תכנון ה-DB

השתמשנו ב-relational decomposition כדי לפרק את הנתונים (מה-API ומקובץ האקסל). במצב ההתחלתי היה שכפול גדול של מידע (redundancy), בזבוז מקום על הדיסק, attributes פחות שימושיים ומפתחות שלא קבעו את כל ה-attributes באופן חד-חד ערכי. הפיצול היה לתתי טבלאות, כאשר העקרונות המנחים היו:

- lossless - כשנפרק לתתי טבלאות ונאחד מחדש, נקבל בדיוק את אותה טבלה
- dependency preserving – כשנפרק לתתי טבלאות, הטבלאות החדשות לא יסתרו את חוקי הטבלה המקורית
- החלטנו על שליפות מרכזיות לדעתנו, ועל בסיסן בחרנו אילו טבלאות להגדיר כדי לאפשר שליפה אופטימלית

שינינו את ה-data types של כל העמודות להיות קטנים ככל שניתן על מנת לחסוך במקום. כמו כן, בשלב יותר מתקדם, השמטנו מה-DB רשומות שהכילו ערכים ריקים עבור עמודות שהן חלק מהשאילתות שהגדרנו, כדי להימנע מתוצאות לא אימפורמטיביות (רשומות ללא imdb_id למשל) ועל מנת לחסוך במקום.

מבנה הטבלאות

table	primary key	foreign keys
movies	imdb_id	
genres	id	
directors	id	
spoken_languages	id	
production_companies	id	origin_country
production_countries	id	
movies_to_genres	id	imdb_id, genres_id
movies_to_directors	id	imdb_id, director_id
movies_to_spoken_languages	id	imdb_id, spoken_languages_id
movies_to_production_companies	id	imdb_id, production_companies_id
movies_to_production_countries	id	imdb_id, production_countries_id

השאילתות

מדף הבית ישנה גישה ל-7 דפים שונים, כל דף עוסק בשאילתא אחרת מבין 7 השאילתות המרכזיות – כאשר על בסיס אינטרקציה עם המשתמש מיוצרת בכל פעם השאילתא הרצויה ומחושב ומוחזר פלט בהתאם. יש לציין שנעשה שימוש ב-3 שאילתות נוספות – שמייצרות רשימה של הז׳אנרים, השפות והמדינות הקיימים נכון להיום ב-DB. הן משמשות לממשק עם המשתמש (כדי להציג לו אפשרויות בחירה). לכאורה ניתן היה להחליף שאילתות אלה ברשימות סטטיות שנשמרות על האתר במנותק מה-DB, אך לדעתנו הדבר שגוי שכן הוא איננו לוקח בחשבון את

האפשרות של שינוי ב-DB שעלול לשנות את הרשימה. כלומר, גם אם הדבר לא יבוא לידי ביטוי בפרויקט כאן, שמירה סטטית לדעתנו היא קונספט שגוי.

ככלל – השתדלנו לצמצם במקום כדי להאיץ את הרצות השאילתות. את מרבית המידע על סרט מחלצים באמצעות "טבלאות מיפוי" – כך למשל כדי למצוא עבור סרט מסוים, עם מזהה tt1100119, מדינה שהוא הופק בה, עלינו לראות בטבלת המיפוי הרלוונטית (movies_to_production_countries) שהקיצור של המדינה הוא AE ואז להבין בטבלת המדינות, שמצומצמת יותר באופן משמעותי, שהכוונה ב-AE היא ל-United Arab Emirates. החסכון במקום כאן הוא משמעותי, למול טבלה מאוחדת שם לא הייתה דחיסה והיה שכפול רב. כך גם נכון ליתר הפיצולים שעשינו. ראוי לציין שפיצולים אלה הם גם מה שאפשר לנו להשיג ביעילות יחסית את הרשימות שמוצגות למשתמש, אותן הזכרנו בסעיף הקודם.

כפועל יוצא מהפיצול על בסיס טבלאות מיפוי, החיפוש על בסיס מזהים של סרטים (imdb_id) היה תכוף מאוד, וחזר בכל שאילתא. הצורך ליעל את תהליך האיתור היה חשוב מאוד, בפרט כשמדובר ב-Join של טבלאות רבות. לכן, אינדקסנו בטבלת הסרטים המרכזית Movies ובכל טבלאות המיפוי את עמודת ה-imdb_id, כנגזרת מכך ש-attribute זה הוגדר כ-primary key או foreign keys בטבלאות אלה. הדבר נכון גם עבור החיפושים על בסיס המזהים לז'אנרים, חברות, מדינות ושפות, בטבלאות בהן הם הוגדרו כמפתחות. מבחינת אינדוקס של ה-full text query: אינדקסנו את עמודת ה-overview ואת עמודת הtitle יחדיו אינדוקס של full text. האינדוקס אפשר לנו לתשאל ביעילות את טבלת Movies שאילתות של full text (כגון שאילתא 5) הבחירה נעשתה מתוך מחשבה ששאילתות מסוג זה יעשו לרוב על עמודות אלו יחדיו.

כללי אצבע נוספים שהנחו אותנו:

- הימנעות משימוש ב-like שמאט את החיפוש – משום שלמשתמש מוצגת רשימה בחירה ללא יכולת שינוי במקרים אלה, ניתן להשתמש באופרטור "=" שהוא יעיל יותר משמעותית.
- הימנעות משימוש ב-OR , והעדפה לשימוש ב-IN (למשל בשאילתא מספר 7) משום שכפי שראינו בשיעור, ב-MYSQL, IN יעיל יותר.
- הימנעות משאילתות מקוננות מיותרות והמרתן במידת האפשר למבנים אחרים (למעשה רק בשאילתא בודדת יש שימוש בשאילתות מקוננות).
- ביצוע projection מתי שניתן והכי מהר שניתן כדי לצמצם את הקלט לשאילתות חיצוניות

השאילתות שמימשנו הינן:

1. עבור קלט מהמשתמש שהוא ז'אנר מתוך רשימת הז'אנרים, טווח תאריכים ומספר הרשומות המקסימלי שיוחזר – מוחזרים כל הסרטים תחת ז'אנר זה שתאריך היציאה שלהם הוא בטווח התאריכים הנ"ל; כאשר מספר הרשומות המוחזר הוא לכל היותר המספר שהזין המשתמש. הסרטים המוחזרים מסודרים מהחדש ביותר לישן ביותר.
2. עבור קלט מהמשתמש שהוא טווח תאריכים – מוחזר עבור כל שפה, כמה סרטים יצאו בטווח תאריכים זה בשפה זו. המיון הוא לפי כמות הסרטים הנ"ל בסדר יורד (כלומר, השפות השכיחות ביותר תחילה).

3. עבור קלט מהמשתמש שהוא מדינה מתוך רשימת המדינות – מוחזר עבור כל חברת הפקות במדינה זו, מה הז׳אנר שהיא הכי מזוהה איתו, כלומר, הז׳אנר הכי נפוץ מבחינת כמות סרטים שהיא הפיקה שסווגו תחת ז׳אנר זה.
4. עבור קלט מהמשתמש שהוא מספר הרשומות המקסימלי שיוחזר – מוחזר עבור כל חברת הפקה וז׳אנר (שהיא הפיקה לפחות סרט אחד שמסווג תחתיו), מה הדירוג הממוצע של הסרטים שהיא הפיקה שמסווגים תחת ז׳אנר זה; כאשר מספר הרשומות המוחזר הוא לכל היותר המספר שהזין המשתמש.
5. עבור קלט מהמשתמש שהוא מילה שהוא רוצה שתופיע בפני עצמה או כתחילית בשם הסרט ו/או תיאור הסרט, מילה שהוא אינו רוצה שתופיע בפני עצמה או כתחילית בשם הסרט ו/או תיאור הסרט, וטווח תאריכים. יוחזרו למשתמש כל הסרטים העונים על הקריטריונים האלו ויוצגו שם ותיאור הסרט עבור המשתמש.
6. עבור קלט מהמשתמש שהוא טווח תקציבים – מוחזרת עבור כל שפה כמות הסרטים בשפה זו בטווח התקציבים הנ״ל. התוצאות מסודרות לפי מספר הסרטים בסדר יורד (כלומר, השפות השכיחות יותר תחילה).
7. עבור קלט מהמשתמש שהוא ז׳אנר מתוך רשימת הז׳אנרים, טווח של משך הזמן (של סרט) בדקות ושתי שפות – האפליקציה מנסה להמליץ למשתמש על סרט מתאים. היא מחזירה את כל הסרטים תחת תנאים אלה (תחת הז׳אנר הנ״ל, עם משך זמן בטווח הנ״ל, ושמדוברת בהם לפחות אחת משתי השפות), כאשר לכל סרט מוצמד הדירוג הממוצע שניתן לו. התוצאות מוחזרות לפי דירוג זה בסדר יורד, כלומר, הסרטים שדורגו גבוה יותר תחילה.