МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни «Організація баз даних та знань»
на тему
«Проектування бази даних інформаційної системи рекомендацій фільмів»

Виконав:
студент групи КН-208
Деревянний А. А.

Викладач:
Мельникова Н.І.

Балів	Дата

Львів – 2020 р.

Зміст

Вступ 3
Логічна схема БД проекту 4
Опис структури БД 5
Фізична модель БД 8
Ділова модель 11
Запити до БД 13
Висновки до розрахункової роботи 16
Список використаних літературних джерел 17

Вступ

«Люди, що не спроможні знайти час для відпочинку, рано чи пізно будуть змушені знайти час для хвороби[2]»

-Джон Вонамейкер – купець, громадський і політичний діяч[3]

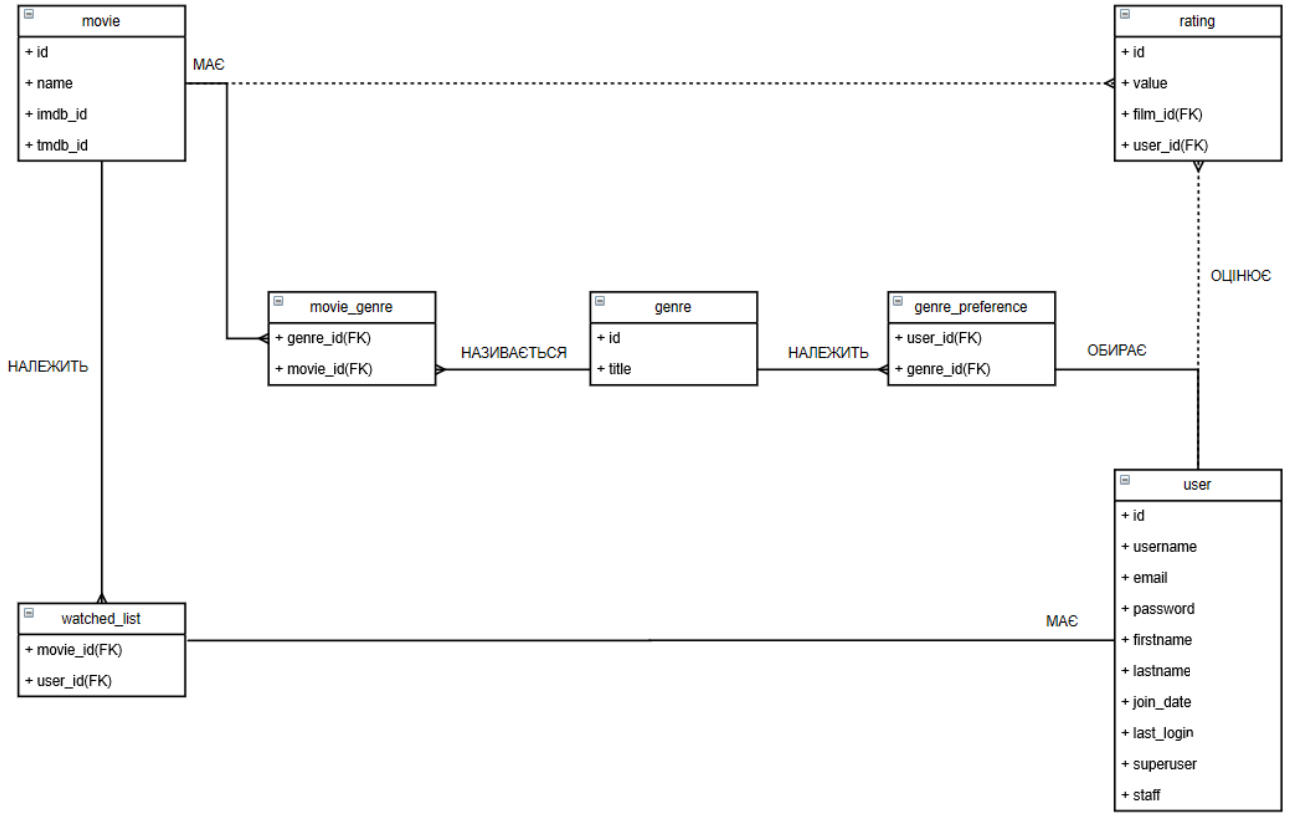
В сучасному світі час цілком можна розцінювати, як валюту, адже зараз, як ніколи, ми перебуваємо у вирі подій, де потрібно стільки всього зробити за день, що коли ти нарешті отримуєш ті пару годин відпочинку - ти не знаєш, що з ними робити, а «тільки робота і ніякого відпочинку перетворює Джека в нудного хлопця[4]». Тут на сцену і виходить наш сервіс – Contip. Що це таке? Contip – це веб-сервіс оптимізації вільного часу через пошук і рекомендацію контенту для споживання. На жаль, на даний момент реалізовано лише частину - рекомендація користувачам фільмів, проте в подальших планах є рекомендація серіалів, музики, книг, відеоігор і, можливо, навіть більше. Даний документ є пояснювальною запискою до одного з ключових частин нашого проекту, а саме – бази даних, але перед тим, як ми зануримося більш детально у модель бази даних і її структуру, давайте поговоримо про актуальність самого проекту.

Contip народився з сильної нелюбові до алгоритмів рекомендацій популярних сервісів – таких, як Netflix[5], YouTube Music і багато інших. Вони заганяють користувачів у інформаційну бульбашку, не виводядь з зони комфорту на стежину нових відкриттів, пропонуючи одне і те саме знову і знову. Хотіли переглянути якийсь поляризуючий серіал – все закінчилося тим, що ви вже всоте переглядаєте Друзі. Хотіли послухати щось ностальгічне, свою улюблену групу, від якої взяли паузу, щоб остаточно не затерти віртуальну платівку, а закінчилося тим, що ви переслуховуєте три альбоми Daughter і ще Ex:Re на додачу. Це не ваша вина - просто алгоритм не вміє читати думки і опирається на ваші останні взаємодії – хай це ситкоми дев’яностих чи спокійна меланхолічна музика, яка дозволяє вам доторкнутися до прекрасного.

То як Contip зможе цьому зарадити? Дуже просто – даючи вибір у руки користувачів. Поки інші сервіси просять заповнити ваші рекомендації лише раз – під час реєстрації, то в нас їх можна змінити у будь який момент. Крім того, користувачі – це і є наш головний рушій, адже саме завдяки їхнім оцінкам і формується список рекомендацій. Це надає неймовірну гнучкість нашому сервісу, дозволяє користувачам завжди отримувати тільки те, що їм потрібно. Щось нове – підходяший саундтрек для прогулянки чи екзистенціальна драма, що змусить думати про все на світі аж до третьої години. Щось старе – група, що змусила тебе вперше спробувати зіграти на гітарі чи фільм, який вичавив з тебе сльозу, але чомусь про це ти геть забув.

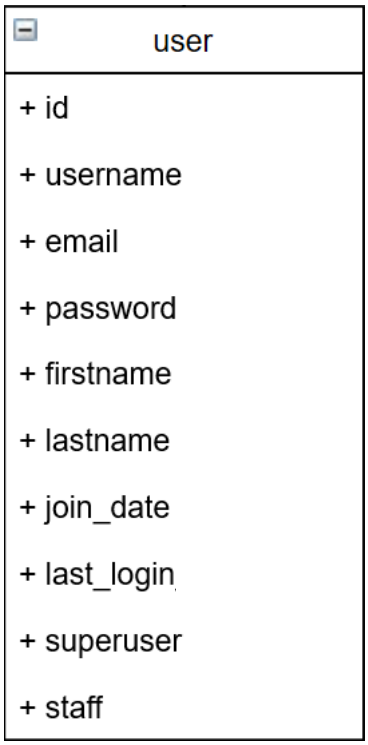
Тепер, коли проект є повністю представлений, перейдемо до головного – бази даних.

Логічна схема БД проекту



Опис структури БД

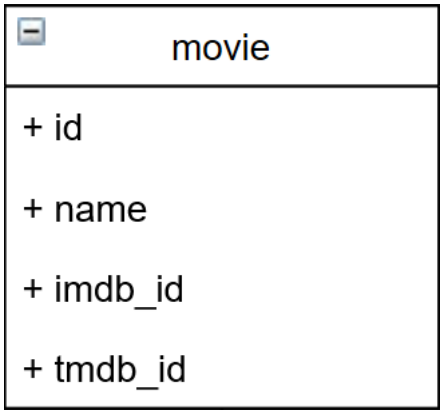
Для того, щоб розглянути структуру БД проекту, поглянемо на її компоненти – таблиці. Почнемо з головної – user:



В цю таблицю входять поля id(integer) – первинний ключ, унікальний індентифікатор кожного кортежа значень у таблиці, username(varchar) – користувацьке ім’я користувача нашого сервісу, email(varchar) – пошта користувача, password(varchar) - пароль користувача, firstname(varchar) – справжнє ім’я користувача, lastname(varchar) - справжнє прізвище користувача, join_date(datetime) - дата і час створення акаунту, last_login(datetime) – дата і час останнього входу в систему, superuser(tinyint) – показує чи має користувач супер-права на сайті, staff(tinyint) – показує чи є користувач адміністратором

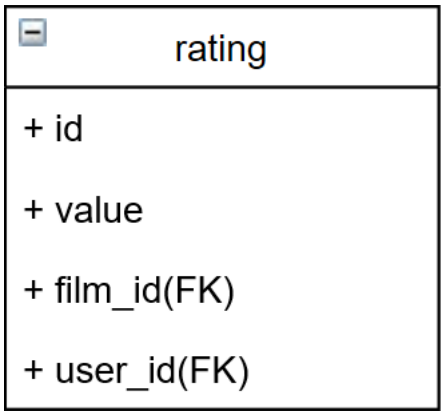
Для поля username також створений індекс, мета якого - пошук і порівняння правильності введених користувачем даних при його авторизації.

Перейдемо до таблиці movie:



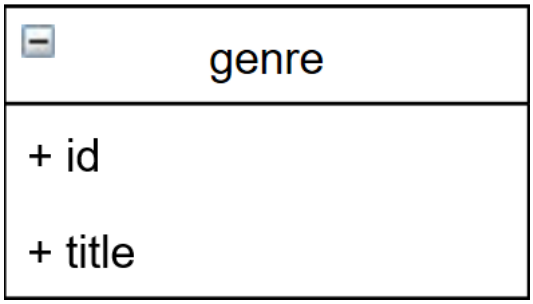
Її атрибутами є: id(integer) – первинний ключ, унікальний індентифікатор кожного кортежа значень у таблиці, name(varchar) – назва фільму, imdb_id(integer) – ідентифікатор сервісу IMDB, tmdb_id(integer) – ідентифікатор сервісу TMDB.

Тепер розглянемо сутність rating:



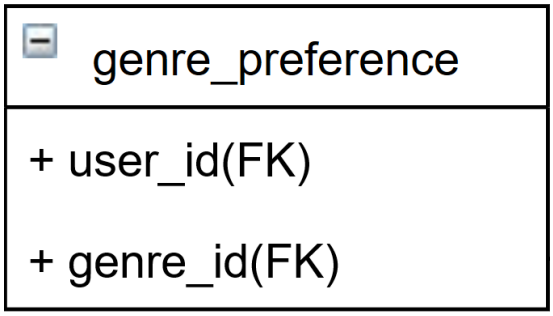
Тут ми бачимо id(integer) – первинний ключ, унікальний індентифікатор кожного кортежа значень у таблиці, value(integer) – значення рейтингу, який поставив певний користувач певному фільму, film_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці Movie, user_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці User.

Наступна таблиця в нашій базі даних – genre:



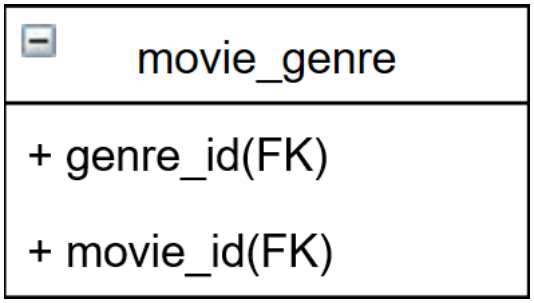
Їй належать наступні поля: id(integer) – первинний ключ, унікальний індентифікатор кожного кортежа значень у таблиці, name(varchar) - назва жанру.

Розглянемо genre_preference - зв’язну таблицю для зв’язка багато до багатьох таблиць user і genre, оскільки користувач може обрати багато жанрів і кожен жанр може бути вибраний у багатьох користувачів:



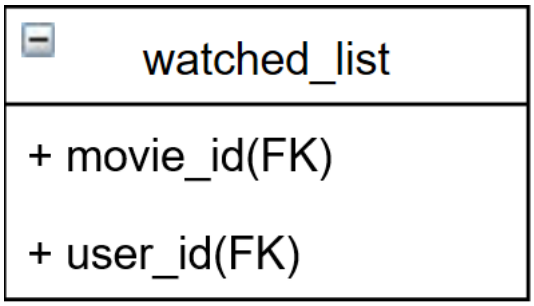
В ній знаходяться genre_id(integer) - зовнішній ключ який відповідає первинному ключу у таблиці genre, user_id(integer) - зовнішній ключ який відповідає первинному ключу у таблиці user.

Також зв'язною таблицею, цього разу для movie і genre, є genre_preference, оскільки фільм може мати багато жанрів і кожен жанр може належати багатьом фільмам:



Їй належать наступні поля: genre_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці genre, movie_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці Movie.

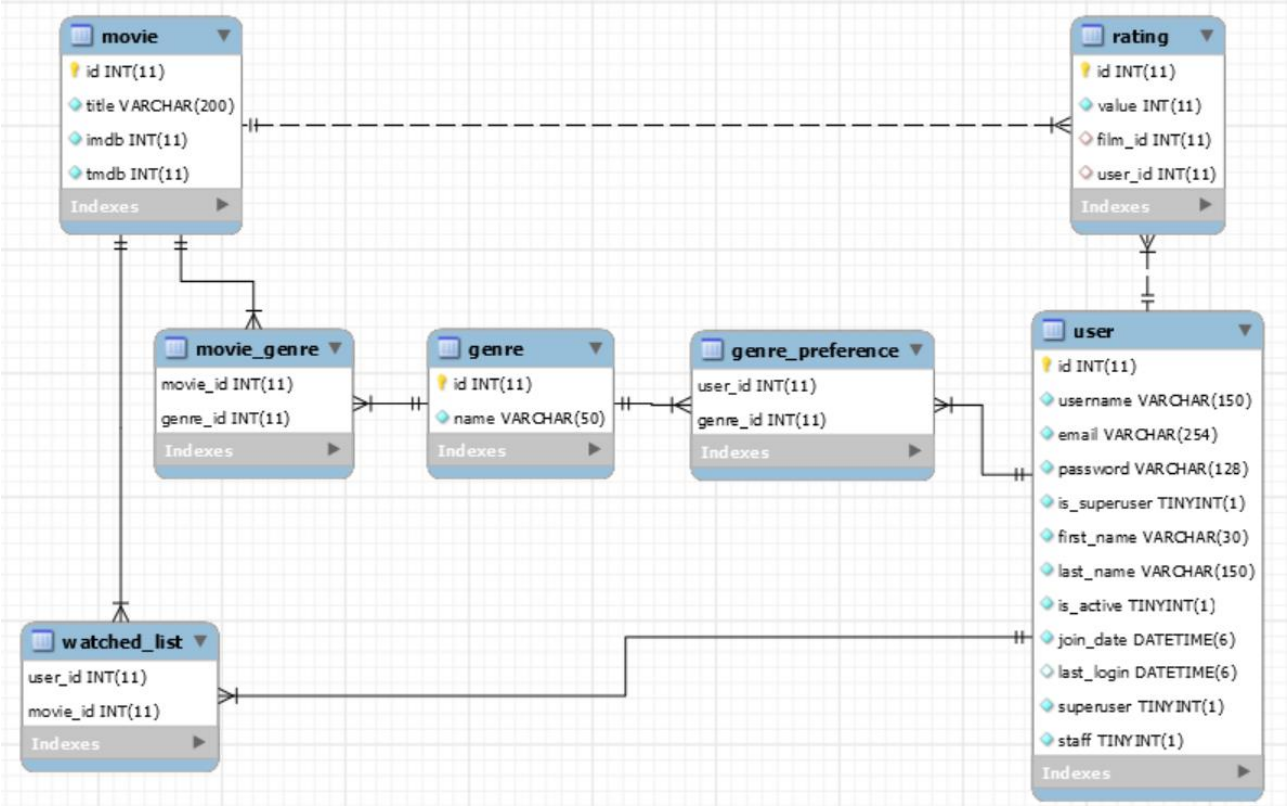
Останньою таблицею в нашій базі даних є зв'язна таблиця між movie і user – watched_list, оскільки користувач може мати багато фільмів у списку переглянутих і кожен фільм може належати багатьом спискам переглянутих:



Її атрибутами є: user_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці User, movie_id(integer) – зовнішній ключ який відповідає первинному ключу у таблиці Movie.

Як ми можемо спостерігати – всі наявні таблиці відповідають першій(є унікальний основний ключ, кожне значення є атомарним і немає полів, в яких позначені різні види одного і того ж), другій(немає даних, що залежать лише від частини ключа) і третій(між неключовими атрибутами немає транзитивної залежності, тобто дані в таблиці залежать винятково від основного ключа) нормальним формам, тому і вся база даних відповідає першій, другій і третій нормальним формам.

Фізична модель БД



MySql-скрипт:

```
drop database if EXISTS `project`;
CREATE DATABASE IF NOT EXISTS `project`;

USE `project`;

CREATE TABLE `genre` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `movie` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(200) NOT NULL,
  `imdb` int(11) NOT NULL,
  `tmdb` int(11) NOT NULL,
  PRIMARY KEY (`id`));
```

```
CREATE TABLE `movie_genre` (
  `movie_id` int(11) NOT NULL,
  `genre_id` int(11) NOT NULL,
  PRIMARY KEY (`movie_id`, `genre_id`),
  CONSTRAINT `movie_genre_genre_id_fk_genre_id` FOREIGN KEY (`genre_id`) REFERENCES
`genre` (`id`),
  CONSTRAINT `movie_genre_movie_id_fk_movie_id` FOREIGN KEY (`movie_id`) REFERENCES
`movie` (`id`)) ;

CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `password` varchar(128) NOT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `first_name` varchar(30) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `join_date` datetime(6) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `superuser` tinyint(1) NOT NULL,
  `staff` tinyint(1) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`));

CREATE TABLE `rating` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `value` int(11) NOT NULL,
  `film_id` int(11) DEFAULT NULL,
  `user_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `rating_film_id_user_uniq` (`film_id`,`user_id`),
  KEY `rating_user_id_fk_user_id` (`user_id`),
  CONSTRAINT `rating_film_id_fk_movie_id` FOREIGN KEY (`film_id`) REFERENCES `movie`
(`id`),
  CONSTRAINT `rating_user_id_fk_user_id` FOREIGN KEY (`user_id`) REFERENCES `user`
(`id`)) ;
```

```
CREATE TABLE `genre_preference` (
  `user_id` int(11) NOT NULL,
  `genre_id` int(11) NOT NULL,
  PRIMARY KEY (`user_id`, `genre_id`),
  CONSTRAINT `user_genr_genre_id_fk_genre` FOREIGN KEY (`genre_id`) REFERENCES `genre` (`id`),
  CONSTRAINT `user_genr_user_id_fk_user` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`));
```

```
CREATE TABLE `watched_list` (
  `user_id` int(11) NOT NULL,
  `movie_id` int(11) NOT NULL,
  PRIMARY KEY (`user_id`, `movie_id`),
  CONSTRAINT `user_watc_user_id_fk_user` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`),
  CONSTRAINT `user_watched_list_movie_id_fk_movie_id` FOREIGN KEY (`movie_id`) REFERENCES `movie` (`id`));
```

Ділова модель

Класи Функції	Користувач	Фільм	Жанр	Рейтинг
Формування рекомендацій	*	*	*	*
Додавання фільму в переглянуті	*	*		
Пошук фільму		*	*	
Керування акаунтом користувача	*	*	*	*
Оцінка фільму користувачем	*	*		*

До інформації про користувача належить
- username
- Ім’я користувача
- Прізвище користувача
- Електронна пошта користувача
- Статус активності користувача
- Дата і час останнього входу в систему
- Дата реєстрації акаунту
- Чи є користувач суперюзером
- Чи є користувач персоналом

До інформації про фільм належить:
- Назва фільму
- IMDB ідентифікатор
- TMDB ідентифікатор

До інформації про жанр фільму належить:
- Назва жанру

До інформації про рейтинг фільму належить:
- Значення рейтингу
- ID фільму
- ID користувача

Розглянемо функціонал запитів до БД:

- Функція «Формування Рекомендацій», як можна зрозуміти з назви, займається формуванням списку рекомендованих фільмів для користувачів. Для цього вона взаємодіє з усіма наявними класами – Користувач, Фільм, Жанр і Рейтинг;
- Функція «Додавання Фільму В Переглянуті», завдяки взаємодії з класами Фільм і Користувач, додає переглянуті фільми в архів;

- Функція «Пошук» дозволяє користувачу знайти фільми в базі даних, опираючись на класи Фільм і Жанр;
- Функція «Керування акаунтом користувача» надає користувачу можливість редагувати свої вподобання і персональні дані, завдяки тому, що взаємодіє з усіма наявними класами – Користувач, Фільм, Жанр і Рейтинг;
- Функція «Оцінка фільму користувачем» є надзвичайно важливою для роботи сервісу, адже саме завдяки оцінкам користувачів і формуються списки рекомендацій фільмів. Для коректної роботи відбувається взаємодія з класами Користувач, Фільм і Рейтинг.

Типові запити при роботі з наявною БД:

- Виведення інформації – демонстрація даних, які заповнюють певну таблицю. Виведення відбувається завдяки команді SELECT * FROM table. Для виведення конкретної інформації, потрібно модифікувати команду. Наприклад, заміною * на column, додавання умови через where, групуванням командою GROUP BY() упорядкуванням командою ORDER BY() і тд. Також, при виведенні інформації з різних таблиць, спочатку їх треба з'єднати між собою. Для цього існує команда join, а саме з'єднання відбувається за допомогою первинних ключів;
- Створення нового користувача – нові дані заповнюються командою INSERT INTO `app_user` (`is_superuser`, `is_staff`, `is_active`, `date_joined`, `username`, `password`, `first_name`, `last_name`, `email`)VALUES (value1, value2, …);. Обов'язково при заповненні заповнити поля, які не можуть бути NULL.
- Додавання уподобань користувача – в цьому запиті ми використовуємо команду INSERT INTO `app_userprofile_genre_preference` (`userprofile_id`, `genre_id`) VALUES (value1, value2). Рекомендується спочатку вивести інформацію про користувачів і жанри, щоб не допустити помилок при заповненні, адже userprofile_id і genre_id – це зовнішні ключі, які належать іншим сутностям(app_user і app_movie_genre відповідно)
- Додавання фільмів у архів – для цього ми використовуємо команду INSERT INTO `app_userprofile_watched_list` (`userprofile_id`, `movie_id`) VALUES (value1, value2). Варто бути обережним, адже, як було вказано вище, `app_userprofile_watched_list` - зв'язна таблиця і в неї входять зовнішні ключі userprofile_id і movie_id – це зовнішні ключі, які належать іншим сутностям(app_user і app_movie_genre відповідно);
- Оцінювання фільмів – щоб додати оцінку користувачів до бази даних, використовується команда INSERT INTO `app_rating` (`value`, `film_id`, `user_id`) VALUES (value1, value2, value3). За цієї операції оцінка вписується в таблицю `app_rating`, їй надається унікальний id, а для подальшого використання надаються зовнішні ключі відповідного користувача і фільму, який оцінювався;

Запити до БД

Спочатку створимо нового користувача. Для цього використаємо команду INSERT INTO table (column1, column2) VALUE (value1, value2), де table –назва таблиці, в яку додаємо дані, а value – дані, що ми вносимо:

```
3 •  INSERT INTO `app_user` (`is_superuser`, `is_staff`, `is_active`, `date_joined`, `username`, `password`, `fi
4    (0, 0, 1, '2020-04-12', "demo", "123", "demo", "demo", "storn@contip.gov.ua");
5
6 •  select * from app_user;
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	password	last_login	is_superuser	username	first_name	last_name	email	is_staff	is_active	date_joine
1	123	NULL	0	storn	Yaroslav	Mokryk	storn@contip.gov.ua	0	1	2020-04-12
2	123	NULL	0	kesha001	Danylo	Berdnyk	kesha@discord.co.uk	0	1	2020-05-08
3	123	NULL	0	Yyr3ll	Yuriy	Tarasenko	yyr3ll@arch.org	0	1	2020-03-25
4	123	NULL	0	Andy	Andriy	Derevianniy	andy@twitch.gg	0	1	2020-04-11
5	123	NULL	0	DanLide	Danylo	Lidzer	dan@anime.xyz	0	1	2019-11-05
6	123	NULL	0	demo	demo	demo	storn@contip.gov.ua	0	1	2020-04-12

Result Grid | Form Editor

Також нашому користувачеві варто задати вподобання. Щоб поглянути на всі наявні жанри, достатньо прописати команду SELECT * FROM table:

```
8 •  select * from app_genre;
```

sult Grid | Filter Rows:

id	name
1	Horror
2	Action
3	Drama
4	Adventure
5	Fantasy
6	Sci-fi
7	Horror
8	Action
9	Drama
10	Adventure
11	Fantasy

Тож нехай наш новий користувач полюбляє переглядати такі фільми, як Drama(id = 3 і 9), Adventure(4 і 10) і Sci-fi(6 і 12). Додамо ці вподобання командою INSERT INTO table (column1, column2) VALUE (value1, value2):

```
10 •  INSERT INTO `app_userprofile_genre_preference` (`userprofile_id`, `genre_id`)
11    VALUES (6, 3), (6, 9), (6, 4), (6, 10), (6, 6), (6, 12);
12
13 •  select * from app_userprofile_genre_preference
14    where userprofile_id like '6';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

userprofile_id	genre_id
6	3
6	4
6	6
6	9
6	10
6	12

Припустимо, що наш новий користувач вже переглянув деякі з рекомендованих фільмів. Це означає, що нам потрібно додати їх в архів – app_userprofile_watched_list. Але спочатку виведемо наявні фільми:

```
16 •  select * from app_movie;
```

Result Grid | Filter Rows: | Edit:

id	title	imdb	tmdb
2	Jumanji (1995)	30	42
3	Toy Story (1995)	58	91
4	Heat (1995)	12	50
5	Star Wars: Episode IV - A New Hope (1977)	25	11
6	Jumanji (1995)	30	42
7	Toy Story (1995)	58	91
8	Heat (1995)	12	50

Нехай наш користувач переглянув такі фільми, як Jumanji(id=2), Heat(4) і Star Wars: Episode IV – A New Hope(5). Додамо ці фільми в список його переглянутих:

```
18 •  INSERT INTO `app_userprofile_watched_list` (`userprofile_id`, `movie_id`)
19    VALUES (6, 2), (6, 4), (6, 5);
20
21 •  select * from app_userprofile_watched_list
22    where userprofile_id like '6';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

userprofile_id	movie_id
6	2
6	4
6	5
NULL	NULL

Також наш новий користувач виставив цим фільмам оцінки: Jumanji(id=2) – 75, Heat(4) - 78 і Star Wars: Episode IV – A New Hope(5) – 73:

```
INSERT INTO `app_rating` (`value`, `film_id`, `user_id`)
VALUES (75, 2, 6), (78, 4, 6), (73, 5, 6);

select * from app_rating
where user_id like '6';
```

Result Grid | Filter Rows: | Edit: | Export/Imp

id	value	film_id	user_id
8	75	2	6
9	78	4	6
10	73	5	6

Тепер давайте виведемо username нашого користувача, а також фільми, які він переглянув, разом з їх жанрами. Через те, що ми оперуємо за межами однієї таблиці, для виведення ми використовуємо команду join:

```
SELECT app_user.username, app_movie.title, GROUP_CONCAT(app_genre.name) FROM app_movie
INNER JOIN app_userprofile_watched_list ON app_userprofile_watched_list.movie_id=app_movie.id
INNER JOIN app_movie_genre ON app_movie.id=app_movie_genre.movie_id
INNER JOIN app_genre ON app_movie_genre.genre_id=app_genre.id
INNER JOIN app_user ON app_userprofile_watched_list.userprofile_id=app_user.id
WHERE app_userprofile_watched_list.userprofile_id=6 GROUP BY app_movie.title;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

username	title	GROUP_CONCAT(app_genre.name)
demo	Heat (1995)	Action
demo	Jumanji (1995)	Adventure,Fantasy
demo	Star Wars: Episode IV - A New Hope (1977)	Action,Adventure,Sci-fi

Висновки до розрахункової роботи

За час виконання даної розрахункової роботи я набув навичок створювання і роботи реляційною базою даних. Я створив модель бази даних, заповнив її даними, і перевірив її на дієздатність простими запитами, які, не зважаючи на свою простоту, цілко можуть використовуватися в повномасштабному проекті.

Список використаних літературних джерел

1. Мельникова Н.І. Вимоги щодо розрахункової роботи з дисципліни «Організація баз даних та знань» — Львів: НУ “ЛП”, 2020. — 4 с.
2. Bret Lowrey. Civilization VI Quotes – The Daily Signal, 2016 - https://lowrey.me/civilization-vi-quotes/
3. Wikipedia. John Wanamaker - https://en.wikipedia.org/wiki/John_Wanamaker
4. Stephen King. The Shining – DoubleDay Publishing, 1977 - 447p.
5. Josefina Blattmann. Netflix: Binging on the Algorithm – UxPlanet, 2018 - https://uxplanet.org/netflix-binging-on-the-algorithm-a3a74a6c1f59
6. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design, and Implementation. 14th ed. – Pearson Education Ltd., 2016. – 638 p.
7. Dewson R. Beginning SQL Server for Developers. 4th ed. – Apress, 2015. – 670 p.
8. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. – Cengage Learning, 2017. – 818 p.
9. Петкович Душан. Microsoft SQL Server 2012. Руководство для начинающих. СПб.: БХВ-Петербург, 2013. — 816 с.
10. Пасічник В.В., Резніченко В.А. Організація баз даних та знань - К.: Видавнича група BHV, 2006. — 384 с.: іл. — ISBN 966-552-156-X.
11. 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.
12. Hernandez M.J. Database Design for Mere Mortals. 3rd Edition. – Addison-Wesley Professional, 2013. – 672 p.
13. Bagui S., Earp R. Database Design Using Entity-Relationship Diagrams. 2nd ed. – CRC Press, 2011. – 362 p.
14. Foster E.C., Godbole S. Database Systems: A Pragmatic Approach. Second Edition. – Apress, 2016. – 619 p.
15. Elmasri R., Navathe S.B. Fundamentals of Database Systems. 7th ed. – Addison Wesley, 2016. – 1272 p.
16. Powell G. Beginning Database Design. – Wrox, 2006. – 500 p.