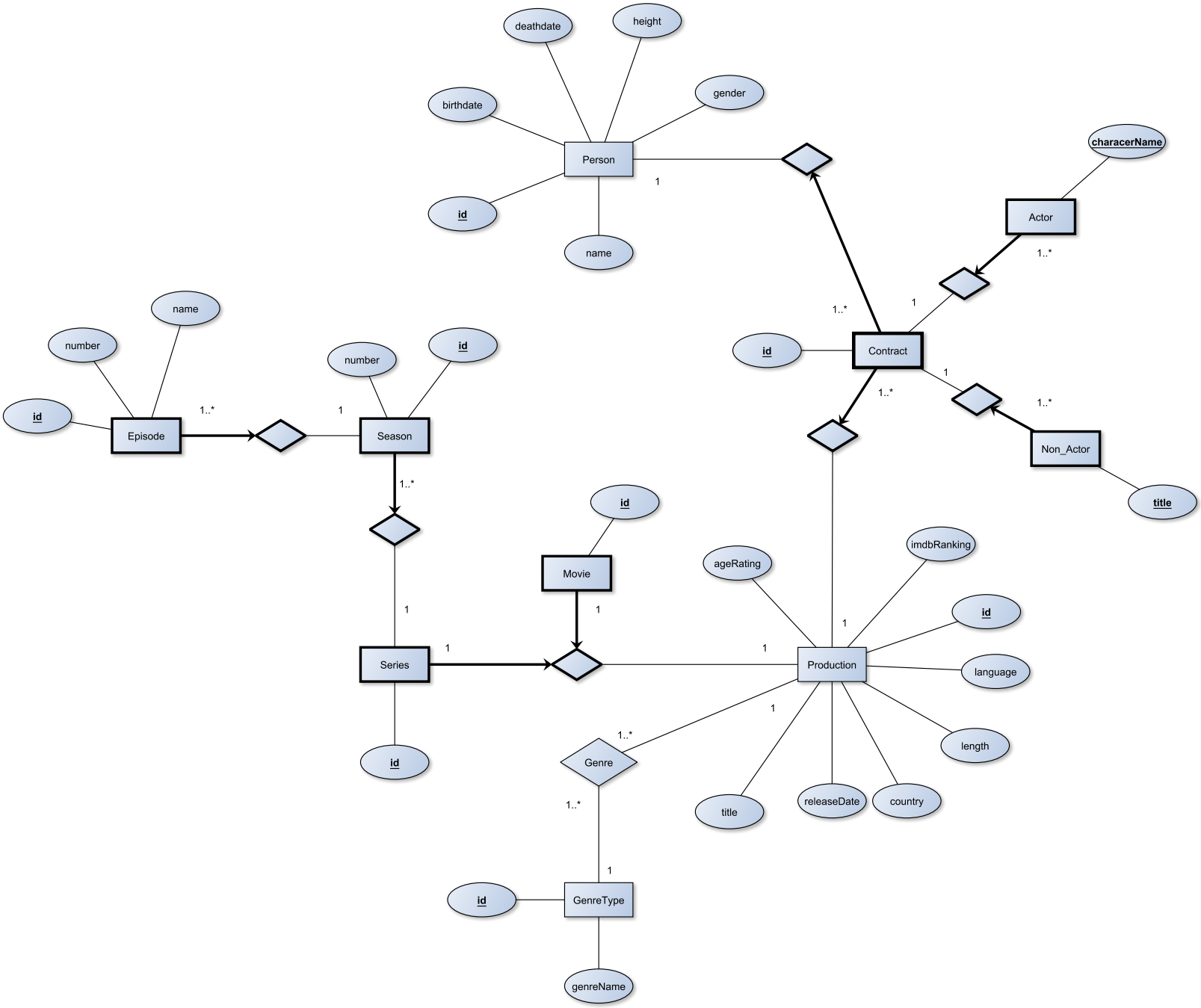
IT-University of Copenhagen

BIDD

Assignment 01

Group: Godfather
Alekxander Binderup(amab@itu.dk)
Kenneth Ry Ulrik(kulr@itu.dk)
Malthe Kirkro(maek@itu.dk)
Mikkel Gaub(mikg@itu.dk)
Omar Khan(omsh@itu.dk)

1 ERD


deathdate
height
birthdate
gender
Person
id
name
characerName
Actor
1..*
1
1..*
id
Contract
1
1..*
1..*
name
number
id
number
id
Episode
1..*
1
Season
Non_Actor
title
1..*
id
Movie
1
imdbRanking
ageRating
id
1
1
Series
1
1
Production
language
1
1..*
Genre
length
title
releaseDate
country
1..*
1
id
GenreType
genreName

2 SQL Schema

```
CREATE TABLE IF NOT EXISTS Production (
        id INT AUTO_INCREMENT,
        length DOUBLE,
        title VARCHAR(80) NOT NULL,
        releaseDate DATETIME,
        country VARCHAR(50),
        language VARCHAR(50),
        imdbRanking DOUBLE,
        ageRating VARCHAR(4),
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS Person (
        id INT AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL,
        birthdate DATETIME,
        deathdate DATETIME,
        gender BOOLEAN,
        height INT,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS Contract (
        id INT AUTO_INCREMENT,
        personId INT,
        productionId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (personId) REFERENCES Person(id),
        FOREIGN KEY (productionId) REFERENCES Production(id)
);

CREATE TABLE IF NOT EXISTS Actor (
        contractId INT,
        characterName VARCHAR(50) NOT NULL,
        PRIMARY KEY (contractId, characterName),
        FOREIGN KEY (contractId) REFERENCES Contract(id)
);

CREATE TABLE IF NOT EXISTS Non_Actor (
        contractId INT,
        title VARCHAR(50) NOT NULL,
        PRIMARY KEY (contractId, title),
        FOREIGN KEY (contractId) REFERENCES Contract(id)
```

```
);

CREATE TABLE IF NOT EXISTS GenreType (
        id INT AUTO_INCREMENT,
        genreName VARCHAR(25) NOT NULL,
        PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS Genre (
        productionId INT,
        genreTypeId INT,
        PRIMARY KEY (productionId, genreTypeId),
        FOREIGN KEY (productionId) REFERENCES Production(id),
        FOREIGN KEY (genreTypeId) REFERENCES GenreType(id)
);

CREATE TABLE IF NOT EXISTS Movie (
        id INT AUTO_INCREMENT,
        productionId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (productionId) REFERENCES Production(id)
);

CREATE TABLE IF NOT EXISTS Series (
        id INT AUTO_INCREMENT,
        productionId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (productionId) REFERENCES Production(id)
);

CREATE TABLE IF NOT EXISTS Season (
        id INT AUTO_INCREMENT,
        seriesId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (seriesId) REFERENCES Series(id)
);

CREATE TABLE IF NOT EXISTS Episode (
        id INT AUTO_INCREMENT,
        seasonId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (seasonId) REFERENCES Season(id)
);
```

3 Assumptions and choices

- Any production is assumed to have a title.
- A person has a name.
- A person is either a man or a woman, represented as a value of either 1 or 0 respectively.
- A contract is made between a single person and a single production.
- A person can have several different roles within a production through a contract. These roles are defined as either an actor or not an actor (ex. writer or director).
- A production can have more than one genre.
- A production is either a movie or a series.
- A series consists of one or more seasons, which consists of one or more episodes.

4 Analysis of DBMS design

Instead of going through every table in the SQL-Schema we have decided to discuss the areas that we believe could seem to interfere with the 'Boyce Codd' normal form.

4.1 Genre

The relationship between the tables GenreType and Production is done with the Genre relationship, which contains references to the aforementioned tables' primary keys, as foreign key constraints. This could seem redundant as a lot of GenreTypes will reoccur in the Genre table but the alternative, which is not having the "Genre" relationship, would force us to repeat genre names multiple times as well as not allowing us to look up the genre types that can be associated with a production. Having the relationship adds this possibility and replaces the duplication of genre names with id values.

4.2 Production

It is possible that multiple attributes such as the 'country' attribute in the relation "Production" should be treated as we treated GenreType, but since it would involve the creation of several additional tables, which would exceed the constraints of the assignment(maximum 15 tables) we have decided to showcase GenreType as an example of optimizing the schema.

4.3 Inheritance

The combination of the relations; Production, Movie, Series (hereunder Season and Episode), contributes to avoid a major amount of redundancy, as the alternative of storing loads of information in each table would introduce. This is solved by containing all common data fields in the Production relation, and having the previously mentioned Movie and Series tables relate to a single entry in the Production relation. Following this, Season and Episode contain little to no additional information, albeit a single Episode will usually have an individual name, setting it apart from the other episodes produced within the series.