WeCookRawData

Members:
Gokul Vamsi Thota - 2019111009
T.H.Arjun - 2019111012
Mukkara Rakesh Reddy - 2019101087

Mini-World: Movie Studio

The mini world that we choose for the Group Project is a Movie Studio Environment, a place which is brimming with life, depicted and portrayed by film artists. The database will be used by a Movie Studio to manage Movie projects including details on movies, sets, actors, directors, producers etc. The users will be the management and marketing teams of Studio. They will be using the data for analytics and managing the studio and the projects.

Entities

Following are **5 entities** of the Mini-World:

1. Movie -

The world to which we escape to when we are stressed / bored with our own lives, and to re-discover our joy, on a screen, is the world of a movie; which is one of the most key components, of both the film studio and entertainment. The studio contains all the data regarding the movies which have been and are being produced by the studio.
It has two key attributes:
Production ID, and a **combination of Director ID and Film Number of Director (Film number = number of films he directed + 1)** which is a **composite key** , which are both unique for every movie. **So it has two key attributes.** The entity 'movies' has the following attributes:

a. Name - It contains the name of the Movie. It is a simple attribute and is represented by an *Alpha-Numeric String. A **constraint** is defined here such that it cannot contain more than 50 characters.*
b. Movie ID- The ID Provided by the studio for each movie project. It is unique, i.e, a key attribute, and can only be *a natural number.*
c. Awards- It describes the awards won by the movie. It is a multi-valued attribute, and contains the awards in *String format.*
d. Director ID- This stores the Director ID which is unique for each director (assigned by studio), i.e, the key attribute of Director entity. If more than one director is directing a movie, then any one director's ID is stored as this attribute's value. *It is a Natural Number, and can be stored in an integer.*
e. Film Number of Director - This stores the Film Number. It is defined as the number this film bears considering the number of works in his career. If there is more than one director, only one is considered, i.e, the corresponding value

associated with the director, whose Director ID was chosen for representing the movie. *It is a natural number.*
The combination of the Director ID and Film No. of Director *acts as a key attribute as it is unique for a movie. It is a composite key.*

f. Average Rating - It is the average rating that the movie received. It is a simple attribute. *For a movie still in production it is NULL. The **constraint** defined on it is that it should have a decimal value (to be stored in float) which is greater than or equal to 0 and less than or equal to 5.*
g. Profit/Loss- It is the profit or loss that the movie incurred to the studio. *It is represented by a float, and is negative if it is a loss and positive if it is a profit. It is **Derived from Budget and Collections.***
h. Cast- It describes the cast which is associated with the movie. It is a multi-valued attribute and the names of casts can be represented in *String format.*
i. Budget- The Budget that was spent on making the Movie. It's a simple attribute, represented by a *Decimal.*
j. Collections- The box office collections that the movie gained. It's a simple attribute, represented by an *Integer.*
k. Release Date- It is a simple attribute, which *contains the release date, if the film is released. Otherwise it is set to NULL.*

2. Director -

If a film is considered a ship, a director is the captain of the ship, who is capable of both sinking the ship or destroying its competitors, by conquering the box office. The 'Director ' entity is the generalisation of ‘main director’ and ‘assistant director’. It contains all data on directors. The entity Director has the following attributes.

a. Name- It contains the name of the Director. *It's a simple attribute represented by a string. A **constraint** is defined here such that it can contain at most 30 characters.*
b. Director ID- It's a key Attribute of the 'Director' Entity. A unique ID given to each director by the studio, represented by an *Integer.*
c. Date of Birth (DOB)- It's a simple attribute, and contains the date of birth of the director.
d. Age- Age of the director. It is *derived from DOB. It's a natural number (simple attribute), represented by an Integer.*
e. Filmography - The number of movies a director has worked/ working on. It's a simple attribute and is always a natural number, thus represented by an *Integer.*
f. Address(es) - It describes the country, state, city and other details of the addresses the director has lived in. *It is a **composite, multi-valued attribute** which is represented by strings.*
g. Accolades - It describes the awards and appreciation received by the director throughout his career. *It is a **multi-valued attribute** and can be represented by strings.*

h. Type- It consists of one of the two values : 1 if the person is a main director, or 0, if the person is an assistant director.

3. Set-

The human eye desires beauty, but it is not always possible to capture some beautiful locations as desired. But creativity and entertainment know no bounds, and hence, the entity 'set' enters the scene, and facilitates this! The entity 'set' provides data on all the sets available in the movie studio. It represents the sets being used by the movies in production. 'Set' is a **weak entity**, as a single location can have multiple desirable sets. The relationship it depends on for identification is the one with movies. A particular set is identified from the movie and Production ID which is the key attribute of a movie and its relationship 'REQUIRES' with movie, is it's Identifying relationship. The set entity has the following attributes:

a. Location Depicted- It gives the name of the Location depicted by the set. It acts as a **partial key** attribute for this entity type, because there might be various kinds of sets which represent a particular location. *It is represented by a String. For example: Let us consider two movies in production "Harry Potter: Part 1" and Fantastic Beasts. Both have Sets depicting "Hogwarts" but both sets are different as there is a gap in the storylines of the movies hence two different depictions of "Hogwarts". But Harry Potter: Part 1 has only 1 Hogwarts, and the same goes for Fantastic Beasts. So even though two sets have the same Location depicted they can be uniquely identified from the movie they are being used in, making it a partial key.*
b. Cost- It describes the cost that was involved in making the set. It is a simple attribute that can be represented by a *decimal value.*
c. Manpower- It describes the manpower required to both construct and maintain the set. It is a simple attribute and a *natural number, and hence can be represented by an Integer.*

4. Actors-

People say 'Seeing is Believing'. But sometimes we don't want to believe everything we see. But there is a class of people who strive hard to make us see and believe their emotions, situations and life; and this class of people is 'Actors'. The entity 'Actors' has the actors working on a particular movie. The specialisations or **Subclasses that are disjoint** are 'lead actor/actress', 'side actor/actress'. **They undergo total participation, as their existence depends on their role in the film.** The attributes of Actors are:

a. Name- It's a simple attribute, which contains the name of the actor/actress. It can be represented by a *string.*
b. Aadhar ID- It contains the Aadhar Number of the Actor. It is a simple attribute as well as the **key attribute** of the actor entity, and can be represented by a *string.*
c. Date Of Birth (DOB)- It's a simple attribute, which contains DOB of the Actor.

d. Age- It is derived from DOB. Hence it's a derived attribute which is always a Natural Number, thus, can be represented by an *integer.*
e. Contact- It contains the phone number of the actor, which is a simple attribute. It can be represented by a *string.*
f. Address(es)- It contains the country, state, city and other details of the addresses the actor/actress has lived in. It is a ***composite, multi-valued attribute*** and can be represented by a *string.*
g. Accolades- It describes the various awards and appreciation received by the actor/actress in his career. It's a ***multi-valued attribute*** and can be represented by *strings.*
h. Remuneration- It contains information about the current remuneration of the actor/actress. It's a simple attribute and can be conveniently represented by a *float.*

5. VFX Studio-

A major part of capturing the attention of the audience, and to ensure that they are glued to the screen, is to stun them by filling their eyes with mesmerizing visuals, which is not always feasible to capture in a real life scenario. A VFX studio does just that. The entity VFX Studio, which is an integral part of a movie, is a **weak entity**, as various VFX studios might have the same name. It is identified by the movie it is associated with, similar to Sets. A **constraint** is defined on a VFX studio that a movie **definitely** works with **one and only one** VFX studio, while a VFX studio can be working on more than one movie simultaneously. The following are the attributes of a typical VFX studio

a. Name- It is a simple attribute which contains the name of the VFX Studio. This is the **partial key** of the VFX Studio entity. *It is represented by a String.*
b. Address - It describes the country, state, city and other details of the location of VFX Studio. *It is a **composite, multi-valued attribute** which is represented by strings.*
c. VFX Director- A VFX director is considered to be the head of the VFX Studio. This simple attribute contains the name of the VFX Director. *It is represented by Strings.*
d. Average Demand- It stores the Average cost demanded by the studio for a particular movie. It is a simple attribute which is a decimal, and hence can be stored in a *float.*

Thus, the **two weak entities** are 'VFX Studio' and 'Sets'. We encountered **specialization** and **generalization** in 'Actors' and 'Directors' entity types respectively.
Movies have **two key attributes**. Further, these **5 entities** have a number of **derived, composite, multi-valued attributes.**

Relations:

The various relationships are described below:

- REQUIRES - It is a **quaternary relationship, i.e, a relationship with degree 4** between 'Movie, Director, Actor and Set' entities. Every entity of this relationship is related to the other three via this relationship type. A movie *requires* actors, sets and a director. A director *requires* film sets, actors and is *required* by a movie. An actor is *required* by a movie, a film set, and director. A film set is *required* by a movie, a director and *requires* an actor.
    a. ‘Movie’ participates in a **m:n relationship** with ‘actor’, as a movie *requires* many actors and an actor can be *required by /* work on any number of movies. A **constraint** is defined here such that an actor can be *required by* **at most 3 movies simultaneously**. It can be realized if a movie is in production by checking its release date attribute, i.e, an actor can be *required by* at most 3 movies whose value for this attribute is currently NULL.
    b. A movie participates in an **m:n relationship** with film sets. But the 'Location Depicted' attribute acts as the partial key in it's **identifying relationship.** Since it is the partial key, a movie should have sets with distinct location depicted, as mentioned in the example of Hogwarts above. A movie might *require* numerous sets and a set might be *required by* numerous movies.
    c. A movie participates in an **m:n relationship** with a director, as a movie might *require* multiple directors and a director might have been *required by* / worked on numerous movies.
    d. A director participates in a **1:n relationship** with film sets, as a director might *require* many sets, while a set is *required by* a single director, when it is being utilized at a point of time. A **constraint** (here, assumption) is defined such that a new set is made specifically for a movie as per requirements and is dismantled after it's usage in that movie.
    e. A director participates in a **m :n relationship** with an actor, as a director *requires* multiple actors to work with him and an actor might be *required by* multiple directors. It is important to note that this is a relationship of central importance as it involves a director selecting the required actors to bring the script to life, and the actors implementing the director's ideas perfectly to strike a chord with the audience.
    f. An Actor participates in an **m:n relationship** with film sets, because an actor might be *required by* multiple film sets (to perform on) and a film set might *require* many actors.
- WORKS_WITH- It is the relationship defined between Assistant Directors and Main Directors, i.e, ‘assistant director’ WORKS_WITH ‘main director’. The coordination between assistant directors and the directors plays a very key role in executing the director's vision flawlessly. It is an **n:1 relationship**, as an Assistant director can only work with one Main Director while a Main Director can have many Assistants working with him. Since both Assistant Director and Main Director are part of 'Director' Entity, It is a **recursive** relationship type with the same participating entity type in **distinct roles.**

- COLLABORATES_FOR- It is the relationship defined between Movies and VFX studios, i.e, a VFX studio COLLABORATES_FOR a movie. Special effects are a key aspect of a film, as it is effective in making the proceedings appealing. It acts as an **identifying relationship** for the weak entity type 'VFX studio'. It is the example of a **1:n relationship**, as a VFX studio could have collaborated for multiple movies, whereas a movie had / has a single VFX studio associated with it.
- COORDINATES_WITH- It is the relationship defined between Directors and VFX studios, i.e, Director COORDINATES_WITH VFX studio. It involves a director explaining his mind to the VFX team, to execute his thoughts with grandeur. It is the example of an **m:n relationship**, as a director can be coordinating / could have coordinated with many VFX studios and vice versa.
- WORKS_ON- It is the relationship defined between VFX Studio and Set, i.e, VFX Studio WORKS_ON Set. It is a **1:n relationship,** as a VFX Studio can be working on / could have worked on multiple sets but a set can have only 1 VFX Studio working on it. A VFX studio working on a set is considered very important because it involves preparing and executing all set effects perfectly, so as to ensure that the audience experience an eye feast.

This is the description of **5 relations** with REQUIRES being a **quaternary relationship (**with a degree greater than 3). The example for a relationship type with the same participating entity type in **distinct roles,** is WORKS_WITH. These relationships have been explained with their corresponding **cardinality constraints.**

Functionalities:

Retrieval:

Query Functions:

1. Selection: 'Select' Function for Movies and other entities. It describes all the data of the entity, i.e, the various entities of different entity types that have associated with the studio.
2. Projection:
    a. Query to display all the movies released in a particular year, by the studio, when a particular year is provided as an input.
    b. Query to display all the movies with average ratings greater than or equal to the value, which is provided as an input.
3. Aggregate function:
    a. Function to display the maximum 'average movie rating of a movie' that has been associated with the studio.
    b. Function to display the minimum 'average movie rating of a movie' that has been associated with the studio.
    c. Function to display the average budget spent on producing a movie.
    d. Function to display the total budget spent on every movie produced in the studio.
4. Search :

a. Partial Text Search for Movie Name to retrieve data on movies matching the string partially.

Analysis:

1. Number of Directors who brought in more than the 'Average Profit per movie' for at least one movie they directed. *It exploits the Director - Movie relationship from REQUIRES.*
2. Number of Actors above a certain threshold of experience: This threshold is defined as- the number of movies in which the particular actor performed multiplied by the number of awards he/she has received. *It exploits the REQUIRES relationship between Actors and Movies.*

Modification:

1. **Insert** : It will add relevant data regarding a movie to all entities and make all the corresponding relations, ensuring that the mentioned constraints are enforced.

2. **Update:** It will update the details of a movie and all the other related data and relations following the constraints. *For example: There might be a change in cast of a movie, altering the movie - actor relationship. Thus, this method comes handy. Another such instance is observed when the remuneration of an 'actor' entity gets updated in accordance to his current fame and demand.*

3. **Delete :** It will delete a movie from the database. This can happen if a movie is shelved / cancelled. Further, updation of details of all the corresponding entities related to that movie, takes place.

   When the Insert, Update or Delete methods are called for a movie, it also inserts, updates or deletes data of its associated entity types. For example:
   a. *If a movie is being produced with actor ‘abc’ who is being associated with the studio for the first time, then his data has to be inserted into the database. So a new actor will be added under the 'actor' entity by calling the 'insert' method on that entity.*
   b. *Consider an Actor ‘A’ received an award for a movie ‘abc’. Then Awards attribute under movie ‘abc’ needs to be updated along with updating actor A's Accolades attribute. Thus, update functionality is used here (to update A's information in 'Actor' entity).*
   c. *Consider the case where a movie gets deleted from the database because it was cancelled. In this case the sets associated with that film alone, and the actors who were working on that film, are deleted from their respective entity tables via the 'delete' functionality.*

   The update, insert, and delete functionalities for each will be implemented or is provided by DBMS.