IT-University of Copenhagen

BIDD

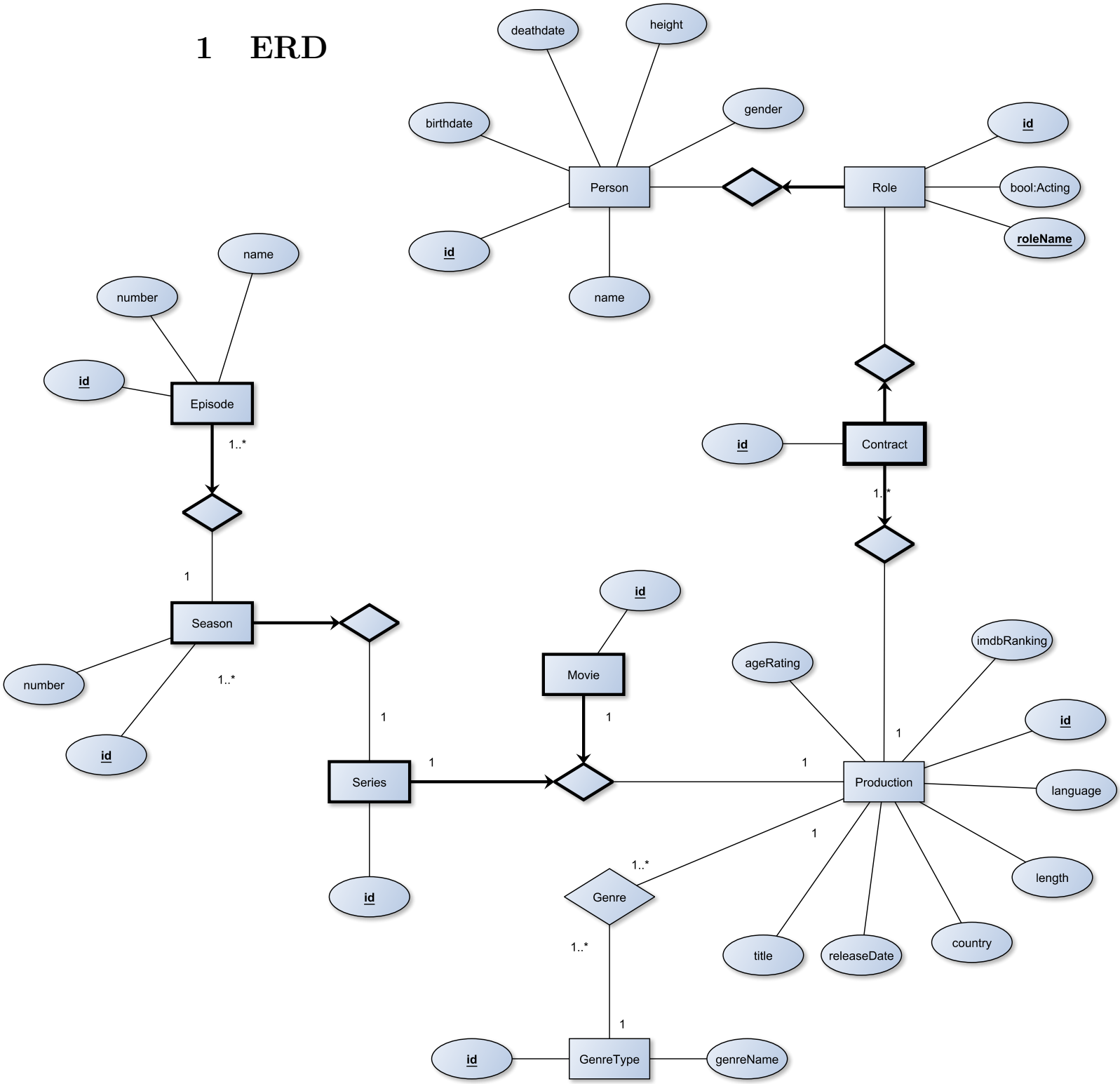
September 17, 2014

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
Role
id
bool:Acting
roleName
name
number
id
Episode
1..*
1
Season
number
1..*
id
id
Contract
1..*
id
Movie
1
1
Series
1
id
ageRating
imdbRanking
id
1
1
Production
language
1
length
1..*
Genre
1..*
title
releaseDate
country
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),
        imdbRating 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 Role (
        id INT AUTO_INCREMENT,
        personId INT,
        acting BOOLEAN NOT NULL,
        roleName VARCHAR(25),
        PRIMARY KEY (id, roleName),
        FOREIGN KEY (personId) REFERENCES Person(id)
);

CREATE TABLE IF NOT EXISTS Contract (
        id INT AUTO_INCREMENT,
        roleId INT,
        productionId INT,
        PRIMARY KEY (id),
        FOREIGN KEY (roleId) REFERENCES Role(id),
        FOREIGN KEY (productionId) REFERENCES Production(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

- A 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 various contracts. These roles are defined as either an actor or not an actor (e.g. writer or director).
- A production can have more than one genre.
- A production is either a movie or a series.
- The country of a production is determined by the origin of the producing film company.
- A series consists of one or more seasons, which consists of one or more episodes.

3.1 Extra notes

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.

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.

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 none additional information, albeit a single Episode will usually have an individual name, setting it apart from the other episodes produced within the series.

4 Analysis of DBMS design

In every single relation in our ERD, every single attribute is functionally dependent on the "id" attribute, as it alone ensures that there can be no absolute dublicates. Simultaneously we need to control, that when not considering the id attribute's existance, that there can be no two entries, where the only attribute taking them apart from each other is the id attribute. Therefore we will go through and identify all possible candidate keys for all our relations, making sure that there is not two candidate keys, involving the same attribute more than once (Again: Excluding id from this candidate key search). Associated assumptions that would have to be made for a candidate key to be valid, will be mentioned in a column of its own. See the table below for the result of this relation search exhaustion.

Relation	**Functional Dependencies**	**(Related) Assumptions**
Production **attributes:** *AgeRating, Title, ReleaseDate, Country, Length, Language, IMDBRating*	Title, ReleaseDate → *All other attributes* Title, Length → *All other attributes*	No 2 Productions with a sharing releasedate and title No 2 Productions with the same length and title
Person **attributes:** *Name, Gender, BirthDate Height, DeathDate*	Name, BirthDate → *All other attributes*	No 2 person share a name and a birthdate

It doesn't make sense to go through any of the other tables, as pretty

much all of them do not work independently, and are thus bound by their stored foreign keys being part of their primary key, making it trivial to look for possible candidate keys that could form functional dependencies.

4.1 Episode-table

The Episode relation is worth taking a look at, as it is a table which nobody references towards. Aside from that, it does contain a few data attributes (title and number), along with containing a reference to the season in which the episode consists. This suggests that one would have to make the primary key (if not using an ID attribute) be a combination of all this data (season_id, title, number).
You need to connect the season id, since otherwise you would assume that no episode would have the same title and sequence number in two different seasons of two different (or the same) series. While the key that combines these 3 attributes (including the foreign key) essentially does the trick, in making our Episode unique, we follow the standard of giving it an id, in the case of database expansion being required in the future, and needing to reference to a specific episode for one reason or another. This is also the explanation for any other case in the database, where a unique identifier in the table was not necessarily required, but was used in order to avoid duplication in tables referring to that table.