Project Phase-1

Team No: 39

Introduction:

The Cinema-Base is a signified database that is used to store and manage data related to the movie industry. It provides a comprehensive structure for organizing and retrieving information about movies, actors, technicians, production houses, and awards. The Database will give access to different users with different capabilities. This Database will be used for analysing the trends in most recent times by Data Analysts and also naïve Users.

Purpose of Database:

The purpose of the Cinema-Base database is to provide a centralized and organized system for storing, managing, and retrieving information related to the movie industry. It aims to:

1. **Organize Movie Data:**
   - Keep track of all movie information, such as names, genres, release dates, budgets, and box office Collections.
2. **Manage Relationships:**
   - To represent the complexity of the film industry, capture and maintain relationships between different entities, including actors, technicians, production houses, awards, and characters.
3. **Support Decision Making:**
   - Provide production houses with accurate data to help them make informed decisions regarding movie production, marketing strategies, and audience engagement.
4. **Facilitate Reporting and Analysis:**
   - Enable data retrieval and analysis to identify trends, measure performance, and assess the impact of different factors on movie success.
5. **Enhance Collaboration:**
   - Serve as a platform for collaboration among various actors, technicians, and production houses, to streamline processes and improve efficiency in movie production.
6. **Preserve Historical Records:**
   - Maintain a record of past movies, awards, and industry milestones for future reference and research.
7. **Improve User Experience:**
   - Provide tools for naïve Users to easily find and access movie-related information.

Users of Database:

a. **Database Administrators (DB Admins):**
   - Responsible for managing and maintaining the database system, ensuring data integrity, security, and performance. They handle user permissions, backups, system upgrades, and troubleshooting any issues that arise.

b. **Production Houses:**
   - Manage movie production details, budgets, and Profit tracking.

c. **Actors and Technicians:**
    - Maintain profiles to track their contributions, roles, and career milestones.

d. **Data Analysts and Researchers:**
    - Extract insights for industry analysis, reporting, and trend research in filmmaking and audience preferences.

e. **Naïve Users:**
    - Naïve users can search for movies, view detailed information about films, Actors, Technicians, Production Houses and access awards data

Applications of Database:

1. **Movie Management Systems**:
    - Track movie production details, box office performance, and revenue generation.
2. **Actor and Technician Databases**:
    - Maintain records of individuals in the film industry, enabling easy access to their profiles.
3. **Award Management Systems**:
    - Organize and track winners for various award categories.
4. **Research and Analytics Tools**:
    - Support researchers and industry analysts by providing data for generating reports, identifying trends, and gaining insights.
5. **Historical Archives:**
    - Serve as a digital archive for preserving the history of cinema, including past films, awards, and notable Personalities for research and reference.

Database Requirements:

a) **Assumptions**

1. Movie Entity has Title as a Composite Attribute and Genre, Released_Languages as Multi-Valued and Profit as Derived (Calculated from Budget and Collections).
2. Every Movie must have atleast One Actor and One Technician
3. Movie must contain a Character
4. Movie will be Produced by only one Production_House
5. Awards are only for Movie
6. A character can be in multiple movies (series of movies) and be played by only one Actor
7. In a series of Movie only the next Movie is considered as sequel for example: MI 3 is sequel of MI 2 Only
8. A Movie may or may not have a Trailer
9. Release_Date, Established_Date, Date_of_Birth, Published_Date is of format DD-MM-YYYY
10. Profit in Movie is calculated by difference of Budget and Collections
11. Duration of a Movie will be in minutes.
12. Number_of_Movies in both Actor and Technician is derived
13. Award has 2 candidate keys and combined key called as Composite Primary Key

b) **Strong Entities**

1. Movie (Super Class)
   Attributes: Movie_ID (INT) (Primary Key), Title {Movie_name (VARCHAR(255)), Tag_name VARCHAR(255)}, Release_Date (Date), Duration (INT) ,Genre (VARCHAR(255)) (Multi-Valued), Budget (DECIMAL(15, 2)) , Collections (DECIMAL(15, 2)), Profit (DECIMAL(15, 2)) (Derived), Released_Languages (VARCHAR(511))(Multi-Valued).

Sub Classes:
All Attributes are Inherited from Movie

- Animated_Movie
  Additional Attributes: Animation_Style (VARCHAR(255))
- Short_Film
- Feature Film
- Documentry
  Additional Attributes: Topic (VARCHAR(255))

2. Production_House
   Attributes: Studio_ID (INT)(Primary Key), Name (VARCHAR(255)), Established_Date (DATE), Location {Street (VARCHAR(255)), Area (VARCHAR(255)), City (VARCHAR(255)), State (VARCHAR(255)), PinCode(INT)}, Owner_Name (VARCHAR(255)).

3. Actor
   Attributes: Actor_ID (INT)(Primary Key), Name (VARCHAR(255)), Date_of_Birth (DATE), Age (INT)(Derived), Nationality (VARCHAR(255)), Number_of_Movies (INT)(Derived).

4. Technician
   Attributes: Technician_ID (INT)(Primary Key), Name (VARCHAR(255)), Date_of_Birth(DATE), Nationality (VARCHAR(255)), Number_of_Movies (INT)(Derived), Department (VARCHAR(255)).

5. Award
   Attributes: Award_Name (VARCHAR(255)) (Candidate Key), Awarded_Year (INT)(Candidate Key), Category (VARCHAR(255)).

c) Weak Entities

1. Character
   Attributes: Character_Name (VARCHAR(255)) (Partial Key), Gender (CHAR(1)), Role_Type (VARCHAR(255)).
   Dependency: Actor, Movie.

2. Trailer
   Attributes: Trailer_ID (INT)(Partial Key), Duration(INT), Published_Date(DATE), Views(INT), URL (VARCHAR(255))
   Dependency: Movie

d) Relationship Types

i. Actors **ACTED_IN** Movie
   Degree: 2
   Participating Entities: Actor, Movie.
   Cardinality Ratio: M:N (Actors : Movie)
   Participation Constraint: Total Participation for Movie and Partial Participation for Actor

ii. Technicians **WORKS_FOR** Movie
   Degree: 2
   Participating Entities: Technician, Movie.
   Cardinality Ratio: M:N (Technician : Movie)
   Participation Constraint: Total Participation for Movie and Partial Participation for Technician

iii. Production_House **PRODUCES** Movie
Degree: 2
Participating Entities: Production_House, Movie.
Cardinality Ratio: 1:N (Production_House : Movie)
Participation Constraint: Total Participation for Movie and Partial Participation for Production_House

iv. Movie **RECEIVED** Award
Degree: 2
Participating Entities: Movie, Award.
Cardinality Ratio: 1:M (Movie : Award)
Participation Constraint: Total Participation for Award and Partial Participation for Movie

v. **PLAYED_BY&IN**
Degree: 3
Participating Entities: Movie, Actor, Character.
Cardinality Ratio: N:K:M (Movie : Actor : Character)
Participation Constraint: Total Participation for Character and Movie and Partial Participation for Actor

vi. Movie **HAS** Trailer
Degree: 2
Participating Entities: Trailer, Movie.
Cardinality Ratio: M:1 (Trailer: Movie)
Participation Constraint: Total Participation for Trailer and Partial Participation for Movie

Degree > 3

vii. **COLLABORATION**
Degree: 4
Participating Entities: Movie, Actor, Technician, Production_House
Cardinality Ratio: 1:M:K:1 (Movie : Actor : Technicians : Production_House)
Participation Constraint: Total Participation for Movie and Partial Participation for Actor, Technician, Production_House

Self-Referencing Relation

viii. **SEQUEL**
Degree: 2
Participating Entities: Movie(Original), Movie(Sequel)
Cardinality Ratio: 1:1
Participation Constraint: Total Participation for Movie(Sequel) and Partial Participation for Movie(Original)

Functional Requirements:

a. **Retrieval**
   1. **Queries**
      i. **Selection :**
         a. Retrieve all characters played by 'Robert Downey Jr.'
         b. Retrieve all actors who have acted in movies of the 'Science Fiction' genre
      ii. **Projection** :
         a. Get the names of all actors who have worked in at least three movies.
         b. Get the names of all technicians working in the 'Editing' department.
      iii. **Aggregate :**
         a. Determine the maximum profit obtained for a movie produced by a production house
         b. Count the total number of movies in each genre.
      iv. **Search :**
         a. Search for technicians with 'Alex' as part of their name.
         b. Find all movies with 'Spiderman' in the title.
   2. **Analysis**
      i. Retrieve the list of production houses that have produced at least three movies, each with collections above the average collection of movies from the past four years.
      ii. Retrieve the number of awards won by movies in each genre, organized by award type.

b. **Modification**

   1. **INSERT:**
      We can insert a record of particular type in corresponding table like a
      - movie in Movie Table
      - actor in Actor Table
      - technician in Technician Table
      - production_house in Production_House Table
      - award in Award Table
      - character in Character Table
      - trailer in Trailer Table
        BY ADMIN.

      Example:

      ```
      INSERT INTO Movie (Movie_ID, Title, Release_Date, Duration, Genre, Budget,
      Collections, Released_Languages) VALUES (101, 'The Adventure Begins', '15-06-2022',
      '120', 'Adventure', 5000000, 8000000, 'English, Spanish');
      ```

      Data Integrity Constraints:

      - Movie_ID is should not present in MOVIE Table before insertion.
      - Movie_ID should not be NULL

   2. **UPDATE**:
      We can update the following attributes for entities
      - MOVIE : Title ,Collections ,Released_Languages (by ADMIN/Production_House)
      - TECHNICIAN: Department (by ADMIN/Technician)
      - PRODUCTION_HOUSE: Name, Owner_Name (by ADMIN/Production_House)
      - TRAILER: Views, Duration (by ADMIN/Production_House)

Example:

`   UPDATE Movie SET Collections = 200000000 WHERE Movie_ID = 101;

Data Integrity Constraints

- Movie_ID should be present in MOVIE Table

3. **DELETE**:

   Delete trailer of a movie (BY ADMIN/Production_House)

   Example:

   DELETE FROM Trailer WHERE Trailer_ID = 10102.

   Data Integrity Constraints:

   - Trailer_ID should be present in Trailer Table

Summary:

The Cinema-Base database offers an organized system for managing complex movie industry data. It meets the needs of various users .This will be helpful in making data management more efficient. Additionally, it supports informed decision-making and preserves historical records.