DATABASE PROJECT REPORT



**21K-3153 21K-3372 21K-4529**

06.12.2023

### 

### 

### 

### ER Diagram:

### 

### Explanation of ER Diagram:

We have our database which contain four tables:

1. Movies
2. User (Contain the details of users)
3. Reviews
4. Watchlist

## **Movies Table:**

The Movies table represents the entity "Movies" and stores information about various movies in the database.

Attributes of the Movies table include ID (as the primary key), Title, ReleaseDate, Genre, Director, and other relevant details about each movie.

## **Reviews Table:**

The Reviews table represents the entity "Reviews" and is used to store information about user reviews for specific movies.

Attributes include ReviewID (as the primary key), MovieID (foreign key referencing Movies table), UserID (foreign key referencing Users table), Rating, and Comments.

## **Users Table:**

The Users table represents the entity "Users" and stores information about the users who interact with the movie database.

Attributes include UserID (as the primary key), Username, Password.

## **Watchlist Table:**

The Watchlist table represents the entity "Watchlist" and is used to keep track of movies that users want to watch.

Attributes may include WatchlistID (as the primary key), UserID (foreign key referencing Users table), MovieID (foreign key referencing Movies table).

## 

## **Relationships:**

The Movies table is likely to have a one-to-many relationship with the Reviews table since one movie can have multiple reviews, but each review is associated with only one movie.

The Users table may have a one-to-many relationship with both the Reviews table (a user can write multiple reviews) and the Watchlist table (a user can have multiple movies in their watchlist).

The Watchlist table is to have a many-to-many relationship with the Movies table, as multiple users can have the same movie in their watchlist, and one user can have multiple movies in their watchlist.

## **Cardinalities:**

Movies to Reviews: One-to-Many (One movie can have many reviews, but each review is for one movie).

Users to Reviews: One-to-Many (One user can write many reviews, but each review is written by one user).

Users to Watchlist: One-to-Many (One user can have many movies in their watchlist, but each movie in the watchlist belongs to one user).

Movies to Watchlist: Many-to-Many (Many users can have the same movie in their watchlist, and one user can have multiple movies in their watchlist).

This ER diagram provides a clear visual representation of how the different entities in the movie database are related to each other, helping in the design and understanding of the database structure.

### 

### 

### NORMALIZATION:

**IMDB\_top\_1000 Table:**

**1NF:**

For 1NF we have an attribute called “Genre” which is multivalued and we need to either break it down to multiple genres (i.e Genre1, Genre2 etc) or we can simply split the table and shift our “Genre” attribute there. We are splitting the table by making the “MovieID” as a referenced key. In the second table, we will be splitting the Genres into multiple tuples which will solve the issue as each cell will contain atomic value.

**Base Table: Movies**

| ID | Poster\_Link | Series\_Ttile | Released\_Year | Certificate | Runtime | IMDB\_Rating | Overview | Meta\_Score | Star1 | Star2 | Star3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

| Star4 | Director | No\_of\_Votes | Gross |
| --- | --- | --- | --- |

**Genres Table:**

| ID | Genres |
| --- | --- |

**2NF:**

Our Table is already in 2NF as it only has one Prime Key Attribute (“ID”) on which all other attributes are dependent hence **Fully Functional Dependency** is achieved.

**3NF:**

**Movies**

| ID | poster\_link | series\_title | released\_year | certificate | runtime | imdb\_rating | overview | meta\_score | no\_of\_votes | gross |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**MoviesStars**

| movie\_id (FK referencing Movies(id)) | star\_id |
| --- | --- |

**Stars**

| **Star\_id (FK of MoviesStars(star\_id))** | **star\_name** |
| --- | --- |

**MovieDirectors**

| movie\_id (FK referencing Movies(id)) | director\_id |
| --- | --- |

**Directors**

| Director\_id  (FK referencingDirectors(director\_id)) | director\_name |
| --- | --- |

**User Table:**

**User**

| id | username | password |
| --- | --- | --- |

**First Normal Form (1NF):**

All columns must contain atomic values (no sets, lists, or nested structures).

The table should have a primary key that uniquely identifies each record.

The "Users" table satisfies the conditions for 1NF with the "id" serving as the primary key.

**Second Normal Form (2NF):**

The table is in 1NF.

There are partial dependencies, meaning that non-prime attributes (attributes not part of the primary key) are fully functionally dependent on the primary key.

In this case, with the "Users" table having only three attributes ("id," "username," and "password"), it is automatically in 2NF because there are no partial dependencies.

**Third Normal Form (3NF):**

The table is in 2NF.

There are no transitive dependencies, meaning that non-prime attributes are not dependent on other non-prime attributes.

Since the"Users" table has only the primary key "id" and non-prime attributes "username" and "password," and there are no transitive dependencies, it is in 3NF.

### Watchlist table:

**Watchlist**

| id | user\_id  (FK referencing to User (id)) | Movie\_id  (FK referencing to Movies (id)) |
| --- | --- | --- |

**First Normal Form (1NF):**

All columns contain atomic values, and the table has a primary key.

The "Watchlist" table meets the conditions for 1NF.

Second Normal Form (2NF):

**Second Normal Form (2NF):**

The table should be in 1NF.

There are no partial dependencies.

In the "Watchlist" table, there are no partial dependencies since the primary key is a composite key (id, user\_id, movie\_id), and both user\_id and movie\_id contribute to the primary key.

The table should be in 2NF.

**Third Normal Form (3NF):**

There should be no transitive dependencies.

The "Watchlist" table does not have transitive dependencies. All non-prime attributes (user\_id and movie\_id) are directly dependent on the primary key (id, user\_id, movie\_id), and there are no indirect dependencies.

In summary, the "Watchlist" table is in the Third Normal Form (3NF).

**Review table:**

**review**

| review\_id | user\_id  (FK referencing to User (id)) | movie\_id  (FK referencing to Movies (id)) | rating | review\_text |
| --- | --- | --- | --- | --- |

**First Normal Form (1NF):**

All columns contain atomic values, and the table has a primary key.

The "Reviews" table meets the conditions for 1NF.

Second Normal Form (2NF):

The table should be in 1NF.

**Second Normal Form (2NF):**

There should be no partial dependencies.

The "Reviews" table does not have partial dependencies. Each non-prime attribute (user\_id, movie\_id, rating, review\_text) is fully functionally dependent on the primary key (review\_id).

The table is in 2NF.

**Third Normal Form (3NF):**

There are no transitive dependencies.

The "Reviews" table does not have transitive dependencies. All non-prime attributes are directly dependent on the primary key (review\_id).

In summary, the "Reviews" table is in the Third Normal Form (3NF).