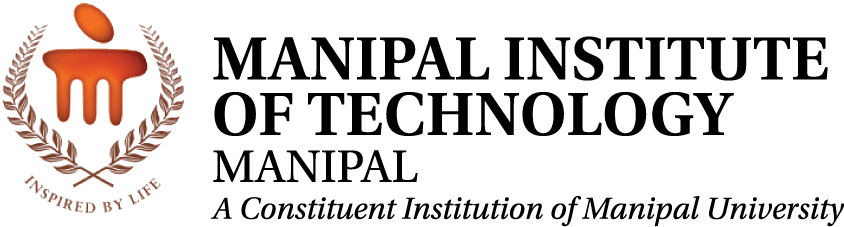
|  |
| --- |
| **Department of Computer Science and Engineering**    **DATABASE SYSTEM LAB – CSE 2262**  ***A mini project report on***    **Database Management System Of CINEMA**  **Group members:**  **Name: Munagala Sai Sandeep**  **Register no: 210962074**  **Section: A**  **Roll no: 17**  **Name: K S Adharsh Reddy**  **Register no: 210962014**  **Section: A**  **Roll no: 05**  **Department of Computer Science and Engineering**  **Manipal Institute of Technology, Manipal.**  **April 2023** |
|  |



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal

07/05/2023

CERTIFICATE

This is to certify that the project titled Database Management System Of CINEMA is a record of the bonafide work done by Student(s) (Reg. No. 210962074 & 210962082) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. Dr Roopalakshmi R, Associate Professor, CSE Dept.
2. Prof. Tanuja Shailesh, Assistant Professor, CSE Dept.

**Content:**

1. Abstract
2. Problem Statement
3. ER Diagram
4. Relational Tables
5. DDL Commands
6. List of SQL Queries
7. PL/SQL Procedures/Functions/Triggers
8. UI DESIGN
9. References

**Introduction:**

A film industry database is a collection of information about the film industry, including films, actors, directors, producers, studios, OTT, and other industry professionals. The database may also contain information about individual actors, directors, producers, and other professionals, such as their biographical details, filmography, awards, and other career achievements.

Film industry databases are commonly used by film studios, production companies, distributors, and other industry professionals to track trends, analyse performance, and make informed decisions about their projects. They may also be used by journalists, researchers, and film enthusiasts to explore the history and evolution of the film industry and discover new and interesting films, actors, and directors.

**Attributes:**

Type 1: Movies – Movie\_ID, Title, Release\_year, Movie\_language, Director\_ID, Genre.

Type 2: Director – Director\_ID, First name, Last name, Gender.

Type 3: Movie Cast – Movie\_ID, Actor\_ID, Charater\_Name.

Type 4: Actor – Actor\_ID, First name, Last name, Gender.

Type 5: Rating – Movie\_ID, Rating.

Type 6: Genre – Movie\_ID, Genre\_Name.

Type 7: Awards – Name, Year\_awarded, Category, Movie\_ID.

**Required software:**

* NOTEPAD
* EXCEL
* MYSQL
* NETBEANS

**Problem statement:**

A database management system (DBMS) is required to manage a movie database. The database will store information about movies, actors, directors, production companies, genres, and ratings.

**Data requirements:**

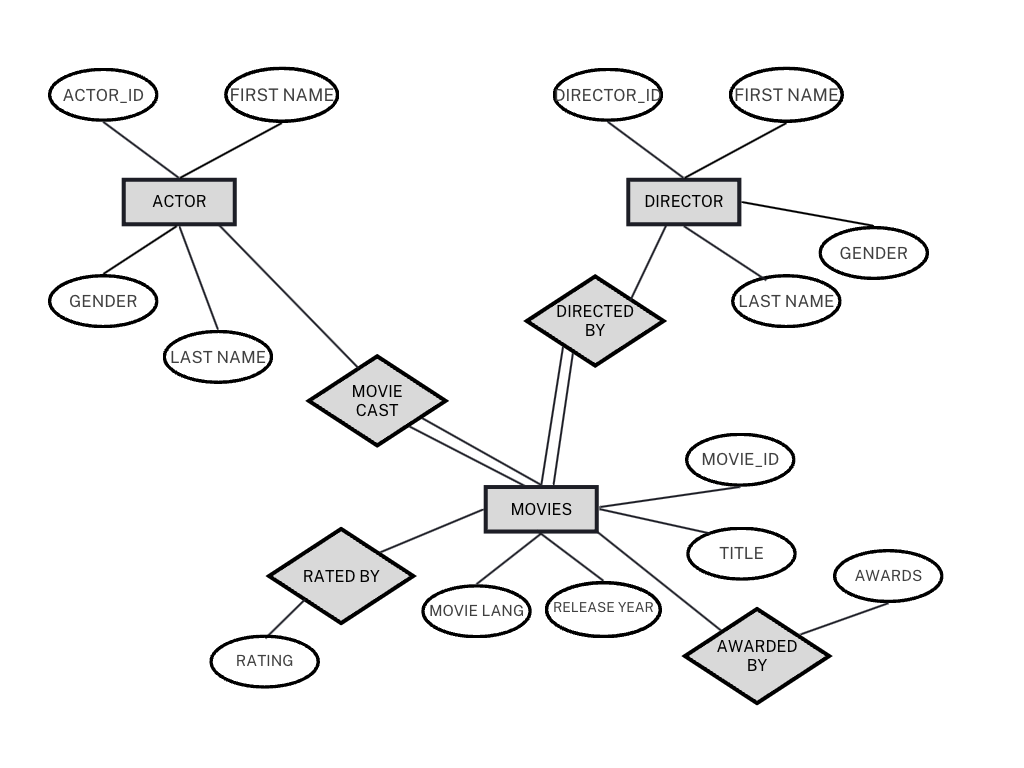
1. Movie data: The database should store information about each movie, including the title, release year.
2. Actor data: The database should store information about each actor, including their name, birth date, and a list of movies they have appeared in.
3. Director data: The database should store information about each director, including their name, birth date, and a list of movies they have directed.
4. Award data: The database should store information about each award, including the name of the movie, category of the award, year of the award.
5. Genre data: The database should store information about each genre, including the name and a list of movies in that genre.
6. Rating data: The database should store information about each movie rating, including the name of the movie, the rating source, and the rating value.

**Functional requirements:**

1. Searching: The database allows users to search for movies based on various criteria, such as title, genre, actor, or director.
2. Updating: The database allows users to update information about movies, actors, directors, awards, genres, and ratings.
3. Adding: The database allows users to add new movies, actors, directors, awards, genres, and ratings to the database.
4. Deleting: The database allows users to delete movies, actors, directors, awards, genres, and ratings from the database.
5. Reports: The database provides reports, such as the top-rated movies, and the most popular actors.
6. Security: The database should provide access control to ensure that only authorized users can access and modify the data.
7. Performance: The database should be optimized for fast queries and efficient storage to handle a large amount of data and a high number of concurrent users.

Overall, the database management system should provide a comprehensive and user-friendly solution for managing a movie database, with the ability to search, update, add, and delete data, as well as generate reports and ensure security and performance.

**ER Model :**



**Relational Tables and DDL Command:**

Use mini\_project

**ACTOR TABLE:**

CREATE TABLE actors (

actor\_id int PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

GENDER char(1)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Actor\_id | First\_name | Last\_name | Gender |
| 1 | Ram | Charan | M |
| 2 | Akshay | Kumar | M |
| 3 | Anushka | Shetty | F |
| 4 | Samantha | Ruth prabhu | F |
| 5 | Shahrukh | khan | M |

**DIRECTOR TABLE:**

CREATE TABLE directors (

director\_id int PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

gender CHAR(1)

);

|  |  |  |  |
| --- | --- | --- | --- |
| Director\_id | First\_name | Last\_name | Gender |
| 1 | SS | Rajamouli | M |
| 2 | Yash | Raj | M |
| 3 | B | Sukumar | M |
| 4 | Sidharth | Anand | M |

**MOVIE TABLE:**

CREATE TABLE movies (

movie\_id INT PRIMARY KEY,

title VARCHAR(100) NOT NULL,

release\_year int,

movie\_language varchar(10),

director\_id int,

genre VARCHAR(50),

FOREIGN KEY (director\_id) REFERENCES directors(director\_id) on delete cascade

);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Movie\_id | Title | Release\_year | Movie\_language | Director\_id | Genre |
| 1 | RRR | 2023 | Telugu | 1 | Action |
| 2 | Mission mangal | 2019 | Hindi | 2 | Sci-fi |
| 3 | Bahubali | 2015 | Telugu | 1 | War |
| 4 | Rangasthalam | 2018 | Telugu | 3 | Drama |
| 5 | Paathan | 2023 | Hindi | 4 | Spy |

**MOVIE CAST TABLE:**

CREATE TABLE Movie\_Cast (

movie\_id INT,

actor\_id INT,

character\_name VARCHAR(255),

PRIMARY KEY (movie\_id, actor\_id),

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) on delete cascade,

FOREIGN KEY (actor\_id) REFERENCES Actors(actor\_id) on delete cascade

);

|  |  |  |
| --- | --- | --- |
| Movie\_id | Actor\_id | Charater\_name |
| 1 | 1 | Sitarama Raju |
| 2 | 2 | Rakesh Dhawan |
| 3 | 3 | Devasena |
| 4 | 1 | Chitti babu |
| 4 | 4 | Ramalakshmi |
| 5 | 5 | Paathan |

**RATING TABLE:**

CREATE TABLE rating (

movie\_id INT primary key,

RATING numeric(6,2),

FOREIGN KEY (movie\_id) REFERENCES movies(movie\_id) on delete cascade

);

|  |  |
| --- | --- |
| Movie\_id | Rating |
| 1 | 9.30 |
| 2 | 8.80 |
| 3 | 8.30 |
| 4 | 9.30 |
| 5 | 8.90 |

**AWARD TABLE:**

CREATE TABLE awards (

name VARCHAR(255) NOT NULL,

year\_awarded INT,

category VARCHAR(255),

movie\_id INT,

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id) on delete cascade

);

|  |  |  |  |
| --- | --- | --- | --- |
| Award\_name | Year\_awarded | Category | Movie\_id |
| Oscar | 2023 | Best original song | 1 |
| Film fare | 2020 | Nominee for best director | 2 |
| Film fare | 2016 | Best film | 3 |
| Film fare | 2016 | Best director | 3 |

**GENRES TABLE:**

CREATE TABLE genres (

movie\_id INT PRIMARY KEY,

genre\_name VARCHAR(50) NOT NULL,

FOREIGN KEY (movie\_id) REFERENCES movies(movie\_id) on delete cascade );

|  |  |
| --- | --- |
| Movie\_id | Genre |
| 1 | Action |
| 2 | Science friction |
| 3 | War |
| 4 | Drama |
| 5 | Spy |

**SQL Queries :**

--1. LIST THE CHARACTER OF THE ACTOR ACTED IN A MOVIE:

select \* from actor natural join movie\_cast where first\_name = 'Ram';

-- 2. ONE OR MORE ACTORS ACTED IN TWO OR MORE MOVIES:

SELECT distinct title from movies natural join movie\_cast where actor\_id in (select actor\_id from movie\_cast group by actor\_id having count(actor\_id)>=2);

--3. LIST ALL ACTORS WHO ACTED BEFORE 2000 AND ALSO AFTER 2017 USING

select f\_name,l\_name,year from (select distinct a.first\_name f\_name, a.last\_name l\_name,m.release\_year year from actors a, movie\_cast mv, movies m where a.actor\_id = mv.actor\_id and m.movie\_id =

mv.movie\_id) where year <=2000 or year>=2017;

**PL/SQL Procedures:**

--1.RETRIVING DATA BY TAKING USER INPUT (MOVIE NAME):

DECLARE  
    v\_movie\_title varchar2(100);  
    v\_movie\_id number;  
    v\_movie\_release\_year int;  
BEGIN  
    v\_movie\_title := '&movie\_title';  
    select movie\_id, release\_year  
    into v\_movie\_id, v\_movie\_release\_year  
    from movies  
    where title = v\_movie\_title;  
  
    DBMS\_OUTPUT.PUT\_LINE('Movie id: ' || v\_movie\_id);  
    DBMS\_OUTPUT.PUT\_LINE('Release year: ' || v\_movie\_release\_year);  
END;  
/

--2.UPDATING MOVIE RATING

DECLARE

v\_movie\_id number;

v\_new\_rating number;

BEGIN

v\_movie\_id := '&movie\_id';

v\_new\_rating := '&rating';

UPDATE rating

SET rating = v\_new\_rating

WHERE movie\_id = v\_movie\_id;

DBMS\_OUTPUT.PUT\_LINE('Rating updated successfully.');

END;

/

--3.RETRIVE TOP RATED MOVIES

DECLARE  
    cursor c\_top\_movies is  
        select title,rating  
        from rating natural join movies  
        order by rating desc;  
BEGIN  
    for movie in c\_top\_movies LOOP  
        DBMS\_OUTPUT.PUT\_LINE('Title: ' || movie.title || ', Rating: ' || movie. rating);  
    END LOOP;  
END;  
/

**Function and Procedure:**

--4.FUNCTION TO RETRIEVE THE NUMBER OF MOVIES RELEASED IN A SPECIFIC YEAR:

create or replace function count\_movies\_in\_year(r\_year in number) return number is v\_movie\_count number;  
BEGIN  
    select count (\*)  
    into v\_movie\_count  
    from movies  
    where release\_year = r\_year;  
  
    RETURN v\_movie\_count;  
END;  
/  
  
declare  
begin  
DBMS\_OUTPUT.PUT\_LINE ('NO OF MOVIES: ' || count\_movies\_in\_year(2019));  
end;  
/  
  
--5.PROCEDURE TO UPDATE THE RATING OF A MOVIE:

create or replace procedure update\_movie\_rating(p\_movie\_id in number,p\_new\_rating in number) is  
BEGIN  
    update rating  
    set rating = p\_new\_rating  
    where movie\_id = p\_movie\_id;  
  
    DBMS\_OUTPUT.PUT\_LINE('Rating updated successfully.');  
END;  
/  
  
declare  
begin  
update\_movie\_rating(2,9.5);  
end;  
/  
  
  
6. FUNCTION TO CALCULATE THE AVERAGE RATING FOR A MOVIE:

create or replace function calculate\_average\_rating return number is v\_avg\_rating number;  
BEGIN  
    select avg(rating)  
    into v\_avg\_rating  
    from rating;  
  
    RETURN v\_avg\_rating;  
END;  
/  
  
declare  
begin  
DBMS\_OUTPUT.PUT\_LINE ('average\_rating: ' || calculate\_average\_rating());  
end;  
/

**TRIGGERS:**

7.TRIGGER TO CHECK MOVIE’s RATING MUST BE BETWEEN 0 AND 10:  
  
create or replace trigger check\_rating\_range  
before insert or update on rating  
for each row  
DECLARE  
    v\_rating rating.rating%TYPE := :new.rating;  
BEGIN  
    IF v\_rating < 0 OR v\_rating > 10 THEN  
        RAISE\_APPLICATION\_ERROR(-20001, 'Rating must be between 0 and 10.');  
    END IF;  
END;  
/

**EXCEPTION:**

8. SEARCHING FOR A MOVIE WITH MOVIE ID :

DECLARE  
    movie\_not\_found EXCEPTION;  
    v\_movie\_id number;  
    v\_movie\_count number := 0;  
BEGIN  
    v\_movie\_id := '&movie\_id';  
    select count(\*)  
    into v\_movie\_count  
    from movies  
    where movie\_id = v\_movie\_id;  
  
    IF v\_movie\_count = 0 THEN  
        raise movie\_not\_found;  
    ELSE  
        DBMS\_OUTPUT.PUT\_LINE('Movie Found.');  
    END IF;  
EXCEPTION  
    when movie\_not\_found then  
        DBMS\_OUTPUT.PUT\_LINE('Movie not found in the database.');  
END;  
/

**UI DESIGN:**

A screen shot of a movie

Description automatically generated with medium confidenceA picture containing screenshot, multimedia software, software, text

Description automatically generated

A screen shot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidence

**Java Functional code:**

JDBC driver for connecting MYSQL Database:**A picture containing text, screenshot

Description automatically generated**

Updating Data:

**A screenshot of a computer program

Description automatically generated with low confidence**

Insertion of Data:

**A screenshot of a computer program

Description automatically generated with low confidence**

**Reference:**

1. JAVA The Complete Reference, Ninth Edition, Herbert Schildt, ORACLE PRESS
2. Silberschatz, Korth, Sudarshan, “ Database System Concepts”, McGraw-Hill, 6th Edition.
3. Iyan Bayross, “SQL, PL/SQL” 2nd/3rd Edition, BPB Publications.