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. 3](#_Toc40868254)

[2. Вступ. 5](#_Toc40868255)

[3. Логічна схема БД проекту. 7](#_Toc40868256)

[4. Опис структури БД. 8](#_Toc40868257)

[5. Фізична модель БД. 12](#_Toc40868258)

[6. Ділова модель. 21](#_Toc40868259)

[7. Запити до БД. 22](#_Toc40868260)

[8. Висновки. 27](#_Toc40868261)

[9. Список використаних джерел інформації. 28](#_Toc40868262)

# 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](https://econom-kino.herokuapp.com).

# 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**.

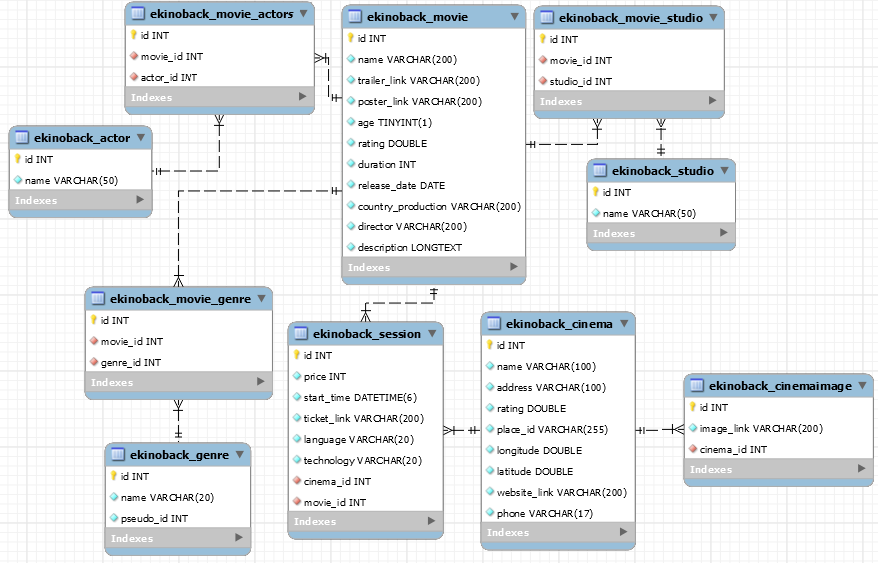
Чим більше людей буде відвідувати наш веб-сервіс, тим прибутковіші будуть кінотеатри, тому ми не заперечуємо їхню співпрацю з нами, квитки у кіно є відносно не дуже дорогим задоволенням, тому ріст аудиторії буде завжди.

* **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\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

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\_id` (`actor\_id` ASC) ,

CONSTRAINT `ekinoback\_movie\_actors\_actor\_id\_51113f26\_fk\_ekinoback\_actor\_id`

FOREIGN KEY (`actor\_id`)

REFERENCES `ekinobase`.`ekinoback\_actor` (`id`),

CONSTRAINT `ekinoback\_movie\_actors\_movie\_id\_ebe45bce\_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\_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\_id`

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.

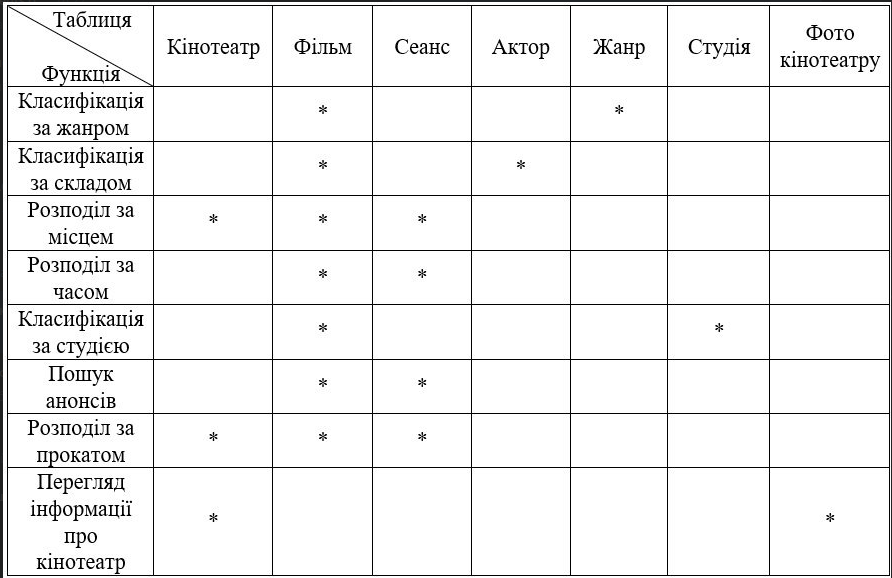
Тут відображена ділова модель бази даних. Тут показано які саме функції відповідають певній сутності у базі.

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.

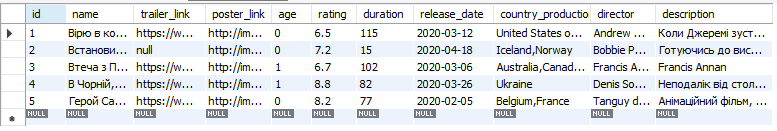
Як видно з моделі то у функції за місцем будуть участь такі сутності як: кінотеатр, за ким саме буде визначатись певне фільтрування, а також присутні сутності фільму та суансу, для показу інформації користувачу.

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



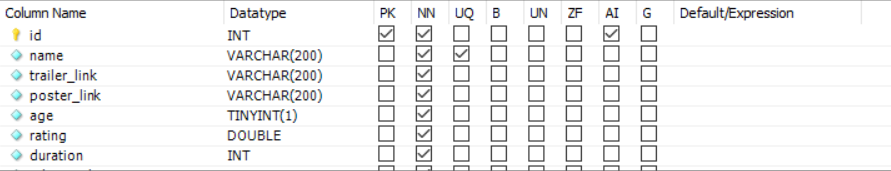
# 7. Запити до БД.

1. Отримаємо всі фільми:

SELECT \* FROM ekinoback\_movie; 

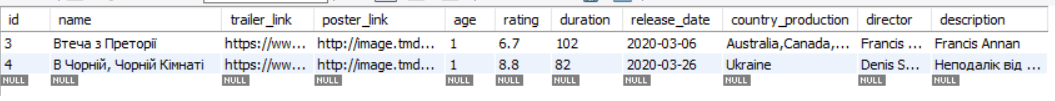
2. Отримаємо всі фільми які мають обмеження у віці(18+):

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



Select \* from ekinoback\_movie

where age = 1;



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



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

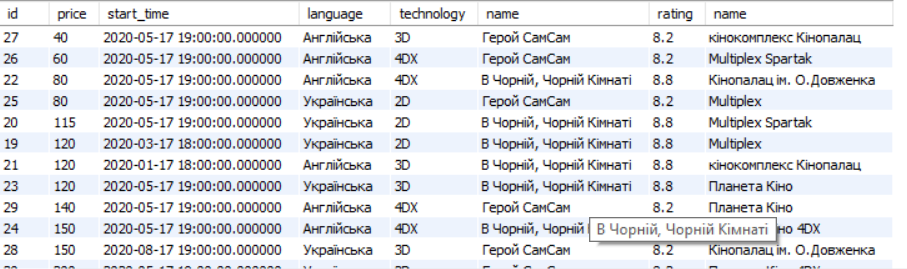
join ekinoback\_cinema as c

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



5. Визначу скільки людей знімалось у певних фільмах:

select count(distinct(a.name)) as amount,

a.name,

m.name

from ekinoback\_movie as m

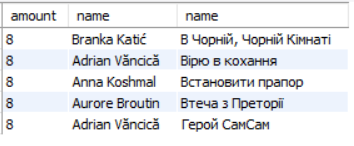
join ekinoback\_movie\_actors as ma

on m.id = ma.movie\_id

join ekinoback\_actor as a

on a.id = ma.actor\_id

group by m.name



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

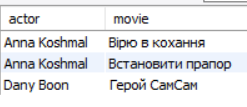
on m.id = ma.movie\_id

join ekinoback\_actor as a

on a.id = ma.actor\_id

where a.name = "Dany Boon"

or a.name = "Anna Koshmal"



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

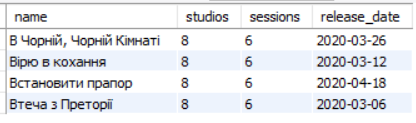
on ms.studio\_id = s.id

join ekinoback\_session as ses

on ses.movie\_id = m.id

where m.release\_date > "2020-03-01"

group by m.name;



# 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-Х.

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>