МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра Систем Штучного Інтелекту

РОЗРАХУНКОВА РОБОТА

з дисципліни «Організація баз даних та знань»

на тему:

«Веб-сервіс пошуку кіно»

Виконав:

студент групи КН-210

Бурак Марко Теодорович

Балів	**Дата**

Викладач:

Мельникова Наталя Іванівна

Львів – 2020

Зміст

1. Тема проекту.

Мета нашого проекту створити веб сервіс, який допомагав би користувачам з легкістю знаходити найоптимальніші варіанти перегляду фільмів у різних кінотеатрах. На нашому сайті можна знайти всі фільми, які перебувають у прокаті, а також анонси майбутніх сеансів.

Коротко про основні завдання нашого сервісу. Веб-сайт “Економ Кіно” дозволяє отримати найкращий сеанс, під цим я маю на увазі зручність сервісу, прості методи сортування, різні цікавинки, які зібрані всі у одному місці, більше того, тут також присутній пошук для пришвидшенн пошуку саме того фільму в якому людина зацікавлена. Також тут пристутні комфортні фільтри, для забезпечення отримання фільмів лише тієї технології сеансу, яка подобається користувачу, наступним ж фільтром є фільтруванн проведення сеансів, а саме, фільтр по кінотеатрах.

Отож, як це все відбувається.Користувач заходить на наш сайт,вибирає певну дату, коли він зацікавлений подивитись певний фільм , після цього йому висвітлюється список фільмів, для яких на сьогодні є сеанси в кінотеатрах Львова. Окрім того , юзер може застосувати пошук, для отримання лише того фільму, в якому він зацікавлений.Опісля йде компонента анонсів, тобто майбутніх фільмів, вони згодом вийдуть в прокат, також можна отримати доступ до цієї компоненти через посилння згори веб-сайту. Також, наш сервіс надає корисну інформацію про кінотеатри, де відбуваються сеанси, за допомогою цього, користувач з комфортом обере саме те місце, яке йому до вподоби.Тут користувач знайде інформацію про рейтинг кінотеатру, фото ззовні та всередині, та відгуки з місцезнаходженням.Друга ж сторінка – це детальна інформація про фільм - його постер, назва, трейлер, опис, рейтинг і тд. Також на цій сторінці користувач зможе вже безпосередньо обрати для себе найкращий сеанс. Тут можна використовувати сортування або фільтрування сеансів за певними критеріями. Після того, як користувач обрав для себе

найкращий варіант, швидко та зручно отримує найоптимальніший варіант приємного проведення часу.

Наш веб-сервіс можна знайти тут посилання.

2. Вступ.

Для даної розрахункової роботи було обрано тему саме веб-сервісів базуючись на поданих перевагах:

- **Зручність**.

Всі люди на сьогодні користуються веб-сервісами, це дуже зручно та кофортно, не потрібно качати додаткових програм, та витрачати час, людина може просто набрати посиланн та отримати доступ до нашого сервісу.

- **Перспективність**.

Кожна компанія, навіть мале підприємство хоче мати свій власний веб-сервіс, тому що люди потребують цього, тому, я вважаю, що цей проект нас навчив навичках Front-end та Back-end, що може нам допомогти навіть при майбутніх проектах в компаніх.

- **Активна, завжди присутня аудиторія**.

Під час вибору кіно, люди часто звертаються до сервісів кінотеатрів чи інших веб-сервісів, ніхто зараз не ходить безпосередньо в кінотеатри чи якимось іншими способами не знаходить інформацію про сеанси, все відбуважться саме через інтернет. На сьогодні є досить багато людей, які люблють кіноіндустрію, тому це ще один плюс, до нашої аудиторії.

- **Платоспроможна аудиторія**.

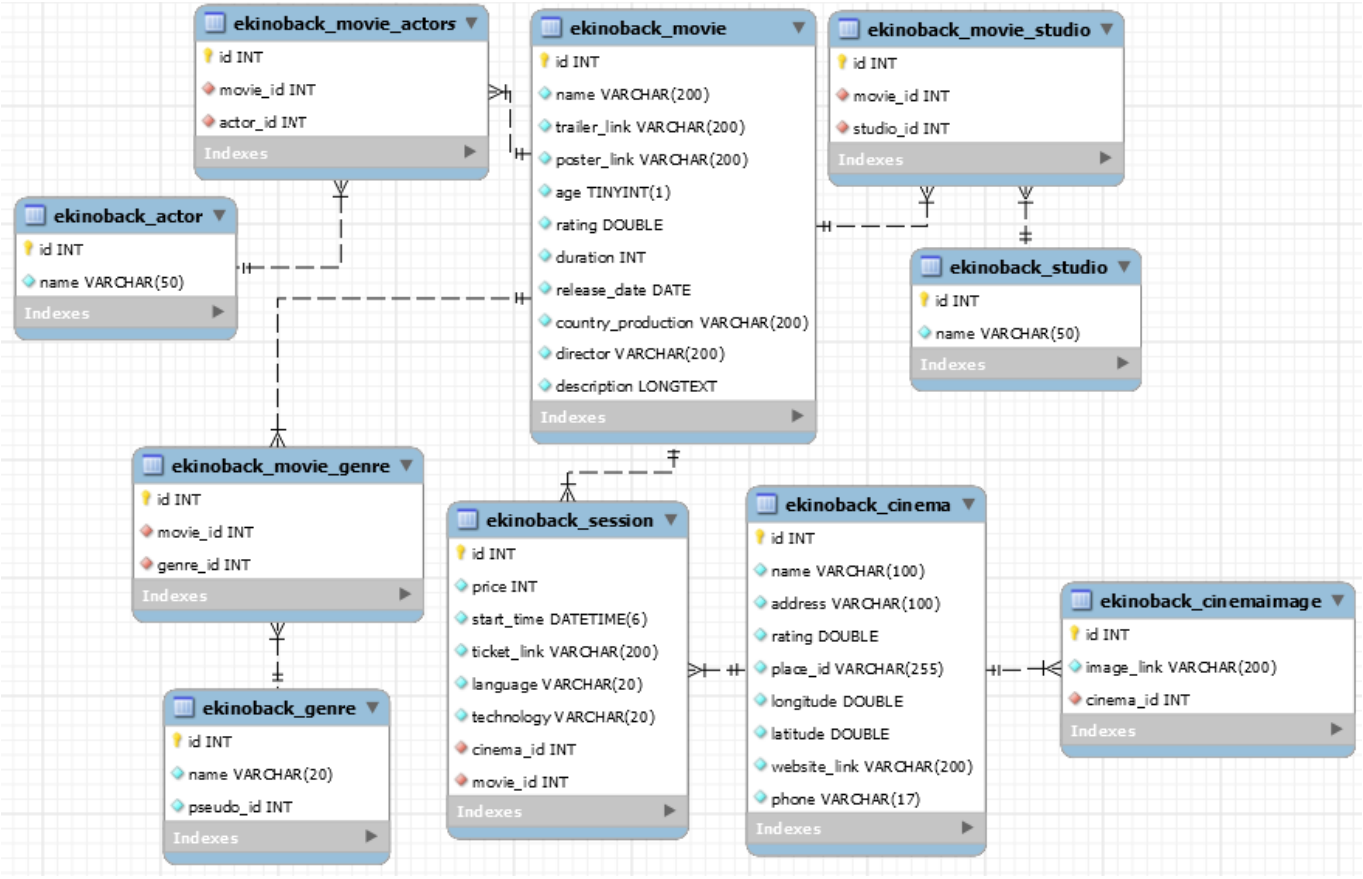
Чим більше людей буде відвідувати наш веб-сервіс, тим прибутковіші будуть кінотеатри, тому ми не заперечуємо їхню співпрацю з нами, квитки у кіно є відносно не дуже дорогим задоволенням, тому ріст аудиторії буде завжди.

- **Слабка конкуренція**.

  На сьогодні існує лише декілька схожих на наш сервіс, проте, ці сервіси не мають інформацію про всі кінотеатри, вони включають лише маловідомі кінотеатри, тому не є сильно популярні. З іншого ж боку наш кінотеатр має цю можливість та надає інформацію про всі кінотеатри.

Порадившись з колегами по команді, ми вирішили зробити цей сервіс, який допоможе з легкістю знайти найкращий сеанс з комфортом та просто.

3. Логічна схема БД проекту.



4. Опис структури БД.

Діаграма до бази даних подана вище. Для розгорнення її можливостей буду описувати таблиці та їх складові. Почну з однієї з головних таблиць у бд – таблиці *ekinoback_movie*. Структура цієї таблиці:

- id INT – Внутрішній ключ, унікальний ідентифікатор. Це поле використовується в кожній таблиці бд, тому в решті таблиць його не розглядатимемо.
- name varchar(200) – Назва фільму. Not Null, обмеження – 200 символів. Значенням поля не може бути NULL,воно повторюється у всіх полях таблиці, тому в решті атрибутів таблиць цього не розглядатимемо.
- trailer_link і poster_link – Посилання на трейлер фільму та на постер фільму. Обмеження – по 200 символів.
- age – Вікове обмеження (True або False) Нуль означає 0+, а один означає 18+.
- rating – Рейтинг фільму за версією IMDB від 0 до 10.
- duration – Тривалість фільму в хвилинах.
- release_date – Дата релізу фільму.
- country_production – Країна-автор фільму.
- director – Режисер або режисери фільму.
- description – Опис фільму.

Також фільм має жанри, актори, які знімались у фільмі, а також студії, які знімали цей фільм. Для цього створено таблиці:

ekinoback_genre:

- name – Назва жанру. Обмеження – 20 символів.
- pseudo_id – Специфічний ідентифікатор жанру, потрібен для правильного функціонування парсерів, які вносять дані у базу даних.

ekinoback_actor:

- name – Ім’я і прізвище актора. Обмеження – 50 символів.

ekinoback_studio:

- name – Назва студії. Обмеження – 50 символів.

Всі ці сутності відносяться до сутності Фільму. Зв’язок між цими таблицями і таблицею movie – Багато до багатьох (many-to-many),тому що в одному фільмі може зніматись багато акторів, і один актор може зніматись в багатьох фільмах. Те саме для студій і жанрів. Тому всі ці зв’язки організовані у вигляді many-to-many за допомогою проміжних таблиць:

- *ekinoback_movie_genre;*
- *ekinoback_movie_studio;*
- *ekinoback_movie_actor;*

які мають однакову структуру – Ідентифікатор фільму та ідентифікатор жанру/студії/актора.

Наступною таблицею, яку ми розглянемо, буде *ekinoback_cinema* – Кінотеатр:

- name – Назва кінотеатру. Обмеження – 100 символів.
- address – Адреса кінотеатру. Обмеження – 100 символів.
- rating – Рейтинг кінотеатру, оцінка цього місця на Google Maps.
- place_id – Ідентифікатор кінотеатру, який використовується в Google Maps. Обмеження – 255 символів.
- longitude – Довгота адреси кінотеатру.
- latitude – Широта адреси кінотеатру.

- website_link – Посилання на веб-сайт кінотеатру.Обмеження – 200 символів.
- phone – Номер телефону кінотуатру. Обмеження – 17 символів.

Ця таблиця майже повністю описує сутність Кінотеатр, проте наш сервіс повинен мати його фотографії кінотеатрів, які також мають зберігатись у базі даних. Для цього створюємо ще одну таблицю, *ekinoback_cinemaimage*, яка відповідатиме за зберігання посилань на фотографії певного кінотеатру:

- image_link – Посилання на фотографію певного кінотеатру. Обмеження – 200 символів.
- cinema_id – Зовнішній ключ, який посилається на Кінотеатр, якому належить це зображення.

Це зв’язок Один до багатьох (one-to-many), який реалізується за допомогою зовнішнього ключа.

Наступна, яка водночас є і останньою, таблиця – це *ekinoback_session*, яка відповідає за сутність Сеанс:

- price – Ціна квитка.
- start_time – Дата і час початку сеансу.
- ticket_link – Посилання на сторінку, де можна придбати квиток саме на цей сеанс. Обмеження – 200 символів.
- language – Мова показу фільму. Обмеження – 20 символів.
- technology – Технологія показу фільму, для прикладу, 2D, 3D, 4DX, Cinetech+ і тд.
- cinema_id – Зовнішній ключ, який посилається на кінотеатр, в якому проводитиметься цей сеанс.

- movie_id – Зовнішній ключ, який посилається на фільм, який показуватиметься під час цього сеансу.

Крім цього, розглянемо індекси, які є у цій базі даних, крім внутрішніх ключів.

- movie.name – цей індекс потрібен тому, що часто здійснюється пошук фільму за назвою.
- movie.rating – цей індекс потрібен тому, що часто проводиться сортування та фільтрація за рейтингом.
- cinema.name та cinema.rating – за тих же самих причин, що й попередні пункти відповідно.
- session.price – цей індекс потрібен, бо часто здійснюється фільтрування або сортування за ціною.

5. Фізична модель БД.

Текст файлу створення БД з оголошенням обмежень, індексів та ключів:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DA
TE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUT
ION';

CREATE SCHEMA IF NOT EXISTS `ekinobase` DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `ekinobase` ;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_actor` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_cinema` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
```

```
  `address` VARCHAR(100) NOT NULL,
  `rating` DOUBLE NOT NULL,
  `place_id` VARCHAR(255) NOT NULL,
  `longitude` DOUBLE NOT NULL,
  `latitude` DOUBLE NOT NULL,
  `website_link` VARCHAR(200) NOT NULL,
  `phone` VARCHAR(17) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name` (`name` ASC),
  INDEX `cinema_rating` (`rating` ASC))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_cinemaimage`
(
  `id` INT NOT NULL AUTO_INCREMENT,
  `image_link` VARCHAR(200) NOT NULL,
  `cinema_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX
`ekinoback_cinemaimage_cinema_id_30f3e107_fk_ekinoback_cinema_
id` (`cinema_id` ASC),
  CONSTRAINT
`ekinoback_cinemaimage_cinema_id_30f3e107_fk_ekinoback_cinema_
id`
```

```
    FOREIGN KEY (`cinema_id`)
    REFERENCES `ekinobase`.`ekinoback_cinema` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_genre` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(20) NOT NULL,
  `pseudo_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name` (`name` ASC) VISIBLE,
  INDEX `ekinoback_genre_pseudo_id_820c15b3` (`pseudo_id` ASC)
)
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_movie` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(200) NOT NULL,
  `trailer_link` VARCHAR(200) NOT NULL,
  `poster_link` VARCHAR(200) NOT NULL,
  `age` TINYINT(1) NOT NULL,
  `rating` DOUBLE NOT NULL,
```

```
  `duration` INT NOT NULL,
  `release_date` DATE NOT NULL,
  `country_production` VARCHAR(200) NOT NULL,
  `director` VARCHAR(200) NOT NULL,
  `description` LONGTEXT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `name` (`name` ASC) ,
  INDEX `movie_rating` (`rating` ASC) )
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS
`ekinobase`.`ekinoback_movie_actors` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `movie_id` INT NOT NULL,
  `actor_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX
`ekinoback_movie_actors_movie_id_actor_id_2f821621_uniq`
(`movie_id` ASC, `actor_id` ASC) ,
  INDEX
`ekinoback_movie_actors_actor_id_51113f26_fk_ekinoback_actor_i
d` (`actor_id` ASC) ,
```

```
  CONSTRAINT
`ekinoback_movie_actors_actor_id_51113f26_fk_ekinoback_actor_i
d`
    FOREIGN KEY (`actor_id`)
    REFERENCES `ekinobase`.`ekinoback_actor` (`id`),
  CONSTRAINT
`ekinoback_movie_actors_movie_id_ebe45bce_fk_ekinoback_movie_i
d`
    FOREIGN KEY (`movie_id`)
    REFERENCES `ekinobase`.`ekinoback_movie` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_movie_genre`
(
  `id` INT NOT NULL AUTO_INCREMENT,
  `movie_id` INT NOT NULL,
  `genre_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX
`ekinoback_movie_genre_movie_id_genre_id_1a7e02ed_uniq`
(`movie_id` ASC, `genre_id` ASC) ,
  INDEX
`ekinoback_movie_genre_genre_id_377ee090_fk_ekinoback_genre_id
` (`genre_id` ASC),
```

```
  CONSTRAINT
`ekinoback_movie_genre_genre_id_377ee090_fk_ekinoback_genre_id
`
    FOREIGN KEY (`genre_id`)
    REFERENCES `ekinobase`.`ekinoback_genre` (`id`),
  CONSTRAINT
`ekinoback_movie_genre_movie_id_0879da91_fk_ekinoback_movie_id
`
    FOREIGN KEY (`movie_id`)
    REFERENCES `ekinobase`.`ekinoback_movie` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_studio` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS
`ekinobase`.`ekinoback_movie_studio` (
  `id` INT NOT NULL AUTO_INCREMENT,
```

```
  `movie_id` INT NOT NULL,
  `studio_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX
`ekinoback_movie_studio_movie_id_studio_id_0ea4d2e7_uniq`
(`movie_id` ASC, `studio_id` ASC),
  INDEX
`ekinoback_movie_studio_studio_id_0c5af49e_fk_ekinoback_studio
_id` (`studio_id` ASC),
  CONSTRAINT
`ekinoback_movie_studio_movie_id_cbf548d0_fk_ekinoback_movie_i
d`
    FOREIGN KEY (`movie_id`)
    REFERENCES `ekinobase`.`ekinoback_movie` (`id`),
  CONSTRAINT
`ekinoback_movie_studio_studio_id_0c5af49e_fk_ekinoback_studio
_id`
    FOREIGN KEY (`studio_id`)
    REFERENCES `ekinobase`.`ekinoback_studio` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback_session` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `price` INT NOT NULL,
```

```
  `start_time` DATETIME(6) NOT NULL,
  `ticket_link` VARCHAR(200) NOT NULL,
  `language` VARCHAR(20) NOT NULL,
  `technology` VARCHAR(20) NOT NULL,
  `cinema_id` INT NOT NULL,
  `movie_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX
`ekinoback_session_cinema_id_d7882bf8_fk_ekinoback_cinema_id`
(`cinema_id` ASC),
  INDEX
`ekinoback_session_movie_id_17d4bf2c_fk_ekinoback_movie_id`
(`movie_id` ASC),
  INDEX `ekinoback_session_price_10262086` (`price` ASC),
  CONSTRAINT
`ekinoback_session_cinema_id_d7882bf8_fk_ekinoback_cinema_id`
    FOREIGN KEY (`cinema_id`)
    REFERENCES `ekinobase`.`ekinoback_cinema` (`id`),
  CONSTRAINT
`ekinoback_session_movie_id_17d4bf2c_fk_ekinoback_movie_id`
    FOREIGN KEY (`movie_id`)
    REFERENCES `ekinobase`.`ekinoback_movie` (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 1
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

6. Ділова модель.

Тут відображена ділова модель бази даних. Тут показано які саме функції відповідають певній сутності у базі.

Наприклад функція класифікація за жанром, ця функція може бути використана у сутності фільмів та жанрів, адже і там і там таке фільтрування можливе. Те ж повторюється і з акторами та студіями, які можуть класифікуватись як за складом так і за студією.

Зрештую існують такі функції як розподіл за місцем та часом.

Як видно з моделі то у функції за місцем будуть участь такі сутності як: кінотеатр, за ким саме буде визначатись певне фільтрування, а також присутні сутності фільму та суансу, для показу інформації користувачу.

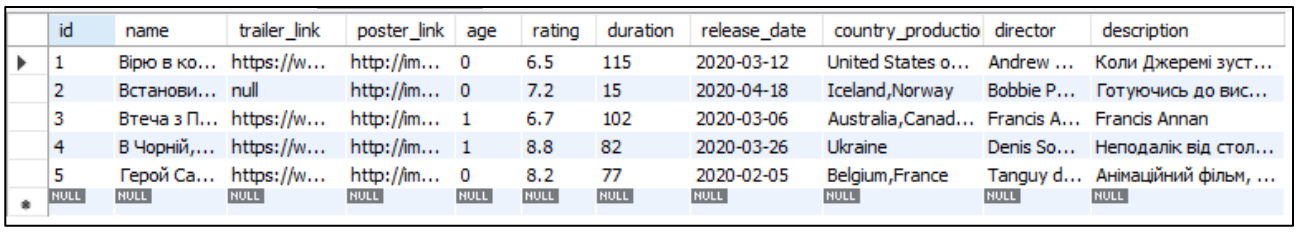
Така ж ситуація відбуваєть з розподілом за часом, проте маніпуляції проводяться над таблицею сеансів.

Таблиця / Функція	Кінотеатр	Фільм	Сеанс	Актор	Жанр	Студія	Фото кінотеатру
Класифікація за жанром		*			*		
Класифікація за складом		*		*			
Розподіл за місцем	*	*	*				
Розподіл за часом		*	*				
Класифікація за студією		*				*	
Пошук анонсів		*	*				
Розподіл за прокатом	*	*	*				
Перегляд інформації про кінотеатр	*						*

7. Запити до БД.

1. Отримаємо всі фільми:

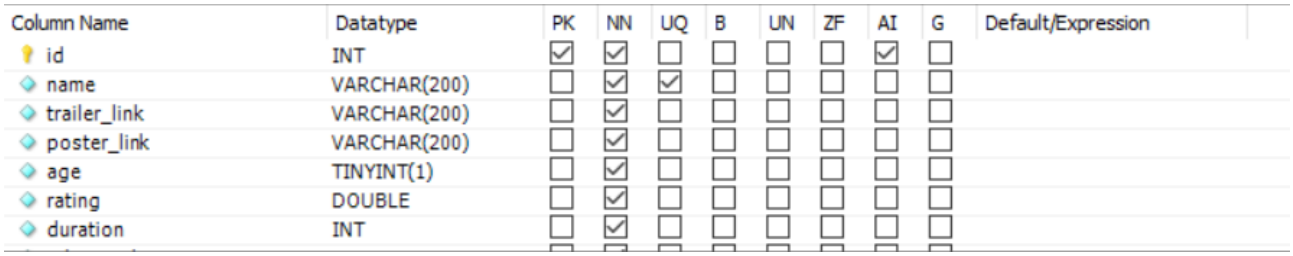
```
SELECT * FROM ekinoback_movie;
```



id	name	trailer_link	poster_link	age	rating	duration	release_date	country_productio	director	description
1	Вірю в ко...	https://w...	http://im...	0	6.5	115	2020-03-12	United States o...	Andrew ...	Коли Джеремі зуст...
2	Встанови...	null	http://im...	0	7.2	15	2020-04-18	Iceland,Norway	Bobbie P...	Готуючись до вис...
3	Втеча з П...	https://w...	http://im...	1	6.7	102	2020-03-06	Australia,Canad...	Francis A...	Francis Annan
4	В Чорній,...	https://w...	http://im...	1	8.8	82	2020-03-26	Ukraine	Denis So...	Неподалік від стол...
5	Герой Са...	https://w...	http://im...	0	8.2	77	2020-02-05	Belgium,France	Tanguy d...	Анімаційний фільм, ...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Отримаємо всі фільми які мають обмеження у віці(18+):

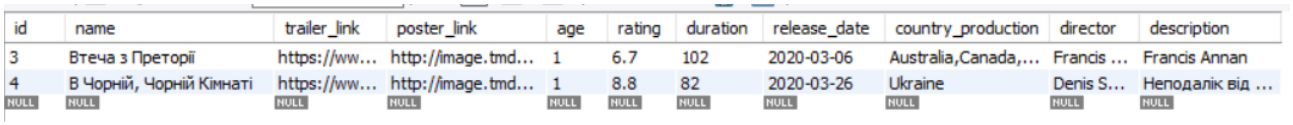
У нашій баці за це відповідає поле age, тут можна бачити, що його значення є Tinyint, де 1 це true, 0 це false.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT	☑	☑	☐	☐	☐	☐	☑	☐	
name	VARCHAR(200)	☐	☑	☑	☐	☐	☐	☐	☐	
trailer_link	VARCHAR(200)	☐	☑	☐	☐	☐	☐	☐	☐	
poster_link	VARCHAR(200)	☐	☑	☐	☐	☐	☐	☐	☐	
age	TINYINT(1)	☐	☑	☐	☐	☐	☐	☐	☐	
rating	DOUBLE	☐	☑	☐	☐	☐	☐	☐	☐	
duration	INT	☐	☑	☐	☐	☐	☐	☐	☐	

Select * from ekinoback_movie

where age = 1;



id	name	trailer_link	poster_link	age	rating	duration	release_date	country_production	director	description
3	Втеча з Преторії	https://ww...	http://image.tmd...	1	6.7	102	2020-03-06	Australia,Canada,...	Francis ...	Francis Annan
4	В Чорній, Чорній Кімнаті	https://ww...	http://image.tmd...	1	8.8	82	2020-03-26	Ukraine	Denis S...	Неподалік від ...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Фільми з обмеженням

3. Запит, який дає змогу отримати інформацію про фільми,а також жанри, тому звертаюсь до таблиці, яка має два зовнішніх ключі на таблицю фільмів, а також жанрів:

```
Select mg.id,
g.name,
m.name,
m.age,m.duration,
m.poster_link from
ekinoback_movie_genre as mg
join ekinoback_movie as m
on mg.movie_id = m.id
join ekinoback_genre as g
on mg.genre_id = g.id
```

id	name	name	age	duration	poster_link
1	Бойовик	Вірю в кохання	0	115	http://image.tmd...
2	Вестерн	Вірю в кохання	0	115	http://image.tmd...
3	Військовий	Вірю в кохання	0	115	http://image.tmd...
4	Детектив	Вірю в кохання	0	115	http://image.tmd...
5	Документальний	Вірю в кохання	0	115	http://image.tmd...
6	Детектив	Встановити прапор	0	15	http://image.tmd...
7	Документальний	Встановити прапор	0	15	http://image.tmd...
8	Драма	Встановити прапор	0	15	http://image.tmd...
9	Жахи	Встановити прапор	0	15	http://image.tmd...
10	Історичний	Встановити прапор	0	15	http://image.tmd...
11	Жахи	Втеча з Преторії	1	102	http://image.tmd...

Тут видно, що кожен фільм має багато жанрів, більше того, тут також показано, що декілька жанрів можуть мати кілька фільмів(детектив), тому ми використовували тут тип зв’язку many-to-many

4. Визначу найоптимальніші сеанси для фільмі “Герой СамСам” та “В Чорній, Чорній Кімнаті”, а саме зроблю вибірку сеансів цих фільмів, посортувавши їх спочатку за ціною, а потім за рейтингом, тоді користувач отримає дешевий та чудовий фільм:

```
select s.id,
s.price,
```

```
s.start_time,
s.language,
s.technology,
m.name,
m.rating,
c.name
from ekinoback_session as s
join ekinoback_movie as m
on s.movie_id = m.id
join ekinoback_cinema as c
on s.cinema_id = c.id
where m.name = "Герой СамСам"
or m.name = "В Чорній, Чорній Кімнаті"
order by s.price , m.rating desc
```

id	price	start_time	language	technology	name	rating	name
27	40	2020-05-17 19:00:00.000000	Англійська	3D	Герой СамСам	8.2	кінокомплекс Кінопалац
26	60	2020-05-17 19:00:00.000000	Англійська	4DX	Герой СамСам	8.2	Multiplex Spartak
22	80	2020-05-17 19:00:00.000000	Англійська	4DX	В Чорній, Чорній Кімнаті	8.8	Кінопалац ім. О.Довженка
25	80	2020-05-17 19:00:00.000000	Українська	2D	Герой СамСам	8.2	Multiplex
20	115	2020-05-17 19:00:00.000000	Українська	2D	В Чорній, Чорній Кімнаті	8.8	Multiplex Spartak
19	120	2020-03-17 18:00:00.000000	Українська	2D	В Чорній, Чорній Кімнаті	8.8	Multiplex
21	120	2020-01-17 18:00:00.000000	Англійська	3D	В Чорній, Чорній Кімнаті	8.8	кінокомплекс Кінопалац
23	120	2020-05-17 19:00:00.000000	Українська	3D	В Чорній, Чорній Кімнаті	8.8	Планета Кіно
29	140	2020-05-17 19:00:00.000000	Англійська	4DX	Герой СамСам	8.2	Планета Кіно
24	150	2020-05-17 19:00:00.000000	Англійська	4DX	В Чорній, Чорній	8.8	но 4DX
28	150	2020-08-17 19:00:00.000000	Українська	3D	Герой СамСам	8.2	Кінопалац ім. О.Довженка

В Чорній, Чорній Кімнаті

5. Визначу скільки людей знімалось у певних фільмах:

```
select  count(distinct(a.name)) as amount,
a.name,
m.name
from ekinoback_movie  as m
join ekinoback_movie_actors as ma
on m.id = ma.movie_id
```

```
join ekinoback_actor as a
on a.id = ma.actor_id
group by m.name
```

amount	name	name
8	Branka Katić	В Чорній, Чорній Кімнаті
8	Adrian Văncică	Вірю в кохання
8	Anna Koshmal	Встановити прапор
8	Aurore Broutin	Втеча з Преторії
8	Adrian Văncică	Герой СамСам

6. Визначу всі фільми у яких знімались певні актори, адже користувач може бути зацікавлений побачити фільм саме з цим актором “Dany Boon” та “Anna Koshmal”:

```
select
a.name as actor,
m.name as movie
from ekinoback_movie  as m
join ekinoback_movie_actors as ma
on m.id = ma.movie_id
join ekinoback_actor as a
on a.id = ma.actor_id
where a.name = "Dany Boon"
or a.name = "Anna Koshmal"
```

actor	movie
Anna Koshmal	Вірю в кохання
Anna Koshmal	Встановити прапор
Dany Boon	Герой СамСам

7. Визначу скільки кожен фільм має студій та сеансів, а також накладу умову, що цей фільм мав вийти пізніше ніж 01-03-2020.

```
select m.name,
count(distinct(s.id)) as studios,
count(distinct(ses.id)) as sessions,
m.release_date
from ekinoback_movie as m
join ekinoback_movie_studio as ms
on ms.movie_id = m.id
join ekinoback_studio as s
on ms.studio_id = s.id
join ekinoback_session as ses
on ses.movie_id = m.id
where m.release_date > "2020-03-01"
group by m.name;
```

name	studios	sessions	release_date
В Чорній, Чорній Кімнаті	8	6	2020-03-26
Вірю в кохання	8	6	2020-03-12
Встановити прапор	8	6	2020-04-18
Втеча з Преторії	8	6	2020-03-06

8. Висновки.

Виконуючи цей проект я навчився працювати з реляційною базою даних, а саме MySQL. Навчився проектувати діаграму, яку синхронізував з самою базою даних. Отже навчився створювати скрипти створення бази даних за допомогою таких команд як Forward Engineering або synchronize model в середовищі MySQL Workbench. Згодом навчився вносити самі дані у базу даних за допомогю insert або ж інтерфейсом. Для отримання впевненості, що дані записались правильно використовував прості скрипти для перевірки.

Щодо того, як ми взаємодіяли з даними у проекті, то ми використовували Django Rest Framework, Python, які були структурою бек-енду. За допомогою цього фреймворку та мови програмування ми могли змогу створювати таблиці, робити вибірки даних та запис. Для виконання розрахункової роботи я створив аналогічну модель бази даних в середовищі MySQL для того щоб показати її роботоздатність.

9. Список використаних джерел інформації.

1. Пасічник В.В., Резніченко В.А. Організація баз даних та знань - К.: Видавнича група BHV, 2006. — 384 с.: іл. — ISBN 966-552-156-X.
2. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. – Cengage Learning, 2017. – 818 p.
3. Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation and Management: Global Edition. – 6th Edition. – Pearson Education, 2015. – 1440 p.
4. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design, and Implementation. 14th ed. – Pearson Education Ltd., 2016. – 638 p.
5. https://www.w3schools.com/sql/
6. https://www.tutorialspoint.com/sql/index.htm
7. http://www.sql-tutorial.ru/
8. https://www.codecademy.com/learn/learn-sql
9. https://www.mysqltutorial.org/
10. https://www.tutorialspoint.com/mysql/index.htm