201 Database Course Work

Topic: Movies

Thomas Pereira - 38866366

Sub-Topic: Science Fiction Movie

Entity domain Paragraph

Sci-Fi movies have a total participation many to many relationships with staff members, as all staff members are required to work on at least one movie and more then one staff member must work on a movie. Sci-Fi movies have a one-to-many relationship with segments. Segments being a weak entity set as many are required to make up a movie like with characters. They also have a partially many to many relationships with characters, multiple character may be participated in a segment but not all segments will need characters. There is a one to many partially relationship between staff members(actors) and their characters.

Normalization:

SciFI_Movies table is 3NF as all attributes are dependent on the movie's name as it's primary key, there is no duplications or transitive dependencies on other elements.

Movie_Staff table is also 3NF as each staff member is given a unique id and no stuff member s repeated twice.

Characters table is 3NF as it is dependent on its movie name and its character id. These two attributes make up its primary key.

Segment table each segment can be identified its primary key the movies it from and the number of the segments. No duplicate segments and all attributes are dependent on the primary key so 3NF.

Works_On table is the many to many relational tables made by the combination of foreign keys one from the staff table and movie table. These two keys make up the tables primary key, making each member unique, therefore 3NF.

Acts_in table another relational table make by foreign keys from character table and segment table. It contains no duplicate or primary key, but combination of foreign key ensures referential integrity so 2NF since the entire entity act as a candidate key.

DDL Statements

```
CREATE TABLE SciFi_Movies(

        Name VARCHAR(100) NOT NULL PRIMARY KEY,

        Duration INT,

        Release_date DATE,

         Budget INT,
        Rating FLOAT,
         Plot TEXT
);
CREATE TABLE Movie_Staff(
        StaffID INT NOT NULL,
        Name TEXT NOT NULL,
        DOB DATE,
        JobTitle Text,
        AvgSalary INT,
        Phone VARCHAR(20),
        Email TEXT,
        PRIMARY KEY (StaffID)
);
CREATE TABLE Characters(
        FilmID VARCHAR(100) NOT NULL,
        CharID INT NOT NULL,
        CharName TEXT NOT NULL,
        ActorID INT,
        Appearance TEXT,
        Background TEXT,
        FOREIGN KEY (FilmID) REFERENCES SciFi_Movies(Name),
        PRIMARY KEY (FilmID,CharID),
        FOREIGN KEY (ActorID) REFERENCES Movie_Staff(StaffID)
);
CREATE TABLE Segments(
        FilmID VARCHAR(100) NOT NULL,
        Number INT NOT NULL,
        Description TEXT,
        Setting TEXT, Duration INT,
        FOREIGN KEY (FilmID) REFERENCES SciFi_Movies(Name),
        PRIMARY KEY (FilmID,Number)
```

```
);

CREATE TABLE IF NOT EXISTS Works_On(

        FilmID VARCHAR(100) NOT NULL,

        StaffID INT NOT NULL,

        FOREIGN KEY (FilmID) REFERENCES SciFi_Movies(Name),

        FOREIGN KEY (StaffID) REFERENCES Movie_Staff(StaffID),

        PRIMARY KEY (FilmID, StaffID)

);

CREATE TABLE Part_In(

        FilmID varchar(100) NOT NULL,

         SegmentNum INT NOT NULL,

        CharID INT NOT NULL,

        FOREIGN KEY (FilmID, CharID) REFERENCES Characters(FilmID,CharID),

        FOREIGN KEY (FilmID, SegmentNum) REFERENCES Segments(FilmID,Number)

);
```

<u>Relational Algebra</u>

1. Delete the movie named Wonder Woman from SciFi_Movies.
   - SciFI_Movies ← SciFi_Movies - σ Name= 'Wonder Woman'(SciFi_Movies)

2. Delete staff member Charles Haney from Movie_Staff table.
   - Movie_Staff ← Movie_Staff - σ Name = 'Charles Haney (Movie_Staff)

3. Names of movies with more than 8 segments.
   - γ FilmID(σ COUNT(*)>8 (π FilmID(Segments)))

4. Display the name of films with less than 8 characters with the work shark in it.
   - σ FilmID LIKE '%Shark%' , COUNT(*) < 8 (γ FilmID (π FilmID(Characters)))

5. Display movies released in 2019 with over a 100000.
   - σ Release_date > '2019-01-01', Release_date <'2019-12-31 (σ Budget > 100000 (Π Name(SciFi_Movies)))

6. Display the cast count and crew count for the movie Frankenstein The True Story.
   - σ FilmID = ' Frankenstein The True Story ' (γ FilmID (π Count(*)(worked_on)))

7. Display the characters and actors of the movie Invitation to Hell.
   - σ FilmID = ' Invitation to Hell ' (π Characters.CharName, Movie_Staff.Name ((Characters) ⋈ Characters.ActorID = Movie_Staff.StaffID (Movie_Staff)))

DML statements

Deletion:

1. DELETE FROM SciFi_Movies WHERE Name = "Wonder Woman";
2. DELETE FROM Movie_Staff WHERE Name = "Charles Haney";

Group by – Having:

3. SELECT FilmID FROM Segments GROUP BY FilmID HAVING COUNT(FilmID) > 8;
4. SELECT FilmID FROM Characters WHERE FilmID LIKE \"Shark%\" GROUP BY FilmID HAVING COUNT(*) < 8;

Others:

5. SELECT Name FROM SciFi_Movies WHERE Budget > 100000 AND Release_date BETWEEN '2019-01-01' AND '2019-12-31';
6. SELECT Count(*) FROM Works_On WHERE FilmID = 'Frankenstein The True Story' Group BY FilmID;
7. SELECT Characters.CharName AS 'Character Name', Movie_Staff.Name AS Actor FROM Characters JOIN Movie_Staff ON Characters.ActorID = Movie_Staff.StaffID WHERE FilmID = 'Invitation to Hell';