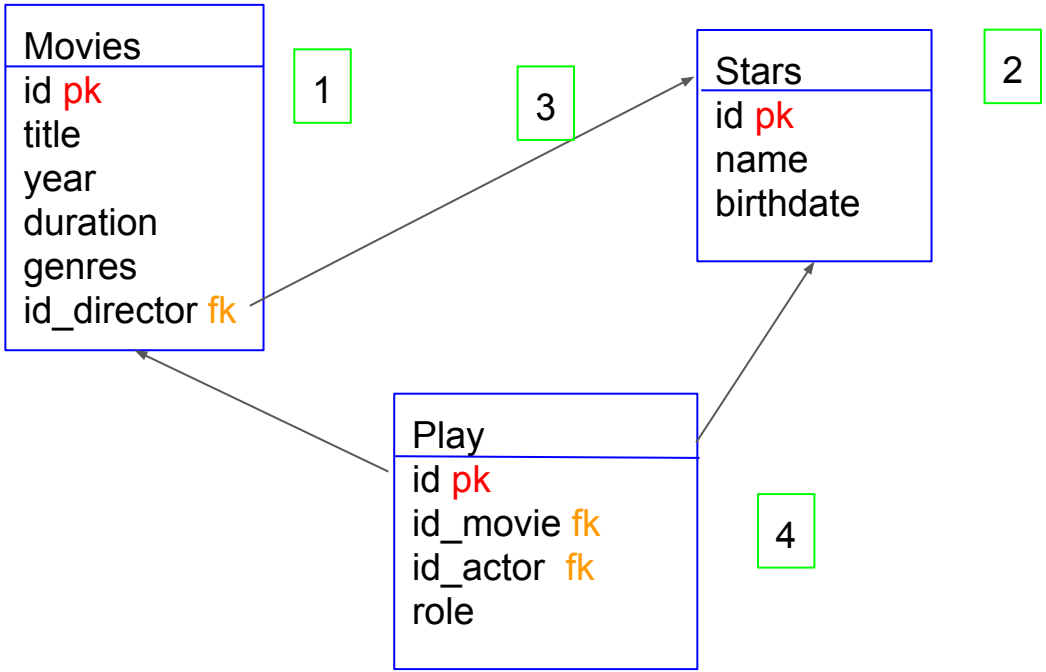
NHibernate : https://nhibernate.info/

Fluent NHibernate : https://github.com/FluentNHibernate/fluent-nhibernate

# 1er Mapping : cinéma



# 2e Mapping : cinéma avec rôle



# Mapping associations

- Association 1,n  (ou 1,1):
  - Many-to-one, (One-to-one) : 1 attribut de l’autre classe : References
  - One-to-many : 1 collection de l’autre côté (IEnumerable, …) : HasMany
  - Unidirectionnel / Bidirectionnel
- Association n,n :
  - Many-to-Many : table d’association (transparente si pas d’attribut) : HasManyToMany

# Requêtes : select

- Id : get/load
- Sql natif : select upper(title), nvl(duration,0) from Movies where duration is not null
  - Aucune vérification / modèle de la bdd / non portabilité => à éviter
- Pseudo SQL : HQL : from Film f where f.Year = 2019
  - Vocabulaire modèle objet / portabilité SGBD (y compris fonctions standard) / typage faible
  - https://nhibernate.info/doc/nhibernate-reference/queryhql.html
- Criteria : Expression.Like(“Title”, “Star%”)
  - Approche objet / vérification au niveau modèle objet
  - https://nhibernate.info/doc/nhibernate-reference/querycriteria.html
- LinQ (.NET) : méthodes objets + fragment de sql
  - https://nhibernate.info/doc/nhibernate-reference/queryling.html
  - https://nhibernate.info/doc/nhibernate-reference/queryqueryover.html

# Mapping d'Héritage

Media : Movie, TVSeries

1. Single-table : 1 seule table pour tte la hiérarchie de classe
   a. Discrimininator pour distinguer le type de chaque ligne
2. Table-per-subclass : 1 seule table par classe fille en répétant les infos communes, pas de table pour la classe mère : Movie, TVSeries
   a. Inconvénient : maj modèle
3. Table-per-class :
   a. 1 table pour la classe parente Media avec les colonnes communes
   b. 1 par fille avec les colonnes spécifiques : Movie et TVSeries
   c. Inconvénient : join

http://notherdev.blogspot.com/2012/01/mapping-by-code-inheritance.html?m=1

# Programmation événementielle

- n Listener(s) : accompagnement
  - Save, Update, SaveUpdate, Load, ...
- 1 Interceptor : stop action pour l’annuler, la reprendre avec modification

https://nhibernate.info/doc/nhibernate-reference/events.html

# Gestion de cache

- Cache L1 (niveau) associé à 1 session => sûre
  - Fin de session : vide le cache associé
- Cache L2 : inter session
  - Objets en lecture seule
  - Objets en modification centralisée par l'application (web)
  - Requête avec ses params: accélération
- Provider :
  - NHibernate : cache par défaut
  - Autres : framework .NET
- Réglage : configuration, mapping, requêtes

# Testing

- Fluent propose de valider le mapping et le cycle save/load

  https://github.com/FluentNHibernate/fluent-nhibernate/wiki/Persistence-specification-testing