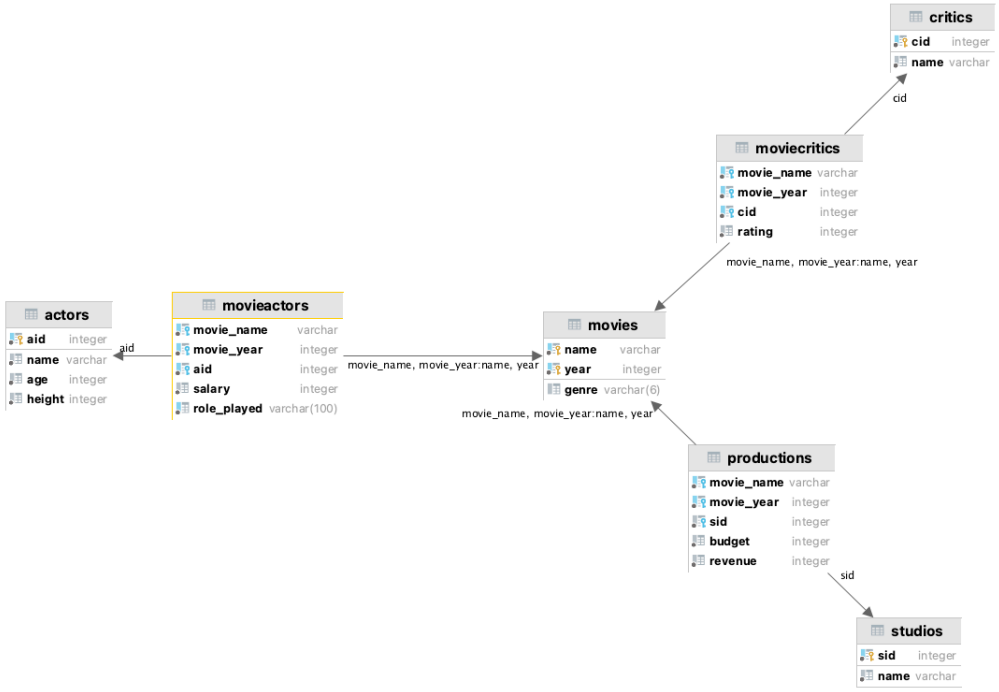
DATABASES HW2

ELIE NEDJAR: 336140116
BENJAMIN SULTAN: 931190987

The database design is described below



We have 4 basic tables which are describing Movies, Actors, Studios, and critics.
For each one the fields are exactly the same as the class described in the file.

Actor:

- aId (Integer): the id of the actor, it's a primary key because first, it must be unique and secondly creating an index on this field will accelerate the queries and because it will be used as a foreign keys in some tables and finally it must not be null.
- Name (Varchar): A not null string describing the name
- Height (Integer): A not null integer describing the height of the actor, with the constraint height > 0
- Age (Integer): A not null integer describing the age of the actor, with the constraint age > 0

Movies:

- Name (Varchar): The name of the movie (more on this one next)
- Year (Integer): The year of the movie's release. A movie is identified with the fields name and year, so we decided to define the pair as the primary for the same reasons as aId in actors
- Genre (Varchar(6)): The string that describe the genre of the movie. We used a constraint to ensure that it's one of the following (Drama, Action, Horror, Comedy)

Studio:

- sId (Integer): The Id of the studio. A studio is identified with his Id, so we define it as a primary key for the same reasons as above.
- Name(Varchar): the name of the studio, define as Not Null

Critics:

- cId (Integer): The Id of the critic. A critic is identified with his Id, so we define it as a primary key for the same reasons as above.
- Name(Varchar): the name of the critic, defined as Not Null

Moreover, as we can see on the diagram, the first described tables are related by some tables that allows us to respond to the needs of the basic API operation such as ActorPlayedInMovie, CriticRatedMovie, StudioProducedMovie (and the ones with "didn't")

MovieActors:

- Movie_name (Varchar): Name of the movie where the actor played
- Movie_year (Integer): Release year of the movie where the actor played, it's a foreign key together with Movie_name, that are referencing name and year of the table Movie
- aId (Integer): Id of the actor that is playing in the corresponding movie. It's a foreign key that is referencing to the aId field of the Actors table
- Salary (Integer): The salary of the actor for this movie, in order to ensure that the salary is not negative we put the constraint Salary > 0 for this field.
- Role_played (Varchar): One of the role that the actor play in the movie

Note: If an actor plays two different roles in the same movie, there will be two different rows for this actors and movie with two, each one having a different role. The unique constraint is on the fields (Movie_name, Movie_year, aId, Role_played)

MovieCritics:

- Movie_name (Varchar): Name of the movie that the critic rated
- Movie_year (Integer): Release year of the movie that the critic rated, it's a foreign key together with Movie_name, that are referencing name and year of the table Movie
- cId (Integer): Id of the critic that rated the corresponding movie. It's a foreign key that is referencing to the cId field of the Critics table
- Rating(Integer): The rate that the critic rated the movie. It's defined to be between 0 to 5

Note: A critic can rate a movie only one time, so the unique constraint is on the fields (Movie_name, Movie_year, cId)

Productions:

- Movie_name (Varchar): Name of the movie that the studio produced
- Movie_year (Integer): Release year of the movie that the studio produced, it's a foreign key together with Movie_name, that are referencing name and year of the table Movie
- sId (Integer): Id of the studio that produced the corresponding movie. It's a foreign key that is referencing to the sId field of the Studios table
- Budget(Integer): The budget of the studio for the movie, the budget isn't negative, so we put the constraint Budget >= 0
- Revenue(Integer): The revenue of the studio for the movie, the revenue isn't negative, so we put the constraint Budget >= 0

Note: A movie can be produced by only one studio, so the unique constraint is on the fields (Movie_name, Movie_year)

To accelerate queries of the APIs, we have created some views:

- ActorMovieNoRoles : It is the same as MovieActors except that each actor appear only once for each movie he plays in, without the roles.

```
'CREATE VIEW ActorsMoviesNoRole AS '
' SELECT movie_name, movie_year,aid,salary '
' FROM movieactors '
' GROUP BY movie_name, movie_year,aid,salary;'
```

- AverageRating : Contains each movies that has been rated at least once and the average grade on of it.

```
'CREATE VIEW AverageRating AS '
'  SELECT genre, movie_name,movie_year, avg(rating) as avg_rating '
'  FROM moviecritics mc '
'  JOIN movies m on mc.movie_name = m.name and mc.movie_year = m.year '
'  group by movie_name, movie_year, genre;',
```

- AvgRatingActors: Contains for each actors that played in at least one movie the average rating of the movie. If the movie wasn't rated the given average is 0

```
'CREATE VIEW avgRatingActors AS '
'    SELECT ma.aid, ma.movie_name, ma.movie_year, avg_rating, ag.genre'
'    FROM averagerating ag '
'    JOIN ActorsMoviesNoRole ma ON ag.movie_year = ma.movie_year AND ag.movie_name = ma.movie_name '
'UNION '
    'SELECT ma.aid, ma.movie_name, ma.movie_year, 0 as avg_rating, m.genre '
    'FROM moviecritics ag '
    'RIGHT OUTER JOIN ActorsMoviesNoRole ma ON ag.movie_year = ma.movie_year AND ag.movie_name = ma.movie_name '
    'JOIN movies m on ma.movie_year = m.year and ma.movie_name = m.name '
    'WHERE ag.rating IS NULL;',
```

- actorStudio: The view contains for each actor that plays in a movie that has been produced by any studio, the actor and the studio.

```
'actorStudio': 'CREATE VIEW actorStudio AS'
               '    SELECT aid, sid '
               '    FROM ActorsMoviesNoRole am '
               '    JOIN productions p on p.movie_name = am.movie_name and p.movie_year = am.movie_year;',
```

- MovieProdActors: The view contains for each movie the budget of the production and the sum of the salaries of the actors, If one of the above fields doesn't exists there is a Null value in the corresponding columns

```
'CREATE VIEW movieprodactors AS'
' select m.name, m.year, budget, sum(salary) as sum_salaries '
' from movies m '
' LEFT OUTER join productions p on m.name = p.movie_name and m.year = p.movie_year '
' LEFT OUTER join actorsmoviesnorole a on p.movie_name = a.movie_name and p.movie_year = a.movie_year'
' group by m.name, m.year, budget;'
```