Entity description

Movie is the base entity, has attributes; movieID (primary key), title, release date and has foreign keys; studioID referencing studio with domain INT, directorID referencing Director with domain INT and awardID referencing Award with domain INT. Multiplicity between Movie-Director, Movie-Studio, Movie-Actor, Movie-Award, Actor-Award is many to many. Cast relation has two foreign keys; movieID and actorID referencing Movie and Actor respectively. All relations except Movie-Award and Actor-Award have a full participation constraint, other two have a partial participation constraint. Studio has attributes; studioID (primary key), name. Director has attributes; directorID (primary key), name, dateOfBirth. Actor has attributes; actorID (primary key), name, dateOfBirth. Award has attributes; awardID (primary key), type.

Process for 1NF

Before:

Movie (movieID: INT, title: TEXT, releaseDate: DATE, directorID: INT, studioID: INT, awardID: INT, PRIMARY KEY: movieID, Foreign Key: directorID referencing Director, Foreign Key: studioID referencing Studio, Foreign Key: awardID referencing Award)

After:

Movie (movieID: INT, title: TEXT, releaseDate: DATE, PRIMARY KEY: movieID)

MoviesMade (studioID: INT, movieID: INT, PRIMARY KEY: (studioID, movieID), Foreign Key: studioID referencing Studio, Foreign Key: movieID referencing Movie)

MoviesDirected (directorID: INT, movieID: INT, PRIMARY KEY: (directorID, movieID), Foreign Key: directorID referencing Director, Foreign Key: movieID referencing Movie)

AwardWinningMovies (movieID: INT, awardID: INT, PRIMARY KEY: (movieID, awardID), Foreign Key: movieID referencing Movie, Foreign Key: awardID referencing Award)

Cast (movieID: INT, actorID: INT, PRIMARY KEY: (movieID, actorID), Foreign Key: movieID referencing Movie, Foreign Key: actorID referencing Actor)

Before:

Director (directorID: INT, name: TEXT, dateOfBirth: DATE, PRIMARY KEY: directorID)

After:

Director (directorID: INT, firstName: TEXT, lastName: TEXT, PRIMARY KEY: directorID)

Before:

Actor (actorID: INT, name: TEXT, dateOfBirth: DATE, awardID: INT, PRIMARY KEY: actorID, Foreign Key: awardID referencing Award)

After:

Actor (actorID: INT, firstName: TEXT, lastNAME: TEXT, dateOFBirth: DATE, PRIMARY KEY: actorID)

AwardWinningActors (actorID: INT, awardID: INT, PRIMARY KEY: (actorID, awardID), Foreign Key: actorID referencing Actor, Foreign Key: awardID referencing Award)

Process for 2NF

Table are in First Normal Form, all attributes for all entities and relations are fully functionally dependant on their respective primary or composite key, therefore all tables are already in Second Normal Form, no changes need to be made.

Process for 3NF

Each table is already in Third Normal Form as they don't have any transitive dependencies, and all non-prime attributes are directly dependent on the primary keys, so no changes need to be made.

DDL Statements

```
CREATE TABLE IF NOT EXISTS Movie (
        movieID INT AUTO_INCREMENT,
        title VARCHAR(255) NOT NULL,
        releaseDate DATE NOT NULL,
        PRIMARY KEY (movieID)
);

CREATE TABLE IF NOT EXISTS Studio (
        studioID INT AUTO_INCREMENT,
        name VARCHAR(255) NOT NULL,
        PRIMARY KEY (studioID),
);
```

```
CREATE TABLE IF NOT EXISTS Director (
        directorID INT AUTO_INCREMENT,
        firstName VARCHAR(255) NOT NULL,
        lastName VARCHAR(255) NOT NULL,
        dateOfBirth CHECK NOT NULL,
        PRIMARY KEY (directorID),
);

CREATE TABLE IF NOT EXISTS Actor (
        actorID INT AUTO_INCREMENT,
        firstName VARCHAR(255) NOT NULL,
        lastName VARCHAR(255) NOT NULL,
        dateOfBirth DATE NOT NULL,
        PRIMARY KEY (actorID),
);

CREATE TABLE IF NOT EXISTS Award (
        awardID INT AUTO_INCREMENT,
        type VARCHAR(255) NOT NULL,
        PRIMARY KEY (awardID)
);

CREATE TABLE IF NOT EXISTS MoviesMade (
        studioID INT,
        movieID INT,
        PRIMARY KEY (studioID, movieID),
        FOREIGN KEY (studioID) REFERENCES Studio(studioID,
        FOREIGN KEY (movieID) REFERENCES Movie(movieID)
);
```

```
CREATE TABLE IF NOT EXISTS MoviesDirected (
        directorID INT,
        movieID INT,
        PRIMARY KEY (directorID, movieID),
        FOREIGN KEY (directorID) REFERENCES Director(directorID),
        FOREIGN KEY (movieID) REFERENCES Movie(movieID)
);

CREATE TABLE IF NOT EXISTS `Cast` (
        movieID INT,
        actorID INT,
        PRIMARY KEY (movieID, actorID),
        FOREIGN KEY (movieID) REFERENCES Movie(movieID) ON DELETE CASCADE,
        FOREIGN KEY (actorID) REFERENCES Actor(actorID) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS AwardWinningMovies (
        movieID INT,
        awardID INT,
        PRIMARY KEY (movieID, awardID),
        FOREIGN KEY (movieID) REFERENCES Movie(movieID) ON DELETE CASCADE,
        FOREIGN KEY (awardID) REFERNCES Award(awardID) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS AwardWinningActors (
        actorID INT,
        awardID INT,
        PRIMARY KEY (actorID, awardID),
        FOREIGN KEY (actorID) REFERENCES Actor(actorID) ON DELETE CASCADE,
        FOREIGN KEY (awardID) REFERNCES Award(awardID) ON DELETE CASCADE
);
```

Rational Algebra Formulations

Query 1: Delete specific director entity set from Director Table.

$\pi_{\text{directorID, firstName, lastName, dateOfBirth}}$ (Director) – $\sigma_{\text{directorID = 2}}$ (Director)

Query 2: Delete specific studio entity set from Studio Table.

$\pi_{\text{studioID, name}}$ (Studio) – $\sigma_{\text{studioID = 5}}$ (Studio)

Query 3: Find movie with the most actors.

R1 = $\sigma_{\text{COUNT() = max(COUNT())}}$ (Movie ⋈ Cast)

π_{title} (R1)

Query 4: Find the actor who has won the most awards.

R1 = $\sigma_{\text{COUNT() = max(COUNT())}}$ (Award ⋈ AwardWinningActors)

$\pi_{\text{firstName, lastName}}$ (R1)

DML Statements

Query 1: Delete specific director entity set from Director Table.

```
DELETE FROM Director WHERE directorID = 2;
```

Query 2: Delete specific studio entity set from Studio Table.

```
DELETE FROM Studio WHERE studioID = 5;
```

Query 3: Find movie with the most actors.

```
SELECT Movie.title FROM Movie JOIN `Cast` ON Movie.movieID = `Cast`.movieID GROUP BY Movie.movieID, Movie.title HAVING COUNT(`Cast`.actorID) = (SELECT MAX (actor_count) FROM (SELECT COUNT(actorID) AS actor_count FROM `Cast` GROUP BY movieID) AS actor_counts);
```

Query 4: Find the actor who has won the most awards.

```
SELECT Actor.firstName, Actor.lastName FROM Actor JOIN AwardWinningActors ON Actor.actorID =
AwardWinningActors.actorID GROUP BY Actor.actorID HAVING
COUNT(AwardWinningActors.awardID) = (SELECT MAX(award_count) FROM (SELECT
COUNT(awardID) AS award_count FROM AwardWinningActors GROUP BY actorID) AS actor_awards);
```