Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

База данных сервиса о кино

БГУИР КП 1-40 01 01 4 ПЗ

Студент: гр. 751001

Бобко А. В.

Руководитель:

асс. Фадеева Е. Е.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc58960756)

[1 Анализ предметной области 6](#_Toc58960757)

[1.1 Обзор аналогов 6](#_Toc58960758)

[1.2 Постановка задачи 8](#_Toc58960759)

[2 Реализация 9](#_Toc58960760)

[2.1 Выбор СУБД и средств разработки 9](#_Toc58960761)

[2.1.1. Типы баз данных 9](#_Toc58960762)

[2.1.2. Сведения о языке SQL 10](#_Toc58960763)

[2.1.3. Выбор СУБД 10](#_Toc58960764)

[2.1.4. Выбор средств разработки 11](#_Toc58960765)

[2.2 Особенности реализации 12](#_Toc58960766)

[2.3 Пользователи системы 13](#_Toc58960767)

[2.4 Обработка данных 14](#_Toc58960768)

[2.5 Взаимодействие приложения с базой данных 16](#_Toc58960769)

[3 Модель предметной области 18](#_Toc58960770)

[3.1 Понятие инфологической модели 18](#_Toc58960771)

[3.2 Выделение объектов предметной области 18](#_Toc58960772)

[3.3 Выделение атрибутов объектов предметной области 18](#_Toc58960773)

[3.4 Бизнес-логика 23](#_Toc58960774)

[3.5 Проверка на нормализацию 24](#_Toc58960775)

[4 Тестирование базы данных 25](#_Toc58960776)

[ЗАКЛЮЧЕНИЕ 26](#_Toc58960777)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc58960778)

[Приложение А Схема базы данных на языке SQL 28](#_Toc58960779)

ВВЕДЕНИЕ

Кинематограф появился в конце XIX века и, начиная от XX века и до наших дней, является одним из самых популярных видов искусства. Регулярно в кинотеатрах показываются премьеры новых фильмов, которые собирают огромное количество зрителей, а гонорары известных актеров могут достигать миллионы долларов. Постоянно проводятся различного рода фестивали. В 2016 году оборот киноиндустрии составил примерно 2 триллиона долларов. Только Голливуд выпускает в среднем около 100-120 фильмов в год. Количество подписчиков онлайн-кинотеатров постоянно растет, регулярно открываются новые студии. У известных стриминговых сервисов число пользователей достигает почти 200 миллионов человек.

Так как популярность кино постоянно набирает популярность, стали появляться различные киносервисы. Сейчас существует куча онлайн-кинотеатров и стриминговых сервисов, которые позволяют смотреть фильмы и сериалы в интернете. Также люди по всему миру активно интересуются новинками кино, ждут премьеры фильмов и читают новости из жизни кинозвезд. Поэтому в интернете существует множество журналов, блогов и информационных сайтов, в которых постоянно публикуются различного рода статьи, обзоры и новости кино.

Сейчас в интернете можно найти большое количество сайтов о кино, которые имеют информацию об колоссальном количестве фильмов, знаменитостей в мире кино, видео контента. Такие сайты стараются покрыть наибольшее количество существующих фильмов, а также предоставить пользователю возможность сохранять, сортировать, выставлять рейтинги и комментировать эти фильмы. Так как существует множество схожих сайтов, каждый сервис старается сделать так, чтобы пользователь захотел пользоваться именно его услугами.

Для таких сервисов необходима удобная база данных, которая сможет хранить огромные объемы информации и предоставлять пользователям необходимую им функциональность на сайте.

Целью данного курсового проекта является изучение основных принципов построения баз данных, а также проектирование реляционной базы данных, которая может использоваться в дальнейшем для создания сервиса о кино, который сможет предоставлять пользователям возможность сохранять информацию о кино, получать о них необходимую информацию, а также высказывать свое мнение и выставлять оценки фильмам и сериалам.

1. Анализ предметной области

В настоящее время на просторах интернета можно найти различные сайты и сервисы, содержащие огромные базы данных о фильмах и людях, работающих в сфере кино. Они предоставляют много различных функций, такие как оценка фильмов, бронирование и покупка билетов, разделение фильмов по папкам, получение уведомлений о новых премьерах. В ходе проектирования были проанализированы и изучены некоторые аналоги, выявлены их положительные и отрицательные стороны. Далее проведено несколько наиболее известных примеров.

* 1. Обзор аналогов

Сервис IMDb, или Internet Movie Database, является крупнейшей в мире базой данных и сайтом о кинематографе. В базе собрана информация о более чем 6,5 миллионах кинофильмов, телесериалов и отдельных серий, а также о 10,4 миллионах персоналий, связанных с кино, — актерах, режиссерах, сценаристах и др. Одной из главных особенностей IMDb является то, что редактировать информацию в базе данных может любой пользователь, прошедший регистрацию и модерацию на сайте. Пример страницы фильма на сайте IMDb отображена на рисунке 1.1.

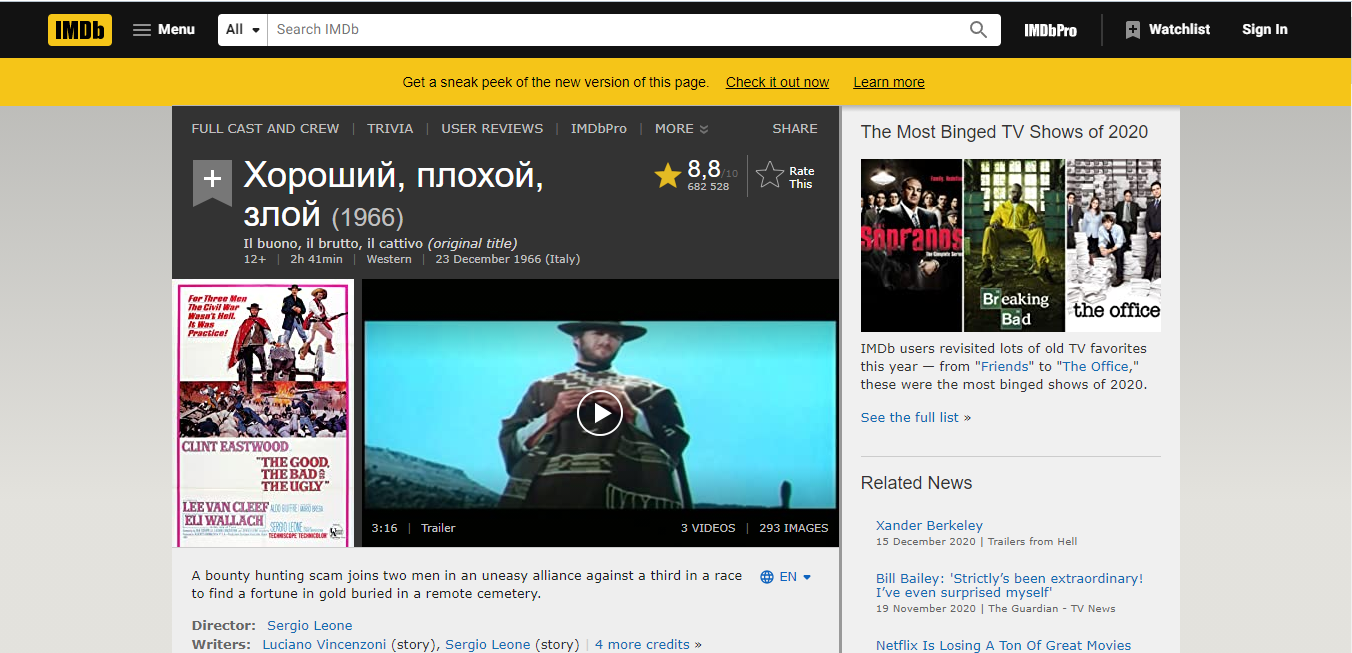


Рисунок 1.1 – страница фильма на сайте IMDb

На сайте содержится много полезной информации о фильме, например, его продолжительность, возрастные ограничения, видео- и фотоматериалы, актерский состав. Каждый пользователь может написать рецензию к фильму и поставить свою оценку фильму, из которой потом формируется средний рейтинг фильма. По этому рейтингу в дальнейшем создается список из наиболее хорошо оцененных фильмов.

Сайт позволяет формировать список фильмов, который пользователь хочет посмотреть, а также создавать свои списки. Также на сайте можно просмотреть список всех сезонов и серий для каждого сериала, а также краткое описание к ним, что является очень полезной функцией.

В качестве минуса можно только отметить наличие на данном сайте помимо кино информации о компьютерных играх, хотя сайт по большей части специализируется именно на кино.

Следующим аналогом является сервис КиноПоиск. Данный сайт очень похож по функциональности на IMDb, но является более популярным в русскоговорящих странах и считается крупнейшим русскоязычным сервисом о кино.

Из особенностей данного сайта можно подчеркнуть наличие онлайн-кинотеатра, что позволяет просматривать фильмы сразу на сайте по подписке.

Страница фильма на данном сайте изображена на рисунке 1.2.

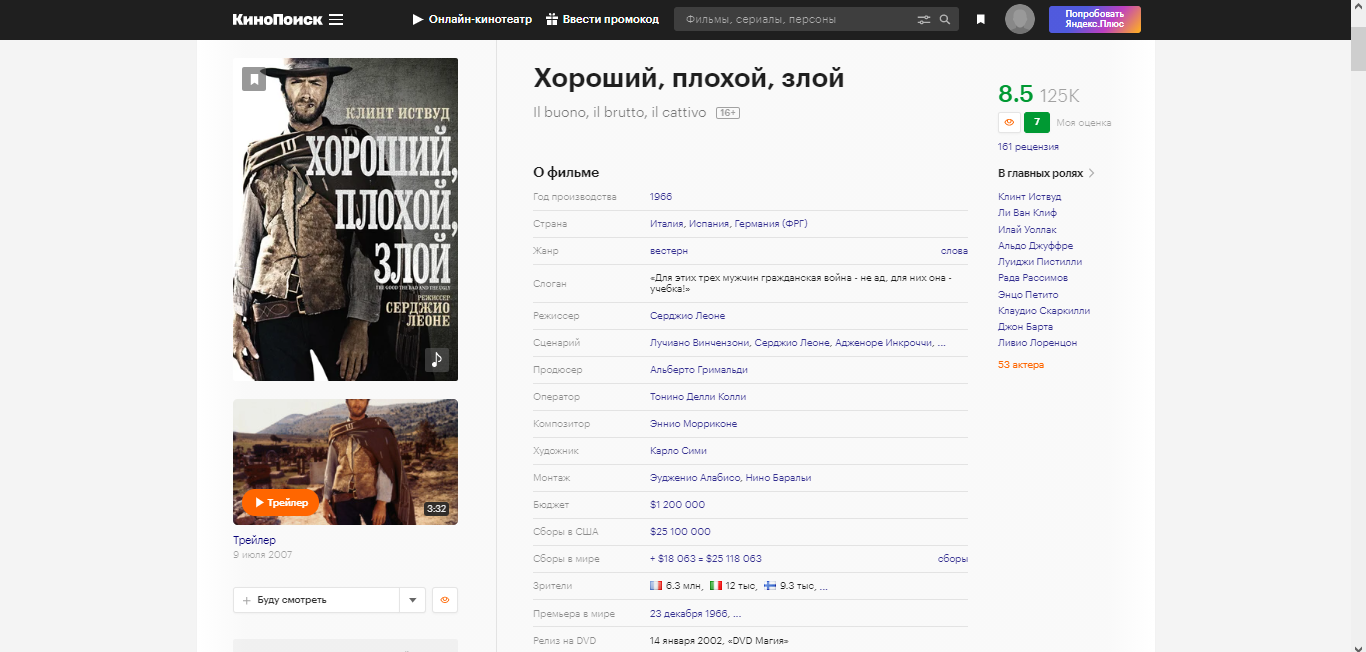


Рисунок 1.2 – пример страницы фильма на сервисе КиноПоиск

Данный сервис позволяет пользователям выставлять оценки фильмам, писать рецензии, а также сортировать фильмы по папкам. Еще в качестве плюса можно отметить наличие статистики для каждого пользователя, которая позволяет анализировать свои предпочтения, а также смотреть людей, с наиболее схожими интересами.

Еще одним из положительных качеств является наличие на сайте информации о кинотеатрах: их местоположение, время работы, сайты, номера телефонов.

В качестве недостатком сайта можно отметить отсутствие подробной информации о каждой серии или сезоне сериала, из-за чего приходится искать такую информацию на других ресурсах.

Последним приведенным аналогом является раздел Кино сайта Афиша TUT.BY. Данный сайт направлен больше на получение информации о киносеансах и обладает не очень большим объемом информации, а также направлен больше на белорусскую аудиторию. Пример страницы фильма показан на рисунке 1.3.

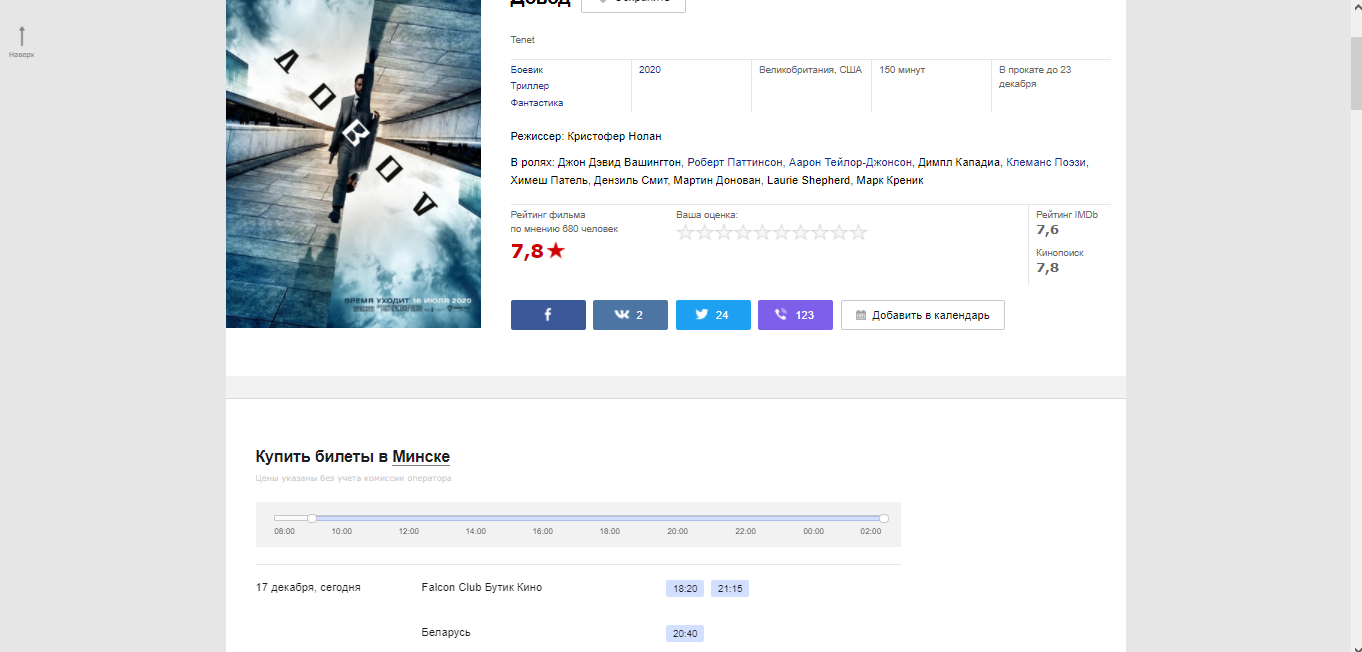


Рисунок 1.3 – пример страницы фильма на сервисе Афиша TUT.BY

Данный сайт содержит только фильмы, которые выходят в прокате. Тут также можно выставлять оценки фильмам и писать рецензии. Несмотря на скудность данных по сравнению с предыдущими аналогами, сервис очень удобен для просмотра киносеансов и покупки билетов в кинотеатры в Беларуси.

* 1. Постановка задачи

В рамках данного курсового проекта планируется спроектировать базу данных, которую в дальнейшем можно будет использовать в различных киносервисах.

При проектировании базы данных планируется выполнить следующие задачи:

– регистрация пользователя в сервисе;

– возможность создания пользователем папок с фильмами;

– возможность пользователю выставлять оценки фильму и оставлять комментарии;

– создание различных хранимых процедур и триггеров для удобной работы с данными;

– создание индексов для ускорения поиска по данным сервиса;

– нормализовать полученную базу данных.

В рамках курсового проекта будет спроектирована и разработана реляционная база данных, содержащая сущности и связи, описывающие предметную область киносервиса (фильм, награда, фестиваль, киностудия и т.д.).

Для проектирования базы данных планируется использовать реляционную систему управления базами данных MySQL, а также инструмент для визуального проектирования баз данных MySQL Workbench.

1. Разработка базы данных
   1. Выбор СУБД и средств разработки
      1. Типы баз данных

Существует множество различных типов баз данных. Необходимый тип выбирается индивидуально для каждой предметной области, в зависимости от ожидаемого функционала и возможностей разрабатываемого приложения.

Некоторые типы баз данных:

– картотеки;

– сетевые базы данных;

– иерархические базы данных;

– объектно-ориентированные базы данных;

– реляционные базы данных;

– дедуктивные базы данных;

– NoSQL базы данных.

Каждый из типов имеет свои преимущества и недостатки.

Наиболее подходящими могут быть следующие типы: картотека, реляционная, объектно-ориентированная или NoSQL базы данных.

Картотека представляет собой упорядоченное (по алфавиту, дате и т.п.) собрание данных в виде записей, каждая из которых предоставляет сведения о каком-то объекте базы данных.

Реляционные базы данных основаны на теоретико-множественной реляционной даталогической модели. Все данные представлены в виде связанных между собой таблиц, разбитых на строки и столбцы.

В объектно-ориентированных базах данных данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями. Это технология напоминает объектно-ориентированное программирование в применении к БД.

База данных NoSQL, или нереляционная база данных, позволяет хранить и обрабатывать неструктурированные или слабоструктурированные данные (в отличие от реляционной базы данных, задающей структуру содержащихся в ней данных). Популярность баз данных NoSQL растет по мере распространения и усложнения веб-приложений.

В рамках данного курсового проекта была выбрана реляционная модель базы данных, так как мы имеем структурированные данные. Реляционная модель понятна и легко визуализируема. Существует множество инструментов для удобной работы с реляционными базами данных. Теоретическая база для данного вида баз данных разрабатывалась долгое время и имеет конкретное математическое обоснование своей эффективности.

* + 1. Язык SQL

Для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных, используется декларативный язык программирования SQL.

SQL является информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. Изначально SQL был основным способом работы пользователя с базой данных, но со временем SQL усложнился – обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) – и стал приобретать черты, свойственные языкам программирования

При всех своих изменениях SQL остаётся самым распространённым лингвистическим средством для взаимодействия прикладного программного обеспечения с базами данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

* + 1. Выбор СУБД

В качестве реляционной системы управления базами данных выбрана MySQL, так как она является свободной и удобной реляционной СУБД. Ее разработку и поддержку которой осуществляет корпорация Oracle.

Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

* + 1. Выбор средств разработки

В качестве средства разработки базы данных был выбран инструмент для визуального проектирования баз данных MySQL Workbench, интегрирующий проектирование, моделирование, создание и эксплуатацию базы данных в единое бесшовное окружение для системы баз данных MySQL. Он очень удобен для проектирования больших баз данных, так как предоставляет визуальный интерфейс для разработки. Пример создания таблиц базы данных в MySQL Workbench представлен на рисунке 2.1.

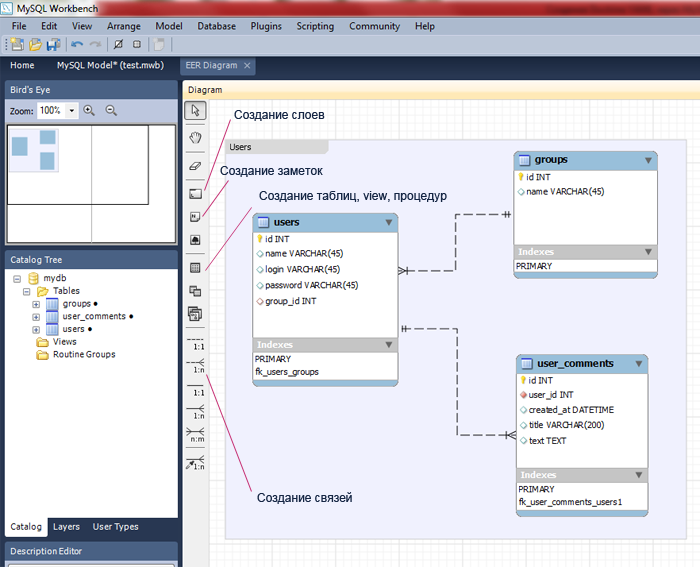


Рисунок 2.1 – пример создания таблиц базы данных в MySQL Workbench

Ниже приведены некоторые преимущества инструмента MySQL Workbench:

– позволяет наглядно представить модель базы данных в графическом виде;

– наглядный и функциональный механизм установки связей между таблицами;

– удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответы в виде таблицы;

– возможность редактирования данных в таблице в визуальном режиме.

* 1. Особенности реализации

Далее перечислен основной функционал, реализованный в базе данных:

– возможность добавления фильма пользователем в одну из его папок;

– возможность выставления оценки фильму;

– вычисление среднего рейтинга фильма;

– вывод 100 лучших фильмов согласно оценкам пользователей;

– вычисление среднего рейтинга фильма;

– выставление эмоции like/dislike статьям, франшизам, странам и жанрам, чтобы в дальнейшем можно было выстраивать рекомендательные системы с помощью существующей базы данных;

– возможность создания ассоциации между статьей и фильмом, персоной, студией;

– возможность создания ассоциации между фото и фильмом, персоной, студией.

Также было добавлено использования индексов для ускорения поиска по названию фильма, фамилии знаменитости, кинотеатра по названию, статьи по заглавию и т.д.

При разработке схемы базы данных было уделено внимание безопасности. Аккаунт пользователя защищен паролем, при этом в базе данных хранится только его хэш.

* 1. Пользователи системы и их роли

Для сервиса о кино было принято выделить несколько основных типов пользователей.

Первым типом пользователей является гость. Это пользователи, зашедшие на сервис и не прошедшие процедуры авторизации. Такие пользователи могут только просматривать существующие фильмы, знаменитостей, награды, фестивали и т.д. Также они могут оставлять анонимные комментарии под фильмом.

Второй тип пользователей – это юзеры. Они являются пользователями, которые когда-то выполнили регистрацию и авторизовались на сайте. Для таких пользователей функционал намного шире, чем для гостей. Они могут создавать свои папки, ставить оценки фильмам, выбирать понравившиеся статьи, жанры, страны. Также они могут отмечать, на какие сеансы они идут. При разработке приложения это позволит добавить функционал напоминания о ближайших киносеансах.

Следующий тип пользователей – модераторы. Они могут не только просматривать информацию, но и редактировать ее. Это позволит сделать информацию наиболее достоверной и понятной.

Последним типом является администратор. Такие пользователи имеют функционал модераторов, при этом они могут сами назначать и удалять модераторов.

1. Модель предметной области
   1. Понятие инфологической модели

Инфологическая модель – ориентированная на человека и не зависимая от типа СУБД модель предметной области, определяющая совокупности информационных объектов, их атрибутов и отношений между объектами, динамику изменений предметной области, а также характер информационных потребностей пользователей.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

* 1. Выделение объектов предметной области

Проанализировав предметную область разрабатываемой базы данных, можно выделить следующие основные объекты:

– видео;

– жанр;

– комментарий;

– киносеанс;

– кинотеатр;

– награда;

– персона;

– пользователь;

– роль пользователя;

– страна;

– студия;

– сериал;

– сезон;

– саундтрек;

– статья;

– фильм;

– фестиваль;

– фото;

– франшиза;

– эмоция;

– эпизод.

* 1. Выделение атрибутов объектов предметной области

Атрибуты основных и вспомогательных сущностей разрабатываемой базы данных представлены в таблице 3.1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Сущность | Атрибут | Тип | Значение | Ограничения |
| 1 | user (пользователь) | id | bigint | идентификатор | первичный ключ |
| image | varchar(255) | адрес изображения |  |
| password | timestamp | хэш пароля | не пустой |
| registration\_date | varchar(255) | дата регистрации |  |
| username | varchar(255) | имя пользователя | не пустой |
| 2 | role (роль) | id | bigint | идентификатор | первичный ключ |
| name | varchar(255) | имя | уникальный |
| 3 | tag (тег) | id | bigint | идентификатор | первичный ключ |
| name | varchar(255) | имя |  |
| description | varchar(255) | описание |  |
| 4 | artist (исполнитель) | id | bigint | идентификатор | первичный ключ |
| name | varchar(255) | название |  |
| description | text | описание |  |
| image | varchar(255) | адрес изображения |  |
| 5 | track (аудиозапись) | id | bigint | идентификатор | первичный ключ |
| artist\_id | bigint | ид. исполнителя | внешний ключ |
| length | int | длина |  |
| media\_file | varchar(255) | адрес mp3-файла |  |
| name | varchar(255) | название |  |
| upload\_date | timestamp | дата загрузки |  |
| uploader\_id | bigint | ид. загрузившего пользователя | внешний ключ |
| 6 | playlist (плейлист) | id | bigint | идентификатор | первичный ключ |
| create\_date | timestamp | дата создания |  |
| name | varchar(255) | название |  |
| owner\_id | bigint | ид. создавшего пользователя | внешний ключ |
| 7 | album (альбом) | id | bigint | идентификатор | первичный ключ |
| name | varchar(255) | название |  |
| release\_date | timestamp | дата выхода |  |
| artist\_id | bigint | ид. исполнителя | внешний ключ |
| label\_id | bigint | ид. лейбла | внешний ключ |
| 8 | playlist\_tracks (аудиозапись в плейлисте) | playlist\_id | bigint | ид. плейлиста | первичный ключ, внешний ключ |
| tracks\_id | bigint | ид. аудиозаписи | первичный ключ, внешний ключ |
| tracks\_order | int | порядковый номер | не пустой |
| 9 | album\_tracks (аудиозапись в альбоме) | album\_id | bigint | ид. альбома | первичный ключ, внешний ключ |
| tracks\_id | bigint | ид. аудиозаписи | первичный ключ, внешний ключ |
| tracks\_order | int | порядковый номер | не пустой |
| 10 | track\_tag (тег аудиозаписи) | track\_id | bigint | ид. аудиозаписи | первичный ключ, внешний ключ |
| tag\_id | bigint | ид. тега | первичный ключ, внешний ключ |
| 11 | playlist\_tag (тег плейлиста) | playlist\_id | bigint | ид. плейлиста | первичный ключ, внешний ключ |
| tag\_id | bigint | ид. тега | первичный ключ, внешний ключ |
| 12 | album\_tag (тег альбома) | album\_id | bigint | ид. альбома | первичный ключ, внешний ключ |
| tag\_id | bigint | ид. тега | первичный ключ, внешний ключ |
| 13 | label (лейбл) | id | bigint | идентификатор | первичный ключ |
| name | varchar(255) | название |  |
| description | text | описание |  |
| country | varchar(255) | страна |  |
| 14 | track\_coment (комментарий к аудиозаписи) | id | bigint | идентификатор | первичный ключ |
| comment\_date | timestamp | дата оставления |  |
| text | text | текст |  |
| track\_timestamp | int | временная отметка аудиозаписи |  |
| track\_id | bigint | ид. аудиозаписи | внешний ключ, не пустой |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |
| 15 | track\_like (лайк аудиозаписи) | id | bigint | bigint | первичный ключ |
| like\_date | timestamp | дата |  |
| track\_id | bigint | ид. аудиозаписи | внешний ключ, не пустой |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |
| 16 | playlist\_like (лайк плейлиста) | id | bigint | bigint | первичный ключ |
| like\_date | timestamp | дата |  |
| playlist\_id | bigint | ид. плейлиста | внешний ключ, не пустой |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |
| 17 | album\_like (лайк плейлиста) | id | bigint | bigint | первичный ключ |
| like\_date | timestamp | дата |  |
| album\_id | bigint | ид. альбома | внешний ключ, не пустой |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |
| 18 | user\_subscriptions (подписка) | user\_id | bigint | ид. пользователя | первичный ключ, внешний ключ |
| subscriptions\_id | bigint | ид. пользователя, на которого подписан данный пользователь | первичный ключ, внешний ключ |
| 19 | user\_roles (роли пользователей) | user\_id | bigint | ид. пользователя | первичный ключ, внешний ключ |
| roles\_id | bigint | ид. роли | первичный ключ, внешний ключ |
| 20 | account\_upgrade\_type (тип расширенного аккаунта) | id | bigint | идентификатор | первичный ключ |
| description | varchar(255) | описание |  |
| name | varchar(255) | название |  |
| 21 | account\_upgrade\_record (запись о расширении аккаунта) | id | bigint | идентификатор | первичный ключ |
| start\_date | datetime(6) | дата начала |  |
| end\_date | datetime(6) | дата окончания |  |
| account\_upgrade\_type\_id | bigint | ид. типа расширения аккаунта | внешний ключ, не пустой |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |
| 22 | blocking\_record (запись о блокировке аккаунта) | id | bigint | идентификатор | первичный ключ |
| blocked\_date | datetime(6) | дата начала блокировки |  |
| blocked\_until | datetime(6) | дата окончания блокировки |  |
| reason | varchar(255) | причина блокировки |  |
| user\_id | bigint | ид. пользователя | внешний ключ, не пустой |

Таблица 3.1 – Представление сущностей и атрибутов базы данных

* 1. Бизнес-логика

Бизнес-логика в базе данных представлена триггерами и процедурами.

Аудиозаписи в системе могут быть удалены. Такая ситуация может произойти по причине обращения правообладателя. Так как пользовательские плейлисты хранят порядковый номер аудиозаписи, при ее удалении целостность данных может быть нарушена – номер аудиозаписи в плейлисте будет пропущен. Чтобы не допустить данной ситуации, был создан триггер fix\_playlists. Он применяется при удалении записи из таблицы track. Если аудиозапись находилась в плейлистах, то после ее удаления для каждого плейлиста порядковые номера аудиозаписей следующей за удаляемой уменьшаются на 1, сохраняя корректность данных.

Аккаунты пользователей также могут быть удалены из системы. При этом комментарии, оставленные пользователями могут представлять ценность, и удалять их нежелательно. Для решения данной проблемы создан триггер set\_deleted\_user, который срабатывает при удалении аккаунта пользователя из системы. Для любых комментариев, оставленных данным пользователем, в качестве владельца устанавливается специальный пользователь с именем “DELETED”. Таким образом, после удаления пользователя, его комментарии будут сохранены.

Для системы рекомендаций может быть полезно выяснить предпочтения пользователя в музыке. В базе данных существует процедура find\_favourite\_track\_tag\_by\_playlists. Ее задача – для указанного пользователя определить любимый тег аудиозаписей. Она работает следующим образом: для всех аудиозаписей из плейлистов, созданных данным пользователем, подсчитывается число каждого тега. Самый популярный тег возвращается выходным параметром.

* 1. Проверка на нормализацию

Проверка на первую нормальную форму. В структуре базы данных отсутствуют сложные атрибуты, а также не наблюдаются многозначные атрибуты, т.е. таблицы приведены к первой нормальной форме.

Проверка на вторую нормальную форму. Просмотрев все отношения, описанные выше, можно сказать, что неключевые атрибуты этих отношений функционально полно зависят от первичных ключей, что является доказательством того, что отношения приведены ко второй нормальной форме.

Проверка на третью нормальную форму. Транзитивные функциональные зависимости отсутствуют, поэтому отношения соответствуют третьей нормальной форме.

1. Тестирование базы данных

Тестирование проводится с целью обеспечить качество разрабатываемого программного продукта. Стандарт ISO/IEC 12207-2003, посвященный описанию систем обеспечения качества программного обеспечения, под качеством понимает совокупность характеристик программного продукта, относящихся к его способности удовлетворять установленные и предполагаемые потребности клиента.

Для тестирования разработанной базы данных будут использоваться следующие запросы:

1. Добавление записи в таблицу.

INSERT INTO user (id, password, username) VALUES

(100, 'password', 'DELETED');

Ожидаемый результат: запись добавлена в таблицу.

1. Изменение записи в таблице.

UPDATE user

SET username = 'john1234'

WHERE id = 1;

Ожидаемый результат: имя пользователя с идентификатором 1 изменено на ‘john1234’.

1. Удаление записей, на которые не ссылаются записи другие из других таблиц.

DELETE FROM label WHERE id = 100;

Ожидаемый результат: запись удалится.

1. Удаление записей, на которые ссылаются записи другие из других таблиц.

DELETE FROM label WHERE id = 101;

Ожидаемый результат: удаление запрещено.

1. Выборка данных

SELECT text, comment\_date FROM track\_comment WHERE

track\_id = 5 ORDER BY comment\_date DESC;

Ожидаемый результат: вывод всех комментариев к аудиозаписи с идентификатором 5, отсортированных по дате в порядке убывания.

Для всех тестовых запросов реальный результат соответствует ожидаемому.

ЗАКЛЮЧЕНИЕ

Аудиосервисы сегодня пользуются большой популярностью. В рамках данной курсовой работы было разработано веб-приложение – аудиосервис Funsymusic. Для разработки использовалась платформа Spring, язык программирования Java и СУБД MySQL.

В процессе разработки были получены навыки создания построения веб-приложений с использованием базы данных и работы с СУБД MySQL.

Приложение реализует основные функции типичного аудиосервиса – такие, как создание аккаунта, авторизация в системе, загрузка своих аудиозаписей на сервис, воспроизведение аудиозаписей, создание, редактирование и воспроизведение плейлистов, возможность оставлять комментарии, ставить лайки, подписываться на других пользователей и просматривать активность подписок.

В приложении были реализованы следующие функции:

– создание аккаунта;

– авторизация в системе;

– возможность установить изображение профиля;

– возможность загрузки своих аудиозаписей на сервис;

– считывание метаданных mp3-файла при загрузке;

– воспроизведение аудиозаписей;

– создание и воспроизведение плейлистов;

– редактирование плейлистов (добавление, удаление аудиозаписей, изменение их порядка);

– возможность оставлять комментарии к аудиозаписи;

– возможность ставить лайки аудиозаписям и плейлистам;

– возможность подписаться на других пользователей;

– отображение активности пользователей;

– статусы аккаунта с дополнительными функциями;

– возможность блокировки аккаунта пользователя администратором сервиса.

Для приложения была спроектирована и разработана реляционная база данных, содержащая сущности и связи, отражающие предметную область музыкального сервиса, а также установлена связь между приложением и базой данных для выполнения запросов и получения их результатов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Брюс Эккель. Философия Java пер. Е. Матвеев. - 4-е полное изд. - Санкт-Петербург [и др.]: Питер, 2015. – 1165 с.
2. Г. Шилдт. Java 8. Полное руководство. 9-е полное издание. — М.: Вильямс, 2015. – 1377 с.
3. Документация Java SE 11 [электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.oracle.com/en/java/javase/11/docs/api/index.html>
4. Документация Spring Framework [электронный ресурс]. – Электронные данные. – Режим доступа: <https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html>
5. Справка MySQL 8.0 [электронный ресурс]. – Электронные данные. – Режим доступа: <https://dev.mysql.com/doc/refman/8.0/en/>
6. SQL – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/>wiki/SQL
7. MySQL – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/>wiki/MySQL
8. Реляционная СУБД – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/>wiki/Реляционная СУБД
9. Нормальная форма – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/>wiki/Нормальная форма

Приложение А  
Схема базы данных на языке SQL

-- MySQL dump 10.13 Distrib 8.0.22, for Win64 (x86\_64)

--

-- Host: localhost Database: funsymusicdatabase

-- ------------------------------------------------------

-- Server version 8.0.22

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `account\_upgrade\_record`

--

DROP TABLE IF EXISTS `account\_upgrade\_record`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `account\_upgrade\_record` (

`id` bigint NOT NULL,

`end\_date` datetime(6) DEFAULT NULL,

`start\_date` datetime(6) DEFAULT NULL,

`account\_upgrade\_type\_id` bigint NOT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_account\_upgrade\_record\_account\_upgrade\_type` (`account\_upgrade\_type\_id`),

KEY `FK\_account\_upgrade\_record\_user` (`user\_id`),

CONSTRAINT `FK\_account\_upgrade\_record\_account\_upgrade\_type` FOREIGN KEY (`account\_upgrade\_type\_id`) REFERENCES `account\_upgrade\_type` (`id`),

CONSTRAINT `FK\_account\_upgrade\_record\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `account\_upgrade\_type`

--

DROP TABLE IF EXISTS `account\_upgrade\_type`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `account\_upgrade\_type` (

`id` bigint NOT NULL,

`description` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `album`

--

DROP TABLE IF EXISTS `album`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `album` (

`id` bigint NOT NULL,

`name` varchar(255) DEFAULT NULL,

`release\_date` timestamp NULL DEFAULT NULL,

`artist\_id` bigint DEFAULT NULL,

`label\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_album\_artist` (`artist\_id`),

KEY `FK\_album\_label` (`label\_id`),

CONSTRAINT `FK\_album\_artist` FOREIGN KEY (`artist\_id`) REFERENCES `artist` (`id`),

CONSTRAINT `FK\_album\_label` FOREIGN KEY (`label\_id`) REFERENCES `label` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `album\_like`

--

DROP TABLE IF EXISTS `album\_like`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `album\_like` (

`id` bigint NOT NULL,

`like\_date` timestamp NULL DEFAULT NULL,

`album\_id` bigint NOT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_album\_like\_album` (`album\_id`),

KEY `FK\_album\_like\_user` (`user\_id`),

CONSTRAINT `FK\_album\_like\_album` FOREIGN KEY (`album\_id`) REFERENCES `album` (`id`),

CONSTRAINT `FK\_album\_like\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `album\_tag`

--

DROP TABLE IF EXISTS `album\_tag`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `album\_tag` (

`album\_id` bigint NOT NULL,

`tag\_id` bigint NOT NULL,

PRIMARY KEY (`album\_id`,`tag\_id`),

KEY `FK\_album\_tag\_tag` (`tag\_id`),

CONSTRAINT `FK\_album\_tag\_album` FOREIGN KEY (`album\_id`) REFERENCES `album` (`id`),

CONSTRAINT `FK\_album\_tag\_tag` FOREIGN KEY (`tag\_id`) REFERENCES `tag` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `album\_tracks`

--

DROP TABLE IF EXISTS `album\_tracks`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `album\_tracks` (

`album\_id` bigint NOT NULL,

`tracks\_id` bigint NOT NULL,

`tracks\_order` int NOT NULL,

PRIMARY KEY (`album\_id`,`tracks\_order`),

KEY `FK\_album\_tracks\_track` (`tracks\_id`),

CONSTRAINT `FK\_album\_tracks\_album` FOREIGN KEY (`album\_id`) REFERENCES `album` (`id`),

CONSTRAINT `FK\_album\_tracks\_track` FOREIGN KEY (`tracks\_id`) REFERENCES `track` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `artist`

--

DROP TABLE IF EXISTS `artist`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `artist` (

`id` bigint NOT NULL,

`name` varchar(255) DEFAULT NULL,

`description` text,

`image` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `blocking\_record`

--

DROP TABLE IF EXISTS `blocking\_record`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `blocking\_record` (

`id` bigint NOT NULL,

`blocked\_date` datetime(6) DEFAULT NULL,

`blocked\_until` datetime(6) DEFAULT NULL,

`reason` varchar(255) DEFAULT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_blocking\_record\_user` (`user\_id`),

CONSTRAINT `FK\_blocking\_record\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `label`

--

DROP TABLE IF EXISTS `label`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `label` (

`id` bigint NOT NULL,

`name` varchar(255) DEFAULT NULL,

`description` text,

`country` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `playlist`

--

DROP TABLE IF EXISTS `playlist`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `playlist` (

`id` bigint NOT NULL,

`create\_date` timestamp NULL DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

`owner\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_playlist\_user` (`owner\_id`),

CONSTRAINT `FK\_playlist\_user` FOREIGN KEY (`owner\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `playlist\_like`

--

DROP TABLE IF EXISTS `playlist\_like`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `playlist\_like` (

`id` bigint NOT NULL,

`like\_date` timestamp NULL DEFAULT NULL,

`playlist\_id` bigint NOT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_playlist\_like\_playlist` (`playlist\_id`),

KEY `FK\_playlist\_like\_user` (`user\_id`),

CONSTRAINT `FK\_playlist\_like\_playlist` FOREIGN KEY (`playlist\_id`) REFERENCES `playlist` (`id`),

CONSTRAINT `FK\_playlist\_like\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `playlist\_tag`

--

DROP TABLE IF EXISTS `playlist\_tag`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `playlist\_tag` (

`playlist\_id` bigint NOT NULL,

`tag\_id` bigint NOT NULL,

PRIMARY KEY (`playlist\_id`,`tag\_id`),

KEY `FK\_playlist\_tag\_tag` (`tag\_id`),

CONSTRAINT `FK\_playlist\_tag\_album` FOREIGN KEY (`playlist\_id`) REFERENCES `playlist` (`id`),

CONSTRAINT `FK\_playlist\_tag\_tag` FOREIGN KEY (`tag\_id`) REFERENCES `tag` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `playlist\_tracks`

--

DROP TABLE IF EXISTS `playlist\_tracks`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `playlist\_tracks` (

`playlist\_id` bigint NOT NULL,

`tracks\_id` bigint NOT NULL,

`tracks\_order` int NOT NULL,

PRIMARY KEY (`playlist\_id`,`tracks\_order`),

KEY `FK\_playlist\_tracks\_track` (`tracks\_id`),

CONSTRAINT `FK\_playlist\_tracks\_playlist` FOREIGN KEY (`playlist\_id`) REFERENCES `playlist` (`id`),

CONSTRAINT `FK\_playlist\_tracks\_track` FOREIGN KEY (`tracks\_id`) REFERENCES `track` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `role`

--

DROP TABLE IF EXISTS `role`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `role` (

`id` bigint NOT NULL,

`name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UK\_role\_name` (`name`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `tag`

--

DROP TABLE IF EXISTS `tag`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `tag` (

`id` bigint NOT NULL,

`name` varchar(255) DEFAULT NULL,

`description` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `track`

--

DROP TABLE IF EXISTS `track`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `track` (

`id` bigint NOT NULL,

`artist\_id` bigint DEFAULT NULL,

`length` int DEFAULT NULL,

`media\_file` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

`upload\_date` timestamp NULL DEFAULT NULL,

`uploader\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `FK\_track\_user` (`uploader\_id`),

KEY `FK\_track\_artist` (`artist\_id`),

CONSTRAINT `FK\_track\_artist` FOREIGN KEY (`artist\_id`) REFERENCES `artist` (`id`),

CONSTRAINT `FK\_track\_user` FOREIGN KEY (`uploader\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'STRICT\_TRANS\_TABLES,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`localhost`\*/ /\*!50003 TRIGGER `fix\_playlists` AFTER DELETE ON `track` FOR EACH ROW BEGIN

DECLARE pl\_id,

or\_num varchar(255);

DECLARE done int DEFAULT 0;

DECLARE cur CURSOR FOR

SELECT

playlist\_id,

tracks\_order

FROM playlist\_tracks

WHERE track\_id = OLD.id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

my\_loop:

LOOP

FETCH cur INTO pl\_id, or\_num;

IF done = 1 THEN

LEAVE my\_loop;

END IF;

UPDATE playlist\_tracks

SET tracks\_order = tracks\_order - 1

WHERE playlist\_id = pl\_id

AND tracks\_order > or\_num;

END LOOP;

CLOSE cur;

SET SQL\_SAFE\_UPDATES = 0;

DELETE

FROM playlist\_tracks

WHERE track\_id = OLD.id;

SET SQL\_SAFE\_UPDATES = 1;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `track\_comment`

--

DROP TABLE IF EXISTS `track\_comment`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `track\_comment` (

`id` bigint NOT NULL,

`comment\_date` timestamp NULL DEFAULT NULL,

`text` text,

`track\_timestamp` int DEFAULT NULL,

`track\_id` bigint NOT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_track\_comment\_track` (`track\_id`),

KEY `FK\_track\_comment\_user` (`user\_id`),

CONSTRAINT `FK\_track\_comment\_track` FOREIGN KEY (`track\_id`) REFERENCES `track` (`id`),

CONSTRAINT `FK\_track\_comment\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `track\_like`

--

DROP TABLE IF EXISTS `track\_like`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `track\_like` (

`id` bigint NOT NULL,

`like\_date` timestamp NULL DEFAULT NULL,

`track\_id` bigint NOT NULL,

`user\_id` bigint NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_track\_like\_track` (`track\_id`),

KEY `FK\_track\_like\_user` (`user\_id`),

CONSTRAINT `FK\_track\_like\_track` FOREIGN KEY (`track\_id`) REFERENCES `track` (`id`),

CONSTRAINT `FK\_track\_like\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `track\_tag`

--

DROP TABLE IF EXISTS `track\_tag`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `track\_tag` (

`track\_id` bigint NOT NULL,

`tag\_id` bigint NOT NULL,

PRIMARY KEY (`track\_id`,`tag\_id`),

KEY `FK\_track\_tag\_tag` (`tag\_id`),

CONSTRAINT `FK\_track\_tag\_tag` FOREIGN KEY (`tag\_id`) REFERENCES `tag` (`id`),

CONSTRAINT `FK\_track\_tag\_track` FOREIGN KEY (`track\_id`) REFERENCES `track` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `user`

--

DROP TABLE IF EXISTS `user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user` (

`id` bigint NOT NULL,

`image` varchar(255) DEFAULT NULL,

`password` varchar(255) NOT NULL,

`registration\_date` timestamp NULL DEFAULT NULL,

`username` varchar(255) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'STRICT\_TRANS\_TABLES,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

/\*!50003 CREATE\*/ /\*!50017 DEFINER=`root`@`localhost`\*/ /\*!50003 TRIGGER `set\_deleted\_user` AFTER DELETE ON `user` FOR EACH ROW BEGIN

UPDATE track\_comment tc

SET tc.user\_id = 100

WHERE tc.user\_id = OLD.id;

END \*/;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

--

-- Table structure for table `user\_roles`

--

DROP TABLE IF EXISTS `user\_roles`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_roles` (

`user\_id` bigint NOT NULL,

`roles\_id` bigint NOT NULL,

PRIMARY KEY (`user\_id`,`roles\_id`),

KEY `FK\_user\_roles\_role` (`roles\_id`),

CONSTRAINT `FK\_user\_roles\_role` FOREIGN KEY (`roles\_id`) REFERENCES `role` (`id`),

CONSTRAINT `FK\_user\_roles\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `user\_subscriptions`

--

DROP TABLE IF EXISTS `user\_subscriptions`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `user\_subscriptions` (

`user\_id` bigint NOT NULL,

`subscriptions\_id` bigint NOT NULL,

PRIMARY KEY (`user\_id`,`subscriptions\_id`),

KEY `FK\_user\_subscriptions\_subsciption` (`subscriptions\_id`),

CONSTRAINT `FK\_user\_subscriptions\_subsciption` FOREIGN KEY (`subscriptions\_id`) REFERENCES `user` (`id`),

CONSTRAINT `FK\_user\_subscriptions\_user` FOREIGN KEY (`user\_id`) REFERENCES `user` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping events for database 'funsymusicdatabase'

--

--

-- Dumping routines for database 'funsymusicdatabase'

--

/\*!50003 DROP PROCEDURE IF EXISTS `find\_favourite\_track\_tag\_by\_playlists` \*/;

/\*!50003 SET @saved\_cs\_client = @@character\_set\_client \*/ ;

/\*!50003 SET @saved\_cs\_results = @@character\_set\_results \*/ ;

/\*!50003 SET @saved\_col\_connection = @@collation\_connection \*/ ;

/\*!50003 SET character\_set\_client = utf8mb4 \*/ ;

/\*!50003 SET character\_set\_results = utf8mb4 \*/ ;

/\*!50003 SET collation\_connection = utf8mb4\_0900\_ai\_ci \*/ ;

/\*!50003 SET @saved\_sql\_mode = @@sql\_mode \*/ ;

/\*!50003 SET sql\_mode = 'STRICT\_TRANS\_TABLES,NO\_ENGINE\_SUBSTITUTION' \*/ ;

DELIMITER ;;

CREATE DEFINER=`root`@`localhost` PROCEDURE `find\_favourite\_track\_tag\_by\_playlists`(IN user\_id varchar(255), OUT favourite\_tag\_id varchar(255))

BEGIN

DECLARE tag\_id\_count int;

SELECT

tt.genre\_id,

COUNT(\*) AS frequency INTO favourite\_tag\_id, tag\_id\_count

FROM track\_tag tt

WHERE tt.track\_id IN (SELECT

track.id

FROM track

JOIN playlist\_tracks pt

ON pt.track\_id = track.id

JOIN playlist

ON playlist.id = pt.playlist\_id

WHERE playlist.owner\_id = user\_id)

GROUP BY tt.tag\_id

ORDER BY frequency DESC

LIMIT 1;

END ;;

DELIMITER ;

/\*!50003 SET sql\_mode = @saved\_sql\_mode \*/ ;

/\*!50003 SET character\_set\_client = @saved\_cs\_client \*/ ;

/\*!50003 SET character\_set\_results = @saved\_cs\_results \*/ ;

/\*!50003 SET collation\_connection = @saved\_col\_connection \*/ ;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;