Osoby, którzy stworzyli projekt: Artem Lytvynenko, Alisa Malakhova, Anhelina Belavezha

Tytuł projektu: Baza danych platformy streamingowej

Cel projektu: Projekt dotyczy stworzenia bazy danych dla platformy streamingowej, umożliwiającej przesyłanie i oglądanie treści w czasie rzeczywistym.

Możliwości projektu: Baza danych ma na celu efektywne zarządzanie informacjami dotyczącymi użytkowników, treści multimedialnych, transmisji nażywo, interakcji społecznościowej oraz innych kluczowych elementów platformy.

Główne założenia projektu:

1. Skupienie się na zapewnieniu efektywnego zarządzania treściami multimedialnymi, umożliwienie śledzenia zmian w czasie, takich jak liczba upodobień, opinie, itp.
2. Wykorzystanie dziedziczenia w projektowaniu bazy danych dla bardziej elastycznego zarządzania różnymi rodzajami treści.
3. Każdy stream maksymalnie może trwać 23:59:59 godzin.

Skrypt tworzący bazę danych: Zawarty w pliku skrypt.sql

Typowe zapytania:

- Liczba obserwatorów danego streamera
- Liczba osób, których dane użytkownik obserwuje
- Top 5 streamerów pod względem liczby wyświetleń
- Najpopularniejsze kategorie streamów
- Top 3 streamerów pod względem liczby subskrypcji
- Statystyki dotyczące ankiet
- Statystyki dotyczące emoji w wiadomościach
- Zbanowani użytkownicy

Schemat bazy danych (diagram relacji)

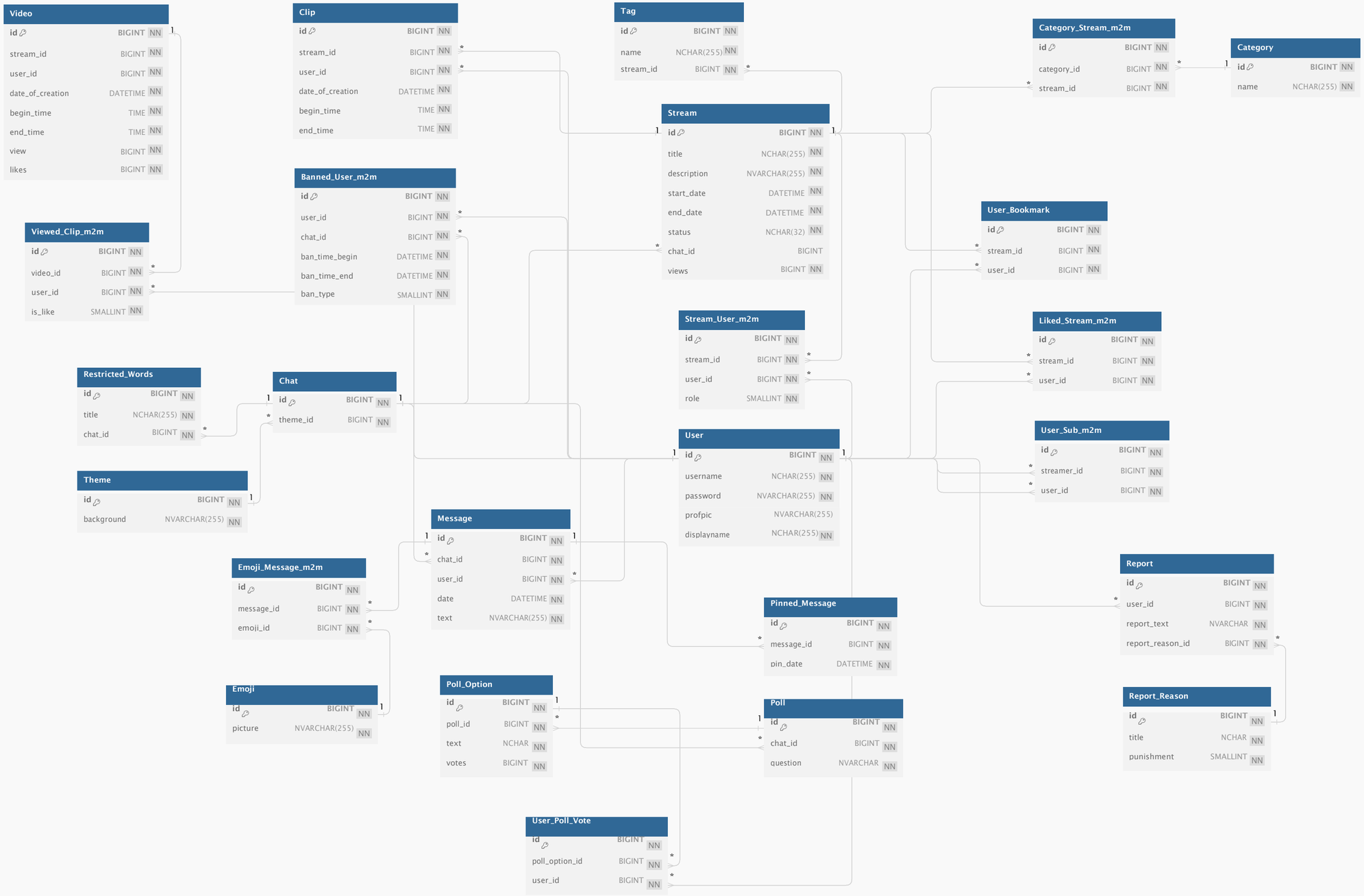
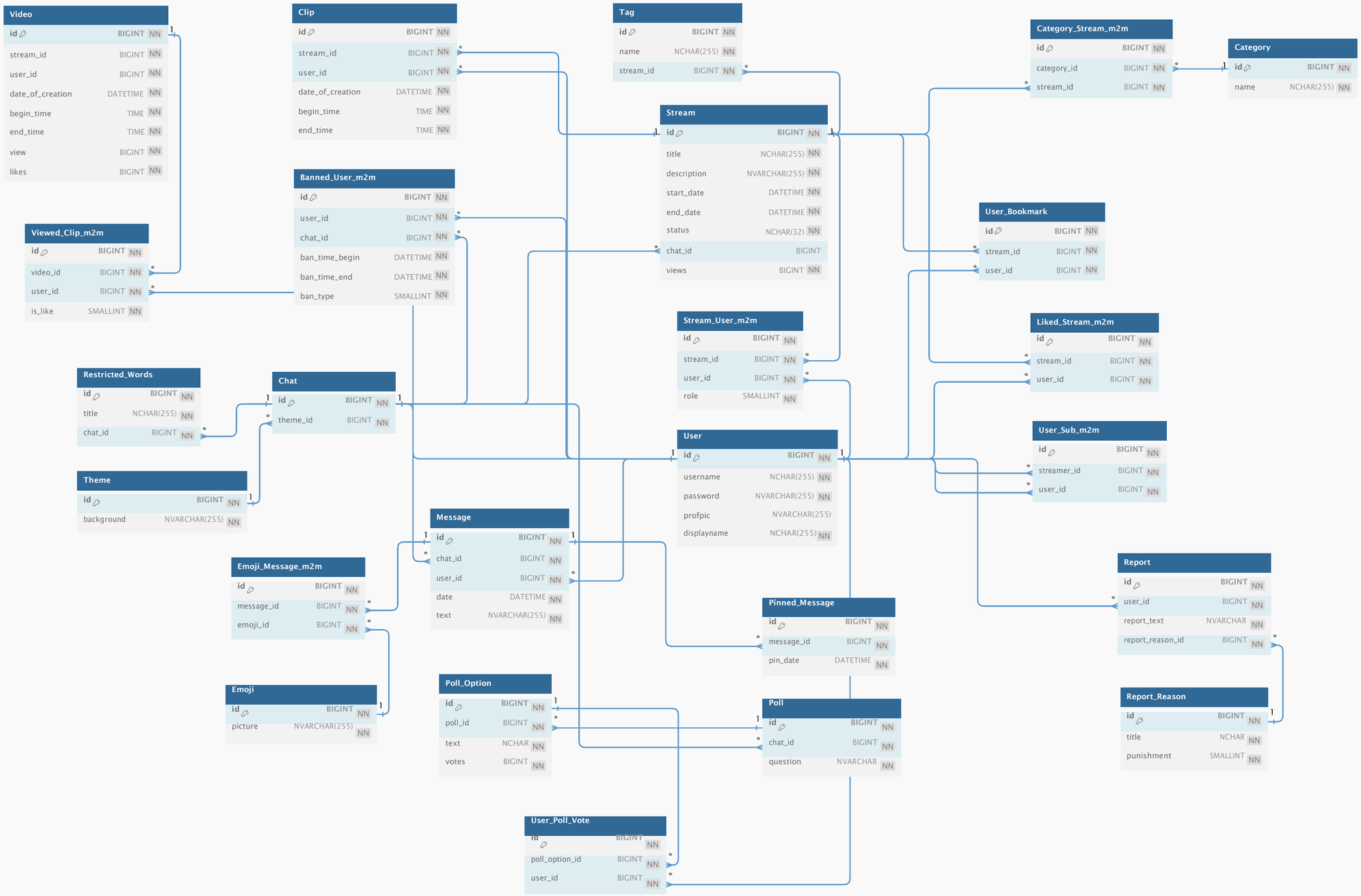


Diagram ER



Opis stworzonych widoków:

1. PopularCategories - Ten widok dostarcza informacji o popularnych kategoriach na podstawie liczby powiązanych strumieni. Pomocny do identyfikacji trendingowych lub popularnych kategorii treści na platformie streamingowej.

```
CREATE VIEW PopularCategories AS
SELECT
   C.id AS Category_ID,
   C.name AS Category_Name,
   COUNT(CS.stream_id) AS Stream_Count
FROM Category C
LEFT JOIN Category_Stream_m2m CS ON C.id = CS.category_id
GROUP BY C.id, C.name
```

2. TopStreamers - Ten widok prezentuje najlepszych nadawców z największą liczbą subskrypcji. Przydatny do rozpoznania najbardziej subskrybowanych nadawców na platformie.

```
CREATE VIEW TopStreamers AS
SELECT U.id, U.username, COUNT(S.streamer_id) AS Subscribtion_Number
FROM [User] U
LEFT JOIN User_Sub_m2m S ON U.id = S.streamer_id
GROUP BY U.id, U.username
```

3. TopStreams - Ten widok wyświetla najlepszych strumieni na podstawie liczby polubień. Idealny do prezentacji najbardziej lubianych strumieni na platformie.

```
CREATE VIEW TopStreams AS
SELECT S.id, S.title, COUNT(*) AS Likes_Number
FROM Stream S
LEFT JOIN Liked_Stream_m2m L ON S.id = L.stream_id
GROUP BY S.id, S.title
```

4. TopClips - Ten widok dostarcza informacji na temat najlepszych klipów na podstawie liczby polubień. Przydatny do wyróżnienia najbardziej lubianych klipów, dając użytkownikom wgląd w popularne i angażujące krótkie treści.

```
CREATE VIEW TopClips AS
SELECT
   C.user_id AS clip_creator_id,
   U.username AS clip_creator_username,
   C.stream_id,
   S.title AS stream_title,
   C.date_of_creation,
   C.begin_time,
   C.end_time,
   COUNT(LC.user_id) AS likes_count
FROM Clip C
JOIN [User] U ON C.user_id = U.id
JOIN Liked_Clip_m2m LC ON C.id = LC.clip_id
JOIN Stream S ON C.stream_id = S.id
GROUP BY
   C.user_id,
   U.username,
   C.stream_id,
   S.title,
   C.date_of_creation,
   C.begin_time, C.end_time
```

5. ReportedUsers - Widok został stworzony w celu dostarczenia informacji o zgłoszonych użytkownikach w systemie.

```
CREATE VIEW ReportedUsers AS
SELECT
   RU.user_id,
   U.username,
   R.title AS report_reason
FROM
   Report RU
JOIN [User] U ON RU.user_id = u.id
JOIN Report_Reason R ON RU.report_reason_id = R.id
```

6. ActiveStreams - Widok został stworzony w celu dostarczenia informacji o aktywnych transmisjach w systemie.

```
CREATE VIEW ActiveStreams AS
SELECT
   S.id,
   S.title,
   U.username AS streamer_username,
   S.start_date,
   S.end_date,
   S.views
FROM
   Stream S
JOIN Stream_User_m2m SU ON S.id = SU.stream_id
JOIN [User] U ON SU.user_id = U.id
WHERE S.status = 1
```

7. BannedUsers - Widok został stworzony w celu dostarczenia informacji o zbanowanych użytkownikach w systemie.

```
CREATE VIEW BannedUsers AS
SELECT
   BU.user_id AS banned_user_id,
   U.username AS banned_username,
   BU.chat_id,
   BU.ban_time_begin,
   BU.ban_time_end,
   BU.ban_type
FROM
   Banned_User_m2m BU
JOIN [User] U ON BU.user_id = U.id
JOIN Chat C ON BU.chat_id = C.id
```

8. PollsStats - Widok został stworzony w celu przedstawienia informacji statystycznych dotyczących ankiet, obejmujących identyfikator ankiety, pytanie, tekst opcji ankiety oraz liczbę głosów uzyskanych dla każdej opcji.

```
CREATE View PollsStats AS
SELECT p.id, p.question, po.text, COUNT(upv.poll_option_id) votes
FROM Poll p
JOIN Poll_Option po ON p.id=po.poll_id
LEFT JOIN User_Poll_Vote upv ON po.id=upv.poll_option_id
GROUP BY po.id,po.text, p.id, p.question
```

9. Emoji_stats - Widok został stworzony w celu dostarczenia statystyk dotyczących użycia emoji w wiadomościach. Korzysta z danych zawartych w tabelach Emoji i Emoji_Message_m2m, aby przedstawić liczbę wystąpień każdego emoji w wiadomościach.

```
CREATE VIEW Emoji_stats
AS
SELECT e.id, COUNT(em.message_id) Messages_number
FROM Emoji e LEFT JOIN Emoji_Message_m2m em ON e.id=em.emoji_id
GROUP BY e.id
```

Opis stworzonych funkcji:

1. SubsOfSubs - Funkcja SubsOfSubs została stworzona w celu zidentyfikowania użytkowników, którzy są subskrybentami subskrybentów danego strumieniującego (streamera). Funkcja ta korzysta z rekurencyjnego zapytania wspólnego typu (Common Table Expression - CTE), aby znaleźć subskrybentów subskrybentów, określając odległość (liczbę pośrednich subskrybentów) między danym strumieniującym a danym użytkownikiem.

```
CREATE FUNCTION SubsOfSubs(@streamer_id bigint)
RETURNS @tab TABLE (follower_id bigint, streamer_id bigint, Distance
bigint)
BEGIN
;WITH CTE_SubsOfSubs (follower_id, streamer_id, Distance)
AS
(
SELECT user_id, streamer_id, 1
FROM User_Sub_m2m
WHERE streamer_id=@streamer_id
UNION ALL
SELECT P.user_id, CTE_SubsOfSubs.streamer_id, Distance + 1
FROM User_Sub_m2m AS P JOIN CTE_SubsOfSubs ON P.streamer_id =
CTE_SubsOfSubs.follower_id
AND P.user_id IS NOT NULL AND P.user_id <>@streamer_id
)
INSERT INTO @tab
SELECT * FROM CTE_SubsOfSubs
RETURN
END
```

2. PinnedMesagesInOrder - Funkcja PinnedMesagesInOrder została stworzona w celu zwrócenia przypiętych wiadomości w określonym porządku dla danego czatu. Funkcja ta korzysta z danych zawartych w tabelach Message i Pinned_Message, aby zidentyfikować wiadomości przypięte w danym czacie i zwrócić je w kolejności od najnowszej do najstarszej.

```
CREATE FUNCTION PinnedMesagesInOrder(@chat_id int)
RETURNS @tab TABLE (message_text NVARCHAR(255), pin_date DATETIME)
BEGIN
INSERT INTO @tab
SELECT m.text, pm.pin_date
FROM [Message] m JOIN Pinned_Message pm ON m.id=pm.message_id
WHERE m.chat_id=@chat_id
ORDER BY pm.pin_date DESC
RETURN
END
```

Opis stworzonych procedur:

1. NUMBER_OF_FOLLOWERS_OF - Ta procedura zwraca liczbę obserwatorów danego streamera na podstawie przekazanego identyfikatora streamera. Liczba ta jest zapisywana do parametru wyjściowego @num.

```
CREATE PROC NUMBER_OF_FOLLOWERS_OF(@streamer_id BIGINT, @num
BIGINT OUTPUT)
AS
SELECT @num=COUNT(*)
FROM [User_Sub_m2m]
WHERE streamer_id=@streamer_id
```

2. PRINT_FOLLOWERS_OF - Ta procedura zwraca identyfikatory użytkowników, którzy obserwują danego streamera na podstawie przekazanego identyfikatora streamera.

```
CREATE PROC PRINT_FOLLOWERS_OF(@streamer_id BIGINT)
AS
SELECT user_id
FROM [User_Sub_m2m]
WHERE streamer_id=@streamer_id
```

3. NUMBER_OF_FOLLOWINGS_OF - Ta procedura zwraca liczbę streamerów, których dany użytkownik obserwuje, na podstawie przekazanego identyfikatora użytkownika. Liczba ta jest zapisywana do parametru wyjściowego @num.

```
CREATE PROC NUMBER_OF_FOLLOWINGS_OF(@user_id BIGINT, @num
BIGINT OUTPUT)
AS
SELECT @num=COUNT(*)
FROM [User_Sub_m2m]
WHERE user_id=@user_id
```

4. PRINT_FOLLOWINGS_OF - Ta procedura zwraca identyfikatory streamerów, których dany użytkownik obserwuje, na podstawie przekazanego identyfikatora użytkownika.

```
CREATE PROC PRINT_FOLLOWINGS_OF(@user_id BIGINT)
AS
SELECT streamer_id
FROM [User_Sub_m2m]
WHERE user_id=@user_id
```

5. GetTopViewedStreamers - Procedura GetTopViewedStreamers została stworzona w celu uzyskania informacji na temat pięciu najczęściej oglądanych strumieniujących użytkowników. Procedura ta korzysta z danych zawartych w tabelach User i Stream, aby zidentyfikować użytkowników i sumować ilość wyświetleń ich strumieni.

```
CREATE PROCEDURE GetTopViewedStreamers AS
BEGIN
   SELECT TOP 5 U.id, U.username, SUM(S.views) AS Views_Number
   FROM [User] U
   LEFT JOIN Stream S ON U.id = S.id
   GROUP BY U.id, U.username
   ORDER BY Views_Number DESC
END
```

Opis stworzonych wyzwalaczy:

1. StreamUpdateViews - Wyzwalacz został stworzony w celu automatycznego aktualizowania liczby wyświetleń (views) strumienia w momencie dodawania nowych wpisów do tabeli Stream_User_m2m. Wyzwalacz ten wykonuje operację inkrementacji wartości views dla odpowiadającego strumienia za każdym razem, gdy nowe dane są wstawiane do tabeli Stream_User_m2m.

```
CREATE TRIGGER StreamUpdateViews ON Stream_User_m2m
AFTER INSERT AS
BEGIN
  DECLARE @currentId INT
  DECLARE InsertedIds CURSOR FOR
  SELECT stream_id FROM inserted
  FOR READ ONLY
  OPEN InsertedIds
  FETCH InsertedIds INTO @currentId
  WHILE(@@FETCH_STATUS=0)
  BEGIN
    UPDATE Stream
      SET views = views+1
      WHERE id = @currentId
    FETCH InsertedIds INTO @currentId
  END
  CLOSE InsertedIds
DEALLOCATE InsertedIds
END
```

2. SubscriptionCheck - Wyzwalacz został stworzony w celu kontrolowania operacji dodawania nowych subskrypcji do tabeli User_Sub_m2m. Działa on zamiast standardowej operacji wstawiania (INSTEAD OF INSERT), co oznacza, że zastępuje domyślne działanie wstawiania nowych rekordów.

```
CREATE TRIGGER SubscriptionCheck ON User_Sub_m2m
INSTEAD OF INSERT
AS
BEGIN
  DECLARE @currStreamerId BIGINT
  DECLARE @currViewerId BIGINT

  DECLARE insertTrack CURSOR FOR
  SELECT streamer_id, user_id FROM inserted
  FOR READ ONLY

  OPEN insertTrack
  FETCH insertTrack INTO @currStreamerId, @currViewerId

  WHILE(@@FETCH_STATUS = 0)
  BEGIN
    IF(@currStreamerId = @currViewerId)
       BEGIN
        DECLARE @ErrorMessage NVARCHAR(255)
        SET @ErrorMessage = FORMATMESSAGE(N'Self subscription is prohibited
(error with ids %I64d and %I64d)', @currStreamerId, @currViewerId)
        RAISERROR(@ErrorMessage, 16, 1)
      END
    FETCH insertTrack INTO @currStreamerId, @currViewerId
  END
  CLOSE insertTrack
  DEALLOCATE insertTrack

  INSERT INTO [User_Sub_m2m](streamer_id,user_id) SELECT streamer_id,user_id
FROM inserted

END
```

3. ViewUpdateLikeCheck - Wyzwalacz został stworzony w celu kontroli procesu aktualizacji danych o wyświetleniach i polubieniach w systemie, gdy nowe dane są wstawiane do tabeli Viewed_Clip_m2m. Głównym celem tego wyzwalacza jest śledzenie, czy dany użytkownik polubił czy wyświetlił już dany klip/wideo, a następnie aktualizacja statystyk wideo w zależności od tego, czy jest to pierwsze wyświetlenie, czy dodatkowe polubienie.

```
CREATE TRIGGER ViewUpdateLikeCheck ON Viewed_Clip_m2m
INSTEAD OF INSERT
AS BEGIN
  DECLARE currData CURSOR FOR
  SELECT video_id, user_id, is_like FROM inserted

  OPEN currData

  DECLARE @currVidId BIGINT
  DECLARE @currUserId BIGINT
  DECLARE @isLike SMALLINT

  FETCH currData INTO @currVidId, @currUserId,@isLike

  WHILE(@@FETCH_STATUS = 0)
  BEGIN
     IF EXISTS(SELECT id FROM Viewed_Clip_m2m WHERE video_id=@currVidId AND
user_id=@currUserId)
     BEGIN
        DECLARE @isLikePrev SMALLINT
        SET @isLikePrev = (SELECT is_like FROM Viewed_Clip_m2m WHERE
video_id=@currVidId AND user_id=@currUserId)

        IF (@isLike = 1 AND @isLikePrev = 0)
          UPDATE Video SET likes=likes+1 WHERE id = @currVidId
          UPDATE Video SET views=views+1 WHERE id = @currVidId
       END
       ELSE
       BEGIN
       IF @isLike = 1
        UPDATE Video SET likes=likes+1 WHERE id = @currVidId
       INSERT INTO Viewed_Clip_m2m(video_id, user_id, is_like) VALUES
       (@currVidId,@currUserId,@isLike)
       END
        FETCH currData INTO @currVidId, @currUserId,@isLike
    END
END
```

4. EnforceStreamLimits - Wyzwalacz został stworzony w celu egzekwowania ograniczeń dotyczących liczby użytkowników w danym strumieniu. Działa zarówno po operacjach wstawiania (AFTER INSERT) jak iaktualizacji (AFTER UPDATE) danych w tabeli Stream_User_m2m. Głównym celem tego wyzwalacza jest zapewnienie, że liczba użytkowników w jednym strumieniu nie przekracza ustalonego limitu, w tym przypadku 15 000 użytkowników. W przypadku przekroczenia limitu, wyzwalacz generuje błąd, informując o niemożności przekroczenia liczby 15 000 użytkowników w jednym strumieniu, i wykonuje operację cofnięcia (ROLLBACK).

```
CREATE TRIGGER EnforceStreamLimitsON [Stream_User_m2m]
AFTER INSERT, UPDATEAS
BEGIN
   --Zamiast 15000 moze być wyliczona maks ilość osób których moze znieść serwer
   IF (SELECT COUNT(*) FROM [Stream_User_m2m] WHERE stream_id = (SELECT
stream_idFROM INSERTED)) > 15000
   BEGIN
      RAISERROR('Cannot exceed 100 users in a stream.', 16, 1);ROLLBACK;
   END
END
```

5. PreventDuplicateStreams - Wyzwalacz został stworzony w celu zapobiegania dodawaniu duplikatów strumieni do tabeli Stream. Głównym celem tego wyzwalacza jest sprawdzanie, czy nowy strumień, który ma zostać dodany, nie jest duplikatem istniejącego aktywnego strumienia. W przypadku wykrycia duplikatu, wyzwalacz generuje błąd, informując o niedozwolonym dodaniu duplikatu strumienia. W przeciwnym razie, gdy nie wykryje duplikatu, dodaje nowy strumień do tabeli Stream.

```
CREATE TRIGGER PreventDuplicateStreams
ON Stream
INSTEAD OF INSERT
AS
BEGIN
   DECLARE currStream CURSOR FOR
   SELECT title FROM Stream

   OPEN currStream
   DECLARE @currTitle NVARCHAR(256)
   FETCH currStream INTO @currTitle

   WHILE(@@FETCH_STATUS = 0)
    BEGIN
    IF EXISTS (SELECT * FROM Stream WHERE title = @currTitle AND [status] =
'Active')
      BEGIN
         RAISERROR('Duplicate stream detected.', 16, 1);
         RETURN;
      END
    ELSE
    BEGIN
        INSERT INTO Stream (title, description, start_date, end_date, status, chat_id,
views)
        SELECT title, description, start_date, end_date, status, chat_id, views FROM
inserted;
      END
    FETCH currStream INTO @currTitle
   END
END
```