1 Постановка задачи

Цели лабораторной работы:

1. Получить физическую модель базы данных для выбранной предметной области.
2. Сформулировать возможные запросы к базе даных.

2 Построение физической модели базы данных

2.1 Физическая модель основных сущностей

Для построения физической модели базы данных возьмем за основу уже построенную в Lab1 логическую модель базы данных.

Определимся с типами атрибутов для сущности произведение киноиндустрии(Title):

1. **titleID** - CHAR(9).
2. **titleType** – ENUM(movie, tvSeries, etc.)
3. **primaryTitle** – VARCHAR.
4. **originalTitle** - VARCHAR.
5. **isAdult** - BOOLEAN (BIT(1)).
6. **startYear** – YEAR.
7. **runtimeMinutes** – TINYINT UNSIGNED NOT NULL.
8. **posterURL** - VARCHAR NULLABLE.
9. **plot** - TEXT NULLABLE.

```
CREATE TABLE `imdb_db`.`title` (
  `title_ID` CHAR(9) NOT NULL,
  `title_type` TEXT NOT NULL,
  `primary_title` TEXT NOT NULL,
  `original_title` TEXT NOT NULL,
  `isAdult` BIT(1) NOT NULL,
  `startYear` YEAR NOT NULL,
  `runtimeMinutes` SMALLINT UNSIGNED NOT NULL,
  `posterURL` TEXT NULL,
  `plot` TEXT NULL,
 PRIMARY KEY (`title_ID`));
```

Определимся с типами атрибутов для сущности рейтинг(Rating):

1. **ratingID** - PK(titleID, ratingType).
2. **titleID** - CHAR(9) NOT NULL.
3. **averageRating** - DOUBLE UNSIGNED NOT NULL.
4. **numVotes** - INT UNSIGNED NOT NULL.
5. **ratingType** - ENUM(IMDB, Metacritic, etc.) NOT NULL.

```
CREATE TABLE `imdb_db`.`rating` (
  `titleID` CHAR(9) NOT NULL,
  `numVotes` INT UNSIGNED NOT NULL DEFAULT 0,
  `averageRating` DOUBLE UNSIGNED NOT NULL DEFAULT 0,
  `ratingType` ENUM('IMDB', 'METACRITIC') NOT NULL,
  PRIMARY KEY (`titleID`, `ratingType`),
  CONSTRAINT `rating_titleID_FK`
    FOREIGN KEY (`titleID`)
    REFERENCES `imdb_db`.`title`(`titleID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Определимся с типами атрибутов для сущности жанр(GENRE):

1. **genreID** - TINYINT AUTOINCREMENT UNSIGNED.
2. **genreType** - ENUM(Drama, comedy, etc.).

```
CREATE TABLE `imdb_db`.`genre` (
  `genreID` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `genre` ENUM('Action', 'Comedy', etc.) NOT NULL,
  PRIMARY KEY (`genreID`),
  UNIQUE INDEX `idgenre_UNIQUE` (`genreID` ASC));
```

Определимся с типами атрибутов для сущности эпизод сериала(SeriesEpisode):

1. **titleID** - CHAR(9).
2. **parentTitleID** - CHAR(9)
3. **seasonNumber** - TINYINT UNSIGNED.
4. **episodeNumber** - TINYINT UNSIGNED.

```
CREATE TABLE `imdb_db`.`series_episode` (
  `titleID` CHAR(9) NOT NULL,
  `parentTitleID` CHAR(9) NOT NULL,
  `seasonNumber` TINYINT UNSIGNED NOT NULL,
  `episodeNumber` TINYINT UNSIGNED NOT NULL,
  PRIMARY KEY (`titleID`),
  CONSTRAINT `series_episode_titleID_FK`
    FOREIGN KEY (`parentTitleID`)
    REFERENCES `imdb_db`.`title`(`titleID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Определимся с типами атрибутов для сущности деятель киноиндустрии(Name):

1. **nameID** - VARCHAR(10).
2. **primaryName** - TEXT.
3. **birthYear** - DATE.
4. **deathYear** - DATE.

```
CREATE TABLE `imdb_db`.`name` (
  `nameID` CHAR(10) NOT NULL,
  `primaryName` TEXT NOT NULL,
  `birthDate` DATE NOT NULL,
  `deathDate` DATE NULL,
  PRIMARY KEY (`nameID`));
```

Определимся с типами атрибутов для сущности профессия(Profession):

1. **professionID** - TINYINT AUTOINCREMENT UNSIGNED.
2. **jobType** - ENUM(director, writer, etc.).

```
CREATE TABLE `imdb_db`.`profession` (
  `professionID` TINYINT NOT NULL AUTO_INCREMENT,
  `profession` ENUM('Director', 'Writer', etc.) NOT NULL,
  PRIMARY KEY (`professionID`));
```

2.2 Реализация связывающих таблиц

Создадим промежуточную таблицу titleGenre для реализации связи многие ко многим между сущностями Title и Genre.

```
CREATE TABLE `imdb_db`.`title_genre` (
  `titleID` CHAR(9) NOT NULL,
  `genreID` TINYINT UNSIGNED NOT NULL,
  PRIMARY KEY (`titleID`, `genreID`),
  CONSTRAINT `title_genre_titleID_FK`
    FOREIGN KEY (`titleID`)
    REFERENCES `imdb_db`.`title`(`titleID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `title_genre_genreID_FK`
    FOREIGN KEY (`genreID`)
    REFERENCES `imdb_db`.`genre`(`genreID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Создадим промежуточную таблицу nameKnownForTitle для реализации связи многие ко многим между сущностями Name и Title.

```
CREATE TABLE `imdb_db`.`name_known_for_title` (
  `nameID` CHAR(10) NOT NULL,
  `titleID` CHAR(9) NOT NULL,
  PRIMARY KEY (`nameID`, `titleID`),
  CONSTRAINT `name_known_for_title_nameID_FK`
  FOREIGN KEY (`nameID`)
    REFERENCES `imdb_db`.`name`(`nameID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `name_known_for_title_titleID_FK`
  FOREIGN KEY (`titleID`)
    REFERENCES `imdb_db`.`title`(`titleID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Создадим промежуточную таблицу nameProfessions для реализации связи многие ко многим между сущностями Name и Profession.

```
CREATE TABLE `imdb_db`.`name_professions` (
  `nameID` CHAR(10) NOT NULL,
  `professionID` TINYINT NOT NULL,
```

```
  PRIMARY KEY (`nameID`, `professionID`),
  CONSTRAINT `name_professions_nameID_FK`
  FOREIGN KEY (`nameID`)
    REFERENCES `imdb_db`.`name`(`nameID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `name_professions_professionID_FK`
  FOREIGN KEY (`professionID`)
    REFERENCES `imdb_db`.`profession`(`professionID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

Создадим промежуточную таблицу principal для реализации связи многие ко многим между сущностями Name, Profession и Title.

```
CREATE TABLE `imdb_db`.`principal` (
  `titleID` CHAR(9) NOT NULL,
  `nameID` CHAR(10) NOT NULL,
  `professionID` TINYINT NOT NULL,
  PRIMARY KEY (`titleID`, `nameID`, `professionID`),
  CONSTRAINT `principal_titleID_FK`
  FOREIGN KEY (`titleID`)
    REFERENCES `imdb_db`.`title`(`titleID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `principal_nameID_FK`
  FOREIGN KEY (`nameID`)
    REFERENCES `imdb_db`.`name`(`nameID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION,
  CONSTRAINT `principal_professionID_FK`
  FOREIGN KEY (`professionID`)
    REFERENCES `imdb_db`.`profession`(`professionID`)
    ON DELETE CASCADE
    ON UPDATE NO ACTION);
```

3 Используемые источники

1. https://dev.mysql.com/doc/refman/8.0/en/charset.html
2. https://dev.mysql.com/doc/refman/8.0/en/data-types.html

3. https://www.mysqldatatypes.com/

4. https://dev.mysql.com/doc/refman/8.0/en/create-table.html

5. https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html

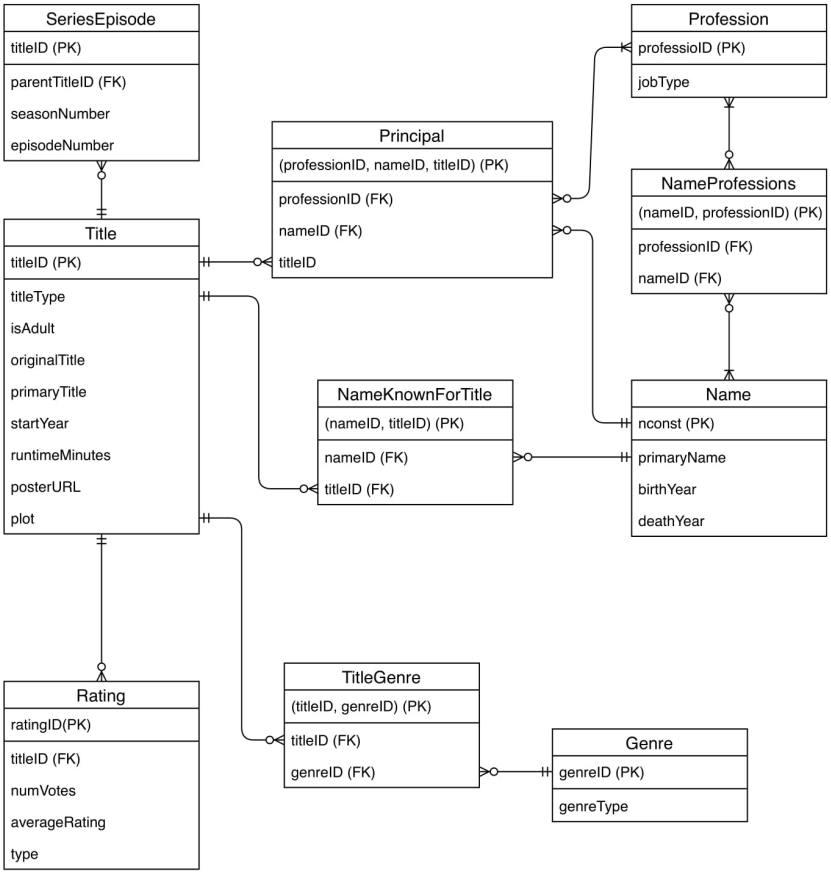

Figure 1: Logical model

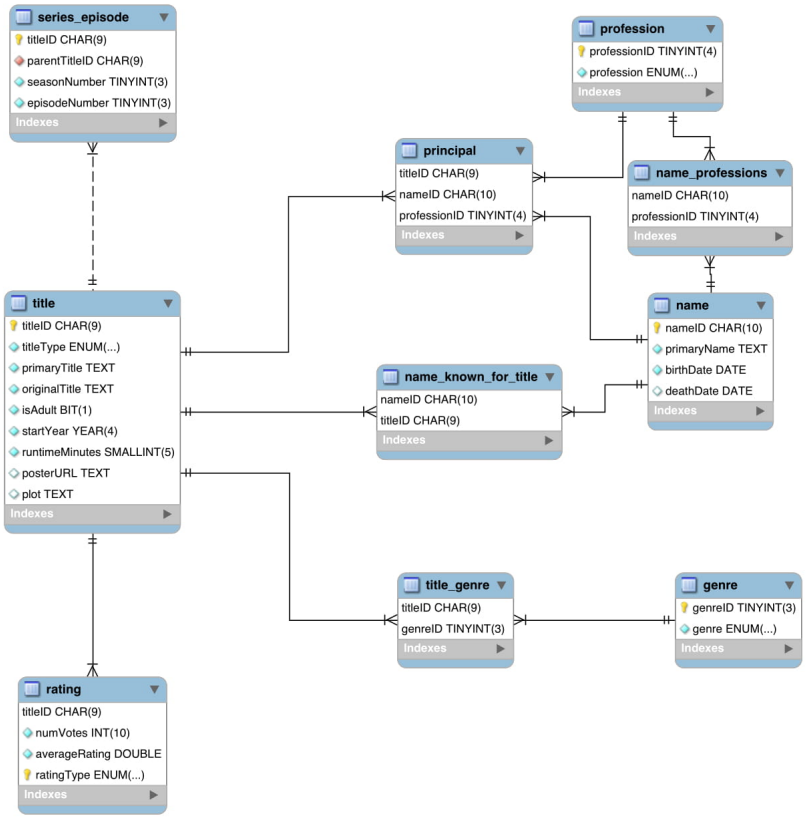

Figure 2: Physical model