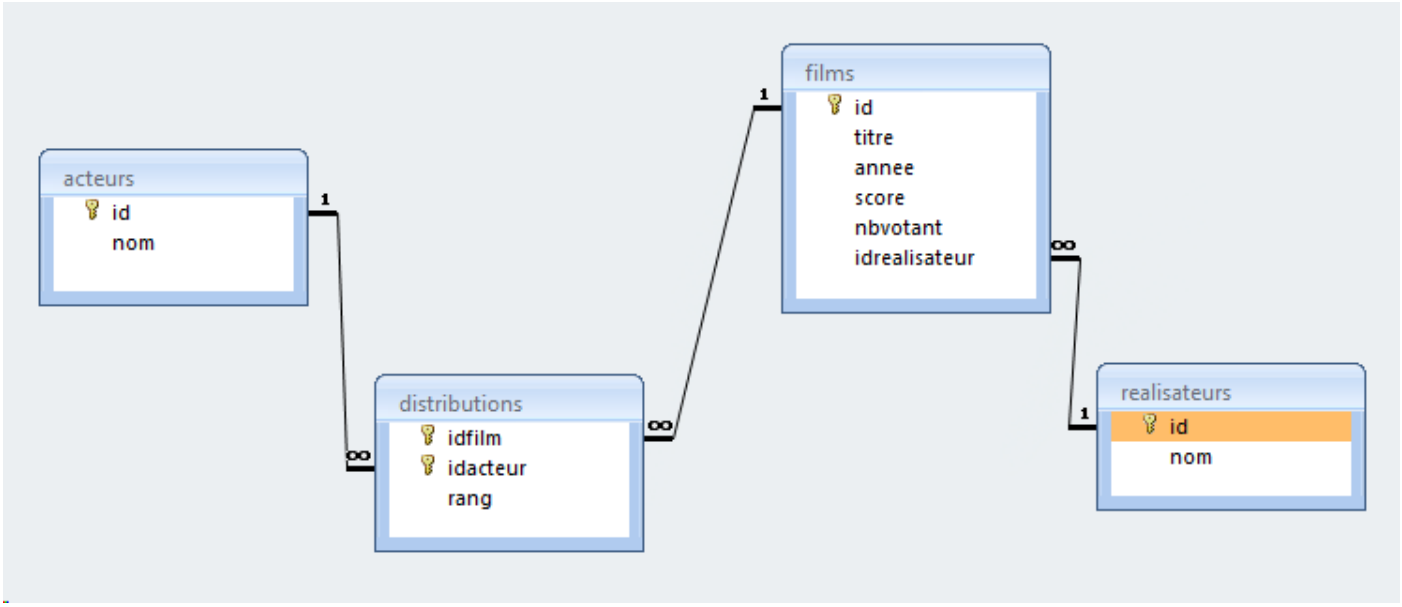
Soit une **base de données de films** et d'informations sur ces films (acteurs, réalisateurs, ...). Cette base est inspirée de ce site. Le schéma relationnel de cette base est décrit ci-dessous :



Il se concrétise dans les relations suivantes dans la base de données du TP (les clés sont soulignées et les clés étrangères sont en gras) :

- films (id, titre, annee, score, nbvotant, **idrealisateur**)
- acteurs (id, nom)
- distributions (**idfilm, idacteur**, rang)
- realisateurs(id, nom)

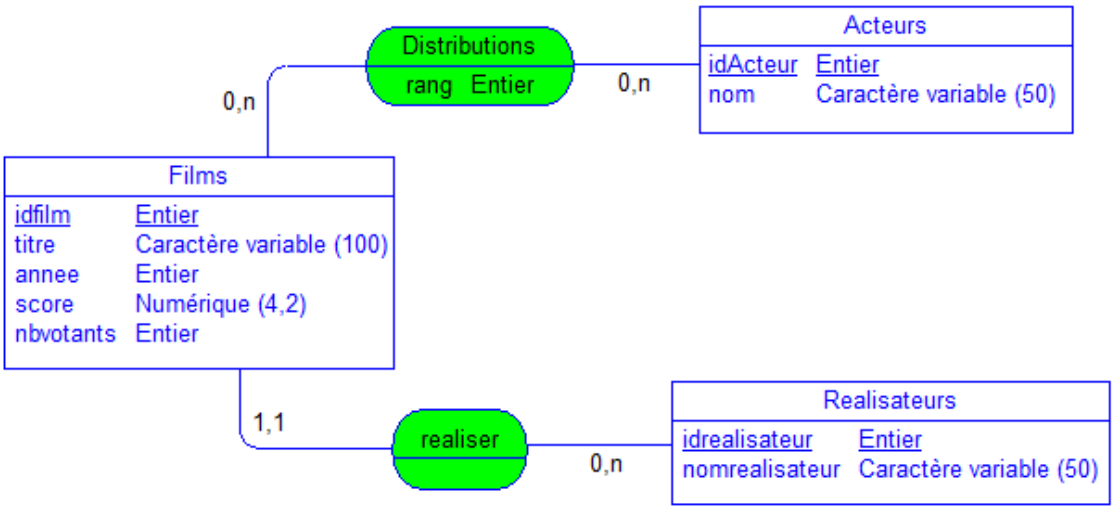
Un **film** est décrit de manière unique par un identifiant (entier, clé de la relation), un titre, une année (première sortie du film), un score (moyenne de tous les votes pour ce film), un nombre de votants et un réalisateur (entier, clé étrangère sur réalisateurs).

Un **acteur** est décrit de manière unique par un identifiant (entier, clé de la relation) et un nom.

Un acteur peut jouer un rôle dans zéro ou plusieurs films et un film a zéro ou plusieurs acteurs dans sa distribution. L'attribut rang indique le statut de l'acteur dans le film (rang=1 indique que l'acteur est la vedette du film). Le couple (idacteur, idfilm) est la clé de la relation **distributions**. idacteur est une clé étrangère sur acteurs et idfilm est une clé étrangère sur films.

Un **réalisateur** est identifié de manière unique par son identifiant (entier, clé de la relation) et un nom.

Pour votre information, le schéma de conception de cette base (schéma entité-association) est le suivant. Un chapitre de cours est consacré à l'étude des schéma entité-association.



1. Questions

1.1 Donner la liste des films (id, titre et annee) triée par titre.

160 tuples

```
SELECT titre
FROM films
ORDER BY titre ASC;
```

1.2 Donner la liste des films avec un score supérieur ou égal à 9 ?

4 tuples

```
SELECT titre
FROM films
WHERE score = 9 OR score > 9;
```

1.3 Quels sont les acteurs principaux (rang=1) des films sortis en 2000 ? On veut l'id et le nom triés par nom.

6 tuples

```
SELECT acteurs.id, acteurs.nom
FROM distributions
JOIN acteurs ON distributions.idacteur = acteurs.id
JOIN films ON distributions.idfilm = films.id
WHERE annee = 2000 AND rang = 1
ORDER BY nom ASC;
```

1.4 Donner les id des films sortis avant 1930 ou bien dans lesquels joue l'acteur de id = 12.

7 tuples (ou 8 pour <=)

```
SELECT DISTINCT  films.id
FROM distributions
JOIN acteurs ON distributions.idacteur = acteurs.id
JOIN films ON distributions.idfilm = films.id
WHERE films.annee < 1930 OR acteurs.id = 12;
```

1.5 Donner le nom des vedettes (rang=1) du film ayant l'id 11, dans l'ordre alphabétique.

1 tuple

```
SELECT acteurs.nom
FROM distributions
JOIN acteurs ON distributions.idacteur = acteurs.id
JOIN films ON distributions.idfilm = films.id
WHERE films.id = 11 AND rang = 1;
```

1.6 Donner les noms des réalisateurs de films sortis avant 1970 et ayant '*the*' dans leur titre.

25 tuples (SQLite n'est pas sensible à la casse de '*the*'. PostgreSQL renverrait 11 tuples, en différenciant '*the*' de '*THE*')

```
SELECT realisateurs.nom
FROM films
JOIN realisateurs ON films.idrealisateur = realisateurs.id
WHERE films.annee < 1970 AND LOWER(films.titre) LIKE '%the%';
```

1.7 Donner les id et titre des films n'ayant aucun acteur dans leur distribution.

6 tuples

```
SELECT films.id, films.titre, COUNT(distributions.idacteur) AS nbActeurs
FROM films
```

```
LEFT JOIN distributions ON distributions.idfilm = films.id
GROUP BY films.id
HAVING nbActeurs = 0;
-- le LEFT JOIN permet d'avoir les distributions de films avec des valeurs null qui, généralement, ne sont pas affiché par le DBMS.
```

1.8 Donner les films (id et titre) sortis en 2000 et ayant au moins deux acteurs dans leur distribution.

6 tuples

```
SELECT films.id, films.titre, COUNT(distributions.idacteur)
FROM films
JOIN distributions ON distributions.idfilm = films.id
WHERE annee = 2000
GROUP BY films.id
HAVING COUNT(distributions.idacteur) > 1;
```

1.9 Donner le(s) film(s) (id, titre) avec le score le plus haut.

4 tuples

```
SELECT id, titre, score
FROM films
WHERE score = (SELECT MAX(score) FROM films);
```

1.10 Donner la moyenne des votants par année de sortie de film.

72 tuples

```
SELECT annee, AVG(nbvotant)
FROM films
GROUP BY annee;
```

1.11 Donner le nombre de films par réalisateur (nom du réalisateur), trié par nombre de films décroissant. 103 tuples

```
SELECT COUNT(films.id), realisateurs.nom
FROM films
JOIN realisateurs ON films.idrealisateur = realisateurs.id
GROUP BY realisateurs.id
ORDER BY COUNT(films.id) DESC;
```

1.12 Donner les noms des acteurs n'ayant jamais joué, trié par nom d'acteur.30 tuples

```
SELECT acteurs.nom
FROM acteurs
LEFT JOIN distributions ON acteurs.id = distributions.idacteur
GROUP BY acteurs.id
HAVING COUNT(distributions.idacteur) = 0;
```

1.13 Donner les noms des acteurs n'ayant jamais été au rang 1. Le résultat sera trié par nom d'acteur. 779 tuples

```
SELECT acteurs.nom
FROM acteurs
LEFT JOIN distributions ON acteurs.id = distributions.idacteur AND distributions.rang = 1 -- seulement inclure les acteurs ayant 1 en rang
WHERE distributions.idacteur IS NULL -- inclus les acteurs qui n'ont pas 1 en valeur de rang
ORDER BY acteurs.nom;
```

1.14 Pour chaque film afficher son titre, son année de sortie, le nom du réalisateur, les noms de chacun des acteurs avec leur rang. Le résultat sera trié par année de sortie du film décroissant, titre du film croissant et par rang de l'acteur croissant. 1135 tuples

```
SELECT films.titre, films.annee, realisateurs.nom, acteurs.nom, distributions.rang
FROM films
JOIN realisateurs ON films.idrealisateur = realisateurs.id
JOIN distributions ON films.id = distributions.idfilm
JOIN acteurs ON distributions.idacteur = acteurs.id
ORDER BY films.annee DESC, films.titre ASC, distributions.rang ASC;
```