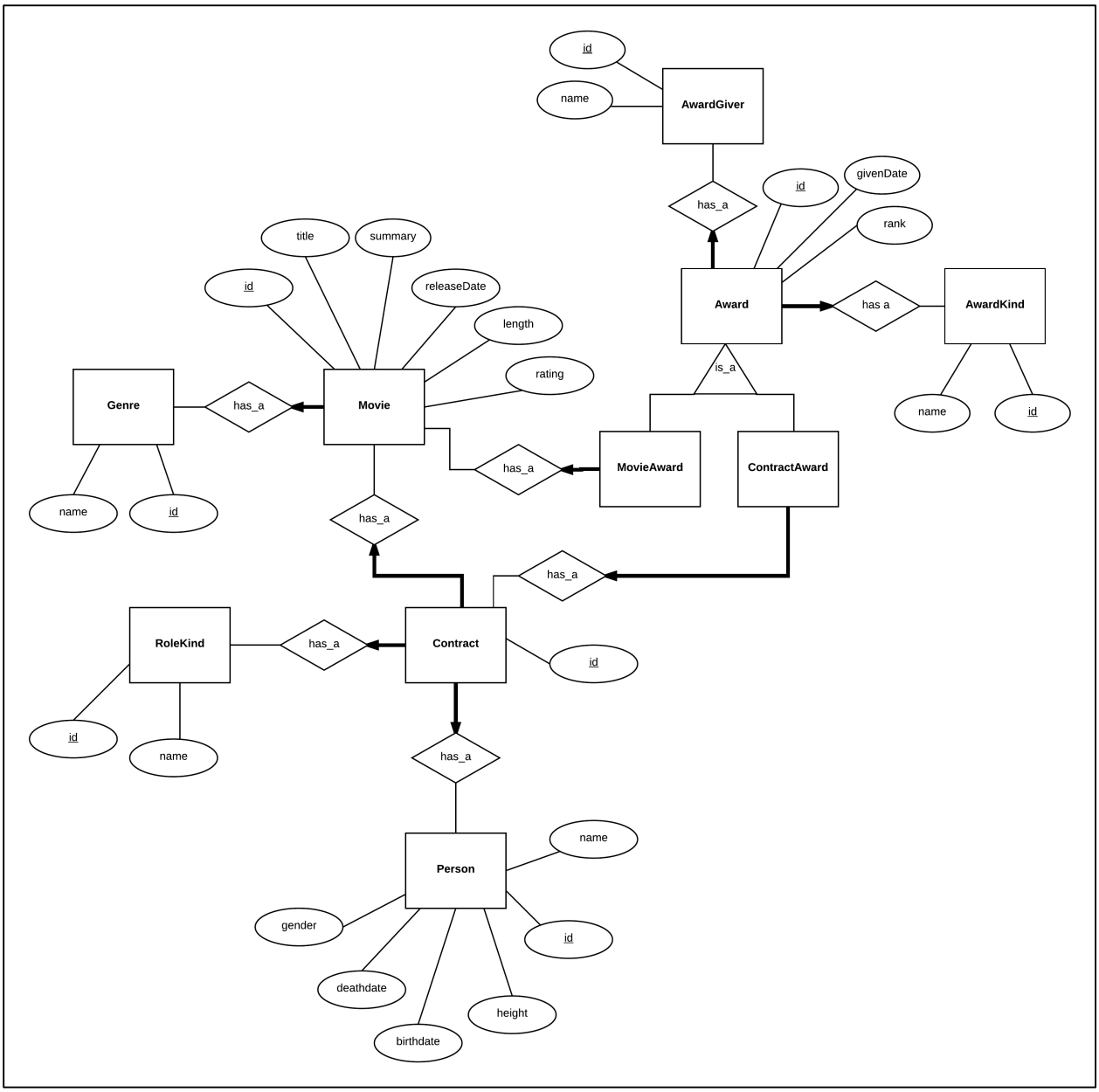
# Group Fight Club - Assignment 1

## Question 1

### ER-model



## Assumptions

- The **rating** attribute on the **Movie** entity is a number from 1 to 10, which is not calculated based on user reviews.
- A **Contract** entity exists for every role a **Person** have had in a given movie, showing for example if the person was camera man, actor or a director.
- The **rank** attribute on the **Award** entity decides the placing of the award that **Contract** or **Movie** won (eg. first place, second place)**.** If the rank is not defined, the **Award** entity simply corresponds to a nomination.
- An **Award** entity can only be awarded to a **Movie** or a **Contract** (a **Person's** role in a movie)**.** Therefore an **Award** can either be a **MovieAward** or a **ContractAward.**
- An **Award** has an **AwardGiver** and an **AwardKind**. The **AwardGiver** is the festival awarding the **Award** (eg. Academy Awards, USA). The **AwardKind** is the kind of the **Award** (eg. Best Motion Picture of the Year).
- An **AwardGiver** can exist without ever giving an **Award.**
- A **Genre** may exist without a **Movie** (eg. if we delete all movies, all the genres should not be deleted).
- A **Movie** may not exist without a **Genre.**
- An **Award** may not exist without an **AwardGiver** or an **AwardKind.**
- A **Contract** may not exist without a **Person**, **RoleKind** or **Movie.**
- For numerous reasons we chose to create the **RoleKind, AwardKind, AwardGiver** and **Genre** entities with **name** attributes. First of all, the entities are all part of zero-to-many relations. As an example, a **RoleKind** may have a relation to more than one **Contract**, so by having the **RoleKind** entity we avoid a lot of redundancy. Secondly, it becomes very easy to distinguish between entities with the same **name** attribute. If we for example created an awardGiverName attribute on the **Award** entity, it would be very hard to find all awards given by the same **AwardGiver**. Thirdly, it is very convenient that we can introduce more attributes later on. If we later wish to store the popularity of the genres or localized the genre, we can simply add an a popularity and german_name attributes to the **Genre** entity.

## Question 2

Person(*id*: int, *deathdate*: Date, *birthdate*: Date, *height*: int, *name*: string, *gender*: Enum)

Contract(*personId*: int, *movieId*: int, *roleKindId:* int)

Genre(id: int, name: string)

RoleKind(id: int, *name*: string)

Movie(*id*: int, *title*: string, *summary*: string, *releaseDate*: Date, *length*: int, *rating*: float, *genreId: int*)

Award(*id*: int, givenDate: Date, *rank*: int, *awardGiverId*: int, *awardKindId: int*)

MovieAward(*awardId*: int, *movieId*: int)

ContractAward(*awardId*: int, *contractId*: int)

AwardGiver(*id*: int, *name*: string)

AwardKind(*id*: int, *name*: string)

## Question 3

```
CREATE TABLE Genre (
    id          INT           PRIMARY KEY AUTO_INCREMENT,
    name        VARCHAR(50)   NOT NULL
);
```

```
CREATE TABLE Movie (
    id            INT           PRIMARY KEY AUTO_INCREMENT,
    title         VARCHAR(100)  NOT NULL,
    summary       TEXT,
    releaseDate   DATE          NOT NULL,
    length        INT           NOT NULL,
    rating        FLOAT         NOT NULL,
    genreId       INT           NOT NULL,
    FOREIGN KEY (genreId) REFERENCES Genre(id)
       ON UPDATE CASCADE
);
```

```
CREATE TABLE Person (
    id            INT              PRIMARY KEY AUTO_INCREMENT,
    name          VARCHAR(100)     NOT NULL,
    height        INT,
    birthdate     DATE,
    deathdate     DATE,
    gender        ENUM('MALE', 'FEMALE', 'OTHER')   NOT NULL
);
```

```
CREATE TABLE RoleKind (
    id          INT             PRIMARY KEY AUTO_INCREMENT,
    name        VARCHAR(50)     NOT NULL
);
```

```
CREATE TABLE AwardGiver (
    id            INT           PRIMARY KEY AUTO_INCREMENT,
    name          VARCHAR(200)  NOT NULL
);
```

```
CREATE TABLE AwardKind (
    id       INT     PRIMARY KEY AUTO_INCREMENT,
    name     INT     NOT NULL
);
```

```
CREATE TABLE Contract (
    movieId       INT          NOT NULL,
    roleKindId    INT          NOT NULL,
    personId      INT          NOT NULL,
    PRIMARY KEY (movieId, roleKindId, personId),
    FOREIGN KEY (movieId) REFERENCES Movie(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (roleKindId) REFERENCES RoleKind(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (personId)   REFERENCES Person(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```
CREATE TABLE Award (
    id              INT          PRIMARY KEY AUTO_INCREMENT,
    awardGiverId    INT          NOT NULL,
    awardKindId     INT          NOT NULL,
    givenDate       DATE         NOT NULL,
    rank            INT,
    FOREIGN KEY (awardGiverId) REFERENCES AwardGiver(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (awardKindId) REFERENCES AwardKind(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```
CREATE TABLE MovieAward (
    movieId       INT          NOT NULL,
    awardId       INT          NOT NULL,
    FOREIGN KEY (awardId) REFERENCES Award(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (movieId) REFERENCES Movie(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

```sql
CREATE TABLE ContractAward (
    contractId    INT          NOT NULL,
    awardId       INT          NOT NULL,
    FOREIGN KEY (awardId) REFERENCES Award(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
    FOREIGN KEY (contractId) REFERENCES Contract(id)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);
```

## Question 4

**Table:** Person
**FDs**: id → deathDate, birthDate, height, name, gender

**Table:** Contract
**FDs:** id → personId, MovieId, roleKind

Represents a persons contract with a movie, and what their role was in the production of the movie (not neccesarily who the played in the movie, but moreso if they were an actor, an instructor, a camera man etc.

**Assumptions:**
If you have two roles in the same film (for example both instructing and acting), you would have 2 records in the table, with a different roleKind. Alle 3 columns combine into a composite key, that uniquely defines your contract on that movie, with that given role, to avoid duplication (SQL Integrity will not let you define two keys that are the same).

**Table:** Movie
**FDs**: id → title, summary, releaseDate, length, rating, genre

**Assumptions:**
One could argue that the movie's title and the releaseDate would also constitute a key, thus not following BCNF. However, it is technically possible for two movies of the exact same name, to be released on the exact same day (ignore copyright issues).

**Table:** Award
**FDs:** id → awardGiver, awardKind, givenDate, rank

Represents a giver (Oscar's for example), a kind of award (Best Male Lead), when it was given, and your placement in the nomination (1st, 2nd, 3rd…).

**Table:** MovieAward
**FDs:** id → movieId, awardId

Represents an award given to the movie as a whole.

**Table:** ContractAward
**FDs:** id → contractId, awardId

Represents an award given to a persons role in (contract with) a movie.

**Table:** Genre
**FDs:** id → name

Represents a genre a movie can have.