GROUP FELLOWSHIP OF THE RING

Assignment 1

Dennis Thinh Tan Nguyen, William Diedrichsen Marstrand, Thor Valentin Olesen, Pernille Lous, Nicklas Johansen

Problem 1

Question 1

1. Change: Thin arrow from department to Works_in to thin line (0..*)
2. Change: Thin arrow from Employee to Works-in to bold Arrow (1..1)
3. Change: employeeID to primary key
4. Change: Bold line from Works on to Project to thin line
5. Change: thin line from Manager to manages to bold line
6. Change: Thin line from researcher to works on to bold arrow
7. Delete: attribute project_name from Project (Because it was not stated in the given properties)

Question 2

1. None:
    a. Because all attributes, relations and entities from the properties description can be found in the E/R Diagram

Question 3

1. Bold, arrow from employee to Works_in
2. Bold, line from Manager to manages
3. Bold, arrow from Researcher to Works on
4. Bold, arrow from manages to Project

Problem 2

Question 4

```
CREATE TABLE exhibition(
exhibitionID int(11) NOT NULL AUTO_INCREMENT,
exhibition_Date timestamp(4) NOT NULL,
room_Number int NOT NULL,
PRIMARY KEY (exhibitionID)
);

CREATE TABLE artist(
name varchar(20) NOT NULL,
artist_number int(11) NOT NULL AUTO_INCREMENT,
ssn int(11) NOT NULL,
PRIMARY KEY (artist_number)
);

CREATE TABLE painting(
paintingID int NOT NULL AUTO_INCREMENT,
artist_number int(11) NOT NULL,
title varchar(20) NOT NULL,
Price float(10,2) NOT NULL,
medium varchar(20) NOT NULL,
FOREIGN KEY (artist_number) REFERENCES artist(artist_number)
ON DELETE CASCADE,
PRIMARY KEY (paintingID)
);
```

```
CREATE TABLE showcase(
exhibitionID int(11) NOT NULL,
paintingID int(11) NOT NULL,
FOREIGN KEY (exhibitionID) REFERENCES exhibition(exhibitionID),
FOREIGN KEY (paintingID) REFERENCES painting(paintingID),
PRIMARY KEY (exhibitionID, paintingID)
);
```

Question 5

- Exhibition: Bold line between exhibition and showcases
  - Add NOT NULL to the foreign key: exhibitionID
- Painting : Bold Arrow between Painting and paints
  - Add NOT NULL to the foreign key: artist_number

Problem 3

Question 5

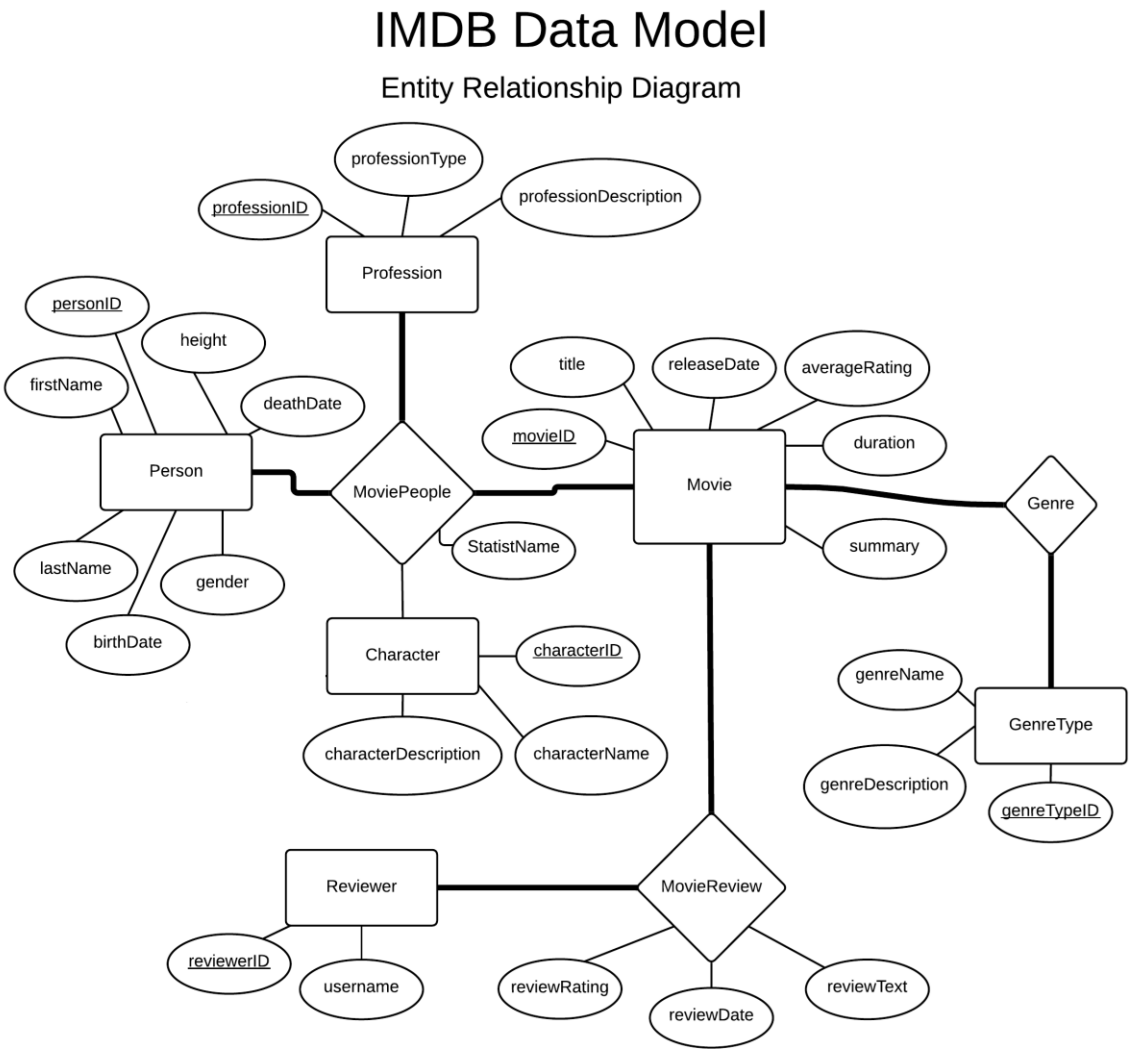
1. –
    a. Yes
    b. Yes
    c. No
    d. Yes
    e. Yes
2. –
    a. C → A
    b. A → C

Question 6

1. Correct answer is (d)
2. –
    a. Yes
    b. No
    c. Yes
    d. No
3. False
4. True

Problem 4

Question 7



Note:

StatistName

StatistName in MoviePeople is an attribute that defines a non-important character (eg. Janitor1). We have added this attribute to avoid redundant characters in the Character table. For an example if movie A and Movie B both have a statist called “Janitor1”, this would require that we in fact have to add multiple entries for “Janitor1”. Given that “Janitor1” in movie A is not the same janitor as “Janitor1” in movie B.

Person

Person defines any credited person working in a movie production.

MoviePeople

Holds the relations between movie, person, character and profession through foreign keys.
The idea behind MoviePeople lies in the fact that it creates an opportunity to retrieve relational data through various queries.

Question 8

```
Person(personId: int, firstName: string, lastName: string, gender: ENUM,
birthDate: date, deathdate: date, height: real)

Character(characterID: int, characterName: string, characterDescription:
string)

Profession(professionID: int, professionDescription: string, professionType:
String)

MoviePeople(personID: int,  professionID: int, movieID: int, characterID: int,
statistName: string)

Movie(movieID: int, title: string, releaseDate: date, averageRating: real,
duration: int, summary: string)

MovieReview(movieID: int, reviewerID: int, reviewText: string, reviewDate:
date, reviewRating: real)

Reviewer(reviewerID: int, username: string)

Genre(genreTypeID: int, movieID: string)

GenreType(genreTypeID: int, genreName: string, genreDescription: string)
```

Question 9

```
CREATE TABLE Person
(
personID INT NOT NULL AUTO_INCREMENT,
firstName VARCHAR(50) NOT NULL,
lastName VARCHAR(50) NOT NULL,
gender ENUM('female', 'male', 'unspecified'),
birthDate DATE,
deathDate DATE,
height REAL,
PRIMARY KEY (personID)
);

CREATE TABLE Character_
(
characterID INT AUTO_INCREMENT,
characterName VARCHAR(100),
characterDescription VARCHAR(255),
PRIMARY KEY (characterID)
);

CREATE TABLE Profession
(
professionID INT NOT NULL AUTO_INCREMENT,
professionDescription VARCHAR(255),
professionType VARCHAR(40),
PRIMARY KEY (professionID)
);

CREATE TABLE Movie
(
movieID INT NOT NULL AUTO_INCREMENT,
```

```
title VARCHAR(100) NOT NULL,
releaseDATE DATE NOT NULL,
averageRating REAL,
duration INT NOT NULL,
summary VARCHAR(2000),
PRIMARY KEY (movieID)
);



CREATE TABLE Reviewer
(
reviewerID INT NOT NULL AUTO_INCREMENT,
userName VARCHAR(20) NOT NULL,
PRIMARY KEY (reviewerID)
);

CREATE TABLE GenreType
(
genreTypeID INT NOT NULL AUTO_INCREMENT,
genreName VARCHAR(50) NOT NULL,
genreDescription VARCHAR(250),
PRIMARY KEY (genreTypeID)
);
```

```
CREATE TABLE Genre
(
genreTypeID INT NOT NULL,
movieID INT NOT NULL,
FOREIGN KEY (genreTypeID) REFERENCES GenreType(genreTypeID)
ON DELETE CASCADE,
FOREIGN KEY (movieID) REFERENCES Movie(movieID)
ON DELETE CASCADE,
PRIMARY KEY(genreTypeID , movieID)
 );

CREATE TABLE MovieReview
(
movieID INT NOT NULL,
reviewerID INT NOT NULL,
reviewText VARCHAR(255),
reviewDate DATE NOT NULL,
reviewRating REAL,
FOREIGN KEY (movieID) REFERENCES Movie(movieID)
ON DELETE CASCADE,
FOREIGN KEY (reviewerID) REFERENCES Reviewer(reviewerID)
ON DELETE CASCADE,
PRIMARY KEY (movieID,reviewerID)
);

CREATE TABLE MoviePeople
(
personID INT NOT NULL,
professionID INT NOT NULL,
movieID INT NOT NULL,
characterID INT,
statist VARCHAR(55),
```

```
FOREIGN KEY (personID) REFERENCES Person(personID)
ON DELETE CASCADE,
FOREIGN KEY (professionID) REFERENCES Profession(professionID)
ON DELETE CASCADE,
FOREIGN KEY (movieID) REFERENCES Movie(movieID)
ON DELETE CASCADE,
FOREIGN KEY (characterID) REFERENCES Character_(characterID)
ON DELETE CASCADE,
PRIMARY KEY (personID, professionID, movieID, characterID)
);
```

Question 10:
The relations are illustrated as tables beneath containing Primary Keys and Functional Dependencies (FD).

Person:

Primary Keys
personID
FD's
personID --> firstName, lastName, birthday, gender, deathDate, height

Profession:

Primary Keys
professionID
FD's
professionID --> professionType, professionDescribtion
professionType --> professionDescription, professionID

Using professionID because it's an int which will save memory in relation MoviePeople

MoviePeople:

Primary Keys
(personID, professionID, movieID, characterID)

No FD's means that the table is in BCNF

Character:

Primary Keys
characterID
FD's
characterID --> characterName, characterDescription
characterName --> characterID, characterDescription

Using characterID because of the same reason as presented in the Profession relation.

Movie:

Primary Keys
movieID
FD's
movieID --> title, releaseDate, averageRating, duration, summary
(title, releaseDate) --> movieID, averageRating, duration, summary

It is assumed the movieID is functional dependent on title and releaseDate, since it is extremely unlikely that multiple movies with the same title are released on the same date.

GenreType:

Primary Keys
genreTypeID
FD's
genreTypeID --> genreName, genreDescription
genreName --> genreDescription

Genre:

The relation genre has been omitted since it only contains foreign keys

MovieReview:

Primary Keys
(reviewerID, movieID)
FD's
(reviewerID, movieID) -->reviewRating, reviewDate, reviewText

Reviewer:

Primary Keys
reviewerID
FD's
reviewerID --> userName

Conclusion:

Since the left hand side of all the FD's in the relations are super keys, they all fulfill the requirements of BCNF. Thus the design of the database is in normal form.