ОТЧЕТ

по самостоятельному заданию

1. Введение

В данном отчете представлена разработка реляционной модели базы данных для онлайн-кинотеатра. Цель работы — спроектировать структуру БД, отражающую ключевые сущности предметной области, их атрибуты и взаимосвязи.

2. Описание предметной области

Онлайн-кинотеатр предоставляет пользователям доступ к фильмам различных жанров. Основные функциональные требования:

- Хранение информации о фильмах, жанрах и пользователях.

- Учет подписок пользователей.

- Фиксация истории просмотров.

3. Проектирование базы данных

3.1. Сущности и их атрибуты

В модели выделены следующие сущности:

1. Фильмы(`Films`):

- `film\_id` (PK) — уникальный идентификатор.

- `title` — название фильма.

- `release\_year` — год выпуска.

- `duration\_min` — длительность в минутах.

- `rating` — рейтинг (от 0 до 10).

- `genre\_id` (FK) — ссылка на жанр.

2. Жанры (`Genres`):

- `genre\_id` (PK) — идентификатор жанра.

- `name` — название (боевик, комедия и т.д.).

- `description` — описание.

3. Пользователи (`Users`):

- `user\_id` (PK) — идентификатор.

- `username` — логин.

- `email` — электронная почта.

- `subscription\_type` — тип подписки (Basic, Premium, VIP).

4. Просмотры (`Views`):

- `view\_id` (PK) — идентификатор просмотра.

- `user\_id` (FK) — пользователь.

- `film\_id` (FK) — фильм.

- `watch\_date` — дата просмотра.

5. Подписки (`Subscriptions`):

- `subscription\_id` (PK) — идентификатор.

- `user\_id` (FK) — пользователь.

- `start\_date`, `end\_date` — срок действия.

3.2. ER-диаграмма

Связи между сущностями:

-Фильм → Жанр (М:1) — у фильма один жанр.

-Пользователь → Просмотр (1:М) — пользователь может смотреть много фильмов.

-Пользователь → Подписка (1:М) — у пользователя может быть несколько подписок.

3.3. SQL-код создания таблиц

```sql

-- Таблица Жанры

CREATE TABLE Genres (

genre\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50) NOT NULL,

description TEXT

);

-- Таблица Фильмы

CREATE TABLE Films (

film\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100) NOT NULL,

release\_year INT CHECK (release\_year >= 1888),

duration\_min INT NOT NULL,

rating DECIMAL(3,1) CHECK (rating BETWEEN 0 AND 10),

genre\_id INT,

FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

-- Таблица Пользователи

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL UNIQUE,

email VARCHAR(100) NOT NULL UNIQUE,

subscription\_type VARCHAR(20) CHECK (subscription\_type IN ('Basic', 'Premium', 'VIP'))

);

-- Таблица Просмотры

CREATE TABLE Views (

view\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

film\_id INT,

watch\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (film\_id) REFERENCES Films(film\_id)

);

-- Таблица Подписки

CREATE TABLE Subscriptions (

subscription\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

CHECK (end\_date > start\_date)

);

```

4. Примеры запросов

1.Фильмы с рейтингом выше 8:

```sql

SELECT title, rating FROM Films WHERE rating > 8 ORDER BY rating DESC;

```

2. \*\*Пользователи с активной подпиской:

```sql

SELECT u.username, s.end\_date

FROM Users u

JOIN Subscriptions s ON u.user\_id = s.user\_id

WHERE s.end\_date >= CURRENT\_DATE;

```

3. \*\*Самые популярные жанры:

```sql

SELECT g.name, COUNT(v.view\_id) AS views\_count

FROM Genres g

JOIN Films f ON g.genre\_id = f.genre\_id

JOIN Views v ON f.film\_id = v.film\_id

GROUP BY g.name

ORDER BY views\_count DESC;

```

5. Заключение

В ходе работы спроектирована реляционная модель БД для онлайн-кинотеатра, включающая:

- 5 нормализованных таблиц с первичными и внешними ключами.

- Ограничения целостности (CHECK, FOREIGN KEY).

- Типовые запросы для анализа данных.

Модель готова к реализации в СУБД (MySQL, PostgreSQL) и масштабированию.

Приложения:

1. ER-диаграмма (в формате PNG/PDF).

2. Полный SQL-скрипт создания БД.