Практика по базам данных
ОТЧЕТ

Мирошников Владислав

Предметная область: «Билеты в кино»

Реализация: Microsoft SQL Server, JetBrains DataGrip

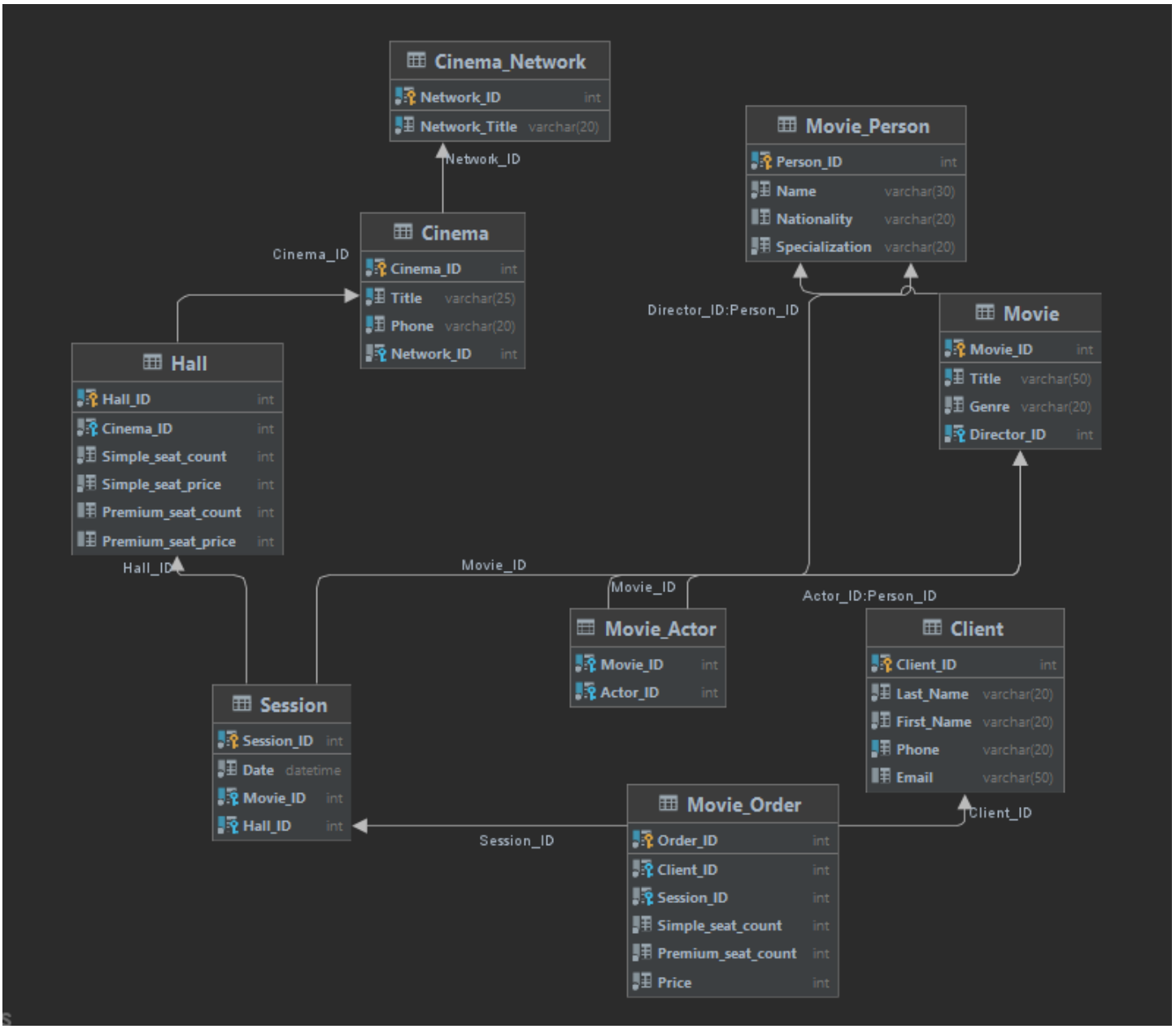
Содержание

ОПИСАНИЕ СИСТЕМЫ

Требования

Система предназначена для заказа билетов на киносеансы в кинотеатрах города.
Программа позволяет клиентам делать заказы, получать информацию о кинотеатрах (название, принадлежность к сети, контактные данные), о репертуаре (включая жанр фильма, режиссера и ведущих актеров), о сеансах.
Делать заказы клиент может только после регистрации в системе (минимальные данные – фамилия, имя, контакт). Один заказ может включать несколько мест, но только на один сеанс.
Некоторые кинотеатры имеют по несколько залов разной вместимости. Места в кинозалах делятся на обычные и повышенного комфорта. Политика цен свободная.

Модель данных



Функциональность

Серверная часть

Хранимые процедуры\функции	*Реализация*	*Комментарии*
Добавление нового клиента	add_new_client_to_db	(фамилия, имя, телефон, почта)
Добавление нового фильма	add_new_movie_to_db	(название, жанр, режиссер)
Оформление заказа		
Добавление нового актера		
Удаление режиссера		
Функция проверки, показывают ли фильм в кинотеатрах, то есть наличие любых сеансов	check_film_in_cinema	(id фильма)
Изменение времени сеанса		
…		
Снятие фильма с проката		

Триггеры	*Реализация*	*Комментарии*
Проверка при добавлении фильма, что у режиссера указана верная специализация	movie_insert	Для киноакадемии
Проверка при покупке билета, что на данный сеанс остались места		
Триггер, срабатывающий после покупки билета пользователем. В нужном зале уменьшает количество обычных и премиум мест.	order_transaction	Для контроля возможности продажи билетов и свободных мест
Проверка при добавлении пары Фильм-Актер, что у актера правильная специализация.	movie_actor_insert	Для хранения правильных данных
...		
Удаление клиента и его истории покупок		
Удаление сеансов		
…		

Представления	*Реализация*	*Комментарии*
Новые фильмы	new_films	Новые фильмы, которые сейчас идут в кинотеатрах города (ID, название, жанр, режиссер)
Сеансы на сегодня	today_sessions	(дата, название фильма, жанр, номер зала, название кинотеатра, телефон кинотеатра)
Список американских актеров и их фильмов		(фамилия имя, национальность, фильм)
Режиссеры с наибольшим количеством фильмов		(фамилия имя, национальность, фильм)
Актуальные клиенты (то есть, у которых есть заказы)	actual_clients	(фамилия имя, телефон, почта)
Клиенты, совершившие больше 1 покупки		-“-
Информация о кинотеатрах с группировкой по сети		(название сети кинотеатра, название кинотеатра, телефон)
Актеры, которые играли в экшн или детективных фильмах		(фамилия имя, национальность, жанр)
…		

Клиентская часть

Экранные формы основные	*дополнительные*	*Реализация* *(запрос)*	*Что здесь можно использовать из серверной части*
База фильмов			
	Новый фильм		add_new_movie_to_db
	Проверка режиссера		movie_insert
	Удалить фильм		

	Фильтры	2. Вывести различные жанры идущих сейчас в кино фильмов за исключением супергеройского 11. Вывести список фильмов, которые идут в кинотеатрах и супергеройского жанра	new_films
База актеров и режиссеров	Добавление пары фильм-актер		movie_actor_insert
	Добавление работника кино (режиссер или актер)		
	Фильтры	4. Вывести всех американских актеров, которые играли в фильмах, где режиссер - тоже американец 7. Список актеров, которые играли в экшн или детективных фильмах 12. Актеры, которые не играли в фильмах, представленных в базе фильмов	
Реестр клиентов	Добавить клиента		add_new_client_to_db
	Удалить клиента		
	Фильтр	3. Клиенты, совершившие больше 1 покупки 8. Вывести список актуальных клиентов, у которых есть Email 9. Информация о клиентах и сумме потраченных денег по возрастанию с разбивкой на кинотеатры в сети Кинонео	actual_clients
	Транзакция (покупка билета)		order_transaction
Реестр кинотеатров			
	Добавить кинотеатр		
	Привязать новый зал с сеансами к кинотеатру		
	Фильтр	10. Залы кинотеатров города с местами повышенного комфорта 5. Информация о кинотеатрах с группировкой по сети	
Список сеансов	Новый сеанс		
	Привязать к сеансу время		
	Фильтр	1. Показать доступные сеансы на сегодня с сортировкой по времени по возрастанию	

	Проверить наличие сеансов на конкретный фильм		check_film_in_cinema
	Сеансы на сегодня		today_sessions
Служебные запросы		6. Общая выручка кинотеатров сети Кинонео	

СКРИПТЫ

Серверная часть

Хранимые процедуры и функции

```
-----------------------------------------------------------
-- Хранимые процедуры и функции
-----------------------------------------------------------

-----------------------------------------------------------
-- Добавление нового клиента
-----------------------------------------------------------
CREATE PROCEDURE add_new_client_to_db(@last_name VARCHAR(20), @first_name VARCHAR(20), @phone
VARCHAR(20),
                                @email VARCHAR(50) = null) as
begin
   declare @result_id int;
   select @result_id = max(Client_ID) + 1 from Client;
   insert into Client(Client_ID, Last_Name, First_Name, Phone, Email)
   values (@result_id, @last_name, @first_name, @phone, @email);
end;

--Пример:
-- EXECUTE add_new_client_to_db @last_name='Petrov', @first_name='Vasya', @phone = '89291776789', @email =
null;

-----------------------------------------------------------
-- Проверка, показывают ли фильм в кинотеатрах, то есть наличие любых сеансов
-----------------------------------------------------------
CREATE FUNCTION check_film_in_cinema(@movie_id int)
   returns int
begin
   if exists(select Session.Movie_ID
           from Movie
                   join Session on Movie.Movie_ID = Session.Movie_ID
           where Session.Movie_ID = @movie_id)
      begin
         return 1;
      end;
   return 0;
end;

--Пример вызова:
--select dbo.check_film_in_cinema(2);

-----------------------------------------------------------
-- Добавление нового фильма
-----------------------------------------------------------
CREATE PROCEDURE add_new_movie_to_db(@title VARCHAR(50), @genre VARCHAR(20), @director_id int) as
begin
   declare @result_id int;
   select @result_id = max(Movie_ID) + 1 from Movie;
   insert into Movie(Movie_ID, Title, Genre, Director_ID)
   values (@result_id, @title, @genre, @director_id);
end;

-- Пример:
--EXECUTE add_new_movie_to_db @title='Spider-Man 3', @genre='superhero', @director_id = 18;
```

Триггеры

```
-----------------------------------------------------------------
-- Триггер, проверяющий при добавлении фильма, что у режиссера правильная специализация
-----------------------------------------------------------------
CREATE TRIGGER movie_insert
   on Movie
   instead of insert as
begin
   if (select director_id
      from inserted) in (select Person_ID
                         from Movie_Person
                         where Specialization = 'Director')
      begin
         insert into Movie(Movie_ID, Title, Genre, Director_ID) select * from inserted;
         return;
      end;
   declare @id int;
   set @id = (select Director_ID from inserted);
   print 'Director with ID ' + cast(@id as VARCHAR(20)) + ' not found!';
   rollback;

end;

--Пример:
-- Неверный insert:
-- INSERT INTO Movie(Movie_ID, Title, Genre, Director_ID)
--  (select max(Movie_ID) + 1, 'Iron Man 2', 'Superhero', 24 from Movie);
-- Верный insert:
-- INSERT INTO Movie(Movie_ID, Title, Genre, Director_ID)
--  (select max(Movie_ID) + 1, 'Iron Man 2', 'Superhero', 15 from Movie);

-----------------------------------------------------------------
-- Триггер, срабатывающий после покупки билета пользователем.
-- В нужном зале уменьшает количество обычных и премиум мест.
-----------------------------------------------------------------
CREATE TRIGGER order_transaction
   on Movie_Order
   after insert as
begin
   update Hall
   set Simple_seat_count -= (select Simple_seat_count from inserted),
      Premium_seat_count -= (select Premium_seat_count from inserted)
   where Hall_ID in (select Hall.Hall_ID
                     from Hall
                           join Session S on Hall.Hall_ID = S.Hall_ID
                     where Session_ID = (select Session_ID from inserted))


end;

-- Пример:
-- INSERT INTO Movie_Order(Order_ID, Client_ID, Session_ID, Simple_seat_count, Premium_seat_count, Price)
-- values (7, 5, 13, 1, default, 210)

-----------------------------------------------------------------
-- Триггер, проверяющий при добавлении пары Фильм-Актер, что у актера правильная специализация
-----------------------------------------------------------------
CREATE TRIGGER movie_actor_insert
   on Movie_Actor
   instead of insert as
begin
```

```
    if (select Actor_ID
        from inserted) in (select Person_ID
                           from Movie_Person
                           where Specialization = 'Actor')
        begin
            insert into Movie_Actor(Movie_ID, Actor_ID) select * from inserted;
            return;
        end;
    declare @id int;
    set @id = (select Actor_ID from inserted);
    print 'Actor with ID ' + cast(@id as VARCHAR(20)) + ' not found!';
    rollback;

end;
--Пример:
-- Неверный insert:
-- insert into Movie_Actor(Movie_ID, Actor_ID) values
--  (4, 18);
-- Верный insert:
--  insert into Movie_Actor(Movie_ID, Actor_ID) values
--  (1, 3);
```

Представления

```
-------------------------------------------------------------
-- Представления
-------------------------------------------------------------

-------------------------------------------------------------
-- Новые фильмы, которые сейчас идут в кинотеатрах города
-------------------------------------------------------------
CREATE VIEW new_films as
select Movie_ID, Title, Genre, Name as Director
from Movie
        join Movie_Person on Director_ID = Movie_Person.Person_ID
where Movie_ID in (select distinct(Movie.Movie_ID)
                   from Movie
                            join Session S2 on Movie.Movie_ID = S2.Movie_ID)

-------------------------------------------------------------
-- Сеансы в кинотеатрах города на сегодня
-------------------------------------------------------------
CREATE VIEW today_sessions as
select Session_ID, Date, M.Title as Movie_Title, Genre, H.Hall_ID, C.Title as Cinema_Title, C.Phone
from Session
        join Movie M on Session.Movie_ID = M.Movie_ID
        join Hall H on H.Hall_ID = Session.Hall_ID
        join Cinema C on H.Cinema_ID = C.Cinema_ID
where (select Convert(date, Session.Date)) = (select Convert(date, GETDATE()))

-------------------------------------------------------------
-- Актуальные клиенты (то есть, у которых есть заказы)
-------------------------------------------------------------
CREATE VIEW actual_clients as
select *
from Client
where Client_ID in (select distinct(Client.Client_ID)
                    from Client
                             join Movie_Order MO on Client.Client_ID = MO.Client_ID)
```

Клиентская часть

```
-----------------------------------------------------------------
-- Запросы
-----------------------------------------------------------------

-----------------------------------------------------------------
-- 1. Показать доступные сеансы на сегодня с сортировкой по времени по возрастанию
-----------------------------------------------------------------
select Movie_Title, Genre, Cinema_Title, Phone, Date
from today_sessions
order by Date
-----------------------------------------------------------------
-- 2. Вывести различные жанры идущих сейчас в кино фильмов за исключением супергеройского
-----------------------------------------------------------------
select distinct(Genre)
from Movie
        join Session S4 on Movie.Movie_ID = S4.Movie_ID
where Genre != 'Superhero'
-----------------------------------------------------------------
-- 3. Клиенты, совершившие больше 1 покупки
-----------------------------------------------------------------
select Last_Name, First_Name, Phone, count(MO3.Client_ID) as Orders_Count
from Client
        join Movie_Order MO3 on Client.Client_ID = MO3.Client_ID
group by Client.Client_ID, Last_Name, First_Name, Phone
having count(Mo3.Client_ID) > 1
-----------------------------------------------------------------
-- 4. Вывести всех американских актеров, которые играли в фильмах, где режиссер - тоже американец
-----------------------------------------------------------------
select Name, Nationality, M2.Title as Movie_Title, M2.Genre
from Movie_Person
        join Movie_Actor on Movie_Actor.Actor_ID = Person_ID
        join Movie M2 on Movie_Actor.Movie_ID = M2.Movie_ID
where (Specialization = 'Actor'
  and Nationality = 'USA'
  and (select Nationality from Movie_Person where Person_ID = M2.Director_ID) = 'USA')
-----------------------------------------------------------------
-- 5. Информация о кинотеатрах с группировкой по сети
-----------------------------------------------------------------
select Network_Title, C2.Title, C2.Phone from Cinema_Network
    join Cinema C2 on C2.Network_ID = Cinema_Network.Network_ID group by Network_Title, C2.Title, C2.Phone
-----------------------------------------------------------------
-- 6. Общая выручка кинотеатров сети Кинонео
-----------------------------------------------------------------
select Title, SUM(O.Price) as Proceeds
from Cinema
        join dbo.Hall H2 on Cinema.Cinema_ID = H2.Cinema_ID
        join Session S3 on H2.Hall_ID = S3.Hall_ID
        join Movie_Order O on S3.Session_ID = O.Session_ID
        join Cinema_Network CN on Cinema.Network_ID = CN.Network_ID
where CN.Network_Title = 'Kinoneo'
group by Cinema.Title
-----------------------------------------------------------------
-- 7. Список актеров, которые играли в экшн или детективных фильмах
-----------------------------------------------------------------
select Name, Nationality, M3.Genre
from Movie_Person
        join Movie_Actor on Person_ID = Movie_Actor.Actor_ID
        join Movie M3 on Movie_Actor.Movie_ID = M3.Movie_ID
group by Name, Nationality, M3.Genre
having M3.Genre = 'Detective'
```

```
UNION
select Name, Nationality, M3.Genre
from Movie_Person
	join Movie_Actor on Person_ID = Movie_Actor.Actor_ID
	join Movie M3 on Movie_Actor.Movie_ID = M3.Movie_ID
where M3.Genre Like 'Action%'
-----------------------------------------------------------------
-- 8. Вывести список актуальных клиентов, у которых есть Email
-----------------------------------------------------------------
select Last_Name, First_Name, Phone, Email from actual_clients where Email IS NOT NULL
-----------------------------------------------------------------
-- 9. Информация о клиентах и сумме потраченных денег по возрастанию с разбивкой на кинотеатры в сети
Кинонео
-----------------------------------------------------------------
select N.Network_Title, C3.Title, Last_Name, First_Name, Client.Phone, SUM(Price) as Proceeds
from Client
	join Movie_Order MO2 on Client.Client_ID = MO2.Client_ID
	join Session S5 on MO2.Session_ID = S5.Session_ID
	join Hall H3 on S5.Hall_ID = H3.Hall_ID
	join Cinema C3 on H3.Cinema_ID = C3.Cinema_ID
	join Cinema_Network N on C3.Network_ID = N.Network_ID
group by N.Network_Title, C3.Title, Last_Name, First_Name, Client.Phone
having N.Network_Title = 'Kinoneo'
order by Proceeds
-----------------------------------------------------------------
-- 10. Залы кинотеатров города с местами повышенного комфорта
-----------------------------------------------------------------
select Title, H4.Hall_ID, H4.Simple_seat_count, H4.Premium_seat_count
from Cinema
	join Hall H4 on Cinema.Cinema_ID = H4.Cinema_ID
where H4.Premium_seat_count > 0
group by Title, H4.Hall_ID, H4.Simple_seat_count, H4.Premium_seat_count
-----------------------------------------------------------------
-- 11. Вывести список фильмов, которые идут в кинотеатрах и супергеройского жанра
-----------------------------------------------------------------
select Title, Director from new_films where Genre = 'Superhero'
-----------------------------------------------------------------
-- 12. Актеры, которые не играли в фильмах, представленных в базе данных
-----------------------------------------------------------------
select Name, Nationality
from Movie_Person
where Specialization = 'Actor'
except
select Name, Nationality
from Movie_Person
	join Movie_Actor on Person_ID = Movie_Actor.Actor_ID
where Specialization = 'Actor'
```

ПРИЛОЖЕНИЕ: Создание и заполнение базы данных, удаление

```
-- CREATE DATABASE movie_tickets;
-- GO
-- USE movie_tickets;

-----------------------------------------------------------------
-- Создание таблиц и PK
-----------------------------------------------------------------
CREATE TABLE Client
(
   Client_ID  INTEGER     NOT NULL,
   Last_Name  VARCHAR(20) NOT NULL,
   First_Name VARCHAR(20) NOT NULL,
   Phone      VARCHAR(20) NOT NULL UNIQUE,
   Email      VARCHAR(50),
   CONSTRAINT Client_PK PRIMARY KEY (Client_ID)
)
;

CREATE TABLE Movie_Order
(
   Order_ID          INTEGER NOT NULL,
   Client_ID         INTEGER NOT NULL,
   Session_ID        INTEGER NOT NULL,
   Simple_seat_count  INTEGER NOT NULL default 0 check (Simple_seat_count >= 0),
   Premium_seat_count INTEGER NOT NULL default 0 check (Premium_seat_count >= 0),
   Price             INTEGER NOT NULL check (Price > 0),
   CONSTRAINT Order_PK PRIMARY KEY (Order_ID)
)
;

CREATE TABLE Session
(
   Session_ID INTEGER                  NOT NULL,
   Date       DATETIME DEFAULT GETDATE() NOT NULL,
   Movie_ID   INTEGER                  NOT NULL,
   Hall_ID    INTEGER                  NOT NULL,
   CONSTRAINT Session_PK PRIMARY KEY (Session_ID)
)
;

-- Кинозалов без обычных мест не может быть
CREATE TABLE Hall
(
   Hall_ID           INTEGER NOT NULL,
   Cinema_ID         INTEGER NOT NULL,
   Simple_seat_count  INTEGER NOT NULL check (Simple_seat_count > 0),
   Simple_seat_price  INTEGER NOT NULL check (Simple_seat_price > 0),
   Premium_seat_count INTEGER check (Premium_seat_count >= 0),
   Premium_seat_price INTEGER check (Premium_seat_price >= 0),
   CONSTRAINT Hall_PK PRIMARY KEY (Hall_ID)
)
;

CREATE TABLE Cinema
(
   Cinema_ID  INTEGER     NOT NULL,
   Title      VARCHAR(25) NOT NULL UNIQUE,
   Phone      VARCHAR(20) NOT NULL UNIQUE,
   Network_ID INTEGER     NOT NULL,
   CONSTRAINT Cinema_PK PRIMARY KEY (Cinema_ID)
```

```
)
;

CREATE TABLE Cinema_Network
(
   Network_ID    INTEGER     NOT NULL,
   Network_Title VARCHAR(20) NOT NULL UNIQUE,
   CONSTRAINT Cinema_Network_PK PRIMARY KEY (Network_ID)
)
;

CREATE TABLE Movie_Person
(
   Person_ID      INTEGER     NOT NULL,
   Name           VARCHAR(30) NOT NULL,
   Nationality    VARCHAR(20),
   Specialization VARCHAR(20) NOT NULL,
   CONSTRAINT Person_PK PRIMARY KEY (Person_ID)
)
;

CREATE TABLE Movie
(
   Movie_ID    INTEGER     NOT NULL,
   Title       VARCHAR(50) NOT NULL UNIQUE,
   Genre       VARCHAR(20) NOT NULL,
   Director_ID INTEGER     NOT NULL,
   CONSTRAINT Movie_PK PRIMARY KEY (Movie_ID)
)
;

CREATE TABLE Movie_Actor
(
   Movie_ID INTEGER NOT NULL,
   Actor_ID INTEGER NOT NULL,
   UNIQUE (Movie_ID, Actor_ID)
)
;

-------------------------------------------------------------
-- Создание FK
-------------------------------------------------------------
ALTER TABLE Movie_Order
   ADD CONSTRAINT FK_Order_Client
      FOREIGN KEY (Client_ID)
         REFERENCES Client (Client_ID) on delete cascade
;

ALTER TABLE Movie_Order
   ADD CONSTRAINT FK_Order_Session
      FOREIGN KEY (Session_ID)
         REFERENCES Session (Session_ID)
;

ALTER TABLE Session
   ADD CONSTRAINT FK_Session_Movie
      FOREIGN KEY (Movie_ID)
         REFERENCES Movie (Movie_ID)
;

ALTER TABLE Session
   ADD CONSTRAINT FK_Session_Hall
      FOREIGN KEY (Hall_ID)
```

```
        REFERENCES Hall (Hall_ID)
;

ALTER TABLE Hall
   ADD CONSTRAINT FK_Hall_Cinema
      FOREIGN KEY (Cinema_ID)
         REFERENCES Cinema (Cinema_ID)
;

ALTER TABLE Cinema
   ADD CONSTRAINT FK_Cinema_Network
      FOREIGN KEY (Network_ID)
         REFERENCES Cinema_Network (Network_ID) on delete cascade
;

ALTER TABLE Movie
   ADD CONSTRAINT FK_Movie_Director
      FOREIGN KEY (Director_ID)
         REFERENCES Movie_Person (Person_ID)
;

ALTER TABLE Movie_Actor
   ADD CONSTRAINT FK_Movie
      FOREIGN KEY (Movie_ID)
         REFERENCES Movie (Movie_ID)
;

ALTER TABLE Movie_Actor
   ADD CONSTRAINT FK_Actor
      FOREIGN KEY (Actor_ID)
         REFERENCES Movie_Person (Person_ID)
;


-----------------------------------------------------------------
-- Заполнение таблиц тестовыми данными
-----------------------------------------------------------------
set dateformat ymd;
INSERT INTO Cinema_Network(Network_ID, Network_Title)
VALUES (1, 'Kinoneo'),
      (2, 'Charlie'),
      (3, 'Cinema Park'),
      (4, 'Karo');

INSERT INTO Cinema(Cinema_ID, Title, Phone, Network_ID)
VALUES (1, 'Kinoneo syzranova', '8(863)432-47-47', 1),
      (2, 'Kinoneo petrovskaya', '8(863)424-35-17', 1),
      (3, 'Charlie Air', '+7(863)053-94-34', 2),
      (4, 'Charlie Caramel', '8(863)112-90-76', 2),
      (5, 'Cinema Park Grand Canyon', '8(800)700-01-11', 3),
      (6, 'Cinema Park Rainbow', '8(812)313-05-28', 3),
      (7, 'KARO 9 Warsaw Express', '8(960)231-47-19', 4);

INSERT INTO Hall(Hall_ID, Cinema_ID, Simple_seat_count, Simple_seat_price, Premium_seat_count,
Premium_seat_price)
VALUES (1, 1, 20, 200, null, null),
      (2, 1, 35, 150, 10, 250),
      (3, 1, 40, 220, 20, 300),
      (4, 2, 56, 180, 14, 230),
      (5, 2, 10, 200, 10, 500),
      (6, 3, 50, 210, 10, 345),
      (7, 4, 30, 350, null, null),
      (8, 4, 35, 100, 10, 145),
```

```
    (9, 5, 80, 200, 20, 290),
    (10, 5, 60, 250, 15, 330),
    (11, 6, 30, 225, 50, 450),
    (12, 7, 90, 100, 30, 375);

INSERT INTO Movie_Person(Person_ID, Name, Nationality, Specialization)
VALUES (1, 'Robert Downey Jr.', 'USA', 'Actor'),
    (2, 'Mark Ruffalo', 'USA', 'Actor'),
    (3, 'Sebastian Stan', 'Romania', 'Actor'),
    (4, 'Chris Evans', 'USA', 'Actor'),
    (5, 'Scarlett Johansson', 'USA', 'Actor'),
    (6, 'Benedict Cumberbatch', 'England', 'Actor'),
    (7, 'Chris Hemsworth', 'Australia', 'Actor'),
    (8, 'Chris Pratt', 'USA', 'Actor'),
    (9, 'Tom Hiddleston', 'England', 'Actor'),
    (10, 'Idris Elba', null, 'Actor'),
    (11, 'Pom Klementieff', 'Canada', 'Actor'),
    (12, 'Tom Holland', 'England', 'Actor'),
    (13, 'Brie Larson', '', 'Actor'),
    (14, 'Jon Favreau', 'USA', 'Director'),
    (15, 'Joe Russo', 'USA', 'Director'),
    (16, 'Alan Taylor', 'USA', 'Director'),
    (17, 'Guy Ritchie', 'Britain', 'Director'),
    (18, 'James Gunn', 'Britain', 'Director'),
    (19, 'Lenny Abramson', 'Ireland', 'Director');


INSERT INTO Movie(Movie_ID, Title, Genre, Director_ID)
VALUES (1, 'Iron Man', 'Superhero', 14),
    (2, 'Captain America: The Winter Soldier', 'Superhero', 15),
    (3, 'Thor: The Dark World', 'Superhero', 16),
    (4, 'Avengers: Age of Ultron', 'Superhero', 15),
    (5, 'Sherlock Holms', 'Detective', 17),
    (6, 'Guardians of the Galaxy', 'Action', 18),
    (7, 'High-rise', 'Drama', 19),
    (8, 'Spider-Man 2', 'Action/Adventure', 18),
    (9, 'Room', 'Thriller', 17);

INSERT INTO Movie_Actor(Movie_ID, Actor_ID)
VALUES (1, 1),
    (1, 2),
    (3, 7),
    (1, 5),
    (2, 3),
    (2, 4),
    (4, 1),
    (4, 2),
    (4, 3),
    (4, 4),
    (4, 5),
    (4, 6),
    (4, 7),
    (4, 8),
    (4, 9),
    (4, 10),
    (4, 11),
    (4, 12),
    (4, 13),
    (5, 6),
    (5, 1),
    (6, 8),
    (6, 11),
    (7, 9),
```

```
    (8, 12),
    (9, 13);

INSERT INTO Session(Session_ID, Date, Movie_ID, Hall_ID)
VALUES (1, '2021-12-21 11:00', 2, 1),
    (2, '2021-12-22 14:00', 2, 3),
    (3, '2021-12-21 15:30', 2, 4),
    (4, '2021-12-24 21:00', 4, 5),
    (5, '2021-12-25 11:00', 5, 7),
    (6, '2021-12-23 14:20', 6, 8),
    (7, '2021-12-26 18:10', 4, 9),
    (8, '2021-12-24 19:00', 2, 3),
    (9, '2021-12-22 20:20', 7, 4),
    (10, '2021-12-20 15:40', 8, 11),
    (11, '2021-12-21 10:00', 7, 12),
    (12, '2021-12-27 11:30', 4, 10),
    (13, '2021-12-26 12:50', 7, 6),
    (14, '2021-12-23 22:00', 5, 5),
    (15, '2021-12-25 23:30', 6, 4),
    (16, '2021-12-24 09:30', 6, 1),
    (17, '2021-12-26 17:35', 5, 12),
    (18, '2021-12-27 13:20', 4, 11),
    (19, '2021-12-21 14:30', 5, 10),
    (20, '2021-12-22 18:20', 8, 6);

INSERT INTO Client(Client_ID, Last_Name, First_Name, Phone, Email)
VALUES (1, 'Miroshnikov', 'Vladislav', '89281060065', null),
    (2, 'Korneev', 'Danil', '89281432133', 'danil.korneev@gmail.com'),
    (3, 'Shatov', 'Michail', '+79521465438', 'michail.shatov@yandex.ru'),
    (4, 'Chernev', 'Alexander', '89321078899', 'alex.chernev2010@jetbrains.com'),
    (5, 'Gordey', 'Vladimir', '89089031485', null);

INSERT INTO Movie_Order(Order_ID, Client_ID, Session_ID, Simple_seat_count, Premium_seat_count, Price)
VALUES (1, 1, 3, 3, default, 540),
    (2, 1, 6, 0, 2, 290),
    (3, 3, 9, 2, 1, 590),
    (4, 4, 16, 4, default, 800),
    (5, 4, 20, 8, 2, 2370),
    (6, 5, 13, 1, default, 210);


-----------------------------------------------------------------
-- Индексы
-----------------------------------------------------------------
CREATE INDEX index_session on Session (Movie_ID, Hall_ID);
CREATE CLUSTERED INDEX index_movie_actor on Movie_Actor (Movie_ID, Actor_ID);
CREATE INDEX index_movie on Movie (Title, Director_ID);

-----------------------------------------------------------------
-- Удаление таблиц
-----------------------------------------------------------------
DROP TABLE Movie_Order;
DROP TABLE Client;
DROP TABLE Movie_Actor;
DROP TABLE Session;
DROP TABLE Hall;
DROP TABLE Movie;
DROP TABLE Cinema;
DROP TABLE Movie_Person;
DROP TABLE Cinema_Network;

-----------------------------------------------------------------
-- Удаление индексов
```

```
-----------------------------------------------------------
DROP INDEX Session.index_session;
DROP INDEX Movie_Actor.index_movie_actor;
DROP INDEX Movie.index_movie;

-----------------------------------------------------------
-- Удаление хранимых процедур и функций
-----------------------------------------------------------
DROP FUNCTION check_film_in_cinema;
DROP PROCEDURE add_new_client_to_db;
DROP PROCEDURE add_new_movie_to_db;

-----------------------------------------------------------
-- Удаление триггеров
-----------------------------------------------------------
DROP TRIGGER movie_insert;
DROP TRIGGER order_transaction;
DROP TRIGGER movie_actor_insert;

-----------------------------------------------------------
-- Удаление представлений
-----------------------------------------------------------
DROP VIEW new_films;
DROP VIEW today_sessions;
DROP VIEW actual_clients;
```