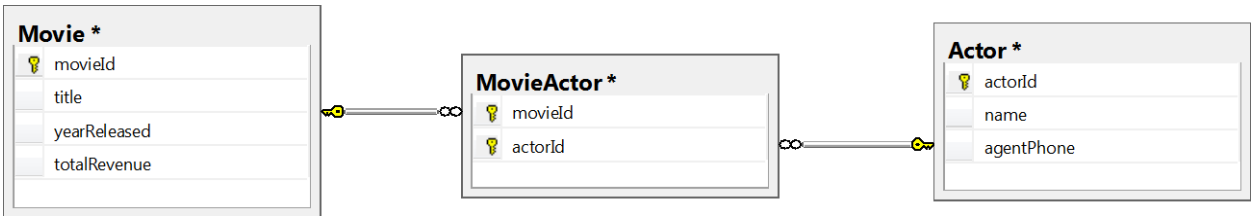
Overview

In the United States, the movie industry has proven to be one of the most resilient to periods of economic recession. Movie studio executives, for example, reported that total U.S. box office revenues exceeded $10 billion for the first time ever during the recession in 2009, with revenues in North America exceeding $11.25 billion in 2016. In an effort to better understand the factors that affect her studio's profitability, one of these executives wants to explore the relationship between a movie's box office revenues and its actors. If certain actors are found to be associated with higher box office revenues, the studio executive might consider hiring analytics personnel to more carefully study the relationships among actors, revenues, salaries, directors, production costs, etc.

Details

To begin the studio executive's project, a database of actors and movies will need to be created. The design that has been chosen for this database is shown in the figure below:



The ***Movie*** table holds information for all of the movies that are being considered. Attributes for the movie table include:

- ***movieID*** – Primary key. A unique integer that uniquely identifies this movie.
- ***title*** – The title of this movie. Maximum length = 100 characters.
- ***yearReleased*** – The calendar year in which this movie was released.
- ***totalRevenue*** – The total worldwide gross revenue (in U.S. dollars) earned by this movie.

The ***Actor*** table holds information for all of the actors that are being considered. Attributes for the actor table include:

- ***actorID*** – Primary key. A unique integer that uniquely identifies this actor.
- ***name*** – The name of this actor. Maximum length = 50 characters.
- ***agentPhone*** – The phone number of the agent who represents this actor (TIP: use a data type that supports text when defining this attribute).

The ***MovieActor*** table serves as an associative entity (intersection table) that enables a many-to-many relationship between the movie and actor tables. This allows a movie to have more than one actor, and allows each actor to appear in more than one movie. The only attributes in this table are the primary keys from the movie and actor tables, which together comprise a composite primary key for the MovieActor table, and individually serve as foreign key links back to their respective parent tables.

Tasks

Your tasks for this assignment are to:

1. Write a series of SQL statements that will fully implement the database depicted above in Microsoft SQL Server, including tables, attributes, data types, keys, and relationships.
2. Write a series of SQL statements that will insert the following data into your database:

Movie

movieID	*title*	*yearReleased*	*totalRevenue*
101	Titanic	1997	2,185,372,302
102	The Avengers	2012	1,511,288,162
103	Star Wars: The Force Awakens	2015	2,068,178,225
104	Indiana Jones 4	2008	786,636,033
105	Star Trek	2009	385,680,446
106	Iron Man 3	2013	1,147,234,000

Actor

actorID	*name*	*agentPhone*
1	Kate Winslet	265-555-3267
2	Robert Downey, Jr.	443-555-2300
3	Harrison Ford	649-555-3992
4	Chris Hemsworth	668-555-7475

MovieActor

movieID	*actorID*
103	3
101	1
105	4
106	2
104	3
102	4
102	2

3. Write a single SQL statement that will select the name of each actor and the average box office revenue for all of the movies in which that actor has appeared. These results should be sorted by average revenue in descending order (largest to smallest).
4. Create all of your SQL statements for tasks 1-3 above in a single SQL Server query window, and save the results as a SQL Server query file (i.e., a ***.sql*** file). Your statements should be properly ordered so that all of the above tasks will be completed in succession when the statements are executed.

   TIP #1: You may want to consider including the following three statements at the very top of your SQL Server query window:

```
IF OBJECT_ID('MovieActor') IS NOT NULL DROP TABLE MovieActor;
IF OBJECT_ID('Movie') IS NOT NULL DROP TABLE Movie;
IF OBJECT_ID('Actor') IS NOT NULL DROP TABLE Actor;
```

   These statements will drop (delete) the MovieActor, Movie, and Actor tables if they already exist in your database, thus allowing you to start with a "clean slate" each time you execute your queries.

   TIP #2: Using an integer (*int*) data type for the *agentPhone* attribute may seem logical, but doing this is actually not a good idea. The *int* data type is a 32-bit signed integer, and can therefore hold a maximum value of 2^{31} - 1 or 2147483647. In the U.S. phone system, using an *int* to store a 10-digit phone number would therefore cause an error for any area code greater than 214.

   TIP #3: Use a floating-point data type (e.g., *float*) for the *totalRevenue* attribute. This will allow you to avoid problems with integer division when computing each actor's average revenue per movie.

Deliverables

To ensure that you receive credit for this assignment, please complete the following tasks:

1. Complete the *Group Assignment Participation Form* for this assignment (available on the course website). **Each group member who contributed to the assignment should complete this task.**
2. Submit the ***.sql*** file that contains your group's SQL statements for tasks 1-3 above using the appropriate link on the course website. Each group should submit just one file. Remember: your SQL statements should be properly ordered so that tasks 1-3 will be completed in succession when the statements are executed.