פרויקט מערכות בסיסי נתונים

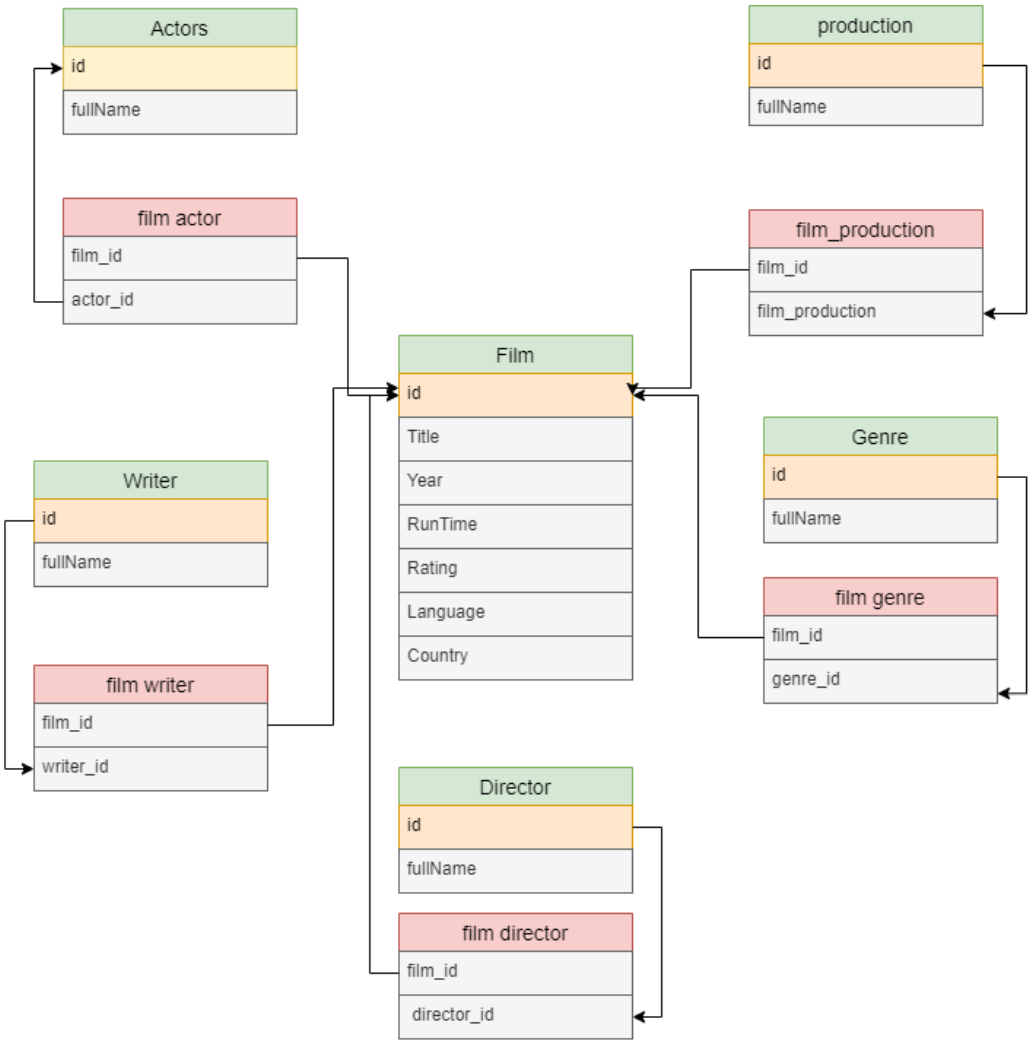
תיעוד התוכנה

1. סכמת ה-DB:
להלן הסכמה:

- Film (id, Title, Year, Runtime, Rating, Language, Country)
- Director(id, fullName)
- Actors(id, fullName)
- Writer(id, fullName)
- Genre(id, fullName)
- Production(id, fullName)
- Film_Actors(Film_id, Actor_id)
- Film_Director(Film_id, Director_id)
- Film_Genre(Film_id, Genre_id)
- Film_Production(Film_id, Production_id)
- Film_Writer(Film_id, Writer_id)

החלטנו לבנות את ה-DB כך משום שרצינו לבנות אפליקציה שמבוססת על סרטים והמידע שקשור אליהם.
ב-Film השארנו את השדות שנקבעים ביחידות על ידי ה-id ושגודלם חסום.
החלטנו לפצל את הז'אנר, ההפקה, הבמאים, הכותבים והשחקנים שקשורים לסרט מסוים לטבלאות משלהם שמקושרות לטבלת Film באמצעות טבלאות ביניים מתאימות, משום שיש בין הטבלאות האלה קשר של רבים לרבים – ה-data של ז'אנר, הפקה, שחקנים, במאים וכותבים עשוי להיות ארוך, ולכן לא הגיוני לייצג אותו בעמודה אחת בטבלת ה-Film.
בנוסף מאפשר חיפוש מהיר על ידי יצירת אינדקסים על המפתחות הראשיים לכל טבלה.
כמו כן, כך אנחנו שומרים על טבלאות קטנות יחסית כפי שהיינו רוצים (זהו תכנון DB נכון).

את שפה וארץ השארנו בטבלה הראשית של Films, כיוון שבשלב זה אין עוד חיפושים המתבצעים על שדות אלו, אך לא רצינו לאבד מידע.



2. <u>אופטימיזציות שביצענו ב-DB:</u>

- לכל טבלה ראשית הגדרנו מפתח ראשי ואינדקס על אותו המפתח. ניתן להניח בעת שבכל התייחסות לטבלה ראשית, נעשה שימוש באינדקס זה, אלא אם צוין אחרת.
- לכל טבלת ביניים הגדרנו מפתחות זרים, שבעזרתם ביצענו קישור בין הטבלאות השונות כפי שניתן לראות בסכמה. ההגדרות הללו עוזרות לביצוע Join מהיר בין הטבלאות.
- יצרנו ארבעה אינדקסים נוספים:
  - אינדקס FULLTEXT בטבלת Director, בעמודת fullName. קיימות שתי שאילתות FullText לטבלה זו.
  - יצרנו אינדקס עבור 3 טבלאות ביניים: Film_Actor, Film_Director, Film_Writer על עמודות Actor_id, Director_id, Writer_id בהתאמה כדי לייעל את פעולת הjoin שאנו עושים בין הטבלאות הללו.
- שימוש בתת שאילתות לשם צמצום הקלט לפעולת Join.

3. <u>תיאור השאילתות שכתבנו</u>:

נמספר השאילתות בהתאם למתודות בקובץ server.py:

query_1:

- השאילתה מחזירה: 20 הז'אנרים/הפקות (לפי קלט) הטובים ביותר, לפי כמות הסרטים עם דירוג 7 ומעלה.
- אופטימיזציות: אינדקסים ראשיים לטבלאות הראשיות
- תמיכה ע"י המבנה: לשתי הטבלאות של הז'אנרים/הפקות יש אותו מבנה ואותן עמודות, לכן ניתן לעבוד עם קלט. באותו אופן גם לטבלאות המקשרות.

query_3:

- השאילתה מחזירה: מחזיר את חמש הז'אנרים עם הכי שחקנים/במאים/כותבים
- אופטימיזציות: אינדקסים ראשיים לטבלאות הראשיות
- תמיכה ע"י המבנה: לשלוש הטבלאות של הז'אנרים/הפקות יש אותו מבנה ואותן עמודות, לכן ניתן לעבוד עם קלט. באותו אופן גם לטבלאות המקשרות.

query_4:

- השאילתה מחזירה: מחזירה את 20 השחקנים ששיחקו בהכי הרבה ז'אנרים ואת כמות הז'אנרים הזו לכל שחקן
- אופטימיזציות: אינדקסים ראשיים לטבלאות הראשיות
- תמיכה ע"י המבנה: הפירוק של טבלאות הביניים מאפשר לחבר את הטבלאות.

query_5:

- השאילתה מחזירה: השאילתה הזו מחזירה 100 שחקנים שעבדו עם הבמאי הנתון.
- אופטימיזציות: שימוש ב FullText אינדקס ובאינדקסים על Film_actor, Film_director.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין במאי לשחקן מתאים בהתאם לסרט בו הם השתתפו, בצורה יעילה.

query_6:

- השאילתה מחזירה: השאילתה הזו מחזירה 100 סרטים טובים ביותר (לפי עמודת הרייטינג) בהינתן במאי.
- אופטימיזציות: שימוש ב FullText אינדקס ובאינדקס על Film_director.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין במאי לסרט המתאים בצורה יעילה.

query_7:

- השאילתה מחזירה: השאילתה הזו מחזירה 100 במאים שביימו יותר מחמישה סרטים מז'אנר שהתקבל כקלט.
- אופטימיזציות: אנו משתמשים בתת שאילתה אשר מחזירה טבלה שמכילה את מספר הסרטים של כל במאי מהז'אנר שהתקבל כקלט. תת שאילתה זו משתמשת באינדקס על Film_Director.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין במאי לסרטים מהז'אנר המתאים בצורה יעילה.

query_8:

- השאילתה מחזירה: השאילתה הזו מחזירה 100 כותבים ואת מספר הסרטים שבהם הם השתתפו (בסדר יורד לפי מספר הסרטים) כך שכל סרט כזה הוא מעל הרייטינג שהתקבל בקלט, ומהז'אנר שהתקבל בקלט.
- אופטימיזציות: אנו יוצרים תת שאילתה אשר מחזירה טבלה שמכילה את כל הסרטים והרייטינג שלהם מהז'אנר שהתקבל בקלט, ומעל הרייטינג שהתקבל בקלט. טבלה זו מצמצמת לנו את הקלט ל - Join בשאילתה החיצונית. בנוסף אנו משתמשים באינדקס על Film_Writer.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין כותב לסרטים מהז'אנר המתאים בצורה יעילה.

query_9:

- השאילתה מחזירה: השאילתה הזו מחזירה את 10 הז'אנרים עם ה-rating הממוצע הכי גבוה (ממוצע על פני כל הסרטים שהם מהז'אנר הזה).
- אופטימיזציות: אנו עושים join על מפתחות ראשיים - id של טבלת Film ושל טבלת Genre.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין ז'אנר לסרטים מהז'אנר הזה בצורה יעילה.

qurey_10:

- השאילתה מחזירה: מספר הסרטים לכל ז'אנר, עם דירוג מעל 8.
- אופטימיזציות: אינדקסים על מפתחות ראשיים.
- תמיכה ע"י המבנה: הפירוק לטבלאות ביניים מאפשר לנו לחבר בין מהז'אנר לסרט.

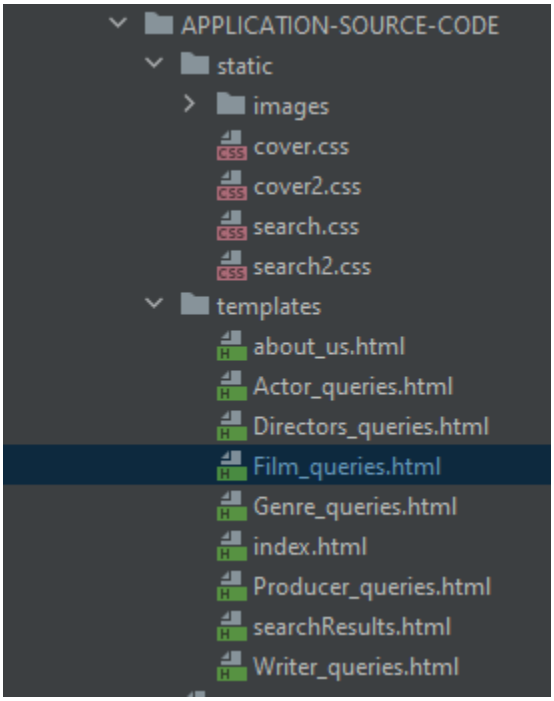
qurey_11:

- השאילתה מחזירה: את הסרט שדורג הכי גבוה בכל שנה, מאז העשור שנבחר על ידי המשתמש (1960 /1970 /1980 /1990 /2000 /2010 /2020), עד 2020 .
- אופטימיזציות: אינדקסים על מפתחות ראשיים.

<u>תיעוד מבנה הקוד</u>

נתאר את החלוקה למודולים:

1. API-DATA-RETRIEVE.py – המודול מתחבר ל-API שבחרנו, ולוקח ממנו מידע בצורת קבצי json (שיורדו אל המחשב שבו המודול רץ). לבסוף הוא מריץ את המודולים BreakJsonToAllTables.py ו - Push_table.py.
2. BreakJsonToAllTables.py – המודול לוקח את קבצי ה-json שנלקחו מה-API, ובאמצעותם יוצר קבצי csv שמתאימים לטבלאות שיש ב-DB. הוא למעשה ממלא את קבצי ה-csv במידע המתאימים לפי ה-attributes שבחרנו בתכנון ה-DB (ושכתובים בסכמת ה-DB).
3. Push_table.py – מודול זה מכניס לטבלאות את קבצי ה-csv שיצרנו באמצעות המודול BreakJsonToAllTables.py.
4. CREATE-DB-SCRIPT.sql – סקריפט sql שיוצר את הטבלאות לפי הסכמה שכתבנו לעיל.
5. Server.py – המודול אחראי על ביצוע שלושה דברים שונים:
   - קבלת קלט מהמשתמש ע"י בקשות HTTP מהאפליקציה.
   - התחברות ל – MySQL Server.
   - ביצוע שאילתות קבועות מראש בהינתן קלט מהאפליקציה.
6. ממשק משתמש: תחת תיקיית APPLICATION-SOURCE-CODE , נמצאות שתי תיקיות הנוגעות לממשק משתמש :
   - תיקיית templates : שם נוכל למצוא את דפי הhtml שבהם השתמשנו:
   - תיקיית static : שם נוכל למצוא את דפי הcss שאחראים על העיצוב של האובייקטים שאנחנו משתמשים בהם בדפי הhtml .



תיאור ה-API

השגת המידע מתחלקת לשני שלבים, את המידע השגנו דרך אתר RAPID-API, שעזר לנו להתחבר ל Movie Database (IMDB Alternative) API, כאשר מגישים בקשה לסרט לפי הID שלו באתר הרשמי של IMDB.

לשם כך, השתמשנו בקבצי tsv חינמים מהאתר הרשמי של IMDB, בהם ניתן להשיג את הID של מיליון יצירות, לאחר סינון לסרטים בשנים הרלוונטיות 1970 ומעלה, שמרנו את רשימת הID הרלוונטית.

באמצעות הקובץ ConnectToAPI.py התחברנו לAPI, כאשר אנחנו עוברים על רשימת הסרטים שהכנו בשלב הקודם. כל סרט נשמר לקובץ JSON.

כמו כן, שמנו לב שלסרטים רבים היה חסר מידע על הדירוג שלהם ולכן השתמשנו בקובץ חיצוני – ratings.csv שהורדנו מ-https://datasets.imdbws.com/ (הקובץ נתון בפורמט tsv והמרנו אותו ל-csv באמצעות כלי אינטרנטי). היכן שהיה חסר מידע, השתמשנו בממוצע הrating כדי לעדכן אותו.

ספריות חיצוניות

עבור צד השרת של האפליקציה (server.py) השתמשנו בספריות הבאות:

1. flask
2. mysql.connector
3. json

במודול BreakJsonToAllTables.py השתמשנו בספריות הבאות:

1. csv
2. json
3. pandas
4. os

במודול API-DATA-RETRIEVE.py השתמשנו בספריות הבאות:

1. requests
2. pathlib
3. os

במודול push_table.py השתמשנו בספריות הבאות:

1. mysql.connector
2. pandas

Flow כללי של האפליקציה

נבחר בעמוד הראשי, עמוד שאילתות לגשת אליו, נבחר שאילתה להפעלה, נקבל תוצאות.