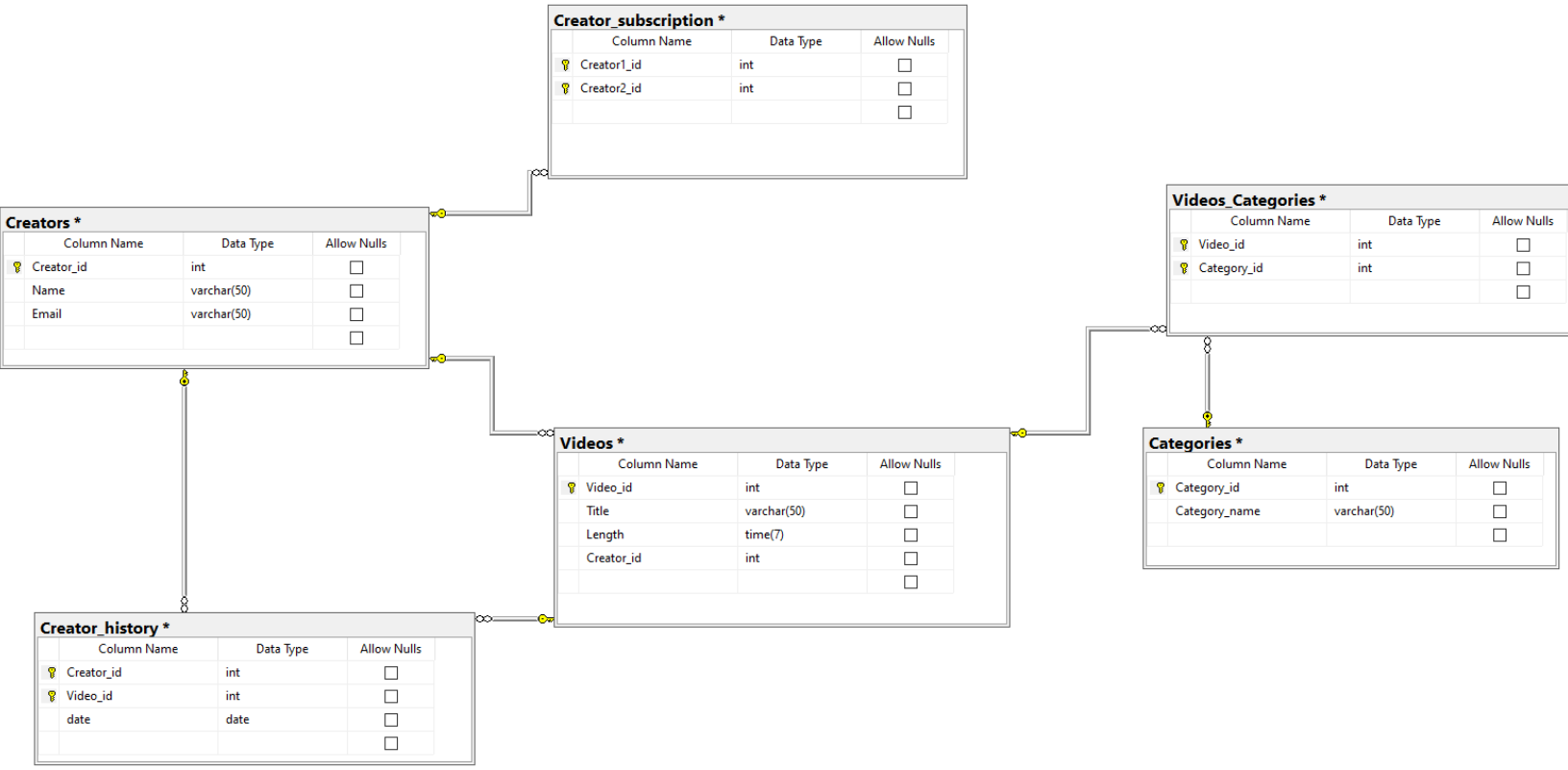
PD2

Wojciech Klusek

Maj 2021

1 Projektowanie bazy danych

Poniżej znajduje się diagram ER zaprojektowanej bazy danych.



Rysunek 1: diagram ER

1.1 Opis tabel

Baza danych posiada 6 tabel.

- Tabela Videos zawiera filmy na platformie, gdzie Video_id to klucz głowny, Title - kolumna zwierająca tytuł filmiku, Length - zawiera długość filmu, Creator_id zawiera id kreatorwa filmiku. Wszystkie te wartości nie są NULL gdyż każdy z filmów musi posiadać tytuł, długość oraz autora.
- Tabela Videos_Categories zawiera spis filmików wraz z ich kategoriami. Video_id oraz Category_id to klucz główny.
- Tabela Categories zawiera kolumny Category_id będąca kluczem głównym oraz Category_name będącą nazwą kategorii. Nazwa kategorii nie może być NULLEM.
- Tabela Creators zawiera kolumny takie jak: Creator_id będącą kluczem głownym, Name będącą nazwą użytkownika/twórcy oraz Email będącą emailem użytkownika twórcy. We wszystkich tych kolumnach nie dopuszczamy wartości NULL.
- Tabela Creator_history zawiera historie oglądania użytkownika. Kluczem głównym są kolumny: Creator_id oraz Video_id. Tabela ta zawiera również kolumne z informacją o dacie odtworzenia danego filmiku przez użytkownika. W żadnej kolumnie nie dopuszczamy wartości NULL.
- Tabela Creator_subscription zawiera informację o subskrybcjach użytkowników. Zawiera ona 2 kolumny: Creator1_id oraz Creator2_id będące kluczem głównym. Przykładowo Creator1_id = 1 i Creator2_id = 2 oznacza, że użytkownik o Creator_id = 1 subskrybuje użytkownika o Creator_id = 2.

1.2 Opis relacji

- Tabela Videos jest połączona z tabelą Categories relacją wiele do wielu przez tabelę Video_Categories. Wynika to z tego, że każdy filmik może mieć wiele kateogrii oraz każda kateogria może zawierać wiele filmików.
- Tabela Videos jest połączona z tabelą Creators relacją jeden do jednego, gdyż jeden filmik może mieć jednego autora.
- Tabela Videos oraz Creators jest połączona relacją wiele do wielu przez tabelę Creator_history informującą o hisotrii oglądania filmików przez danych użytkowników. Jest to tak relacja gdyż filmik może zostać obejrzany przez wielu użytkowników oraz użytkownik może obejrzeć wiele filmików.
- Tabela Creators jest połączona z tabelą Creator_subscription relacją wiele do wielu gdyż użytkowik może subskrybować wielu innych użytkownników oraz danego użytkownika może subskrybować wielu innych użytkowników.

2 Skrypt tworzący bazę

2.1 Tworzenie bazy danych

Poniżej znajduje się skrypt tworzący bazę danych.

```
CREATE DATABASE pd2
GO

USE pd2
GO

CREATE TABLE Videos(
    Video_id INT NOT NULL,
    Title VARCHAR(50) NOT NULL,
    Length TIME(7) NOT NULL,
    Creator_id INT NOT NULL,
);

CREATE TABLE Videos_Categories(
    Video_id INT NOT NULL,
    Category_id INT NOT NULL,
);

CREATE TABLE Categories(
    Category_id INT NOT NULL,
    Category_name VARCHAR(50) NOT NULL,
);

CREATE TABLE Creators(
    Creator_id INT NOT NULL,
    Name VARCHAR(20) NOT NULL,
    Email VARCHAR(20) NOT NULL,
);

CREATE TABLE Creator_subscription(
    Creator1_id INT NOT NULL,
    Creator2_id INT NOT NULL,
);

CREATE TABLE Creator_history(
    Creator_id INT NOT NULL,
    Video_id INT NOT NULL,
    date DATE NOT NULL
);
```

Rysunek 2: skrypt 1/3

```
--klucze glowne

ALTER TABLE Videos
ADD
    CONSTRAINT PK_Videos_key PRIMARY KEY(Video_id)
GO

ALTER TABLE Categories
ADD
    CONSTRAINT PK_Categories_key PRIMARY KEY(Category_id)
GO

ALTER TABLE Videos_Categories
ADD
    CONSTRAINT PK_Videos_Categories_key PRIMARY KEY(Video_id, Category_id)
GO

ALTER TABLE Creators
ADD
    CONSTRAINT PK_Creators_key PRIMARY KEY(Creator_id)
GO

ALTER TABLE Creator_subscription
ADD
    CONSTRAINT PK_Creator_subscription_key PRIMARY KEY(Creator1_id, Creator2_id)
GO

ALTER TABLE Creator_history
ADD
    CONSTRAINT PK_Creator_history_key PRIMARY KEY(Creator_id, Video_id)
GO
```

Rysunek 3: skrypt 2/3

```
--klucze obce i relacje

ALTER TABLE Videos
ADD
    CONSTRAINT FK_Videos_Creators FOREIGN KEY (Creator_id) REFERENCES Creators(Creator_id)
GO

ALTER TABLE Videos_Categories
ADD
    CONSTRAINT FK_Videos_Categories_Videos FOREIGN KEY (Video_id) REFERENCES Videos(Video_id),
    CONSTRAINT FK_Videos_Categories_Categories FOREIGN KEY (Category_id) REFERENCES Categories(Category_id)
GO

ALTER TABLE Creator_subscription
ADD
    CONSTRAINT FK_Creator_subscription_Creators1 FOREIGN KEY (Creator1_id) REFERENCES Creators(Creator_id),
    CONSTRAINT FK_Creator_subscription_Creators2 FOREIGN KEY (Creator2_id) REFERENCES Creators(Creator_id)
GO

ALTER TABLE Creator_history
ADD
    CONSTRAINT FK_Creator_history_Creators FOREIGN KEY (Creator_id) REFERENCES Creators(Creator_id),
    CONSTRAINT FK_Creator_history_Videos FOREIGN KEY (Video_id) REFERENCES Videos(Video_id)
GO
```

Rysunek 4: skrypt 3/3

2.2 Wstawianie danych

Poniżej znajduje się skrypt wstawiający przykładowe dane do tabeli.

```
INSERT INTO Creators
VALUES(1,'PewDiePie', 'pewdie@hotmail.com'), (2,'SebastianSzwed', 'ugh_boy@gmail.com'),
      (3, 'Huston Jones', 'hj200@bestmail.com'), (4, 'Larry Wheels', 'larygary@u2.uk'),
      (5, 'Eric Rosen', 'ericche22@yt.com'), (6, 'Maciej Je', 'jemhoho@hihi.pl')

INSERT INTO Videos
VALUES(1,'Sebastian Szwed nie jest brzydki', '00:02:10', 2), (2,'My new car', '00:10:24', 1),
      (3,'Calf kicked by UFC Fighter', '00:11:32', 3), (4,'AM I READY', '00:28:20', 4),
      (5,'I was bribed to play the Wayward Queen Attack', '00:08:20', 5),
      (6,'Duze pieniądze za najdroższe jedzenie', '00:16:16', 6), (7,'fifa z widzami', '00:05:02', 2),
      (8,'Steki w Dubaju', '00:12:42', 6)

INSERT INTO Creator_subscription
VALUES (1,2), (1,3), (1,5), (2,1), (2,6),(3,5), (3,1),(4,1), (4,2), (5,4), (5,6), (6,2), (6,3)

INSERT INTO Categories
VALUES (1, 'Gaming'), (2,'Educational'), (3, 'Movie'), (4, 'Music'), (5, 'Action'), (6, 'Tutorial')

INSERT INTO Videos_Categories
VALUES (1,2), (2,1), (3, 2), (4,5), (5,6), (6,2), (7,6)

INSERT INTO Creator_history
VALUES (1,1, '2008-12-11'), (1,3, '2019-03-05'), (1,4, '2020-02-07'), (1,6, '2015-04-09'),
       (2, 1, '2018-09-05'), (2,5, '2008-03-03'), (2,2, '2017-09-03'), (2,6, '2019-11-06'),
       (3,2, '2015-06-09'), (3,4, '2019-07-08'), (3, 5, '2012-02-01'),
       (4,1, '2006-04-30'), (4,2, '2002-03-05'), (4,3, '2014-02-16'), (4, 6, '2019-08-04'),
       (5,1, '2019-07-24'), (5,2, '2009-09-24'), (5,3,'2020-03-27'), (5,4, '2018-05-18'),
       (6,2, '2015-06-19'), (6, 4, '2001-09-09'), (6, 7, '2001-10-09'), (6, 1, '2001-10-12')
```

Rysunek 5: wstawianie danych

2.3 Modyfikowanie danych

Poniżej znajduje się skrypt modyfikujący dane w wybranej tabeli.

```
UPDATE Creators
 SET Email = 'pewdie12@hotmail.com'
 WHERE Creator_id = 1


UPDATE Creators
 SET Name = 'Kidraman'
 WHERE Creator_id = 2

UPDATE Creators
 SET Name = 'Houston Jones'
 WHERE Creator_id = 3
```

Rysunek 6: modyfikacja danych

3 Indeksy

Poniżej znajdują się zaproponowane indeksy dla poszczególnych tabel.

```
--indeksy

CREATE CLUSTERED INDEX PK_Videos_new_key
 ON Videos(Title)
 GO

CREATE INDEX PK_Creator_history
 ON Creator_history(date)
 GO

CREATE CLUSTERED INDEX PK_Videos_Categories_new_key
 ON Videos_Categories(Category_id)
 GO

CREATE CLUSTERED INDEX PK_Creators_new_key
 ON Creators(name)
 GO
```

Rysunek 7: Indeksy

- Dla tabeli Videos został stworzony indeks typu clustered dla kolumny o nazwie Title, będącej tytułem filmu. Indeks jest typu clustered ponieważ główną operacją jaka będzie wykorzystywana w tabeli Videos będzie operacja wyszukiwania. Operacje wstawiania i usuwania będą stosunkowo rzadkie. Wyszukiwanie odbywa się zwykle przy pomocy tytułu filmu dlatego też wykorzystana została kolumna Title.
- Dla tabeli Creator_history wykorzytsany został indeks typu non clustered ponieważ przeważające będą operacje wstawiania do tej tabeli, a czasem potrzebne będzie również wyszukiwanie w hisotrii oglądania użytkowanika filmów obejrzanych w konkretnych dniach.
- Dla tabeli Videos_Categories wykorzystany został indeks typu clustered dla kolumny Category_name z tych samych względów co dla tabeli Videos. Najczęściej wykorzystywaną operacją będzie wyszukiwanie filmów w danej kategorii.
- Dla tabeli Creators wykorzystany został indeks typu clustered dla kolumny name, ponieważ użytkowicy częsciej będą wyszukiwani niż dodawani czy też usuwani, a wyszukiwanie najczęsciej odbywa się po nazwie użytkowanika.

4 SQL SELECT

1. Łączna długość filmów obejrzanych przez użytkowników na danym kanale. Wynik polecenia:

```
SELECT c.Name as viewer, c2.Name as creator, time
FROM(
    SELECT ch.Creator_id AS viewer, v.Creator_id as creator,
    CONVERT(VARCHAR(8),DATEADD(ms, SUM(DATEDIFF(ms, '00:00:00', v.Length)), '00:00:00'),108) as time
    FROM Creator_history CH
    JOIN Videos v on v.Video_id = CH.Video_id
    GROUP BY V.Creator_id, ch.Creator_id) gr
JOIN Creators c on gr.viewer = c.Creator_id
JOIN Creators c2 on gr.creator = c2.Creator_id
```

Rysunek 8: sql select 1/5

	viewer	creator	time
1	PewDiePie	Kidraman	00:02:10
2	PewDiePie	Houston Jones	00:11:32
3	PewDiePie	Larry Wheels	00:28:20
4	PewDiePie	Maciej Je	00:16:16
5	Kidraman	PewDiePie	00:10:24
6	Kidraman	Kidraman	00:02:10
7	Kidraman	Eric Rosen	00:08:20
8	Kidraman	Maciej Je	00:16:16
9	Houston Jones	PewDiePie	00:10:24
10	Houston Jones	Larry Wheels	00:28:20
11	Houston Jones	Eric Rosen	00:08:20
12	Larry Wheels	PewDiePie	00:10:24
13	Larry Wheels	Kidraman	00:02:10
14	Larry Wheels	Houston Jones	00:11:32
15	Larry Wheels	Maciej Je	00:16:16
16	Eric Rosen	PewDiePie	00:10:24
17	Eric Rosen	Kidraman	00:02:10
18	Eric Rosen	Houston Jones	00:11:32
19	Eric Rosen	Larry Wheels	00:28:20
20	Maciej Je	PewDiePie	00:10:24
21	Maciej Je	Kidraman	00:07:12
22	Maciej Je	Larry Wheels	00:28:20

Rysunek 9: sql select rezultat 1/5

I sprawdźmy na przykład wiersz nr. 21. Maciej Je ma Creator_id = 6 czyli interesuje nas ostatni wiersz na poniższym rysunku. Obejrzał on filmy o Viedo_id = 2, 4, 7, 1. Filmy Kidramana to filmy o Video_id = 1, 7, a ich łączna długość wynosi 7:12 min.

```
INSERT INTO Creator_history
VALUES (1,1, '2008-12-11'), (1,3, '2019-03-05'), (1,4, '2020-02-07'), (1,6, '2015-04-09'),
       (2, 1, '2018-09-05'), (2,5, '2008-03-03'), (2,2, '2017-09-03'), (2,6, '2019-11-06'),
       (3,2, '2015-06-09'), (3,4, '2019-07-08'), (3, 5, '2012-02-01'),
       (4,1, '2006-04-30'), (4,2, '2002-03-05'), (4,3, '2014-02-16'), (4, 6, '2019-08-04'),
       (5,1, '2019-07-24'), (5,2, '2009-09-24'), (5,3,'2020-03-27'), (5,4, '2018-05-18'),
       (6,2, '2015-06-19'), (6, 4, '2001-09-09'), (6, 7, '2001-10-09'), (6, 1, '2001-10-12')
```

Rysunek 10: insert

2. Wyszukać użytkowników, którzy obserwują kanał, ale obejrzeli na nim mniej niż trzy filmy. Wynik polecenia:

```
--Wyszukać użytkowników, którzy obserwują kanał, ale obejrzeli na nim mniej niż trzy filmy.

SELECT C.Name as viewer, C2.Name as subscibesTo, GR.liczba as ilosc_filmow
FROM(
    SELECT c.Creator_id as viewer, C2.Creator_id as subscribesTo, COUNT(*) as liczba
    FROM Creator_subscription cs
    JOIN Creators C on c.Creator_id = cs.Creator1_id
    JOIN Creators C2 on c2.Creator_id = cs.Creator2_id
    JOIN Creator_history ch on C.Creator_id = ch.Creator_id
    JOIN Videos V ON V.Video_id = CH.Video_id AND C2.Creator_id = V.Creator_id
    GROUP BY C.Creator_id, C2.Creator_id) GR
JOIN Creators c ON GR.viewer = C.Creator_id
JOIN Creators c2 ON GR.subscribesTo = C2.Creator_id
WHERE GR.liczba < 3
```

Rysunek 11: sql select 2/5

	viewer	subscibesTo	ilosc_filmow
1	Kidraman	PewDiePie	1
2	Houston Jones	PewDiePie	1
3	Larry Wheels	PewDiePie	1
4	PewDiePie	Kidraman	1
5	Larry Wheels	Kidraman	1
6	Maciej Je	Kidraman	2
7	PewDiePie	Houston Jones	1
8	Eric Rosen	Larry Wheels	1
9	Houston Jones	Eric Rosen	1
10	Kidraman	Maciej Je	1

Rysunek 12: sql select rezultat 2/5

Możemy spojrzeć 3 wiersz w powyższym rysunku. Widzimy, że Larry Wheels objerzał jeden film na kanale PewDiePie i zgadza się to z rysunekiem nr. 10 gdyż film PewDiePie ma Viedo_id = 2 a Larry Wheels ma Creator_id = 4. Na rysunku nr, 10 widzimy dodany rekord (4, 2, '2002-03-05') a Larry Wheels subskrybuje PewDiePie zgodnie z Rysunkiem nr. 5 gdzie przy Creator_subscription mamy (4,1) gdzie 4 to Creator_id Larrego Wheelsa a 4 to Creator_id PewDiePie to 1.

3. Dla każdego gatunku wyszukać film cieszący się największą popularnością. Wynik zapytania:

```
--Dla każdego gatunku wyszukać film cieszący się największą popularnością

SELECT c.Category_name as CategoryName, MAX(v.Title) as VideoTitle,
MAX(gr.liczba_odtworzen) as TimesPlayed
FROM(
    SELECT vc.Category_id as category_id, v.Video_id as video_id,
    COUNT(*) as liczba_odtworzen
    FROM Videos_Categories vc
    JOIN Categories c on vc.Category_id = c.Category_id
    JOIN Videos v on vc.Video_id = v.Video_id
    JOIN Creator_history cr on v.Video_id = cr.Video_id
    GROUP BY vc.Category_id, v.Video_id) GR
JOIN Categories C on c.Category_id = gr.category_id
JOIN Videos v on v.Video_id = gr.video_id
GROUP BY c.Category_name
```

Rysunek 13: sql select 3/5

	CategoryName	VideoTitle	TimesPlayed
1	Action	AM I READY	4
2	Educational	Sebastian Szwed nie jest brzydki	5
3	Gaming	My new car	5
4	Tutorial	I was bribed to play the Wayward Queen Attack	2

Rysunek 14: sql select rezultat 3/5

Weźmy na przykład wiersz nr. 2. Tytuł filmu to Sebastian Szwed nie jest brzydki, został oberjrzany 5 razy i jest w kategorii Education. Film ten ma Video_id = 1 i liczba odtworzeń zgadza się z rysunkiem nr. 10. Zauważmy, że jest również najczęściej odtwarzanym filmem w kategorii Education gdyż inne filmy w tej kategorii mają odpowiednio 2 i 3 wyświetlenia zgodnie z rysunkiem nr. 5.

4. Dla każdego kanału wyświetlić stosunek liczby wyświetleń do liczby opublikowanych filmów. Wynik zapytania:

```
--Dla każdego kanału stosunek liczby wyświetleń do liczby opublikowanych filmów

SELECT C.Name AS name, gr.Stosunek as ratio
FROM (
    SELECT c.Creator_id, COUNT(*)/MAX(gr.LiczbaFilmow) AS Stosunek
    FROM Creators c
    JOIN Videos v on v.Creator_id = c.Creator_id
    JOIN Creator_history ch on ch.Video_id = v.Video_id
    JOIN (
        SELECT v.Creator_id as creator_id, COUNT(*) AS LiczbaFilmow
        FROM Videos V
        GROUP BY V.Creator_id) gr on gr.creator_id = c.Creator_id
    GROUP BY c.Creator_id) gr
JOIN Creators C ON C.Creator_id = GR.Creator_id
ORDER BY GR.Stosunek DESC
```

Rysunek 15: sql select 4/5

	name	ratio
1	PewDiePie	5
2	Larry Wheels	4
3	Kidraman	3
4	Houston Jones	3
5	Eric Rosen	2
6	Maciej Je	1

Rysunek 16: sql select rezultat 4/5

Weźmy wiersz 3 i Kidramana. Posiada on 2 filmy zgodnie z rysunkiem nr. 5 ich Video_id to 1 i 7. Zgodnie z rysunkiem nr. 10 jego filmy zostały obejrzane 6 razy. Czyli stosunek się zgadza.

5. Wyświetlić filmy, które są dostępne w ramach kanału, który ma najwięcej subskrybentów. Wynik zapytania:

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

SELECT C.Name, V.Title, GR.SubCount
FROM Creators C
JOIN (
    SELECT TOP 1 *
    FROM (
        SELECT CS.Creator2_id AS CreatorId, COUNT(*) AS SubCount
        FROM Creator_subscription CS
        GROUP BY CS.Creator2_id) GR
    ORDER BY GR.SubCount DESC) GR ON C.Creator_id = GR.CreatorId
JOIN Videos V ON V.Creator_id = GR.CreatorId
```

Rysunek 17: sql select 5/5

	Name	Title	SubCount
1	Kidraman	fifa z widzami	3
2	Kidraman	Sebastian Szwed nie jest brzydki	3

Rysunek 18: sql select rezultat 5/5

Kidraman jest najczęściej subskrybowanym kanałem z 3 subskrybcjami co wynika z rysunku nr. 5 i posiada on dwa filmy.

5 Procedura składowana

Poniżej znajduje się procedura przenosząca starsze niż n dni rekordy z tabeli Creator_history do nowopowstałej tabeli HistoryArchive. Procedura ta jedocześnie modyfikuje nowo dodaną kolumna ViewsCnt w tabeli Videos. Poniżej kod:

```
--PROCEDURA SKLADNIOWA

USE PD2
GO
    if OBJECT_ID('HistoryArchive') IS NOT NULL  drop table HistoryArchive;
    select * into HistoryArchive from Creator_history where 0=1;

    ALTER TABLE Videos
    ADD
        ViewsCnt int
    GO

    ALTER TABLE HistoryArchive
    ADD
        CONSTRAINT PK_HistoryArchive_key PRIMARY KEY(Creator_id, Video_id)
    GO
GO

IF OBJECT_ID('arch') IS NOT NULL
    DROP PROC arch;
GO
CREATE PROCEDURE arch
    @DaysCount int
AS
BEGIN
    --set Identity_insert dbo.HistoryArchive ON;

    DECLARE @archiveDate  datetime;
    DECLARE @currentDate  datetime;
    DECLARE @archivedOrderIDs TABLE(Creator_id int, Video_id int);

    SET @currentDate = getdate();
    SET @archiveDate = dateadd(dd, -@DaysCount, @currentDate);

    IF (SELECT COUNT(*) FROM Creator_history WHERE date <= @archiveDate) = 0
    BEGIN
        print 'There are no records to archive... ';
        RETURN;
    END;
```

Rysunek 19: procedura 1/2

```
    print 'Archiving records older than ' + CONVERT(varchar(50),@DaysCount) + ' days';

    set transaction isolation level serializable;

    begin transaction
    begin try

        INSERT INTO HistoryArchive([Creator_id], [Video_id], [date])
        OUTPUT INSERTED.Creator_id, INSERTED.Video_id INTO @archivedOrderIDs
        SELECT * FROM Creator_history WHERE date <= @archiveDate;

        DELETE FROM Creator_history
        FROM Creator_history CH JOIN @archivedOrderIDs AO
        ON CH.Creator_id = AO.Creator_id AND CH.Video_id = AO.Video_id;

        UPDATE Videos
        SET  Videos.ViewsCnt = gr.numb
        FROM (SELECT COUNT(*) as numb, CH.Video_id AS video_id FROM Creator_history CH GROUP BY CH.Video_id) GR
        WHERE Videos.Video_id = gr.video_id

        UPDATE Videos
        SET Videos.ViewsCnt = 0
        WHERE Videos.ViewsCnt IS NULL

    commit transaction
    END TRY

    BEGIN CATCH
        ROLLBACK transaction

    END CATCH

END
```

Rysunek 20: procedura 2/2

W czasie tej procedury używamy tranzakcji o isolation level typu serializable czyli zapewiamy, że żadne dane nie zostaną zmienione czy też odczytane w czasie działania tranzakcji.

	Creator_id	Video_id	date
1	6	4	2001-09-09
2	6	7	2001-10-09
3	6	1	2001-10-12
4	4	2	2002-03-05
5	4	1	2006-04-30
6	2	5	2008-03-03
7	1	1	2008-12-11
8	5	2	2009-09-24
9	3	5	2012-02-01
10	4	3	2014-02-16
11	1	6	2015-04-09
12	3	2	2015-06-09
13	6	2	2015-06-19
14	2	2	2017-09-03
15	5	4	2018-05-18
16	2	1	2018-09-05
17	1	3	2019-03-05
18	3	4	2019-07-08
19	5	1	2019-07-24
20	4	6	2019-08-04
21	2	6	2019-11-06
22	1	4	2020-02-07
23	5	3	2020-03-27

Rysunek 21: Przed procedurą

Następnie wywołujemmy procedurę EXEC arch 1000; aby przenieść wszystkie rekordy starsze niż 1000 dni do tabeli HistoryArchive i usunąć je z tabeli Creator_history a następnie uaktualnić kolumnę ViewsCnt w tabeli Videos. Tabela Creator_history po procedurze:

	Creator_id	Video_id	date
1	2	1	2018-09-05
2	1	3	2019-03-05
3	3	4	2019-07-08
4	5	1	2019-07-24
5	4	6	2019-08-04
6	2	6	2019-11-06
7	1	4	2020-02-07
8	5	3	2020-03-27

Rysunek 22: Po procedurze

A poniżej tabela Videos po zastosowaniu procedury:

	Video_id	Title	Length	Creator_id	ViewsCnt
1	4	AM I READY	00:28:20.0000000	4	2
2	3	Calf kicked by UFC Fighter	00:11:32.0000000	3	2
3	6	Duze pieniądze za najdroższe jedzenie	00:16:16.0000000	6	2
4	7	fifa z widzami	00:05:02.0000000	2	0
5	5	I was bribed to play the Wayward Queen Attack	00:08:20.0000000	5	0
6	2	My new car	00:10:24.0000000	1	0
7	1	Sebastian Szwed nie jest brzydki	00:02:10.0000000	2	2
8	8	Steki w Dubaju	00:12:42.0000000	6	0

Rysunek 23: Po procedurze