13.01.2022r.

Raport

Projekt - Bazy danych

Patryk Świątek

Wydział Matematyki i Nauk Infromacyjnych
Politechnika Warszawska

Spis treści

1 Wstęp

Celem projektu było zaprojektowanie modelu bazy danych dla platformy streamingowej, w tym:

- stworzenie relacyjnej bazy danych
- wstawienie przykładowych rekordów i ich modyfikacja
- zaprojektowanie indeksów
- przygotowanie zapytań *SQL SELECT*
- utworzenie procedury składowanej

W raporcie przedstawione zostaną wyniki zapytań, których kod źródłowy jest w dołączonych plikach:

* **Patryk_Swiatek_tworzenie_bazy_indeksy.sql** (kod tworzący bazę wraz z indeksami, wstawiający i modyfikujący rekordy)
* **Patryk_Swiatek_SQL_queries.sql** (kod zapytaniami *SQL SELECT*)
* **Patryk_Swiatek_procedura_skladowana.sql** (kod z procedurą składaną)

2 Projekt bazy danych

2.1 Opis bazy

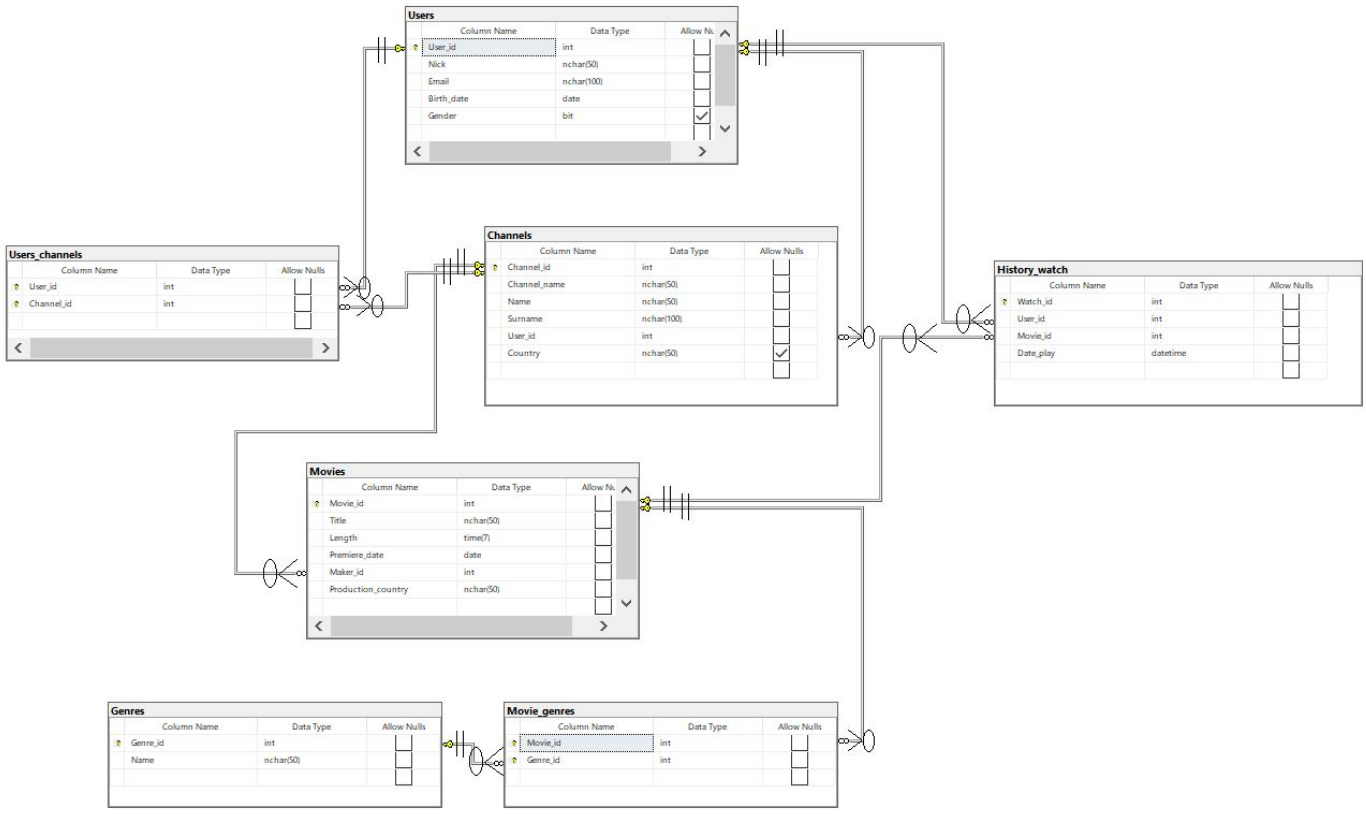
Baza danych składa się z siedmiu tabel ze zmiennymi połączonymi relacjami różnego typu widocznymi w następnym podrozdziale na schemacie ER. Tabele *Movies* z kluczem głównym *Movie_id*, *Users* z kluczem głównym *User_id* oraz *Channels* z kluczem głównym *Channel_id* przedstawiają informacje odpowiednio o: filmach zamieszczonych na platformie streamingowej (m.in. tytuł, długość, data premiery czy producent), użytkownikach platformy (np. unikalny Nick, email czy płeć) oraz twórcach filmów prowadzących swoje kanały (nazwa kanału, imię i nazwisko twórcy czy jego kraj pochodzenia). Tworząc ostatnią z nich przyjęto założenie, że prowadzący kanał musi być wcześniej użytkownikiem platformy i posiadać swój *User_id*.

Stworzona została również tabela **History_watch** przedstawiająca obejrzane filmy przez wszystkich użytkowników. Utworzona została zmienna *Watch_id* będąca kluczem głównym wspomnianej tabeli. Koncepcja ta została przyjęta przez uwagę na to, że ten sam użytkownik może obejrzeć dany film kilkukrotnie.

Przez wgląd na relacje *"wiele do wielu"* poszczególnych zmiennych, w skład bazy wchodzą też tabele informujące o obserwowanych kanałach przez użytkowników (*Users_channels*) oraz o gatunkach filmów (zgodnie z wytycznymi każdy film należy do przynajmniej jednego gatunku, istotnie również jeden gatunek reprezentuje wiele filmów). Tabela *Genres* informuje o szczegółach z góry określonych gatunków.

2.2 Schemat ER

Poniżej załączony został schemat ER wygenerowany w programie *Microsoft SQL Server Management Studio 18* z poprawkami ułatwiającymi odczytanie relacji pomiędzy encjami.



2.3 Zdefiniowane relacje

W ramach bazy ustalone są relacje pomiędzy następującymi kluczami:

- Users(User_id) — Channels(Channel_id) - relacja *wele do wielu*
- Channels(Channel_id) — Movies(Maker_id) - relacja *jeden do wielu*
- Movies(Movie_id) — History_watch(Movie_id) - relacja *jeden do wielu*
- Users(User_id) — History_watch(User_id) - relacja *jeden do wielu*
- Users(User_id) — Channels(User_id) - relacja *jeden do wielu*
- Genres(Genre_id) — Movies(Movie_id) - relacja *wiele do wielu*

3 Propozycje indeksów

Dla każdej tabeli w programie Microsoft SQL Server Management Studio automatycznie tworzone są indeksy dla kolumn będących kluczami głównymi. Mając na celu zwiększenie wydajności przeszukiwania tabel, na przykład przy zapytaniach *SQL SELECT*, zaproponowane zostały dodatkowe indeksy na wybranych tabelach. Wszystkie z nich są typu *NONCLUSTERED*.

Stworzone zostały unikalne indeksy dla tabel:

- *Channels* - dla kolumny *Channel_name* z nazwami kanałów
- *Users* - dla kolumny *Nick* przechowującej nick użytkowników
- *Genres* - dla nazw gatunków (kolumna *Name*)

Powyższe indeksy zostały stworzone ze względu na unikalność kolumn oraz potencjalnie częste wyszukiwanie przy zapytaniach *SQL SELECT*, szczególnie przy łączeniu tabel używane mogą być nazwy kanałów czy nick użytkowników. Z kolei ze względu na to, że nazwy gatunków są podane odgórnie przez operatora platformy, można założyć, że rekordy tej tabeli nie będą podlegały modyfikacjom. Mało prawdopodobne jest też dodawanie nowych rekordów, zatem można pominąć te sytuacje, w których stosowanie indeksu jest czasami nieefektywne. Z kolei indeks może pomóc na przykład przy wyszukiwaniu danych rekordów (np. używając polecenia *WHERE* czy *IN)*

Kolejne indeksy, które nie są już unikalne, zostały zaproponowane dla tabel:

- *History_watch* - dla kolumn:
  - *Movie_id*
  - *User_id*
  - *Date_play*
- *Channels* - dla kolumn z imieniem i nazwiskiem twórców (*Name,Surname*)
- *Movies* - dla:
  - *Maker_id*
  - *Production_country*

Powodem zaprogramowania powyższych indeksów również jest ich potencjalnie częste używanie przy łączeniu różnych tabel, czy wyszukiwaniu danej grupy rekordów (np. wyszukując filmy po imieniu i nazwisku twórcy, kraju produkcji czy analizując historię oglądania poszczególnych użytkowników lub oglądalność danego filmu). Dodatkowo indeksy stworzone dla daty odtwarzania filmu przez użytkownika zwiększą wydajność wykonywania procedury składowanej zawartej w poleceniu (przy przenoszeniu rekordów ze starszą datą oglądania niż określona liczba dni).

4 Zapytania SQL SELECT

W ramach projektu zostało napisane pięć zapytań tworzących raporty na podstawie wyszukiwanych rekordów spełniających dane warunki. Zapytania zawarte są w pliku .sql, w raporcie przedstawione zostaną jedynie testy.

4.1 Zapytanie 1

Łączna długość filmów obejrzanych przez użytkowników na danym kanale. Kolumny wynikowe: nick, łączny czas, nazwa kanału.

Sprawdzenie poprawności działania i użycia funkcji *DATEPART* (Oblicza długość trwania filmu w minutach)

```
SELECT u.Nick,m.Movie_id,
m.Length, ch.Channel_name,
DATEPART(SECOND, m.Length)/60 +
DATEPART(MINUTE, m.Length) +
60 * DATEPART(HOUR, m.Length)
 as 'Total minutes'
FROM History_watch h
JOIN Users u ON h.User_id = u.User_id
JOIN Movies m ON h.Movie_id = m.Movie_id
JOIN Channels ch ON ch.Channel_id = m.Maker_id
```

Wynikowa tabela:

	Nick	Movie_id	Length	Channel_name	Total minutes
6	hello123	2	02:22:00.0000000	famousmovies	142
7	hello123	5	02:42:00.0000000	bestdirector	162
8	hello123	6	03:21:00.0000000	lordofthering	201
9	moviefreak123	5	02:42:00.0000000	bestdirector	162
10	wannapassdatabases1	5	02:42:00.0000000	bestdirector	162
11	wannapassdatabases1	2	02:22:00.0000000	famousmovies	142
12	wannapassdatabases1	4	03:20:00.0000000	bestdirector	200
13	patrick567	5	02:42:00.0000000	bestdirector	162
14	frankbestdirector	5	02:42:00.0000000	bestdirector	162
15	amanda2308	5	02:42:00.0000000	bestdirector	162
16	amanda2308	3	02:55:00.0000000	bestdirector	175
17	patrick567	1	02:22:00.0000000	famousfrench	142
18	amanda2308	4	03:20:00.0000000	bestdirector	200
19	amanda2308	4	03:20:00.0000000	bestdirector	200
20	martinmovies	5	02:42:00.0000000	bestdirector	162
21	fordfrancais	5	02:42:00.0000000	bestdirector	162
22	moviefreak123	2	02:22:00.0000000	famousmovies	142
23	patrick567	6	03:21:00.0000000	lordofthering	201
24	olivieroscars	6	03:21:00.0000000	lordofthering	201
25	moviefreak123	6	03:21:00.0000000	lordofthering	201

Funkcja poprawnie zlicza czas trwania filmu. Można zauważyć, że po zgrupowaniu po nazwie kanału i nicków użytkowników oraz po zsumowaniu w grupach czasu trwania filmu wynik będzie ten sam, co w zapytaniu docelowym.

4.2 Zapytanie 2

Wyszukać użytkowników, którzy obserwują kanał, ale obejrzeli na nim mniej niż trzy filmy. Raport ma zawierać informację o użytkowniku, o kanale oraz liczbę filmów jaką użytkownik obejrzał na danym kanale.

Zapytanie zliczające ilość obejrzanych filmów na każdym kanale:

```
SELECT Maker_id,
User_id,
COUNT(User_id) AS Films_watched
FROM (SELECT DISTINCT User_id, Maker_id, m.Movie_id
FROM History_watch h
JOIN Movies m ON h.Movie_id = m.Movie_id) AS Movies_distinct
GROUP BY Maker_id, User_id
```

Wynik:

	Maker_id	User_id	Films_watched
1	2	1	1
2	4	1	1
3	5	1	1
4	6	1	1
5	2	2	1
6	5	2	2
7	6	2	1
8	5	3	2
9	6	3	1
10	2	4	1
11	4	4	1
12	5	4	1
13	5	5	3
14	2	6	1
15	5	6	3
16	5	7	1
17	2	8	1
18	5	9	1

Kanały obserwowane przez użytkowników:

```
SELECT * FROM Users_channels
```

Wynik:

	User_id	Channel_id
1	1	2
2	2	1
3	2	2
4	3	1
5	3	5
6	4	1
7	4	2
8	5	1
9	5	2
10	6	2
11	6	3

Na tak małym zbiorze można zobaczyć, że z pierwszej tabeli wynikowej odrzucone zostaną jedynie dwa rekordy (użytkownicy z trzema obejrzanymi filmami), jednak po uwzględnieniu obserowanych kanałów, pozostaną jedynie użytkownicy wynikowi z zapytania docelowego.

4.3 Zapytanie 3

Dla każdego gatunku wyszukać film cieszący się największą popularnością.

Podzapytanie zliczające ilość obejrzanych filmów na każdym kanale:

```
SELECT
g.Name,
m.Title,
COUNT(*) AS Views_number
FROM History_watch h
JOIN Movies m ON h.Movie_id = m.Movie_id
JOIN Movie_genres mg ON mg.Movie_id = m.Movie_id
JOIN Genres g ON g.Genre_id = mg.Genre_id
GROUP BY g.Name, m.Title
```

Wynik:

	Name	Title	Views_number
1	Comedy drama	Intouchables	3
2	Drama	The Godfather I	3
3	Drama	The Godfather II	4
4	Drama	The Godfather III	8
5	Fantasy	The Lord of the Rings: The Return of the King	5
6	Drama	The Shawshank Redemption	2

Podzapytanie zlicza oglądalność dla wszystkich filmow z danych gatunkow. Można zauważyć, że po wyborze filmów z największą oglądalnością dla każdego gatunku otrzymany zostanie wynik zapytania zawartego w pliku .sql.

4.4 Zapytanie 4

Dla każdego kanału stosunek liczby wyświetleń do liczby opublikowanych filmów. Wyniki posortować malejąco po tej wartości.

Podzapytanie zliczające ilość wyświetleń na danym kanale:

```
SELECT Channel_id,
COUNT(*) AS Viewings
FROM History_watch h
JOIN Movies m ON h.Movie_id = m.Movie_id
JOIN Channels ch ON ch.Channel_id = m.Maker_id
GROUP BY Channel_id
```

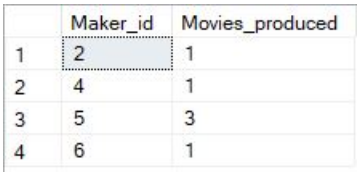
Wynik:

	Channel_id	Viewings
1	2	5
2	4	2
3	5	15
4	6	3

Podzapytanie zliczające ilość wyprodukowanych filmów na danym kanale:

```
SELECT Maker_id,
COUNT(*) AS Movies_produced
FROM Movies
GROUP BY Maker_id
```

Wynik:



	Maker_id	Movies_produced
1	2	1
2	4	1
3	5	3
4	6	1

Za pomocą elementarnego dzielenia można zauważyć, że stosunki pokrywają się z zapytaniem wynikowym, zatem tabele zostały połączone poprawnie i zapytanie zawarte w pliku .sql zwraca pożądany wynik.

4.5 Zapytanie 5

Wyświetlić filmy, które są dostępne w ramach kanału, który ma najwięcej subskrybentów.

W rozwiązaniu zdefiniowany został widok obrazujący tabelę będącą wynikiem następującego zapytania:

```
SELECT Channel_id, COUNT(*) AS sub
FROM Users_channels
GROUP BY Channel_id)
```

Wynik:

	Channel_id	sub
1	1	4
2	2	5
3	3	1
4	5	1

Najwięcej subskrybentów ma kanał o ID równym 2. Wystarczy sprawdzić teraz oferowane filmy przez ten kanał. Zapytanie:

```
SELECT Title
FROM Movies
WHERE Maker_id=2
```

Zwrócony został jeden tytuł filmu, mianowicie: *"The Lord of the Rings: The Return of the King"*, czyli wynik pokrywa się z wynikiem zapytania zawartym w pliku .sql.

5 Procedura składowana

Procedura została przetestowana dla parametru $DaysCount = 476$. Otrzymane wyniki:

Tabela **History_watch**:

	Watch_id	User_id	Movie_id	Date_play
1	4	6	6	2021-07-08 00:35:22.000
2	10	3	5	2021-09-07 00:35:27.000
3	11	3	2	2021-08-29 19:35:37.000
4	12	3	4	2021-09-30 21:05:27.000
5	18	5	4	2021-02-04 22:05:27.000
6	19	5	4	2021-02-07 22:25:27.000
7	20	6	5	2021-02-28 12:15:27.000
8	21	9	5	2020-12-31 02:15:27.000
9	22	1	2	2020-12-06 07:15:27.000

Tabela **HistoryArchive**:

	Watch_id	User_id	Movie_id	Date_play
1	1	1	1	2020-01-22 04:35:27.000
2	2	6	3	2019-11-25 03:32:21.000
3	3	6	4	2019-11-26 18:12:21.000
4	5	2	3	2019-04-01 19:44:27.000
5	6	2	2	2018-05-22 00:35:37.000
6	7	2	5	2020-06-07 15:41:27.000
7	8	2	6	2020-03-09 16:17:17.000
8	9	1	5	2020-01-13 17:35:27.000
9	13	4	5	2019-10-03 18:15:27.000
10	14	7	5	2019-11-06 17:35:37.000
11	15	5	5	2018-10-16 14:35:23.000
12	16	5	3	2019-07-19 14:25:27.000
13	17	4	1	2018-01-14 03:45:17.000
14	23	4	6	2019-09-24 05:35:27.000
15	24	8	6	2018-04-17 13:35:27.000
16	25	1	6	2019-06-08 01:28:27.000

Zaktualizowana tabela **Movies**:

	Movie_id	Title	Length	Premiere_date	Maker_id	Production_country	ViewsCnt
1	1	The Shawshank Redemption	02:22:00.0000000	1994-09-10	4	USA	2
2	2	Intouchables	02:22:00.0000000	2011-09-23	6	France	1
3	3	The Godfather I	02:55:00.0000000	1972-03-15	5	USA	3
4	4	The Godfather II	03:20:00.0000000	1974-12-12	5	USA	1
5	5	The Godfather III	02:42:00.0000000	1990-12-20	5	USA	5
6	6	The Lord of the Rings: The Return of the King	03:21:00.0000000	2003-12-05	2	New Zealand	4

Za pomocą poniższego zapytania obliczono datę o 476 dni wcześniej od bieżącej daty:

```
SELECT DATEADD(DAY, -476, getdate())
```

Co daje wynikową datę: *2020-09-24 05:48:56.337*. Analizując zmienną *Date_play* w dwóch tabelach wynikowych, można stwierdzić, że w tabeli *HistoryArchive* są starsze rekordy niż powyższa data, z kolei w tabeli *History_watch* pozostały jedynie rekordy nowsze od tej daty.

Można też zauważyć, że powstała nowa zmienna w tabeli *Movies* - *ViewsCnt*, która rzeczywiście zlicza oglądalność filmu, ale tylko dla przenoszonych rekordów. Pokazuje to wynik następującego zapytania:

```
SELECT Movie_id,
COUNT(*) AS ViewsCnt
FROM HistoryArchive
GROUP BY Movie_id
```

Wynik:

	Movie_id	ViewsCnt
1	1	2
2	2	1
3	3	3
4	4	1
5	5	5
6	6	4

Wyniki są zgodne z zaktualizowanymi wartościami zmiennej w tabeli *Movies*, co pokazuje poprawność napisanej procedury.