FORMATION DEV WEB WEB MOBILE
CCP 2 - DÉVELOPPER LA PARTIE BACK-END D'UNE APPLICATION WEB

LES BASES DE DONNEES RELATIONNELLES

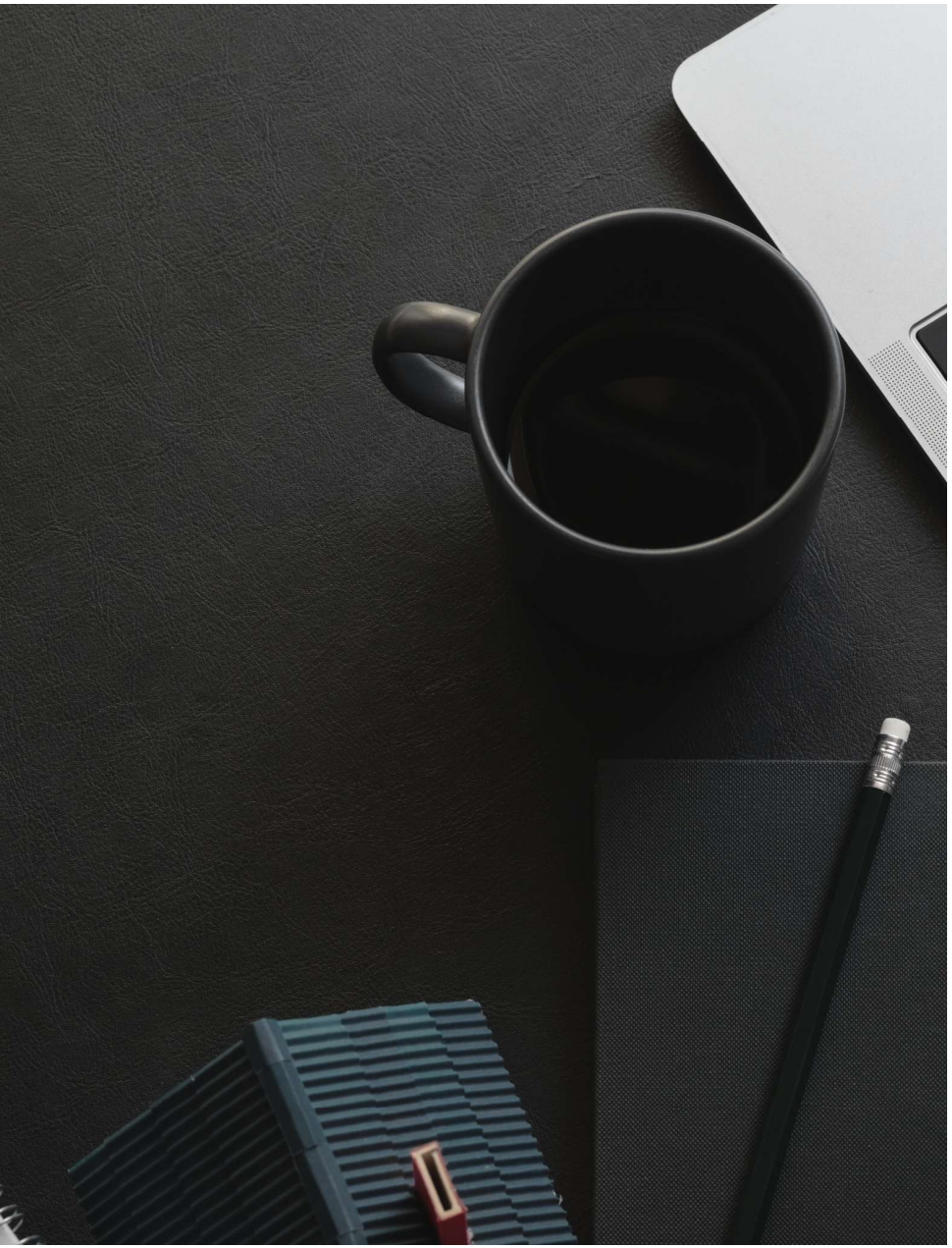
JOUR 2 : LES RELATIONS ET LA MODELISATION DEA

PRESENTE PAR NATACHA DESSE

AFEC

DEROULE MODULE

- Découverte des BDD et Installation
- **Les relations et modélisation (DEA)**
- Contraintes et schéma physique
- Jeu d’essai, insertion et agrégations, requêtes avancées
- Jointures et sous requêtes
- Sauvegardes SQL et gestion des rôles



LES ASSOCIATIONS EN BASE DE DONNEES

LES OBJECTIFS

Comprendre la notion d'associations entre tables en base de données relationnelle

Identifier les différents types de relations : 1-1, 1-N, N-N

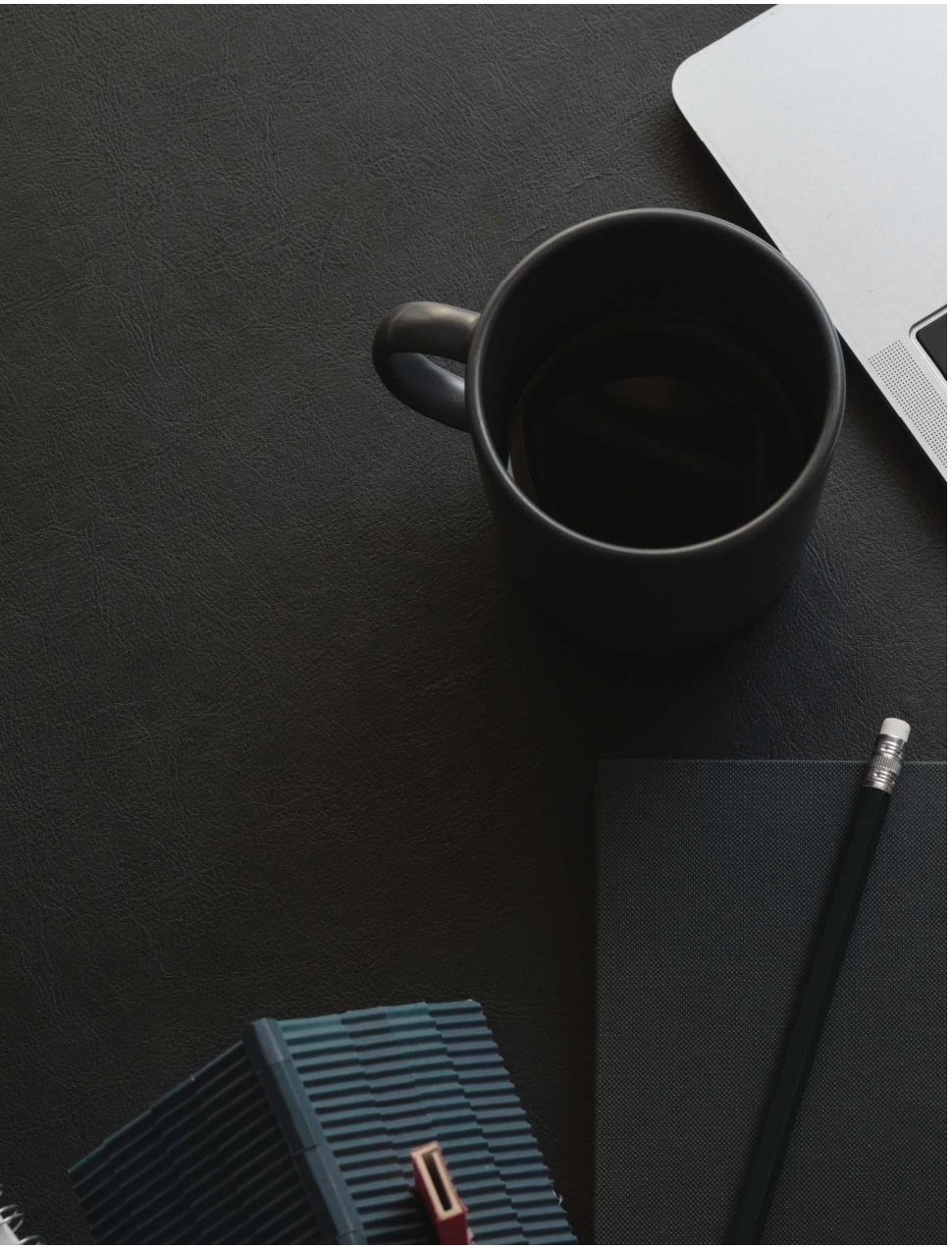
Modéliser les relations entre entités sous forme de diagramme entité-association (DEA)

Passer d'un DEA à une structure de tables SQL

NUAGE DE MOTS

Vous avez retenu quoi sur les requêtes SQL ?

https://app.wooclap.com/UYERSA?from=event-page

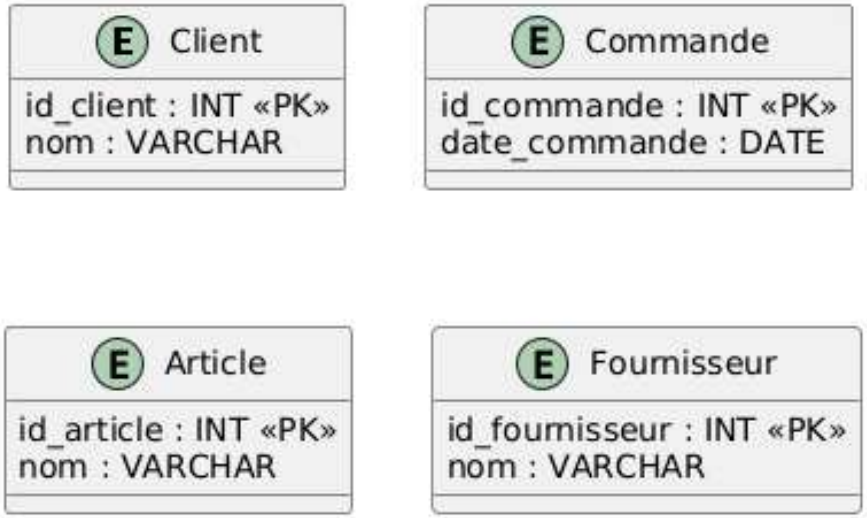


LES NOTIONS

QU'EST-CE QU'UNE ENTITE ?

Une **entité** est une population d'individus **homogène**.

Par exemple, les produits ou les articles vendus par une entreprise peuvent être regroupés dans une même entité articles, car d'un article à l'autre, les informations ne changent pas de nature (à chaque fois, il s'agit de la désignation, du prix unitaire, etc.).



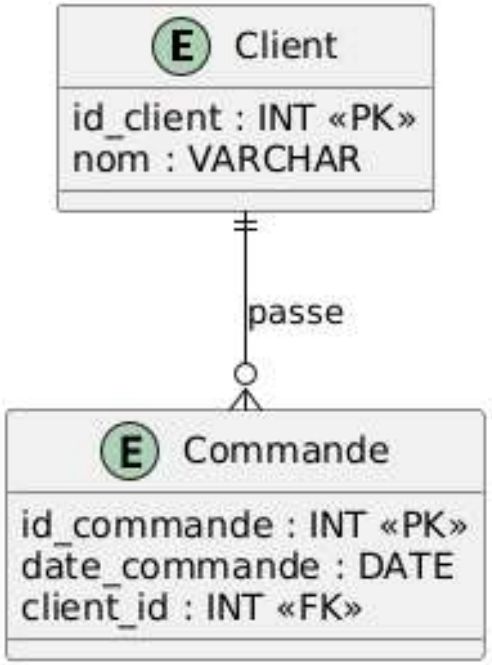
Par contre, les articles et les clients ne peuvent pas être regroupés : leurs informations ne sont pas homogènes (un article ne possède pas d'adresse et un client ne possède pas de prix unitaire). Il faut donc leur réserver deux entités distinctes : l'entité articles et l'entité clients.

QU'EST-CE QU'UNE ASSOCIATION ?

Une **association** (ou **liaison**) en base de données relationnelle représente un **lien logique entre deux entités (tables)**.

Elle permet de **modéliser les relations réelles** entre les objets du monde (ex. : un client passe des commandes) et se traduit par une **clé étrangère** dans la structure des tables.

Ces liaisons assurent la **cohérence**, **la normalisation** et facilitent la **recherche croisée des données**.



CARDINALITES

La cardinalité d'un lien entre une entité et une association précise le **minimum** et le **maximum** de fois qu'un individu de l'entité peut être **concerné** par l'**association**.

Exemple : un client a au moins commandé un article et peut commander n articles (n étant indéterminé), tandis qu'un article peut avoir été commandé entre 0 et n fois (même si ce n'est pas le même n que précédemment). On obtient alors le schéma entités-associations complet.

Une cardinalité minimale de 1 doit se justifier par le fait que les individus de l'entité en question ont besoin de l'association pour exister (un client n'existe pas avant d'avoir commandé quoi que ce soit, donc la cardinalité minimale de l'entité clients dans l'association commander est 1). Dans tous les autres cas, la cardinalité minimale vaut 0 (c'est le cas pour une liste préétablie d'articles, par exemple).

ZOOM SUR LES CARDINALITES

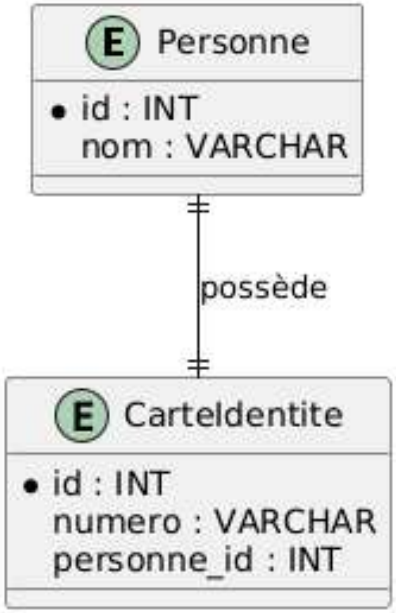
RELATION 1-1 (ONE TO ONE)

Chaque ligne d'une table est associée à une seule ligne d'une autre table.

Exemple :

- Une personne possède une carte d'identité
- Une carte d'identité est liée à une personne

Personne (1,1) —— (1,1) CarteIdentite



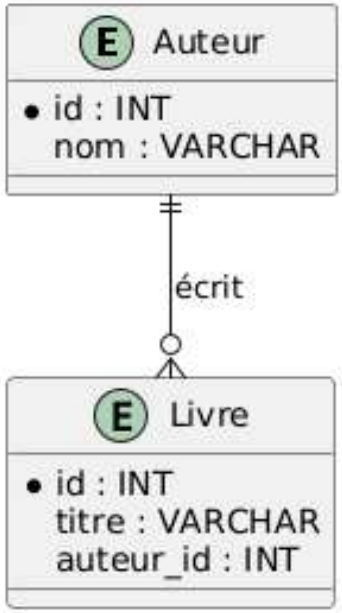
RELATION 1-N (ONE TO MANY)

Une ligne d'une table est associée à plusieurs lignes d'une autre table.

Exemple :

- Un auteur peut écrire plusieurs livres
- Un livre est écrit par un seul auteur

Auteur (1,1) —— (0,N) Livre



RELATION N-N (MANY TO MANY)

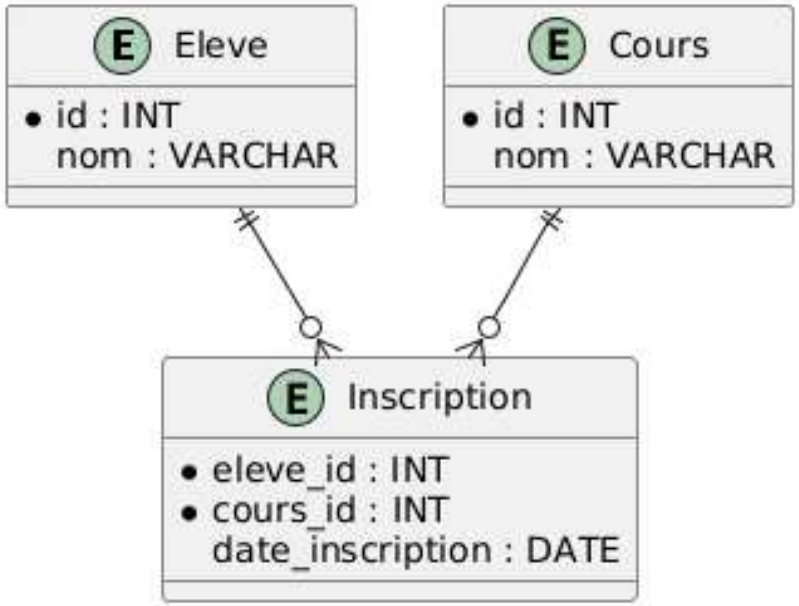
Plusieurs lignes d'une table peuvent être associées à plusieurs lignes d'une autre.

Exemple :

- Un élève peut suivre plusieurs cours
- Un cours peut être suivi par plusieurs élèves

Élève (0,N) —— (0,N) Cours

→ Cette relation nécessite une table de liaison.



LES BONNES PRATIQUES

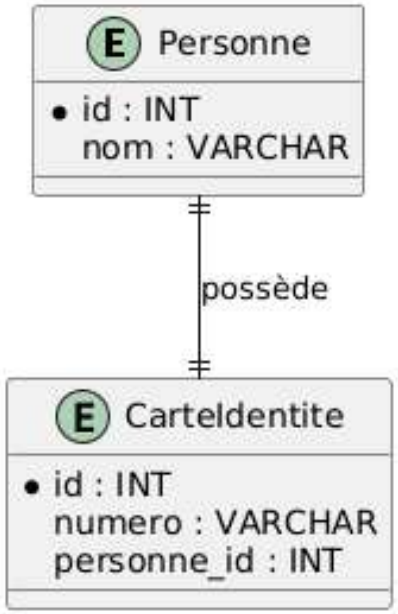
NOMMAGE

Normalisation des entités (importante) : toutes les entités qui sont remplaçables par une association doivent être remplacée

Normalisation des noms : le nom d'une entité, d'une association ou d'un attribut doit être unique et explicite.

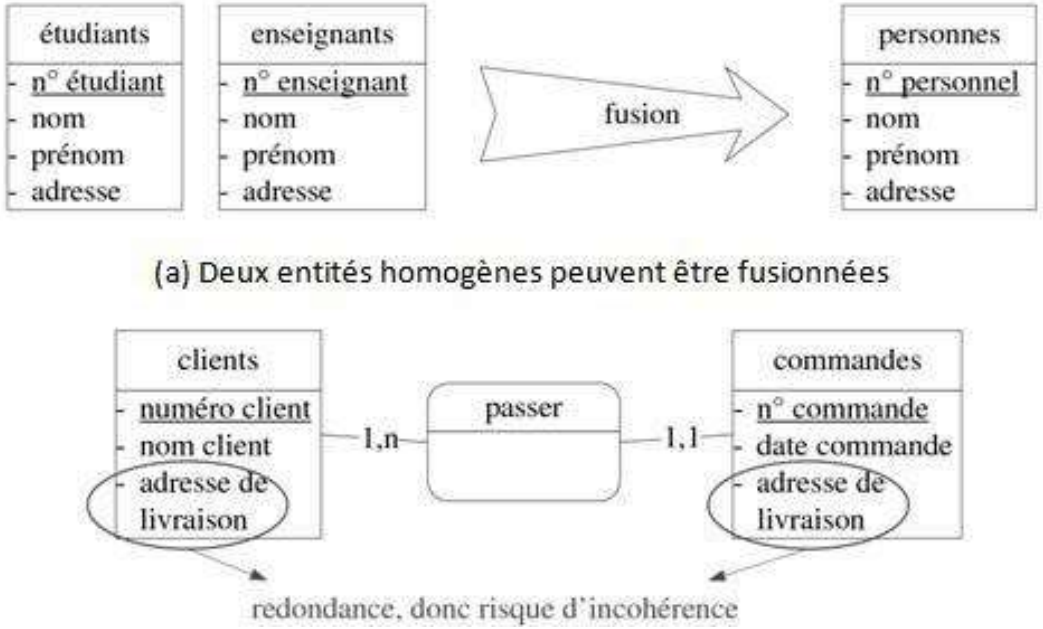
Conseils :

- Le singulier est recommandé pour nommer les entités et les tables avec une majuscule ou PascalCase si c'est une combinaison de mots
- pour les attributs on utilise du snake_case



COHERENCE

- Deux entités homogènes peuvent être fusionnées.
- Si deux attributs contiennent les mêmes informations, alors la redondance induit non seulement un gaspillage d'espace, mais également un grand risque d'incohérence : ici, les adresses risquent de ne pas être les mêmes et dans ces conditions, où faut-il livrer ?



CLE PRIMAIRE ET ETRANGERES

- Clé primaire nommée id
- Clé étrangère nommé entite_id par exemple personne_id dans Carteldentité
- Utiliser DELETE ON CASCADE pour retirer les références dans les tables associées

ATELIER

Modélisation d’une base de données “cineclub”

CONSIGNES

Voici le scénario

Une petite salle de cinéma souhaite gérer sa programmation via une application simple. Le personnel a besoin de stocker, consulter et analyser les films projetés. Chaque film doit être caractérisé par :

- *Titulaire d'un identifiant unique*
- *Titre*
- *Année de sortie*
- *Durée en minutes*
- *Genre*
- *Note critique (sur 10, avec un decimal)*

CONSIGNES

Le personnel souhaite également :

- *Savoir **qui** a réalisé chaque film (un seul **réalisateur** par film), avec nom, prénom, date de naissance*
- *Enregistrer les **acteurs** principaux, avec leur nom, prénom, et date de naissance.*
- *Savoir quel **rôle** chaque acteur a joué dans chaque film (nom du personnage).*
- *Classer les **films** dans un ou plusieurs **genres** (comédie, drame, action, etc.).*
- *Connaître les dates de **projection** des films dans la salle.*

ETAPE 1 : Définir les entités et leurs attributs

AIDE

Film
- *Titre*
- *Année de sortie*
- *Durée (en minutes)*
- *Note critique (sur 10, avec un décimal)*
- *Genre*

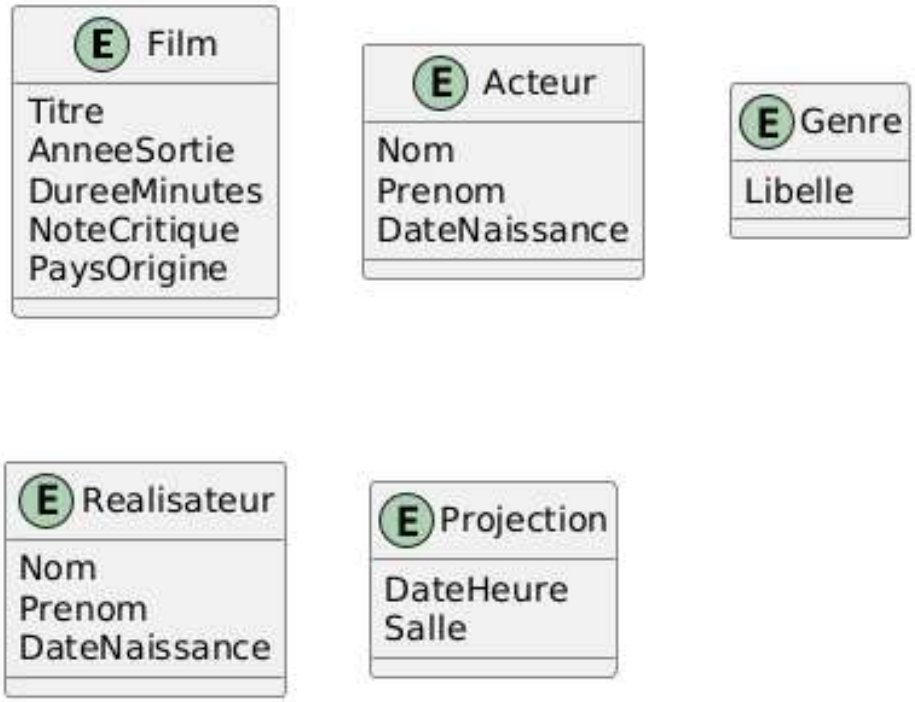
Acteur
- *Nom*
- *Prénom*
- *Date de naissance*

Genre
- Libellé

Réalisateur
- Nom
- Prénom
- Date de naissance

Projection
- Date et heure de la séance
- Salle

ENTITES ET ATTRIBUTS



ETAPE 2 : Définir les associations

AIDE

Actions entre relations :

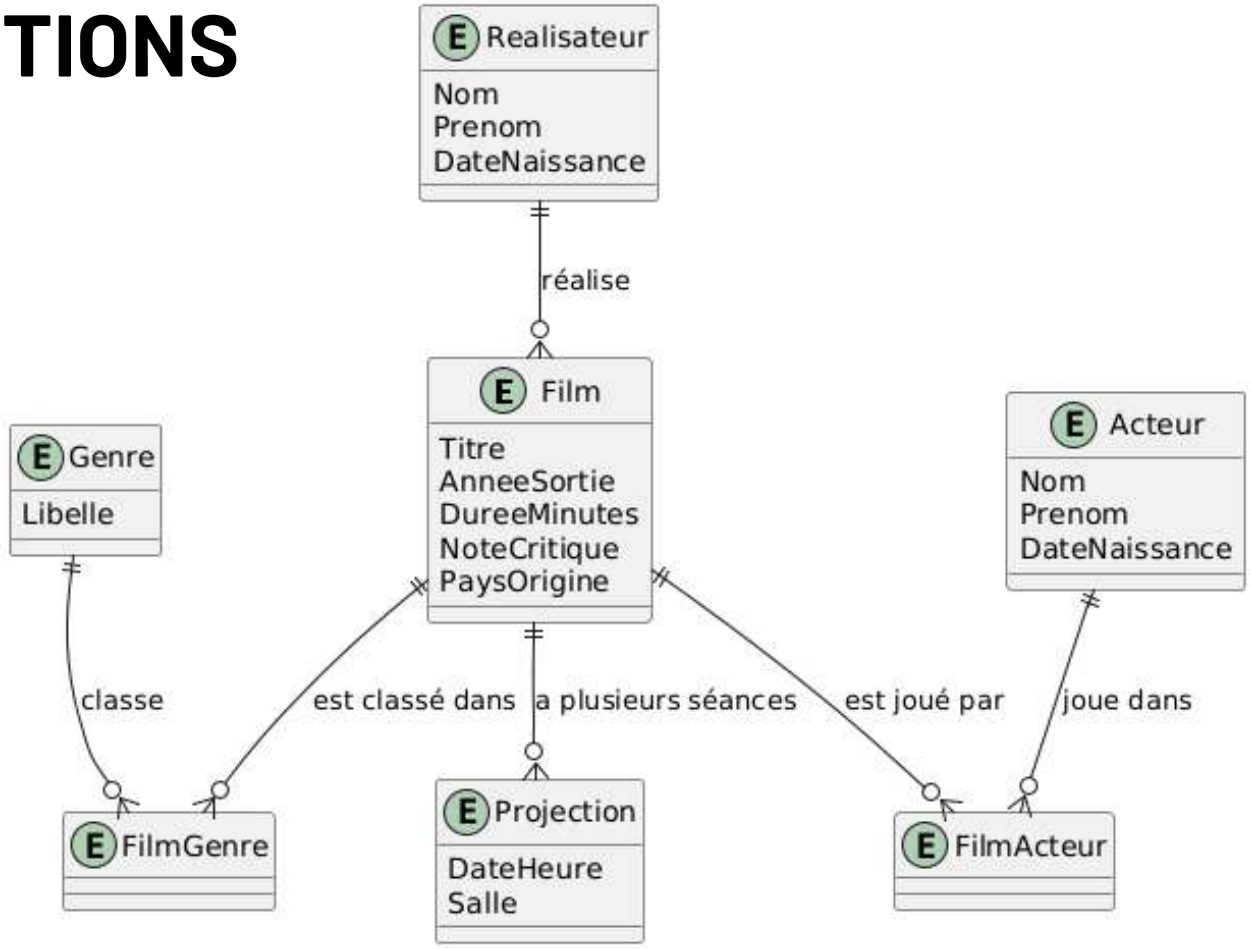
- *Un Film est joué par plusieurs Acteurs*
- *Un Acteur joue dans plusieurs Films*

- *Un Film est classé dans plusieurs Genres*
- *Un Genre classe plusieurs Films*

- *Un Réalisateur réalise un ou plusieurs Films*
- *Un film est réalisé par 1 réalisateur*

- *Un Film a plusieurs séances Projection*
- *Une projection n'a qu'un film de projeté*

ASSOCIATIONS

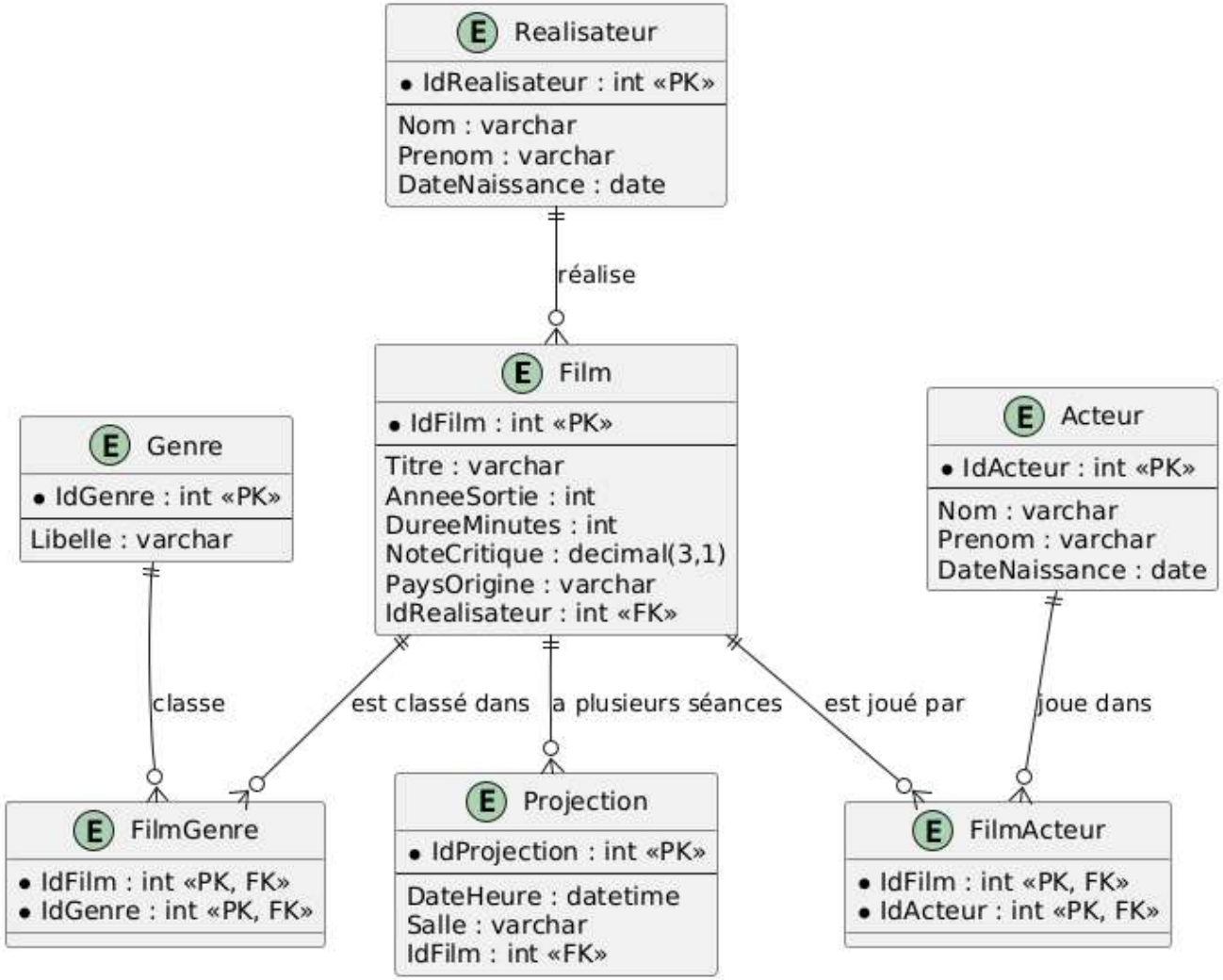


ETAPE 3 : On passe au MLD, on donne les clés primaires par “PK” et on type les données

AIDE

DATE TYPE	SPEC	DATA TYPE	SPEC
CHAR	String (0 - 255)	INT	Integer (-2147483648 to 214748-3647)
VARCHAR	String (0 - 255)	BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 - 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 - 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 - 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 - 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 - 16777215)	DATETIME	YYYY-MM-DD HH:MM:SS
LONGTEXT	String (0 - 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 - 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (-128 to 127)	ENUM	One of preset options
SMALLINT	Integer (-32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (-8388608 to 8388607)	BOOLEAN	TINYINT(1)

MLD

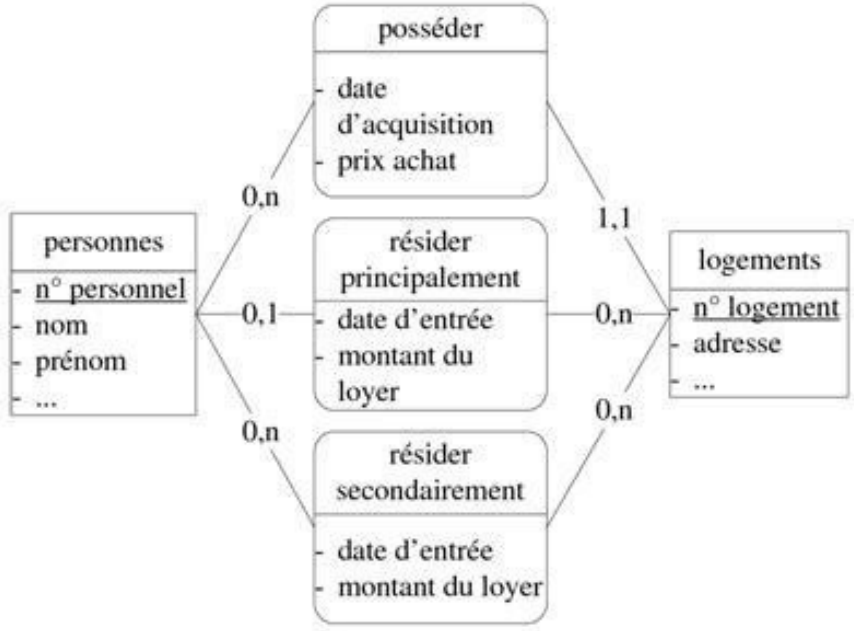

Realisateur
• IdRealisateur : int «PK»
Nom : varchar
Prenom : varchar
DateNaissance : date
réalise
Film
• IdFilm : int «PK»
Titre : varchar
AnneeSortie : int
DureeMinutes : int
NoteCritique : decimal(3,1)
PaysOrigine : varchar
IdRealisateur : int «FK»
Genre
• IdGenre : int «PK»
Libelle : varchar
Acteur
• IdActeur : int «PK»
Nom : varchar
Prenom : varchar
DateNaissance : date
classe
est classé dans
a plusieurs séances
est joué par
joue dans
FilmGenre
• IdFilm : int «PK, FK»
• IdGenre : int «PK, FK»
Projection
• IdProjection : int «PK»
DateHeure : datetime
Salle : varchar
IdFilm : int «FK»
FilmActeur
• IdFilm : int «PK, FK»
• IdActeur : int «PK, FK»

ENCORE PLUS D'ASSOCIATIONS

ASSOCIATIONS PLURIELLES

Deux mêmes entités peuvent être plusieurs fois en association

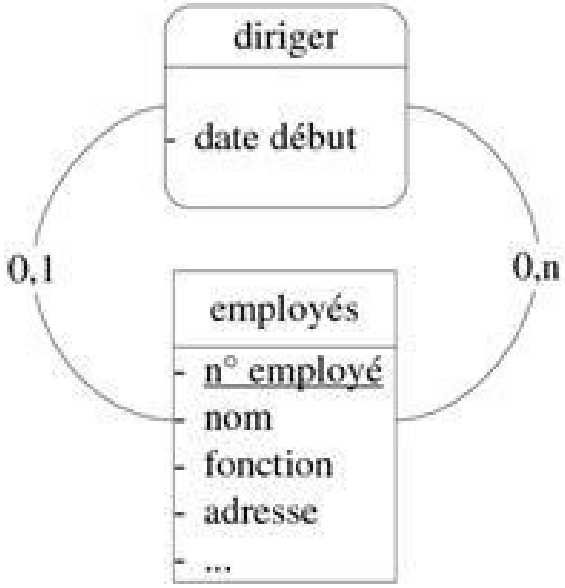
Dans cet exemple issu d'une agence immobilière, une personne peut être propriétaire, résider principalement ou résider secondairement dans un logement géré par l'agence.

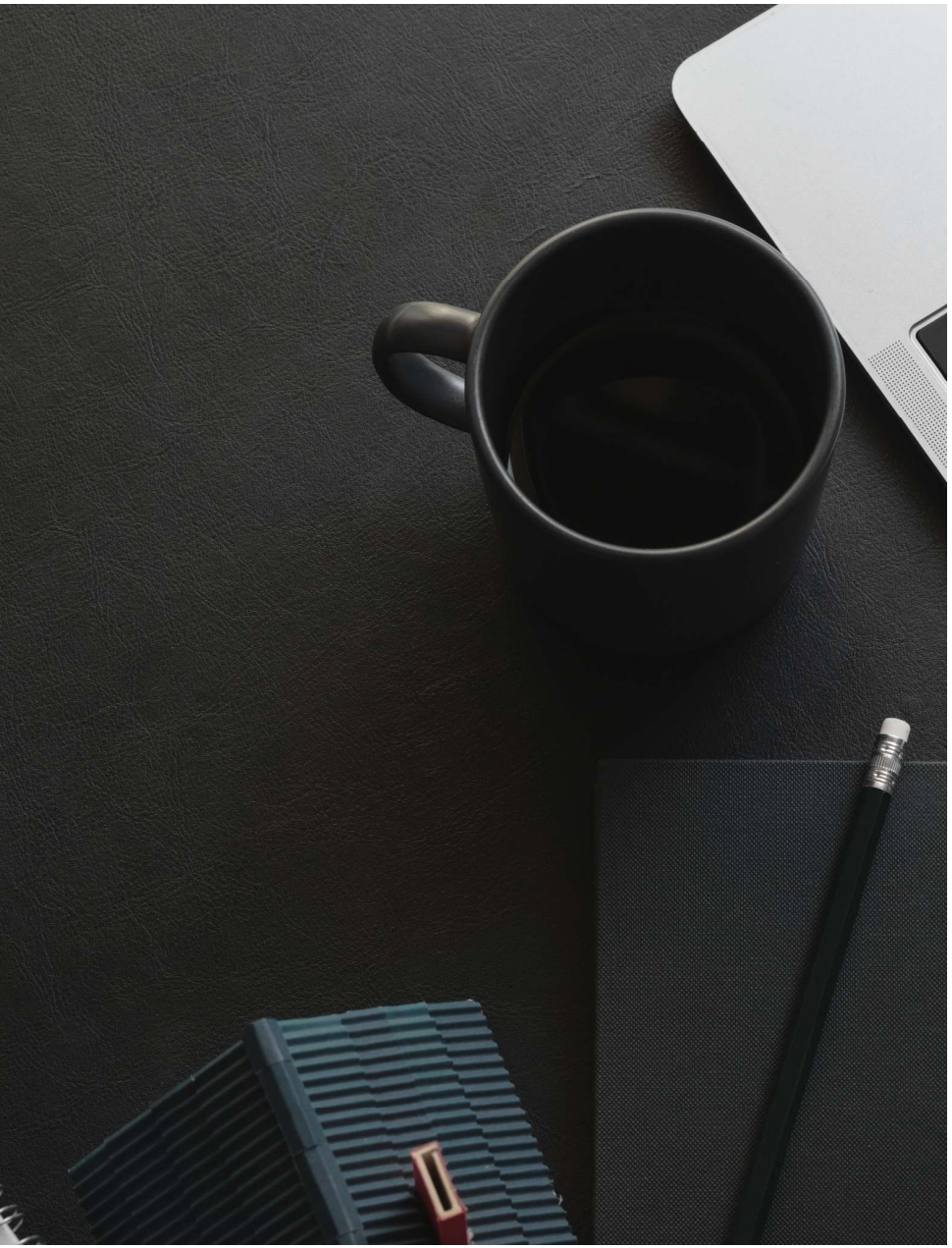


ASSOCIATIONS REFLEXIVES

Il est permis à une association d'être branchée plusieurs fois à la même entité, comme l'association binaire reflexive

Dans cet exemple, tout employé est dirigé par un autre employé (sauf le directeur général) et un employé peut diriger plusieurs autres employés, ce qui explique les cardinalités sur le schéma.





IMPLEMENTATION EN SQL

RAPPEL - CREATION DE TABLE

```sql
-- CRÉATION D'UNE BASE DE DONNÉES
CREATE DATABASE ma_base;

-- UTILISER UNE BASE DE DONNÉES
USE ma_base;

-- SUPPRIMER UNE BASE DE DONNÉES
DROP DATABASE ma_base;

-- CRÉATION D'UNE TABLE
CREATE TABLE utilisateurs (
    id INT PRIMARY KEY AUTO_INCREMENT, -- identifiant unique
    nom VARCHAR(50),
    prenom VARCHAR(50),
    email VARCHAR(100) UNIQUE,
    date_naissance DATE,
    actif BOOLEAN DEFAULT TRUE
);
```

RAPPEL - INSERT INTO

```sql
-- VOIR LA STRUCTURE D'UNE TABLE
DESCRIBE utilisateurs;

-- INSÉRER DES DONNÉES
INSERT INTO utilisateurs (nom, prenom, email, date_naissance)
VALUES ('Dupont', 'Jean', 'jean.dupont@email.com', '1990-05-10');

-- INSÉRER PLUSIEURS LIGNES
INSERT INTO utilisateurs (nom, prenom, email, date_naissance) VALUES
('Martin', 'Alice', 'alice.martin@email.com', '1988-03-20'),
('Bernard', 'Paul', 'paul.bernard@email.com', '1992-07-11');
```

RAPPEL - SELECT

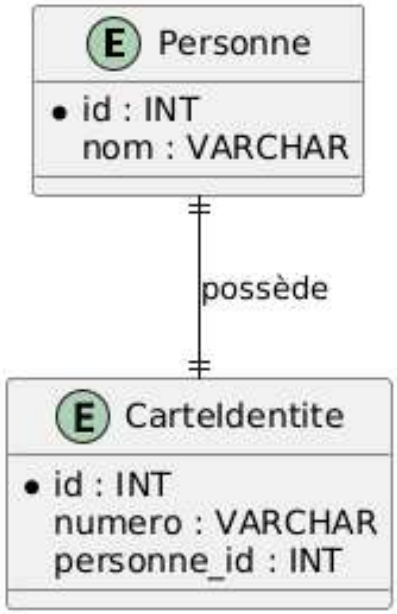
```sql
-- RÉCUPÉRER DES DONNÉES
SELECT * FROM utilisateurs;

SELECT nom, email FROM utilisateurs;

-- FILTRER DES DONNÉES
SELECT * FROM utilisateurs WHERE actif = TRUE;

SELECT * FROM utilisateurs WHERE nom = 'Martin';

-- FILTRER AVEC DES CONDITIONS
SELECT * FROM utilisateurs WHERE actif = TRUE AND date_naissance <
'1990-01-01';

-- TRIER LES RÉSULTATS
SELECT * FROM utilisateurs ORDER BY nom ASC;

-- LIMITER LE NOMBRE DE RÉSULTATS
SELECT * FROM utilisateurs LIMIT 5;

```

RAPPEL - MODIFIER LES DONNEES

```sql
-- SUPPRIMER DES DONNÉES
DELETE FROM utilisateurs WHERE id = 2;

-- AJOUTER UNE COLONNE
ALTER TABLE utilisateurs ADD telephone VARCHAR(20);

-- MODIFIER UNE COLONNE
ALTER TABLE utilisateurs MODIFY telephone VARCHAR(30);

-- SUPPRIMER UNE COLONNE
ALTER TABLE utilisateurs DROP COLUMN telephone;
```

RELATION 1-1 (ONE TO ONE)

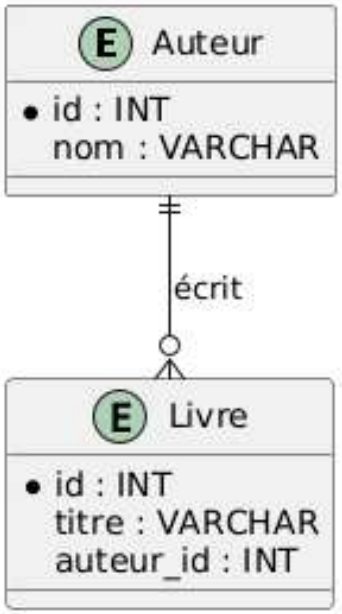
Personne (1,1) —— (1,1) CarteIdentite




```
CREATE TABLE Personne (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100),
 prenom VARCHAR(100)
);

CREATE TABLE CarteIdentite (
  id INT AUTO_INCREMENT PRIMARY KEY,
  numero VARCHAR(20) NOT NULL UNIQUE,
  personne_id INT NOT NULL UNIQUE,
  FOREIGN KEY (personne_id) REFERENCES
Personne(id)
);
```

UNIQUE sur personne_id garantit qu'une carte est liée à une seule personne et qu'une personne n'a qu'une seule carte.

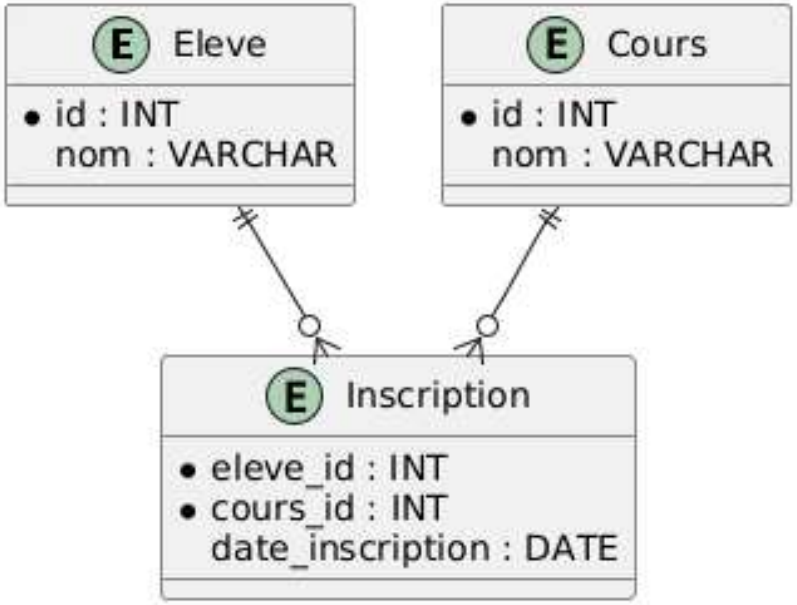
RELATION 1-N (ONE TO MANY)




```
CREATE TABLE Auteur (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100),
 prenom VARCHAR(100)
);

CREATE TABLE Livre (
  id INT AUTO_INCREMENT PRIMARY KEY,
  titre VARCHAR(200)  NOT NULL,
  auteur_id INT NOT NULL,
  FOREIGN KEY (auteur_id ) REFERENCES Auteur(id)
);
```

RELATION N-N (MANY TO MANY)

Élève (0,N) —— (0,N) Cours

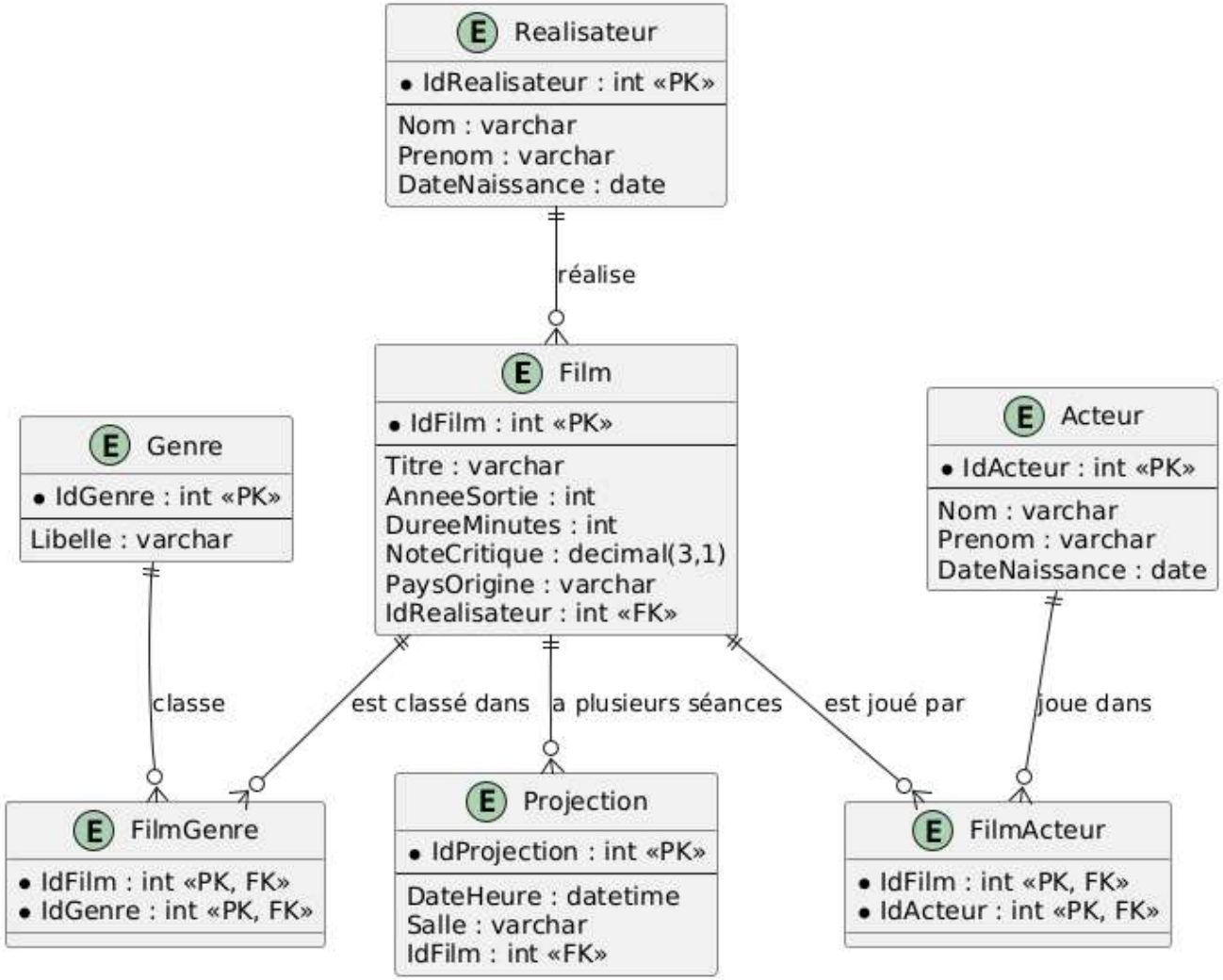



```sql
CREATE TABLE Eleve (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100),
);

CREATE TABLE Cours(
 id INT AUTO_INCREMENT PRIMARY KEY,
 nom VARCHAR(100),
);

CREATE TABLE Inscription (
  eleve_id INT NOT_NULL,
  cours_id INT  NOT NULL,
  date_inscription DATE,
  PRIMARY_KEY (eleve_id, cours_id),
  FOREIGN KEY (eleve_id) REFERENCES Eleve(id),
  FOREIGN KEY (cours_id) REFERENCES Cours(id)
);
```

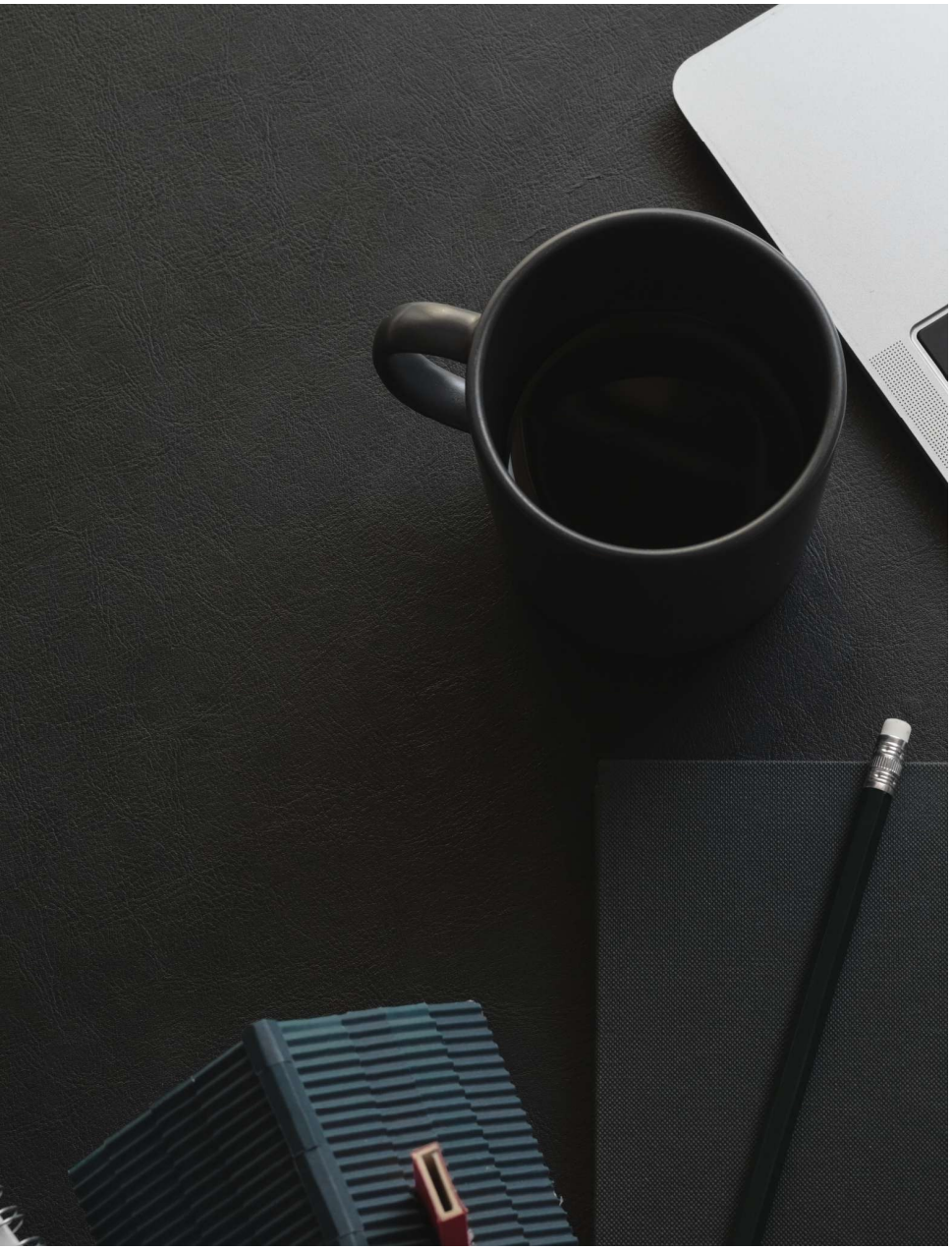
ATELIER

Implémentation de la base de données
“cineclub”

MLD


Realisateur
• IdRealisateur : int «PK»
Nom : varchar
Prenom : varchar
DateNaissance : date
réalise
Film
• IdFilm : int «PK»
Titre : varchar
AnneeSortie : int
DureeMinutes : int
NoteCritique : decimal(3,1)
PaysOrigine : varchar
IdRealisateur : int «FK»
Genre
• IdGenre : int «PK»
Libelle : varchar
Acteur
• IdActeur : int «PK»
Nom : varchar
Prenom : varchar
DateNaissance : date
classe
est classé dans
a plusieurs séances
est joué par
joue dans
FilmGenre
• IdFilm : int «PK, FK»
• IdGenre : int «PK, FK»
Projection
• IdProjection : int «PK»
DateHeure : datetime
Salle : varchar
IdFilm : int «FK»
FilmActeur
• IdFilm : int «PK, FK»
• IdActeur : int «PK, FK»

PARTIE 2 : REALISATION DE REQUÊTES



QUIZZ