BAZA DANYCH

„KINO”

DOKUMENTACJA

Wykonanie: Patrycja Baczewska

1. Zawartość dokumentacji

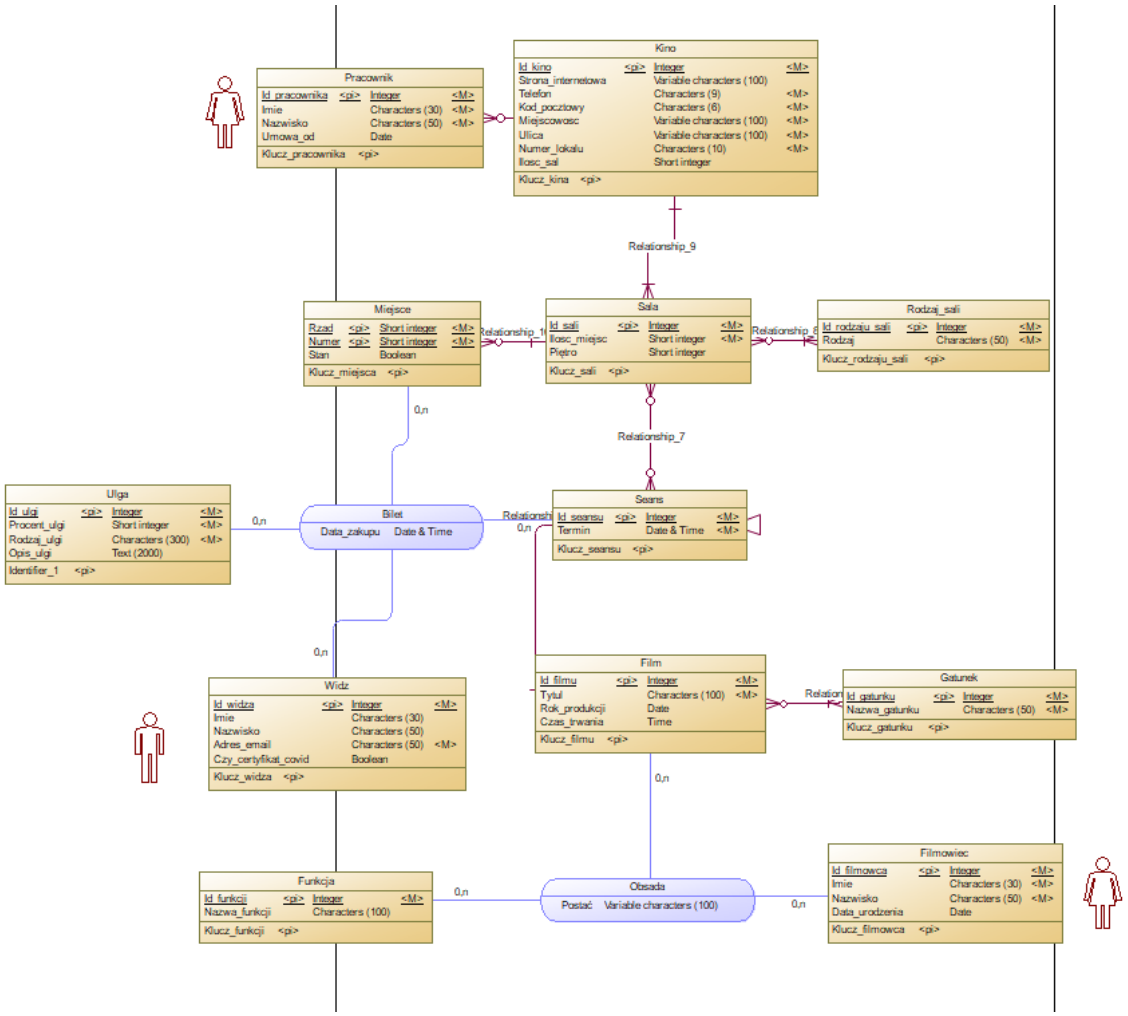
- Opis słowny bazy danych
- Model konceptualny bazy danych
- Model fizyczny relacyjnej bazy danych
- Opis oraz listing SQL poniższych elementów:

    - 3 widoki (w tym jeden widok zmaterializowany)
    - 3 procedury wyzwalane (triggery)
    - 3 procedury wbudowane (minimum 1 mechanizm kursora i transakcji)
    - 3 funkcje wbudowane (minimum 1 mechanizm kursora i transakcji)

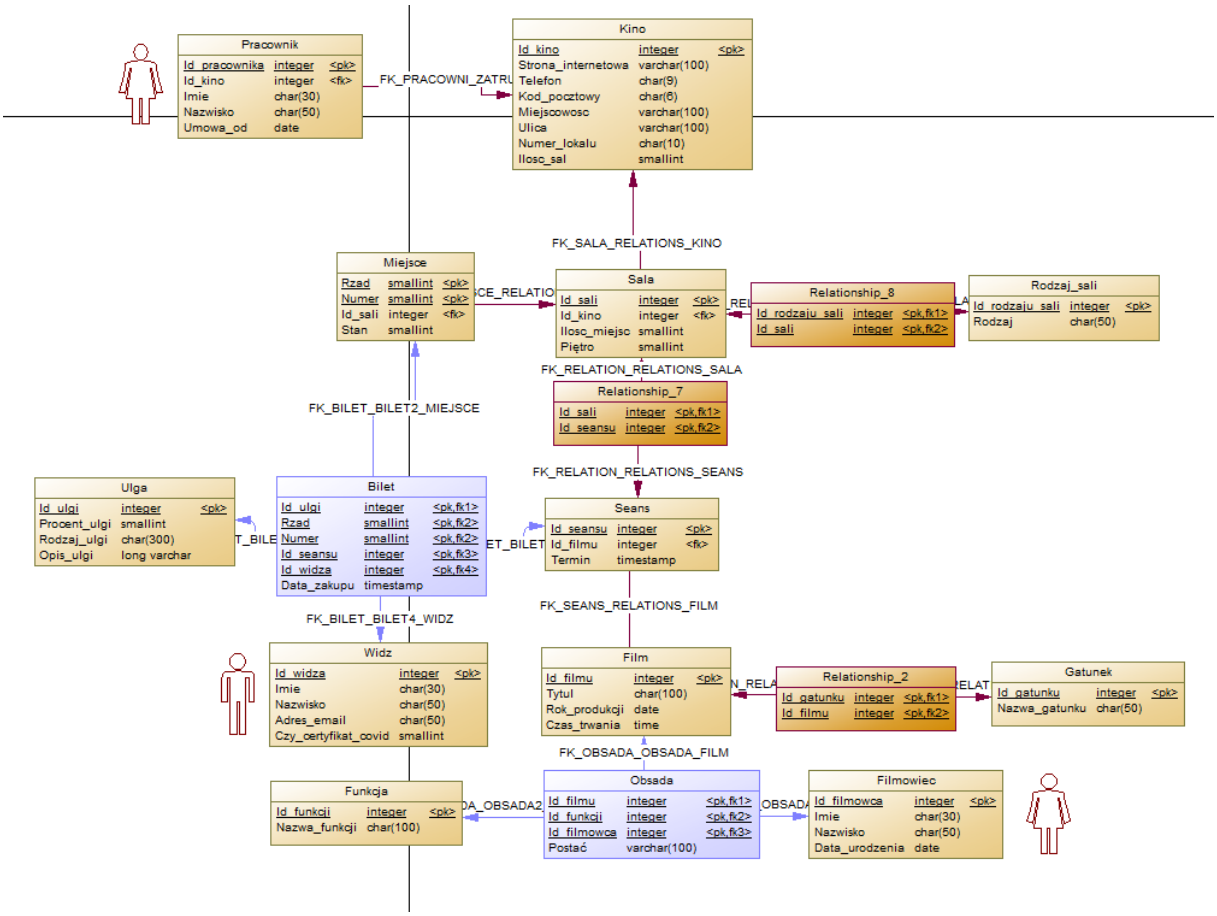
2. Opis słowny bazy danych

Baza danych „Kino” przechowuje kompletne informacje o sieci kin oraz ich działaniu. Zawiera w sobie łącznie 12 tabel oraz 2 encji. Tabela „Kino” składa się z następujących kolumn: „Id_kino” (klucz główny), „Strona_internetowa”, „Telefon”, „Kod_pocztowy”, „Miejscowość”, „Ulica”, „Numer_lokalu”, „Ilość_sal”. Tabela kino połączona jest z tabelą „Pracownik” relacją jeden do wielu – każdy z pracowników jest zatrudniony w określonym kinie. Tabela „Pracownik” złożona jest z kolumn: „Id_pracownika” (klucz główny), „Imie”, „Nazwisko”, „Umowa_od”. Tabela „Kino” jest również połączona relacją jeden do wielu z tabelą „Sala”, składającą się z kolumn: „Id_Sali” (klucz główny), „Ilosc_miejsc”, „Piętro”. Tabela „Sala” połączona jest relacją wiele do wielu z tabelą „Rodzaj_sali", która zawiera w sobie kolumny: „Id_rodzaju_Sali” (klucz główny) oraz „Rodzaj”. Tabela „Sala” ma również relację wiele do wielu z tabelą Seans, która składa się z kolumn: „Id_seansu” (klucz główny) oraz „Termin”. Ostatnia relacja tabeli „Sala” to połączenie jeden do wielu z tabelą „Miejsce”. Tabela „Miejsce” zawiera następujące kolumny: „Rząd” (klucz główny 1), „Numer” (klucz główny 2), „Stan”. Tabela „Film” składa się z kolumn: „Id_filmu” (klucz główny), „Tytul”, „Rok_produkcji”, „Czas trwania”. Tabel film ma relację jeden do wielu z tabelą „Seans” oraz relację wiele do wielu z tabelą „Gatunek”, złożoną z kolumn: „Id_gatunku” (klucz główny) oraz „Nazwa gatunku”. W bazie danych występuje jeszcze tabela „Funkcja” o następujących kolumnach: „Id_funkcji” (kluz główny), „Nazwa_funkcji”, tabela „Filmowiec” złożona z kolumn: „Id_filmowca” (klucz główny), „Imie”, „Nazwisko”, „Data_urodzenia”, tabela „Widz” – kolumny: „Id_widza” (klucz główy), „Imie”, „Nazwisko”, „Adres_email”, „Czy_certyfikat_covid”, tabela „Ulga” – kolumny: „Id_ulgi” (klucz główny), „Procent_ulgi”, „Rodzaj_ulgi”, „Opis_ulgi”. Pomiędzy tabelami „Film”, „Funkcja” oraz „Filmowiec” określona jest encja „Obsada” z atrybutem „Postac”. W bazie danych występuje też encja „Bilet” będąca połączeniem tabeli „Miejsce”, „Seans”, „Ulga” oraz „Widz”, występuje w niej atrybut „Data_zakupu”.

3. Model konceptualny bazy danych



4. Model fizyczny relacyjnej bazy danych



5. Opis słowny zawartości tabel

Lp.	Nazwa tabeli	Opis zawartości tabeli
1.	Kino	Tabela „Kino” zawiera w sobie szczegóły kina – adres, dane kontaktowe, ilość sal.
2.	Pracownik	Tabela „Pracownik” przechowuje dane zatrudnionych pracowników – imię oraz nazwisko, w którym kinie pracują oraz od kiedy mają podpisaną umowę.
3.	Sala	Tabela „Sala” określa id każdej sali kinowej, zawiera w sobie informację o ilości miejsc, piętrze na którym jest umiejscowiona oraz w jakim znajduje się kinie.
4.	Rodzaj_sali	Tabela „Rodzaj_sali" przechowuje id rodzaju Sali oraz słownie opisany rodzaj który reprezentuje.
5.	Miejsce	Tabela „Miejsce” zawiera w sobie spis dostępnych miejsc w danej sali, przechowywana jest również informacja o stanie miejsca – czy jest ono wolne czy zajęte.
6.	Seans	Tabela „Seans” przydziela id każdemu połączeniu id filmu oraz terminu.
7.	Film	Tabela „Film” przechowuje id filmu, jego tytuł, rok produkcji oraz czas trwania.
8.	Gatunek	Tabela „Gatunek” składa się z id gatunku oraz słownej nazwy gatunku – tabela służy do określania filmów.
9.	Filmowiec	Tabela „Filmowiec” zawiera spis ludzi kina nadając każdej z nich id filmowca, przechowując również imię i nazwisko oraz datę urodzenia.
10.	Funkcja	Tabela „Funkcja” jest to id funkcji oraz jej nazwa – służy do określenia funkcji filmowca w filmie.
11.	Widz	Tabela „Widz” przechowuje dane o widzach kupujących bilety – znajduje się tam imię i nazwisko oraz e-mail, a także pole określające czy widz posiada certyfikat COVID. Każdy widz ma swoje id widza.
12.	Ulga	Tabela „Ulga” przechowuje w sobie id ulgi, procent zniżki jaki ta ulga zapewnia, słowną nazwę tej ulgi oraz jej opis.

6. Widok 1: „Ilosc_biletow”

Widok „Ilosc_biletow” zlicza bilety każdego z widzów w bazie kina. Składa się z kolumn „Id_widza”, „Imie”, „Nazwisko” „Adres_email”, „Ile_biletow”. W ostatnie kolumnie sumują się wszystkie bilety zakupione przez określonego widza.

Listing widoku:

```
ALTER VIEW "Pati"."Ilosc_biletow"

AS

SELECT Widz.Id_widza, Widz.Imie, Widz.Nazwisko,
Widz.Adres_email, count(Bilet.Id_widza) Ile_biletow

FROM Bilet

LEFT OUTER JOIN Widz

ON Widz.Id_widza=Bilet.Id_widza

GROUP BY Widz.Id_widza, Widz.Imie, Widz.Nazwisko,
Widz.Adres_email

ORDER BY Ile_biletow DESC
```

Wywołanie widoku:

```
SELECT *

FROM Ilosc_biletow
```

7. Widok 2 (zmaterializowany): „Ilosc_zatrudnionych”

Widok „Ilosc_zatrudnionych” zlicza przedstawia ile jest osób zatrudnionych w każdym z kin znajdującym się w bazie danych. Widoczne są kolumny „Id_kino”, „Telefon”, „Kod_pocztowy”, „Miejscowosc”, „Ulica”, „Numer_lokalu”, „Ilu_pracownikow”. Ostatnia kolumna przedstawia wartość liczbową określającą liczbę zatrudnionych osób w wybranym kinie.

Listing widoku:

```
CREATE MATERIALIZED VIEW "Pati"."Ilosc_zatrudnionych"()

IN "system" AS

SELECT Pati.Kino.Id_kino, Pati.Kino.Telefon,
Pati.Kino.Kod_pocztowy, Pati.Kino.Miejscowosc,

Pati.Kino.Ulica, Pati.Kino.Numer_lokalu,
count(Pati.Pracownik.Id_pracownika) Ilu_pracownikow

FROM Kino

LEFT OUTER JOIN Pracownik

ON Kino.Id_kino=Pracownik.Id_kino

GROUP BY Pati.Kino.Id_kino, Pati.Kino.Telefon,
Pati.Kino.Kod_pocztowy, Pati.Kino.Miejscowosc,

Pati.Kino.Ulica, Pati.Kino.Numer_lokalu

HAVING count(Pati.Pracownik.Id_pracownika) > 0

ORDER BY count(Pati.Pracownik.Id_pracownika), Kino.Id_kino
```

Wywołanie widoku:

```
SELECT *

FROM Ilosc_zatrudnionych
```

8. Widok 3: „Repertuar_kin”

Widok „Repertuar_kin” wyświetla seanse grane aktualne we wszystkich kinach. Widok ten składa się z następujących kolumn: „Id_kino”, „Tytul”, „Nazwa_gatunku”, „Czas_trwania”, „Termin”, „Id_sali". Ułatwia to sortowanie seansów np. po czasie trwania czy gatunku filmu.

Listing widoku:

```
ALTER VIEW "Pati"."Repertuar_kin"

AS

SELECT Kino.Id_kino, Film.Tytul, Gatunek.Nazwa_gatunku,
Film.Czas_trwania, Seans.Termin, Sala.Id_sali

FROM Seans

LEFT OUTER JOIN Film

ON Film.Id_filmu=Seans.Id_filmu

LEFT OUTER JOIN Relationship_7

ON Seans.Id_seansu=Relationship_7.Id_seansu

LEFT OUTER JOIN Sala

ON Sala.Id_sali=Relationship_7.Id_sali

LEFT OUTER JOIN Kino

ON Sala.Id_kino=Kino.Id_kino

LEFT OUTER JOIN Relationship_2

ON Film.Id_filmu=Relationship_2.Id_filmu

LEFT OUTER JOIN Gatunek

ON Gatunek.Id_gatunku=Relationship_2.Id_gatunku

GROUP BY Kino.Id_kino, Film.Tytul, Gatunek.Nazwa_gatunku,
Film.Czas_trwania, Seans.Termin, Sala.Id_sali

ORDER BY Seans.Termin ASC
```

Wywołanie widoku:

```
SELECT *
FROM Repertuar_kin
```

9. Widok 4: „Wolne_miejsca”

Widok „Wolne_miejsca” zlicza pozostałe wolne miejsca na seanse. Widok złożony jest z tabel „Id_seansu”, „Tytul”, „Termin”, „Ilosc_miejs”. Ostatnia kolumna jest to suma pozostałych miejsc na wybrany seans. Widok ten wyświetla również wiersze w których nie pozostały żadne wolne miejsca.

Listing widoku:

```
ALTER VIEW "Pati"."Wolne_miejsca"()
AS
SELECT Seans.Id_seansu, Film.Tytul, Seans.Termin,
count(Miejsce.Numer) AS Ilosc_miejsc
FROM Seans
LEFT OUTER JOIN Relationship_7
ON Seans.Id_seansu = Relationship_7.Id_seansu
LEFT OUTER JOIN Sala
ON Sala.Id_sali = Relationship_7.Id_sali
LEFT OUTER JOIN Miejsce
ON Sala.Id_sali = Miejsce.Id_sali
LEFT OUTER JOIN Film
ON Seans.Id_filmu = Film.Id_filmu
WHERE Miejsce.Stan = 0
GROUP BY Seans.Id_seansu, Seans.Termin, Film.Tytul
```

Wywołanie widoku:

```
SELECT *
FROM Wolne_miejsca
```

10. Procedura 1 (z użyciem kursora): „Aktualizacja_seansów”

Procedura „Aktualizacja_seansów” przeprowadza operację usuwania nieaktualnych seansów. Sprawdzana jest data wyświetlania każdego z nich, następnie seanse które już się odbyły są usuwane z bazy, a pozostają jedynie te aktualne. Procedura nie przyjmuje żadnych argumentów.

Listing procedury:

```
ALTER PROCEDURE "Pati"."Aktualizacja_seansow"()
BEGIN
    FOR petla AS kursor CURSOR FOR
        SELECT Termin, Id_seansu as Ids
        FROM Seans
        DO
        IF Termin < getdate() THEN
            DELETE FROM Seans
            WHERE Id_seansu = Ids
        ENDIF;
    END FOR;
END
```

Wywołanie procedury:

```
CALL Aktualizacja_seansow()
```

11. Procedura 2: „Nowy_widz”

Procedura „Nowy_widz” ułatwia dodawanie nowego widza do bazy danych. Jeśli nastąpi powtórzenie numeru id widza lub powtórzenie adresu e-mail procedura nie wykona się. W argumentach należy podać „Id_widza”, „Imie”, „Nazwisko”, „Adres_email”, „Czy_certyfikat_covid” – wartości 0 i 1;

Listing procedury:

```
ALTER PROCEDURE "Pati"."Nowy_widz"(
@Id_widza INTEGER,
@Imie CHAR(30),
@Nazwisko CHAR(50),
@Adres_email CHAR(50),
@Czy_certyfikat_covid SMALLINT)
AS
BEGIN
    DECLARE @czy_powtorzenie INTEGER
    SELECT @czy_powtorzenie = COUNT(*)
    FROM Widz
    WHERE Widz.Id_widza = @Id_widza OR Widz.Adres_email =
@Adres_email
    IF @czy_powtorzenie = 0
    BEGIN
        INSERT INTO Widz (Id_widza, Imie, Nazwisko,
Adres_email, Czy_certyfikat_covid)
        VALUES (@Id_widza, @Imie, @Nazwisko, @Adres_email,
@Czy_certyfikat_covid)
    END
END
```

Wywołanie procedury:

```
CALL Nowy_widz(4, ‘Patrycja’, ‘Baczewska’, ‘email@gmail.com, 0)
```

12. Procedura 3 (z użyciem transakcji): „Zmiana_miejsca”

Procedura „Zmiana_miejsca” umożliwia zmianę miejsca wybranego widza na określony seans pod warunkiem, że wybierane miejsce nie jest zajęte. Procedura przyjmuje w argumentach „Id_widza”, „Id_seansu”, „Rzad”, „Numer”, „Sala”. Transakcja polega na wykonaniu poszczególnych czynności: aktualizacji w szczegółach biletu nowego miejsca, ustawienia stanu starego miejsca na wolne, ustawienia stanu nowego miejsca na zajęte.

Listing procedury:

```
ALTER PROCEDURE "Pati"."Zmiana_miejsca"(
@Id_widza INTEGER,
@Id_seansu INTEGER,
@Rzad SMALLINT,
@Numer SMALLINT,
@Sala SMALLINT)
AS
BEGIN
     DECLARE @ilosc_wolnych INTEGER
    DECLARE @czy_wybrane_wolne INTEGER
    SET @ilosc_wolnych = (  SELECT count(Miejsce.Numer) AS
Ilosc_miejsc
                            FROM Seans
                            LEFT OUTER JOIN Relationship_7
                            ON Seans.Id_seansu =
Relationship_7.Id_seansu
                            LEFT OUTER JOIN Sala
                            ON Sala.Id_sali =
Relationship_7.Id_sali
                            LEFT OUTER JOIN Miejsce
```

```
                            ON Sala.Id_sali = Miejsce.Id_sali
                            WHERE Miejsce.Stan = 0 AND
Seans.Id_seansu=@Id_seansu
                            GROUP BY Seans.Id_seansu,
Seans.Termin )
    SET @czy_wybrane_wolne = (  SELECT Miejsce.Stan
                                FROM Seans
                                LEFT OUTER JOIN Relationship_7
                                ON Seans.Id_seansu =
Relationship_7.Id_seansu
                                LEFT OUTER JOIN Sala
                                ON Sala.Id_sali =
Relationship_7.Id_sali
                                LEFT OUTER JOIN Miejsce
                                ON Sala.Id_sali =
Miejsce.Id_sali
                                WHERE Miejsce.Rzad = @Rzad AND
Miejsce.Numer = @Numer AND Seans.Id_seansu=@Id_seansu )
    BEGIN TRAN transakcja


        UPDATE Bilet
        SET Bilet.Rzad = @Rzad, Bilet.Numer = @Numer
        WHERE Bilet.Id_widza = @Id_widza AND Bilet.Id_seansu =
@Id_seansu AND Bilet.Rzad = @Rzad AND Bilet.Numer = @Numer


        UPDATE Miejsce
        SET Miejsce.Stan = 1
        WHERE Miejsce.Rzad = @Rzad AND Miejsce.Numer = @Numer
AND Miejsce.Id_sali = @Id_sali


        IF @czy_wybrane_wolne = 0
```

```
            BEGIN
                ROLLBACK TRAN transakcja
            END
        ELSE
            BEGIN
                COMMIT TRAN
            END
END
```

Wywołanie procedury:

```
CALL Zmiana miejsca(2, 3, 1, 2, 4)
```

13. Funkcja 1 (z użyciem transakcji): „Anulowanie_biletu”

Funkcja „Anulowanie_biletu” pozwala w prosty sposób anulować bilet, otrzymując w argumentach „Id_widza”, „Id_seansu”, „Rzad”, „Numer”. Transakcja polega na zaktualizowaniu stanu miejsca zajmowanego przez widza i ustawieniu jego stanu na 0 – wolne, następnie usuwany jest odpowiedni rekord z tabeli bilet.

Listing funkcji:

```
ALTER FUNCTION "Pati"."Anulowanie_biletu"(
@Id_widza INTEGER,
@Id_seansu INTEGER,
@Rzad INTEGER,
@Numer INTEGER)
RETURNS CHAR(30)
AS
BEGIN
    DECLARE @czy_usunieto CHAR(30)
    DECLARE @sala INTEGER
    DECLARE @czy_miejsce INTEGER
    SET @czy_usunieto = 'NIE'
    SET @sala = (   SELECT TOP 1 Id_sali
                    FROM Relationship_7
                    WHERE Relationship_7.Id_seansu =
@Id_seansu
                    ORDER BY Id_seansu ASC)
    SET @czy_miejsce = (SELECT Miejsce.Stan
                        FROM Seans
                        LEFT OUTER JOIN Relationship_7
```

```
                        ON Seans.Id_seansu =
Relationship_7.Id_seansu
                        LEFT OUTER JOIN Sala
                        ON Sala.Id_sali =
Relationship_7.Id_sali
                        LEFT OUTER JOIN Miejsce
                        ON Sala.Id_sali = Miejsce.Id_sali
                        WHERE Miejsce.Rzad = @Rzad AND
Miejsce.Numer = @Numer AND Seans.Id_seansu=@Id_seansu)
    BEGIN TRAN
    SAVE TRAN transakcja


        UPDATE Miejsce
        SET Miejsce.Stan = 0
        WHERE Miejsce.Rzad = @Rzad AND Miejsce.Numer = @Numer
AND Miejsce.Id_sali = @sala


        DELETE FROM Bilet
        WHERE Bilet.Rzad = @Rzad AND Bilet.Numer = @Numer AND
Bilet.Id_widza=@Id_widza AND Bilet.Id_seansu = @Id_seansu


        SET @czy_usunieto = 'TAK'


        IF @czy_miejsce = 1
            BEGIN
                ROLLBACK TRAN transakcja
            END


    COMMIT TRAN
     RETURN @czy_usunieto
```

```
END
```

Wywołanie funkcji:

```
SELECT Anulowanie_biletu(3, 2, 4, 5)
```

14. Funkcja 2: „Najmniej_popularny_film”

Funkcja „Najmniej_popularny_film” zlicza wolne miejsca na poszczególnych filmach i wybiera film na który zostało kupione najmniej miejsc. Tytuł ten wyświetlany jest jako najmniej popularny film.

Listing funkcji:

```
ALTER FUNCTION "Pati"."Najmniej_popularny_film"()
RETURNS CHAR(50)
NOT DETERMINISTIC
BEGIN
     DECLARE "Najmniej popularny film" CHAR(50);
    SELECT TOP 1 Wolne_miejsca.Tytul INTO "Najmniej popularny
film"
    FROM Wolne_miejsca
    WHERE Ilosc_miejsc = (  SELECT max(Ilosc_miejsc)
                            FROM Wolne_miejsca);
     RETURN "Najmniej popularny film";
END
```

Wywołanie funkcji:

```
SELECT Najmniej_popularny_film()
```

15. Funkcja 3 (z użyciem kursora): „Średnia_czasu_zatrudnienia”

Funkcja „Średnia_czasu_zatrudnienia” oblicza za pomocą kursora średni czas zatrudnienia wszystkich pracowników kina.

Listing funkcji:

```
ALTER PROCEDURE "Pati"."Aktualizacja_seansow"()
BEGIN
    FOR petla AS kursor CURSOR FOR
        SELECT Termin, Id_seansu as Ids
        FROM Seans
        DO
        IF Termin < getdate() THEN
            DELETE FROM Seans
            WHERE Id_seansu = Ids
        ENDIF;
    END FOR;
END
```

Wywołanie funkcji:

```
CALL Aktualizacja_seansow()
```

16. Trigger 1: „Certyfikat_covid”

Trigger „Certyfikat_covid” wyzwalany jest przy dodawaniu oraz aktualizowaniu tabeli „Widz”. Zapewnia on wpisanie poprawnej wartości kolumny „Czy_certyfikat_covid” (0 lub 1) – wyrzuci błąd gdy wartość pola będzie różna od akceptowalnej bądź gdy wartość pola pozostanie NULL.

Listing triggera:

```
ALTER TRIGGER "Certyfikat_covid" AFTER INSERT, UPDATE
ORDER 1 ON "Pati"."Widz"
REFERENCING OLD AS old_name NEW AS new_name
FOR EACH ROW
BEGIN
    DECLARE Niepoprawna_wartosc_covid EXCEPTION FOR SQLSTATE
'99990';
    IF (new_name.Czy_certyfikat_covid<>1 AND
new_name.Czy_certyfikat_covid<>0)
    THEN IF new_name.Czy_certyfikat_covid IS NOT NULL THEN
            SIGNAL Niepoprawna_wartosc_covid;
        ENDIF;
    ENDIF;
    IF (new_name.Czy_certyfikat_covid=1 OR
new_name.Czy_certyfikat_covid=0)
    THEN
        IF new_name.Czy_certyfikat_covid IS NULL THEN
            SIGNAL Niepoprawna_wartosc_covid;
        ENDIF;
    ENDIF;
END
```

17. Trigger 2: „Dodawanie_seansow”

Trigger „Dodawanie_seansow” wyzwalany jest bezpośrednio po dodaniu bądź aktualizacji wiersza w tabeli „Seans”. Dostajemy błąd w momencie gdy chcemy dodać seans o terminie wcześniejszym niż obecny – dozwolone jest jedynie dodawanie seansów przyszłych. Trigger również zwróci wyjątek gdy wartość terminu będzie równa NULL.

Listing triggera:

```
ALTER TRIGGER "Dodawanie_seansow"
AFTER INSERT, UPDATE
ORDER 1 ON "Pati"."Seans"
REFERENCING OLD AS old_name NEW AS new_name
FOR EACH ROW
BEGIN
    DECLARE Niewlasciwy_termin_seansu EXCEPTION FOR SQLSTATE
'99991';
    IF (new_name.Termin<getdate() AND
new_name.Termin=getdate())
    THEN  IF new_name.Termin IS NOT NULL THEN
            SIGNAL Niewlasciwy_termin_seansu;
        ENDIF;
    ENDIF;
    IF (new_name.Termin>getdate()) THEN
        IF new_name.Termin IS NULL THEN
            SIGNAL Niewlasciwy_termin_seansu;
        ENDIF;
    ENDIF;
END
```

18. Trigger 3: „Historia_zatrudnienia”

Trigger „Historia_zatrudnienia” uzupełnia tabelę „Historia_zatrudnienia” w przypadku dodania danych nowego pracownika, zmiany wiersza bądź usunięcia pracownika. Przechowuje ona również informację jaki rodzaj czynności był wykonywany.

Listing triggera:

```
ALTER TRIGGER "Historia_zatrudnienia"
AFTER INSERT, DELETE, UPDATE
ORDER 1 ON "Pati"."Pracownik"
REFERENCING OLD AS old_name NEW AS new_name
FOR EACH ROW
BEGIN
    DECLARE rodzaj_operacji char(1);
        SET rodzaj_operacji = 'U';
    IF DELETING THEN
        SET rodzaj_operacji = 'D';
    ELSEIF INSERTING THEN
        SET rodzaj_operacji = 'I';
    ENDIF;
    INSERT Historia_zatrudnienia (Id_pracownika, Imie,
Nazwisko, Umowa_od, Termin_operacji, Rodzaj_operacji,
Uzytkownik)
    VALUES (new_name.Id_pracownika, new_name.Imie,
new_name.Nazwisko, new_name.Umowa_od, getdate(),
rodzaj_operacji, connection_property('UserID'))
END
```

19. Listing bazy danych „Kino”

```
/*==============================================================*/
/* DBMS name:      Sybase SQL Anywhere 12                       */
/* Created on:     06.05.2022 00:20:12                          */
/*==============================================================*/



if exists(select 1 from sys.sysforeignkey where
role='FK_BILET_BILET_ULGA') then
    alter table Bilet
       delete foreign key FK_BILET_BILET_ULGA
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_BILET_BILET2_MIEJSCE') then
    alter table Bilet
       delete foreign key FK_BILET_BILET2_MIEJSCE
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_BILET_BILET3_SEANS') then
    alter table Bilet
       delete foreign key FK_BILET_BILET3_SEANS
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_BILET_BILET4_WIDZ') then
    alter table Bilet
```

```
       delete foreign key FK_BILET_BILET4_WIDZ
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_MIEJSCE_RELATIONS_SALA') then
    alter table Miejsce
       delete foreign key FK_MIEJSCE_RELATIONS_SALA
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_OBSADA_OBSADA_FILM') then
    alter table Obsada
       delete foreign key FK_OBSADA_OBSADA_FILM
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_OBSADA_OBSADA2_FUNKCJA') then
    alter table Obsada
       delete foreign key FK_OBSADA_OBSADA2_FUNKCJA
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_OBSADA_OBSADA3_FILMOWIE') then
    alter table Obsada
       delete foreign key FK_OBSADA_OBSADA3_FILMOWIE
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_PRACOWNI_ZATRUDNIE_KINO') then
    alter table Pracownik
```

```
       delete foreign key FK_PRACOWNI_ZATRUDNIE_KINO
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_GATUNEK') then
    alter table Relationship_2
       delete foreign key FK_RELATION_RELATIONS_GATUNEK
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_FILM') then
    alter table Relationship_2
       delete foreign key FK_RELATION_RELATIONS_FILM
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_SEANS') then
    alter table Relationship_7
       delete foreign key FK_RELATION_RELATIONS_SEANS
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_SALA') then
    alter table Relationship_7
       delete foreign key FK_RELATION_RELATIONS_SALA
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_RODZAJ_S') then
    alter table Relationship_8
```

```
       delete foreign key FK_RELATION_RELATIONS_RODZAJ_S
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_RELATION_RELATIONS_SALA') then
    alter table Relationship_8
       delete foreign key FK_RELATION_RELATIONS_SALA
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_SALA_RELATIONS_KINO') then
    alter table Sala
       delete foreign key FK_SALA_RELATIONS_KINO
end if;


if exists(select 1 from sys.sysforeignkey where
role='FK_SEANS_RELATIONS_FILM') then
    alter table Seans
       delete foreign key FK_SEANS_RELATIONS_FILM
end if;


drop index if exists Bilet.Bilet4_FK;


drop index if exists Bilet.Bilet3_FK;


drop index if exists Bilet.Bilet2_FK;


drop index if exists Bilet.Bilet_FK;


drop table if exists Bilet;
```

```
drop index if exists Film.Film_PK;

drop table if exists Film;

drop index if exists Filmowiec.Filmowiec_PK;

drop table if exists Filmowiec;

drop index if exists Funkcja.Funkcja_PK;

drop table if exists Funkcja;

drop index if exists Gatunek.Gatunek_PK;

drop table if exists Gatunek;

drop index if exists Kino.Kino_PK;

drop table if exists Kino;

drop index if exists Miejsce.Relationship_11_FK;

drop index if exists Miejsce.Miejsce_PK;

drop table if exists Miejsce;

drop index if exists Obsada.Obsada3_FK;
```

```
drop index if exists Obsada.Obsada2_FK;

drop index if exists Obsada.Obsada_FK;

drop table if exists Obsada;

drop index if exists Pracownik.Zatrudnienie_FK;

drop index if exists Pracownik.Pracownik_PK;

drop table if exists Pracownik;

drop index if exists Relationship_2.Relationship_3_FK;

drop index if exists Relationship_2.Relationship_2_FK;

drop table if exists Relationship_2;

drop index if exists Relationship_7.Relationship_12_FK;

drop index if exists Relationship_7.Relationship_7_FK;

drop table if exists Relationship_7;

drop index if exists Relationship_8.Relationship_9_FK;

drop index if exists Relationship_8.Relationship_8_FK;

drop table if exists Relationship_8;
```

```
drop index if exists Rodzaj_sali.Rodzaj_sali_PK;

drop table if exists Rodzaj_sali;

drop index if exists Sala.Relationship_10_FK;

drop index if exists Sala.Sala_PK;

drop table if exists Sala;

drop index if exists Seans.Relationship_4_FK;

drop index if exists Seans.Seans_PK;

drop table if exists Seans;

drop index if exists Ulga.Ulga_PK;

drop table if exists Ulga;

drop index if exists Widz.Widz_PK;

drop table if exists Widz;

/*==============================================================
=*/
/* Table: Bilet
*/
```

```
/*==============================================================
=*/

create table Bilet

(

   Id_ulgi              integer                        not
null,

   Rzad                 smallint                       not
null,

   Numer                smallint                       not
null,

   Id_seansu            integer                        not
null,

   Id_widza             integer                        not
null,

   Data_zakupu          timestamp                      null,

   constraint PK_BILET primary key (Id_ulgi, Rzad, Numer,
Id_seansu, Id_widza)

);


/*==============================================================
=*/

/* Index: Bilet_FK
*/

/*==============================================================
=*/

create index Bilet_FK on Bilet (

Id_ulgi ASC

);


/*==============================================================
=*/
```

```
/* Index: Bilet2_FK
*/

/*==============================================================
=*/

create index Bilet2_FK on Bilet (

Rzad ASC,

Numer ASC

);


/*==============================================================
=*/

/* Index: Bilet3_FK
*/

/*==============================================================
=*/

create index Bilet3_FK on Bilet (

Id_seansu ASC

);


/*==============================================================
=*/

/* Index: Bilet4_FK
*/

/*==============================================================
=*/

create index Bilet4_FK on Bilet (

Id_widza ASC

);


/*==============================================================
=*/
```

```
/* Table: Film
*/

/*==============================================================
=*/

create table Film

(

   Id_filmu             integer                        not
null,

   Tytul                char(100)                      not
null,

   Rok_produkcji        date                           null,

   Czas_trwania         time                           null,

   constraint PK_FILM primary key (Id_filmu)

);


/*==============================================================
=*/

/* Index: Film_PK
*/

/*==============================================================
=*/

create unique index Film_PK on Film (

Id_filmu ASC

);


/*==============================================================
=*/

/* Table: Filmowiec
*/

/*==============================================================
=*/

create table Filmowiec
```

```
(
   Id_filmowca          integer                        not
null,
   Imie                 char(30)                       not
null,
   Nazwisko             char(50)                       not
null,
   Data_urodzenia       date                           null,
   constraint PK_FILMOWIEC primary key (Id_filmowca)
);


/*==============================================================
=*/
/* Index: Filmowiec_PK
*/
/*==============================================================
=*/
create unique index Filmowiec_PK on Filmowiec (
Id_filmowca ASC
);


/*==============================================================
=*/
/* Table: Funkcja
*/
/*==============================================================
=*/
create table Funkcja
(
   Id_funkcji           integer                        not
null,
   Nazwa_funkcji        char(100)                      null,
```

```
   constraint PK_FUNKCJA primary key (Id_funkcji)
);


/*==============================================================
=*/
/* Index: Funkcja_PK
*/
/*==============================================================
=*/
create unique index Funkcja_PK on Funkcja (
Id_funkcji ASC
);


/*==============================================================
=*/
/* Table: Gatunek
*/
/*==============================================================
=*/
create table Gatunek
(
   Id_gatunku           integer                        not
null,
   Nazwa_gatunku        char(50)                       not
null,
   constraint PK_GATUNEK primary key (Id_gatunku)
);


/*==============================================================
=*/
/* Index: Gatunek_PK
*/
```

```
/*==============================================================
=*/
create unique index Gatunek_PK on Gatunek (
Id_gatunku ASC
);


/*==============================================================
=*/
/* Table: Kino
*/
/*==============================================================
=*/
create table Kino
(
   Id_kino              integer                        not
null,
   Strona_internetowa   varchar(100)                   null,
   Telefon              char(9)                        not
null,
   Kod_pocztowy         char(6)                        not
null,
   Miejscowosc          varchar(100)                   not
null,
   Ulica                varchar(100)                   not
null,
   Numer_lokalu         char(10)                       not
null,
   Ilosc_sal            smallint                       null,
   constraint PK_KINO primary key (Id_kino)
);


comment on primary key on Kino is
```

```
'Klucz kina ...';


/*==============================================================
=*/
/* Index: Kino_PK
*/
/*==============================================================
=*/
create unique index Kino_PK on Kino (
Id_kino ASC
);


/*==============================================================
=*/
/* Table: Miejsce
*/
/*==============================================================
=*/
create table Miejsce
(
   Rzad                 smallint                       not
null,
   Numer                smallint                       not
null,
   Id_sali              integer                        not
null,
   Stan                 smallint                       null,
   constraint PK_MIEJSCE primary key (Rzad, Numer)
);


/*==============================================================
=*/
```

```
/* Index: Miejsce_PK
*/
/*==============================================================
=*/
create unique index Miejsce_PK on Miejsce (
Rzad ASC,
Numer ASC
);


/*==============================================================
=*/
/* Index: Relationship_11_FK
*/
/*==============================================================
=*/
create index Relationship_11_FK on Miejsce (
Id_sali ASC
);


/*==============================================================
=*/
/* Table: Obsada
*/
/*==============================================================
=*/
create table Obsada
(
   Id_filmu             integer                        not
null,
   Id_funkcji           integer                        not
null,
```

```
   Id_filmowca          integer                        not
null,
   Postać               varchar(100)                   null,
   constraint PK_OBSADA primary key (Id_filmu, Id_funkcji,
Id_filmowca)
);


/*==============================================================
==*/
/* Index: Obsada_FK
*/
/*==============================================================
==*/
create index Obsada_FK on Obsada (
Id_filmu ASC
);


/*==============================================================
==*/
/* Index: Obsada2_FK
*/
/*==============================================================
==*/
create index Obsada2_FK on Obsada (
Id_funkcji ASC
);


/*==============================================================
==*/
/* Index: Obsada3_FK
*/
```

```
/*==============================================================
=*/
create index Obsada3_FK on Obsada (
Id_filmowca ASC
);


/*==============================================================
=*/
/* Table: Pracownik
*/
/*==============================================================
=*/
create table Pracownik
(
   Id_pracownika        integer                        not
null,
   Id_kino              integer                        not
null,
   Imie                 char(30)                       not
null,
   Nazwisko             char(50)                       not
null,
   Umowa_od             date                           null,
   constraint PK_PRACOWNIK primary key (Id_pracownika)
);


/*==============================================================
=*/
/* Index: Pracownik_PK
*/
/*==============================================================
=*/
```

```
create unique index Pracownik_PK on Pracownik (
Id_pracownika ASC
);


/*==============================================================
=*/
/* Index: Zatrudnienie_FK
*/
/*==============================================================
=*/
create index Zatrudnienie_FK on Pracownik (
Id_kino ASC
);


/*==============================================================
=*/
/* Table: Relationship_2
*/
/*==============================================================
=*/
create table Relationship_2
(
   Id_gatunku           integer                        not
null,
   Id_filmu             integer                        not
null,
   constraint PK_RELATIONSHIP_2 primary key (Id_gatunku,
Id_filmu)
);


/*==============================================================
=*/
```

```
/* Index: Relationship_2_FK
*/
/*==============================================================
=*/
create index Relationship_2_FK on Relationship_2 (
Id_gatunku ASC
);


/*==============================================================
=*/
/* Index: Relationship_3_FK
*/
/*==============================================================
=*/
create index Relationship_3_FK on Relationship_2 (
Id_filmu ASC
);


/*==============================================================
=*/
/* Table: Relationship_7
*/
/*==============================================================
=*/
create table Relationship_7
(
   Id_sali              integer                        not
null,
   Id_seansu            integer                        not
null,
   constraint PK_RELATIONSHIP_7 primary key (Id_sali,
Id_seansu)
```

```
);


/*==============================================================
=*/
/* Index: Relationship_7_FK
*/
/*==============================================================
=*/
create index Relationship_7_FK on Relationship_7 (
Id_sali ASC
);


/*==============================================================
=*/
/* Index: Relationship_12_FK
*/
/*==============================================================
=*/
create index Relationship_12_FK on Relationship_7 (
Id_seansu ASC
);


/*==============================================================
=*/
/* Table: Relationship_8
*/
/*==============================================================
=*/
create table Relationship_8
(
   Id_rodzaju_sali      integer                        not
null,
```

```
   Id_sali              integer                         not
null,

   constraint PK_RELATIONSHIP_8 primary key (Id_rodzaju_sali,
Id_sali)

);


/*==============================================================
=*/

/* Index: Relationship_8_FK
*/

/*==============================================================
=*/

create index Relationship_8_FK on Relationship_8 (

Id_rodzaju_sali ASC

);


/*==============================================================
=*/

/* Index: Relationship_9_FK
*/

/*==============================================================
=*/

create index Relationship_9_FK on Relationship_8 (

Id_sali ASC

);


/*==============================================================
=*/

/* Table: Rodzaj_sali
*/

/*==============================================================
=*/
```

```
create table Rodzaj_sali
(
   Id_rodzaju_sali      integer                        not
null,
   Rodzaj               char(50)                       not
null,
   constraint PK_RODZAJ_SALI primary key (Id_rodzaju_sali)
);


/*==============================================================
==*/
/* Index: Rodzaj_sali_PK
*/
/*==============================================================
==*/
create unique index Rodzaj_sali_PK on Rodzaj_sali (
Id_rodzaju_sali ASC
);


/*==============================================================
==*/
/* Table: Sala
*/
/*==============================================================
==*/
create table Sala
(
   Id_sali              integer                        not
null,
   Id_kino              integer                        not
null,
```

```
   Ilosc_miejsc         smallint                       not
null,
   Piętro               smallint                       null,
   constraint PK_SALA primary key (Id_sali)
);


/*==============================================================
=*/
/* Index: Sala_PK
*/
/*==============================================================
=*/
create unique index Sala_PK on Sala (
Id_sali ASC
);


/*==============================================================
=*/
/* Index: Relationship_10_FK
*/
/*==============================================================
=*/
create index Relationship_10_FK on Sala (
Id_kino ASC
);


/*==============================================================
=*/
/* Table: Seans
*/
/*==============================================================
=*/
```

```
create table Seans
(
   Id_seansu            integer                        not
null,
   Id_filmu             integer                        not
null,
   Termin               timestamp                      not
null,
   constraint PK_SEANS primary key (Id_seansu)
);


/*==============================================================
=*/
/* Index: Seans_PK
*/
/*==============================================================
=*/
create unique index Seans_PK on Seans (
Id_seansu ASC
);


/*==============================================================
=*/
/* Index: Relationship_4_FK
*/
/*==============================================================
=*/
create index Relationship_4_FK on Seans (
Id_filmu ASC
);
```

```
/*==============================================================
=*/
/* Table: Ulga
*/
/*==============================================================
=*/
create table Ulga
(
   Id_ulgi              integer                        not
null,
   Procent_ulgi         smallint                       not
null,
   Rodzaj_ulgi          char(300)                      not
null,
   Opis_ulgi            long varchar                   null,
   constraint PK_ULGA primary key (Id_ulgi)
);


/*==============================================================
=*/
/* Index: Ulga_PK
*/
/*==============================================================
=*/
create unique index Ulga_PK on Ulga (
Id_ulgi ASC
);


/*==============================================================
=*/
/* Table: Widz
*/
```

```
/*==============================================================
=*/
create table Widz
(
   Id_widza             integer                        not
null,
   Imie                 char(30)                       null,
   Nazwisko             char(50)                       null,
   Adres_email          char(50)                       not
null,
   Czy_certyfikat_covid smallint                       null,
   constraint PK_WIDZ primary key (Id_widza)
);


/*==============================================================
=*/
/* Index: Widz_PK
*/
/*==============================================================
=*/
create unique index Widz_PK on Widz (
Id_widza ASC
);


alter table Bilet
   add constraint FK_BILET_BILET_ULGA foreign key (Id_ulgi)
      references Ulga (Id_ulgi)
      on update restrict
      on delete restrict;


alter table Bilet
```

```
   add constraint FK_BILET_BILET2_MIEJSCE foreign key (Rzad,
Numer)
      references Miejsce (Rzad, Numer)
      on update restrict
      on delete restrict;


alter table Bilet
   add constraint FK_BILET_BILET3_SEANS foreign key (Id_seansu)
      references Seans (Id_seansu)
      on update restrict
      on delete restrict;


alter table Bilet
   add constraint FK_BILET_BILET4_WIDZ foreign key (Id_widza)
      references Widz (Id_widza)
      on update restrict
      on delete restrict;


alter table Miejsce
   add constraint FK_MIEJSCE_RELATIONS_SALA foreign key
(Id_sali)
      references Sala (Id_sali)
      on update restrict
      on delete restrict;


alter table Obsada
   add constraint FK_OBSADA_OBSADA_FILM foreign key (Id_filmu)
      references Film (Id_filmu)
      on update restrict
```

```
      on delete restrict;


alter table Obsada
   add constraint FK_OBSADA_OBSADA2_FUNKCJA foreign key
(Id_funkcji)
      references Funkcja (Id_funkcji)
      on update restrict
      on delete restrict;


alter table Obsada
   add constraint FK_OBSADA_OBSADA3_FILMOWIE foreign key
(Id_filmowca)
      references Filmowiec (Id_filmowca)
      on update restrict
      on delete restrict;


alter table Pracownik
   add constraint FK_PRACOWNI_ZATRUDNIE_KINO foreign key
(Id_kino)
      references Kino (Id_kino)
      on update restrict
      on delete restrict;


alter table Relationship_2
   add constraint FK_RELATION_RELATIONS_GATUNEK foreign key
(Id_gatunku)
      references Gatunek (Id_gatunku)
      on update restrict
      on delete restrict;
```

```
alter table Relationship_2
   add constraint FK_RELATION_RELATIONS_FILM foreign key
(Id_filmu)
      references Film (Id_filmu)
      on update restrict
      on delete restrict;


alter table Relationship_7
   add constraint FK_RELATION_RELATIONS_SEANS foreign key
(Id_seansu)
      references Seans (Id_seansu)
      on update restrict
      on delete restrict;


alter table Relationship_7
   add constraint FK_RELATION_RELATIONS_SALA foreign key
(Id_sali)
      references Sala (Id_sali)
      on update restrict
      on delete restrict;


alter table Relationship_8
   add constraint FK_RELATION_RELATIONS_RODZAJ_S foreign key
(Id_rodzaju_sali)
      references Rodzaj_sali (Id_rodzaju_sali)
      on update restrict
      on delete restrict;


alter table Relationship_8
```

```
   add constraint FK_RELATION_RELATIONS_SALA foreign key
(Id_sali)
      references Sala (Id_sali)
      on update restrict
      on delete restrict;


alter table Sala
   add constraint FK_SALA_RELATIONS_KINO foreign key (Id_kino)
      references Kino (Id_kino)
      on update restrict
      on delete restrict;


alter table Seans
   add constraint FK_SEANS_RELATIONS_FILM foreign key
(Id_filmu)
      references Film (Id_filmu)
      on update restrict
      on delete restrict;
```