**Brayden Sampson**

**Assignment 3**

The Movie table is a table holding data on all the various movies within the TOED database. The attributes within the table are MovieID, M\_name, M\_year, M\_genre, M\_duration, TOED\_rating, and User\_rating. The primary key is MovieID, which is the unique number given to each entry (movie) within the database. There are no foreign keys in the movie table. The Movie table has relationships with Movie\_Director and Movie\_Role tables, which are both join tables to Director and Actor respectively. Because these are join tables, the Movie table has a 1:M relationship with both. These join tables were created to eliminate the M:N relationships within the database. For example, a movie can have many movie directors or roles, but a movie role or director can only be associated with one movie.

The next table moving clockwise is the Movie\_Role table which is the join table between Movie and Actor. The attributes within this table are MovieID and ActorID which together form the composite key for the table. This table has a relationship with Movie and Actor which are both 1:M relationships towards the join table. The issues were that many actors and directors could be associated with many shows or movies.

Next is the Actor table which holds all data on actors and actresses within all movies and tv shows in the database. The attributes within the table are ActorID, A\_fname, A\_lname, and A\_DOB. The primary key is ActorID, which is the unique attribute in the table, in this case a unique number given to each actor and actress. There are no foreign keys in the Actor table. The table is related to Movie\_Role and Show\_role which are join tables to Movie and Show respectively. This helps eliminate the issue of M:N relationships between the tables. Since these relationships are to join tables, they are both 1:M.

The Show\_Role table is the next join table connecting Actor to Show. The attributes within this join table are ShowID and ActorID, which together form the composite key for this table. There are no other attributes within the table (no foreign key either). This join table has 1:M relationships with both Actor and Show around it, pointing towards the join table.

Next is the Show table which like Movie, is holding similar data on all tv shows within the database. The attributes in the table are ShowID, S\_name, S\_year, S\_genre, S\_episodes, TOED\_rating, and User\_rating. The primary key is ShowID, which gives each show in the database a unique ID number. There are no foreign keys within the table. Like the other tables in the ERD, show is connected to the join tables Show\_Director and Show\_Role, which are the join tables connecting to Director and Actor respectively. This results in 1:M relationships pointing to the join tables.

The Show\_Director table is another join table joining Director and Show. This table has the attributes ShowID and DirectorID which form the composite key for this table. The table does not have a foreign key. This table has relationships with Director and Show, which are both 1:M relationships towards the join table.

Next is the Director table which stores all data on the directors of shows and movies within the database. The attributes in the table are DirectorID, D\_fname, D\_lname, and D\_DOB. The primary key is DirectorID which a unique number associated with each director. The table does not have a foreign key. This table is too connected to the join tables Movie\_Director and Show\_Director which connect to Movie and Show respectively. Like all the other join tables, they are there to solve the issue with the M:N relationships between the main 4 tables. The new 1:M relationships are towards the join tables.

The last table is the join table Movie\_Director which has the attributes MovieID and DirectorID forming the composite key for the table. Ther is no foreign key. The table connects Movie to Director with 1:M relationships pointing inwards to the join table, successfully solving the issue of the M:N relationships.

Diagram

Description automatically generated