

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ
по индивидуальному домашнему заданию
по дисциплине «Базы данных»
Тема 32: Разработка базы данных для обеспечения продажи билетов в
кинотеатр

Студентка гр. 2308

Рымарь М.И.

Преподаватель

Новакова Н.Е.

Санкт-Петербург

2024

ЗАДАНИЕ НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

Студентка Рымарь М.И.

Группа 2308

Тема работы: проектирование базы данных «Кинотеатр»

Исходные данные:

База данных (БД) для продажи билетов в кинотеатр обеспечивает в режиме диалога доступ к информации о фильмах, сеансах, залах, доступных местах и покупателях, а также автоматизирует процесс бронирования и продажи билетов.

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 27.10.2024

Дата сдачи реферата: 05.12.2024

Дата защиты реферата: 10.12.2024

Студентка

Рымарь М.И.

Преподаватель

Новакова Н.Е.

АННОТАЦИЯ

В процессе выполнения индивидуального домашнего задания была спроектирована и создана база данных «Кинотеатр». При проектировании и создании были использованы знания, полученные в процессе изучения курса «Базы данных». В рамках работы были созданы таблицы и ограничения целостности, разработаны объекты промежуточного слоя (представления, хранимые процедуры, UDF-ы), построена диаграмма базы данных, а также выбрана стратегия резервного копирования.

SUMMARY

As part of the individual assignment, the "Cinema" database was designed and implemented. The design and implementation process were based on knowledge gained during the "Databases" course. The work included creating tables and integrity constraints, developing intermediate layer objects (views, stored procedures, UDFs), constructing a database diagram, and selecting a backup strategy.

СОДЕРЖАНИЕ

	Введение	5
1.	Реализация БД	7
1.1	Проектирование БД	7
1.2	Создание БД	11
1.3	Создание таблиц и ограничений целостности	13
2.	Заполнение таблиц данными	21
3.	Создание объектов промежуточного слоя	27
3.1	Создание представлений	27
3.2	Создание хранимых процедур	29
3.3	Создание UDF	32
4.	Резервное копирование	37
5.	Процедура восстановления БД	39
	Заключение	40
	Список использованных источников	42

ВВЕДЕНИЕ

База данных (БД) «Кинотеатр» предназначена для автоматизации процессов продажи билетов и управления информацией о фильмах, сеансах, залах и клиентах. Она предоставляет доступ в режиме диалога к информации о доступных местах на сеансах, бронированиях, покупателях и текущих продажах.

В рамках предметной области учитываются сведения о фильмах (название, жанр, продолжительность, возрастные ограничения), залах (номер, вместимость, схема мест) и сеансах (дата, время, цена билета, зал). Также реализован учет информации о клиентах и их бронированиях. В процессе выполнения работы были созданы объекты промежуточного слоя (представления, хранимые процедуры, UDF), а также разработана стратегия резервного копирования для обеспечения сохранности данных.

Цель: Закрепить теоретические знания, полученные в рамках курса «Базы данных», и приобрести практические навыки проектирования и разработки баз данных. Основное внимание уделено созданию структуры данных, объектов промежуточного слоя и разработке стратегии резервного копирования.

Формулировка задания: Для выполнения работы выбрана база данных «Кинотеатр». Необходимо выполнить следующие этапы:

1. Спроектировать структуру базы данных.
2. Создать базу данных.
3. Реализовать таблицы и ограничения целостности.
4. Заполнить таблицы тестовыми данными.
5. Создать объекты промежуточного слоя (представления, хранимые процедуры, UDF).
6. Разработать стратегию резервного копирования.

Краткое описание предметной области: Предметная область представляет собой управление продажей билетов в кинотеатр. Необходимо учитывать информацию о фильмах (название, жанр, возрастные ограничения, продолжительность), залах (номер зала, схема мест, вместимость), и сеансах (дата, время, стоимость билета). Для каждого сеанса хранится информация о доступных местах и текущих бронированиях. Клиенты могут бронировать билеты на определенные места. БД проектируется с учетом следующих условий:

- Один фильм может демонстрироваться в нескольких залах на разных сеансах.
- Один клиент может бронировать несколько билетов на один или несколько сеансов.
- Каждый зал имеет уникальную схему размещения мест.

Такой подход позволит оптимизировать работу кинотеатра, ускорить процесс бронирования и повысить удобство для клиентов.

1. РЕАЛИЗАЦИЯ БД

1.1. ПРОЕКТИРОВАНИЕ БД

Для проектирования базы данных «Кинотеатр» будем использовать метод ER-диаграмм. Выделим основные сущности и их атрибуты:

1. **ФИЛЬМ** (номер_фильма*, название, жанр, возрастное_ограничение, продолжительность);
2. **ЗАЛ** (номер_зала*, вместимость);
3. **МЕСТО** (номер_зала*, ряд*, место*, статус);
4. **СЕАНС** (номер_сеанса*, дата, время, цена_билета, номер_зала, номер_фильма);
5. **КЛИЕНТ** (номер_клиента*, ФИО, телефон, email);
6. **БРОНИРОВАНИЕ** (номер_бронирования*, номер_сеанса, номер_клиента, место).

Рассмотрим связи между сущностями:

1. **СЕАНС** и **ФИЛЬМ** (рисунок 1). Один фильм может быть показан на нескольких сеансах. Каждый сеанс связан только с одним фильмом. Следовательно, это связь **1:М**, которая преобразуется в две таблицы:

ФИЛЬМ (номер_фильма*, название, жанр, возрастное_ограничение, продолжительность);

СЕАНС (номер_сеанса*, дата, время, цена_билета, номер_зала, номер_фильма).



Рисунок 1 – ER диаграмма для связи сеанс-фильм

2. **СЕАНС** и **ЗАЛ** (рисунок 2). Каждый сеанс проводится в одном зале, но один зал может быть использован для множества сеансов. Это также связь **1:M**, преобразуемая в две таблицы:

ЗАЛ (номер_зала*, вместимость);

СЕАНС (номер_сеанса*, дата, время, цена_билета, номер_зала, номер_фильма).

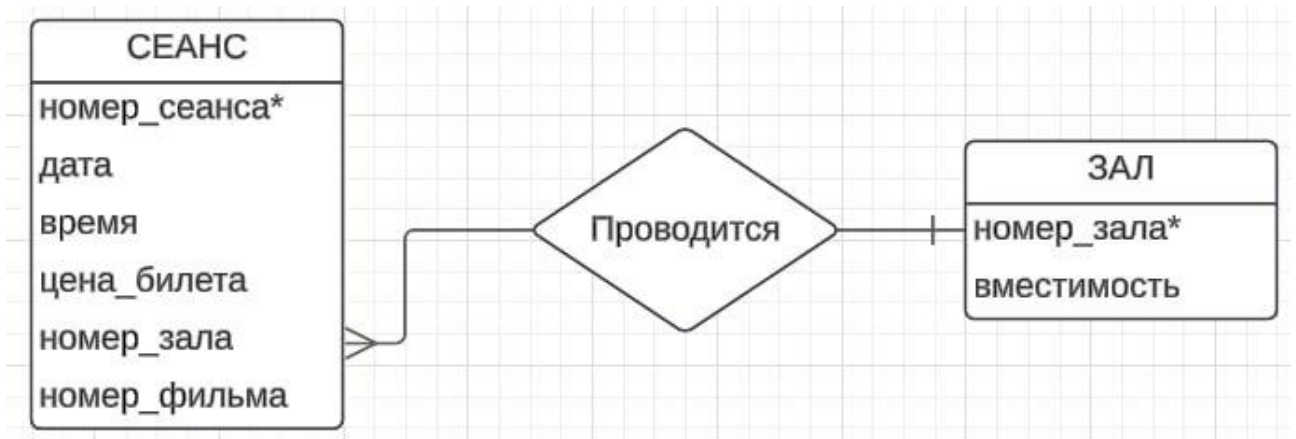


Рисунок 2 – ER-диаграмма для связи сеанс-зал

3. **ЗАЛ** и **МЕСТО** (рисунок 3). Один зал содержит множество мест, и каждое место уникально идентифицируется рядом и номером. Это связь **1:M**, которая преобразуется в две таблицы:

ЗАЛ (номер_зала*, вместимость);

МЕСТО (номер_зала*, ряд*, место*, статус).



Рисунок 3 – ER-диаграмма для связи зал-место

4. **СЕАНС** и **КЛИЕНТ** через **БРОНИРОВАНИЕ** (рисунок 4). Один клиент может забронировать несколько мест на один или несколько сеансов. Одно место на сеансе может быть забронировано только одним клиентом. Это

связь **M:N**, которая преобразуется в три таблицы (связь реализуется через дополнительную таблицу):

СЕАНС (номер_сеанса*, дата, время, цена_билета, номер_зала, номер_фильма).

КЛИЕНТ (номер_клиента*, ФИО, телефон, email).

БРОНИРОВАНИЕ (номер_бронирования*, номер_сеанса, номер_клиента, номер_зала, ряд, место).

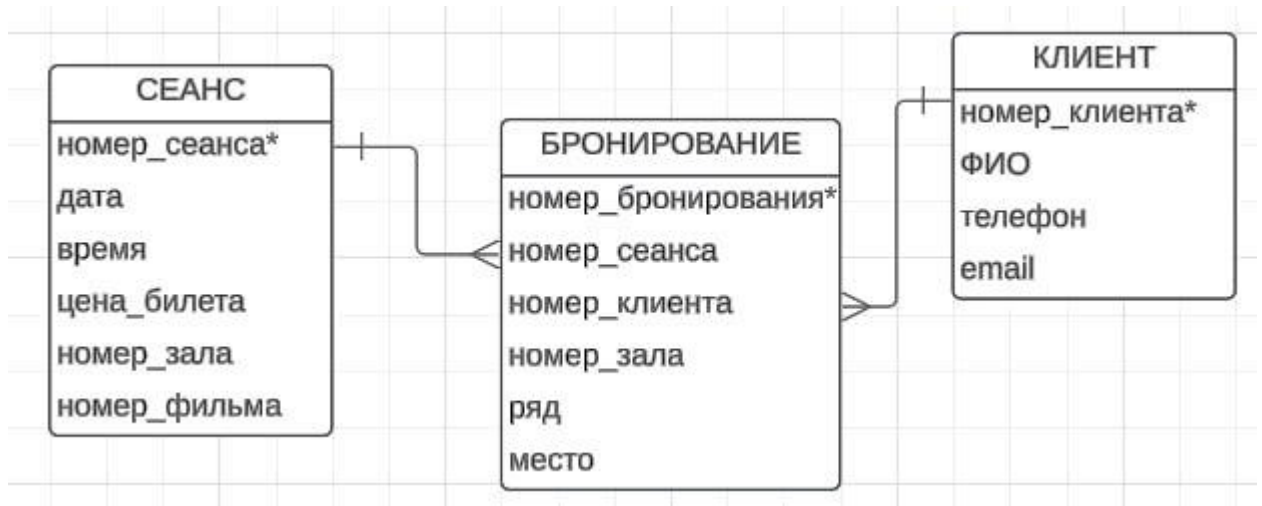


Рисунок 4 – ER-диаграмма для связи сеанс-клиент через бронирование

5. **БРОНИРОВАНИЕ** и **МЕСТО** (рисунок 5). Одна бронь содержит одно место, а каждое место может быть забронировано несколько раз для разных сеансов. Это связь **1:M**, которая преобразуется в две таблицы:

БРОНИРОВАНИЕ (номер_бронирования*, номер_сеанса, номер_клиента, номер_зала, ряд, место);

МЕСТО (номер_зала*, ряд*, место*, статус).

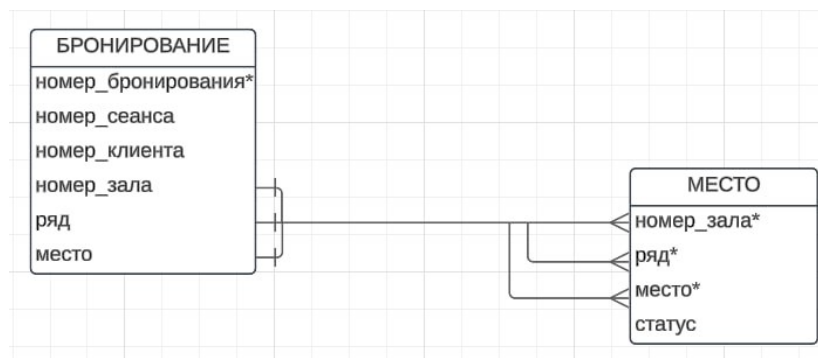


Рисунок 5 – ER-диаграмма для связи бронирование-место

Итоговая структура базы данных:

- **ФИЛЬМ** (номер_фильма*, название, жанр, возрастное_ограничение, продолжительность);
- **ЗАЛ** (номер_зала*, вместимость);
- **МЕСТО** (номер_зала*, ряд*, место*, статус);
- **СЕАНС** (номер_сеанса*, дата, время, цена_билета, номер_зала, номер_фильма);
- **КЛИЕНТ** (номер_клиента*, ФИО, телефон, email);
- **БРОНИРОВАНИЕ** (номер_бронирования*, номер_сеанса, номер_клиента, номер_зала, ряд, место).

Полная ER-диаграмма представлена на рисунке 6.

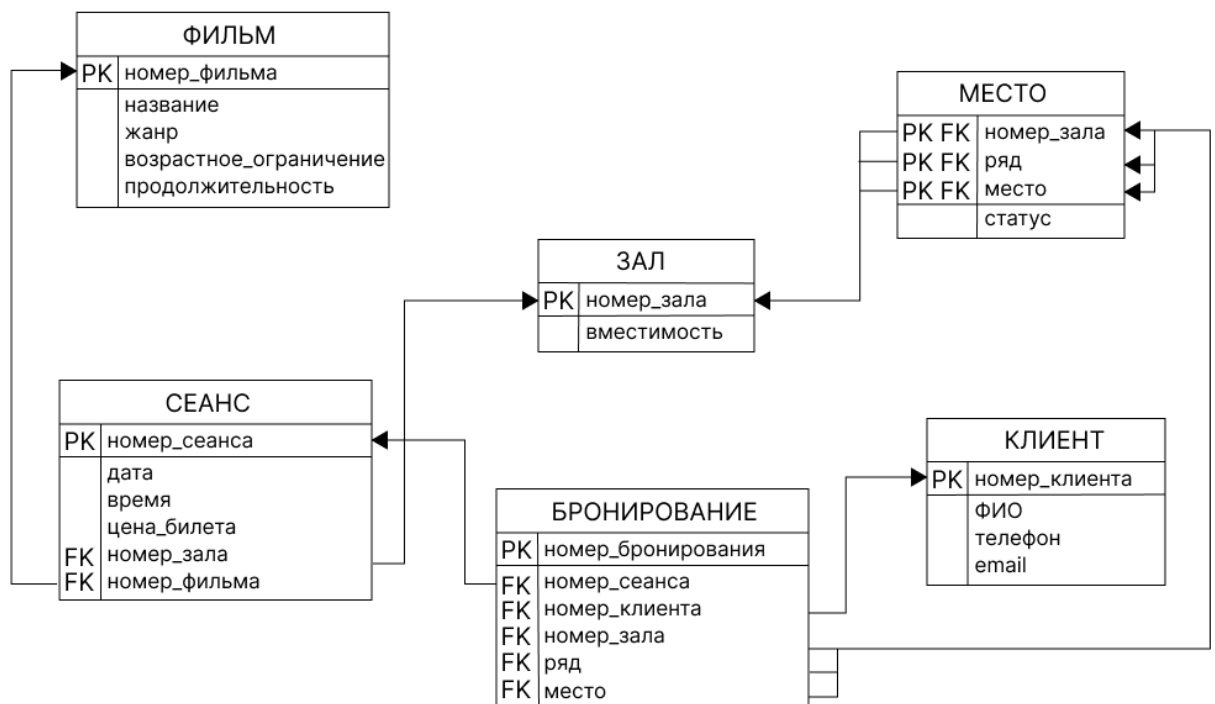


Рисунок 6 – ER-диаграмма всех связей базы данных

1.2. СОЗДАНИЕ БД

Создание базы данных следующей командой:

```
CREATE database Кинотеатр;
```

Результат выполнения представлен на рисунке 7.

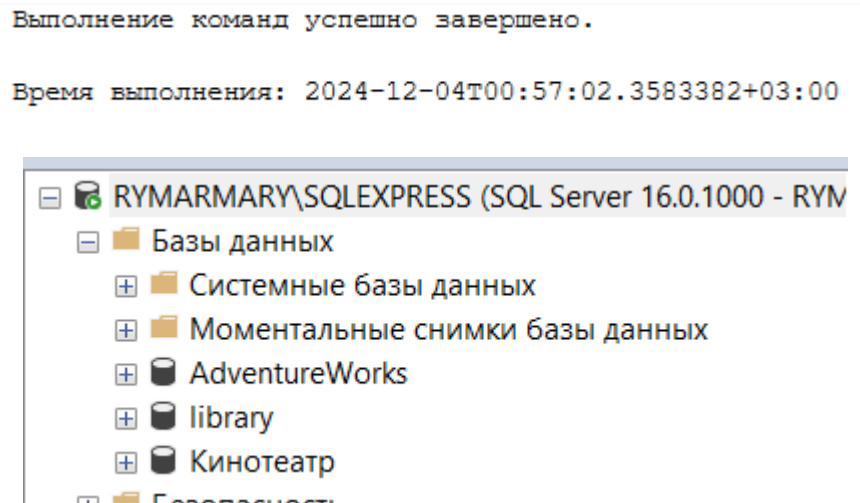


Рисунок 7 – Создание базы данных Кинотеатр

Далее создадим схему, чтобы в названиях таблиц не было dbo и проверим наличие созданной схемы. Результат показан на рисунке 8.

```
CREATE SCHEMA Кинотеатр;  
GO  
SELECT * FROM sys.schemas  
WHERE name = 'Кинотеатр';
```

	name	schema_id	principal_id
1	Кинотеатр	5	1

(затронута одна строка)

Время выполнения: 2024-12-11T21:13:22.0721269+03:00

Рисунок 8 – Создание и проверка схемы Кинотеатр

Далее, чтобы было удобно работать из-под любого пользователя, сделаем схему публичной. Результат выполнения команд показан на рисунке 9.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON  
SCHEMA::Кинотеатр TO public;
```

Выполнение команд успешно завершено.

Время выполнения: 2024-12-04T01:23:24.5269686+03:00

Рисунок 9 – Результат выполнения команд

1.3. СОЗДАНИЕ ТАБЛИЦ И ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Для проверки создадим первую таблицу в базе данных ФИЛЬМ со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры фильма		Имя таблицы: ФИЛЬМ	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 1					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Уникальный номер фильма	номер_фильма	int	P	IDENTITY(1,1)
2	Название фильма	название	nvarchar(2 55)		NOT NULL
3	Жанр фильма	жанр	nvarchar(1 00)		NOT NULL
4	Возрастное ограничение	возрастное_ограничение	tinyint		CHECK (возрастное _ограничен ие BETWEEN 0 AND 18)
5	Продолжительность фильма	Продолжительность	int		CHECK (продолжи тельность > 0)

Создание таблицы ФИЛЬМ:

```
CREATE TABLE Кинотеатр.ФИЛЬМ (  
    номер_фильма INT IDENTITY(1,1) PRIMARY KEY,  
    название NVARCHAR(255) NOT NULL,  
    жанр NVARCHAR(100),
```

```

        возрастное_ограничение TINYINT CHECK
(возрастное_ограничение BETWEEN 0 AND 18),
        продолжительность INT CHECK (продолжительность >
0)
);

```

Результат выполнения команд и проверки созданной таблицы представлен на рисунке 10.

Выполнение команд успешно завершено.

Время выполнения: 2024-12-04T01:24:17.5517667+03:00

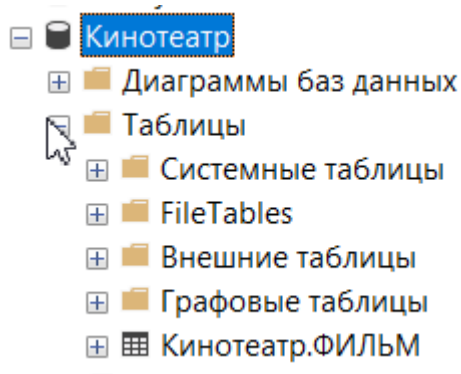


Рисунок 10 – Результат создания таблицы ФИЛЬМ

Так как первая таблица была успешно создана, можно создать оставшиеся таблицы базы данных.

Создание таблицы ЗАЛ со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры зала		Имя таблицы: ЗАЛ	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 2					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Уникальный номер зала	номер_зала	int	P	IDENTITY(1,1)
2	Вместимость зала	вместимость	int		NOT NULL CHECK (вместимос ть > 0)

Создание таблицы ЗАЛ:

```
CREATE TABLE Кинотеатр.ЗАЛ (
    номер_зала INT IDENTITY(1,1) PRIMARY KEY,
    вместимость INT NOT NULL CHECK (вместимость > 0)
);
```

Создание таблицы СЕАНС со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры сеанса		Имя таблицы: СЕАНС	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 3					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Уникальный номер сеанса	номер_сеанса	int	P	IDENTITY(1,1)
2	Дата сеанса	дата	date		NOT NULL
3	Время сеанса	время	time		NOT NULL
4	Цена билета	цена_билета	decimal(1 0,2)		CHECK (цена_биле та > 0)
5	Номер зала	номер_зала	int	F	NOT NULL
6	Номер фильма	номер_фильма	int	F	NOT NULL

Создание таблицы СЕАНС:

```
CREATE TABLE Кинотеатр.СЕАНС (
    номер_сеанса INT IDENTITY(1,1) PRIMARY KEY,
    дата DATE NOT NULL,
    время TIME NOT NULL,
    цена_билета DECIMAL(10, 2) CHECK (цена_билета >
0),
    номер_зала INT NOT NULL,
    номер_фильма INT NOT NULL,
    CONSTRAINT FK_СЕАНС_ЗАЛ FOREIGN KEY (номер_зала)
REFERENCES Кинотеатр.ЗАЛ(номер_зала)
ON DELETE CASCADE ON UPDATE CASCADE,
```


`CONSTRAINT FK_СЕАНС_ФИЛЬМ FOREIGN KEY`

`(номер_фильма) REFERENCES Кинотеатр.ФИЛЬМ(номер_фильма)`
`);`

Создание таблицы КЛИЕНТ со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры клиента		Имя таблицы: КЛИЕНТ	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 4					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Уникальный номер клиента	номер_клиента	int	P	IDENTITY(1,1)
2	ФИО клиента	ФИО	nvarchar(2 55)		NOT NULL
3	Телефон клиента	телефон	nvarchar(2 0)		
4	Электронная почта клиента	email	nvarchar(2 55)		CHECK (email LIKE '% @%.%')

Создание таблицы КЛИЕНТ:

```
CREATE TABLE Кинотеатр.КЛИЕНТ (
    номер_клиента INT IDENTITY(1,1) PRIMARY KEY,
    ФИО NVARCHAR(255) NOT NULL,
    телефон NVARCHAR(20),
    email NVARCHAR(255) CHECK (email LIKE '%@%.%')
);
```

Создание таблицы БРОНИРОВАНИЕ со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры бронирования		Имя таблицы: БРОНИРОВАНИЕ	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 5					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Уникальный номер бронирования	номер_бронирования	int	P	IDENTITY(1,1),
2	Номер сеанса	номер_сеанса	int	F	
3	Номер клиента	номер_клиента	int	F	
4	Номер зала	номер_зала		F	
5	Номер ряда	ряд	int	F	
6	Номер места	место	int	F	

Создание таблицы БРОНИРОВАНИЕ:

```
CREATE TABLE Кинотеатр.БРОНИРОВАНИЕ (
    номер_бронирования INT PRIMARY KEY IDENTITY(1,1),
    номер_сеанса INT,
    номер_клиента INT,
    номер_зала INT,
    ряд INT,
    место INT,
    FOREIGN KEY (номер_сеанса) REFERENCES
    Кинотеатр.СЕАНС(номер_сеанса),
    FOREIGN KEY (номер_клиента) REFERENCES
    Кинотеатр.КЛИЕНТ(номер_клиента),
    FOREIGN KEY (номер_зала, ряд, место) REFERENCES
    Кинотеатр.МЕСТО(номер_зала, ряд, место)
```

);

Создание таблицы МЕСТО со следующим описанием:

Описание структуры таблицы БД		Наименование таблицы БД: Таблица общие параметры места		Имя таблицы: МЕСТО	
Дата разработки: 03.12.2024					
Порядковый номер таблицы: 6					
№ п/п	Наименование поля	Спецификация данных			
		Имя поля	Тип данных	Ключ	Ограничения целостности
1	Номер зала	номер_зала	int	P, F	NOT NULL
2	Номер ряда	ряд	int	P	NOT NULL
3	Номер места	место	int	P	NOT NULL
4	Статус	статус	nvarchar(50)		CHECK (статус IN ('свободно', 'занято', 'зарезервир овано'))

Создание таблицы МЕСТО:

```
CREATE TABLE Кинотеатр.МЕСТО (  
    номер_зала INT,  
    ряд INT,  
    место INT,  
    статус NVARCHAR(50) CHECK (статус IN ('свободно',  
'занято', 'зарезервировано')),  
    PRIMARY KEY (номер_зала, ряд, место),  
    CONSTRAINT FK_МЕСТО_ЗАЛ FOREIGN KEY (номер_зала)  
REFERENCES Кинотеатр.ЗАЛ(номер_зала)  
ON DELETE CASCADE ON UPDATE CASCADE
```

) ;

Результат создания таблиц посмотрим в обозревателе объектов (см. рисунок 11).

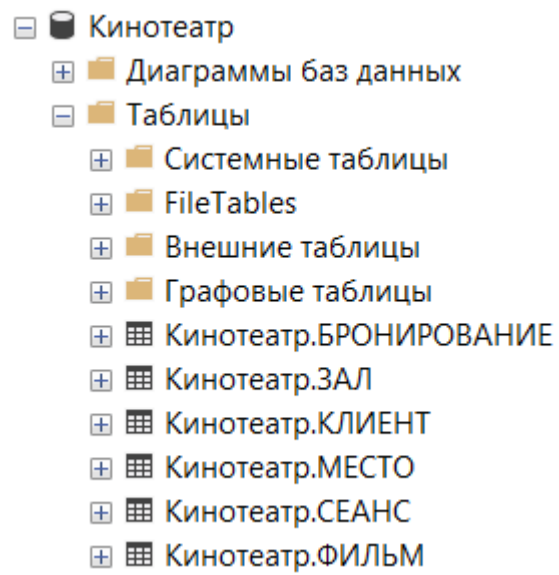


Рисунок 11 – Созданные таблицы

2. ЗАПОЛНЕНИЕ ТАБЛИЦ ДАННЫМИ

Далее заполним созданные таблицы данными.

Заполнение таблицы ФИЛЬМ:

```
INSERT INTO Кинотеатр.ФИЛЬМ (название, жанр,  
возрастное_ограничение, продолжительность)  
VALUES  
( 'Интерстеллар', 'Фантастика', 12, 169),  
( 'Одержимость', 'Драма', 16, 107),  
( 'Мстители: Финал', 'Экшн', 12, 181),  
( 'Начало', 'Фантастика', 16, 148),  
( 'Темный рыцарь', 'Боевик', 16, 152),  
( 'Крестный отец', 'Криминал', 18, 175),  
( 'Титаник', 'Драма', 12, 195),  
( 'Властелин колец: Братство кольца', 'Фэнтези', 12,  
178),  
( 'Матрица', 'Фантастика', 16, 136),  
( 'Джокер', 'Драма', 18, 122);  
SELECT * FROM Кинотеатр.ФИЛЬМ;
```

Результат представлен на рисунке 12.

номер_фильма	название	жанр	возрастное_ограничение	продолжительность
1	Интерстеллар	Фантастика	12	169
2	Одержимость	Драма	16	107
3	Мстители: Финал	Экшн	12	181
4	Начало	Фантастика	16	148
5	Темный рыцарь	Боевик	16	152
6	Крестный отец	Криминал	18	175
7	Титаник	Драма	12	195
8	Властелин колец: Братство кольца	Фэнтези	12	178
9	Матрица	Фантастика	16	136
10	Джокер	Драма	18	122

Рисунок 12 – Результат заполнения таблицы ФИЛЬМ

Заполнение таблицы ЗАЛ:

```
INSERT INTO Кинотеатр.ЗАЛ (местимость)
VALUES
(50) ,
(75) ,
(100) ,
(40) ,
(60) ,
(90) ,
(120) ,
(80) ,
(110) ,
(130) ;

SELECT * FROM Кинотеатр.ЗАЛ;
```

Результат представлен на рисунке 13.

номер_зала	местимость
1	50
2	75
3	100
4	40
5	60
6	90
7	120
8	80
9	110
10	130

Рисунок 13 – Результат заполнения таблицы ЗАЛ

Заполнение таблицы МЕСТО:

```
INSERT INTO Кинотеатр.МЕСТО (номер_зала, ряд, место,
статус)
VALUES
(1, 1, 1, 'Свободно') ,
(1, 1, 2, 'Занято') ,
```

```
(1, 2, 1, 'Свободно'),
(1, 2, 2, 'Свободно'),
(1, 2, 3, 'Занято'),
(2, 1, 1, 'Свободно'),
(2, 1, 2, 'Занято'),
(2, 2, 1, 'Свободно'),
(2, 2, 2, 'Свободно'),
(2, 2, 3, 'Свободно');
```

```
SELECT * FROM Кинотеатр.МЕСТО;
```

Результат представлен на рисунке 14.

номер_зала	ряд	место	статус
1	1	1	Свободно
1	1	2	Занято
1	2	1	Свободно
1	2	2	Свободно
1	2	3	Занято
2	1	1	Свободно
2	1	2	Занято
2	2	1	Свободно
2	2	2	Свободно
2	2	3	Свободно

Рисунок 14 – Результат заполнения таблицы МЕСТО

Заполнение таблицы СЕАНС:

```
INSERT INTO Кинотеатр.СЕАНС (дата, время,
цена_билета, номер_зала, номер_фильма)
```

```
VALUES
```

```
('2024-12-05', '18:00', 350, 1, 1),
('2024-12-05', '20:30', 400, 2, 2),
('2024-12-06', '15:00', 300, 3, 3),
('2024-12-06', '17:30', 350, 4, 4),
('2024-12-07', '19:00', 450, 5, 5),
('2024-12-07', '21:00', 500, 6, 6),
('2024-12-08', '14:00', 250, 7, 7),
```

```
( '2024-12-08' , '16:00' , 350 , 8 , 8 ) ,
( '2024-12-09' , '18:30' , 400 , 9 , 9 ) ,
( '2024-12-09' , '20:00' , 450 , 10 , 10 ) ;
```

```
SELECT * FROM Кинотеатр.СЕАНС;
```

Результат представлен на рисунке 15.

номер_сеанса	дата	время	цена_билета	номер_зала	номер_фильма
1	2024-12-05	18:00:00.0000000	350.00	1	1
2	2024-12-05	20:30:00.0000000	400.00	2	2
3	2024-12-06	15:00:00.0000000	300.00	3	3
4	2024-12-06	17:30:00.0000000	350.00	4	4
5	2024-12-07	19:00:00.0000000	450.00	5	5
6	2024-12-07	21:00:00.0000000	500.00	6	6
7	2024-12-08	14:00:00.0000000	250.00	7	7
8	2024-12-08	16:00:00.0000000	350.00	8	8
9	2024-12-09	18:30:00.0000000	400.00	9	9
10	2024-12-09	20:00:00.0000000	450.00	10	10

Рисунок 15 – Результат заполнения таблицы СЕАНС

Заполнение таблицы КЛИЕНТ:

```
INSERT INTO Кинотеатр.КЛИЕНТ (ФИО, телефон, email)
VALUES
```

```
( 'Иванов Иван Иванович' , '79991234567' ,
'ivanov@example.com' ) ,
( 'Петров Петр Петрович' , '79876543210' ,
'petrov@example.com' ) ,
( 'Сидорова Анна Васильевна' , '79998765432' ,
'sidorova@example.com' ) ,
( 'Кузнецов Алексей Викторович' , '79871234567' ,
'kuznetsov@example.com' ) ,
( 'Смирнова Ольга Александровна' , '79994567890' ,
'smirnova@example.com' ) ,
( 'Морозов Сергей Дмитриевич' , '79879879812' ,
'morozov@example.com' ) ,
```



```

('Волкова Екатерина Владимировна', '79987654321',
'volkova@example.com'),
('Попов Дмитрий Олегович', '79876567890',
'popov@example.com'),
('Семенов Николай Сергеевич', '79984321678',
'semenov@example.com'),
('Григорьева Татьяна Николаевна', '79879865432',
'grigorieva@example.com');

SELECT * FROM Кинотеатр.КЛИЕНТ;

```

Результат представлен на рисунке 16.

номер_клиента	ФИО	телефон	email
1	Иванов Иван Иванович	79991234567	ivanov@example.com
2	Петров Петр Петрович	79876543210	petrov@example.com
3	Сидорова Анна Васильевна	79998765432	sidorova@example.com
4	Кузнецов Алексей Викторович	79871234567	kuznetsov@example.com
5	Смирнова Ольга Александровна	79994567890	smirnova@example.com
6	Морозов Сергей Дмитриевич	79879879812	morozov@example.com
7	Волкова Екатерина Владимировна	79987654321	volkova@example.com
8	Попов Дмитрий Олегович	79876567890	popov@example.com
9	Семенов Николай Сергеевич	79984321678	semenov@example.com
10	Григорьева Татьяна Николаевна	79879865432	grigorieva@example.com

Рисунок 16 – Результат заполнения таблицы КЛИЕНТ

Заполнение таблицы БРОНИРОВАНИЕ:

```

INSERT INTO Кинотеатр.БРОНИРОВАНИЕ (номер_сеанса,
номер_клиента, ряд, место)
VALUES
(1, 1, 1, 1),
(1, 2, 1, 2),
(2, 3, 2, 1),
(2, 4, 2, 2),
(1, 5, 2, 2),
(1, 6, 2, 1),
(2, 7, 2, 3),

```

(1, 8, 1, 1),

(2, 9, 1, 2),

(1, 10, 1, 1);

SELECT * FROM Кинотеатр.БРОНИРОВАНИЕ;

Результат представлен на рисунке 17.

номер_бронирования	номер_сеанса	номер_клиента	ряд	место	номер_зала
16	1	1	1	1	1
17	1	2	1	2	1
18	2	3	2	1	1
21	2	4	2	2	1
24	1	6	2	2	2
25	1	5	2	1	2
26	2	7	2	3	2
27	1	8	1	1	1
28	2	9	1	2	1
29	1	10	1	1	2

Рисунок 17 – Результат заполнения таблицы БРОНИРОВАНИЕ

3. СОЗДАНИЕ ОБЪЕКТОВ ПРОМЕЖУТОЧНОГО СЛОЯ

3.1. СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ

Представление 1: Актуальные_сеансы

Это представление показывает информацию о сеансах, которые еще не прошли. Оно включает дату, время, название фильма, номер зала и цену билета. Проверка представления показана на рисунке 18.

```
-- Создание представления
CREATE VIEW Кинотеатр.Актуальные_сеансы AS
SELECT
    СЕАНС.дата,
    СЕАНС.время,
    ФИЛЬМ.название AS фильм,
    СЕАНС.номер_зала,
    СЕАНС.цена_билета
FROM
    Кинотеатр.СЕАНС
JOIN
    Кинотеатр.ФИЛЬМ
ON
    СЕАНС.номер_фильма = ФИЛЬМ.номер_фильма
WHERE
    СЕАНС.дата >= CAST(GETDATE() AS DATE);
-- Проверка
SELECT * FROM Кинотеатр.Актуальные_сеансы;
```

дата	время	фильм	номер_зала	цена_билета
2024-12-06	15:00:00.0000000	Мстители: Финал	3	300.00
2024-12-06	17:30:00.0000000	Начало	4	350.00
2024-12-07	19:00:00.0000000	Темный рыцарь	5	450.00
2024-12-07	21:00:00.0000000	Крестный отец	6	500.00
2024-12-08	14:00:00.0000000	Титаник	7	250.00
2024-12-08	16:00:00.0000000	Властелин колец: Братство кольца	8	350.00
2024-12-09	18:30:00.0000000	Матрица	9	400.00
2024-12-09	20:00:00.0000000	Джокер	10	450.00

Рисунок 18 – Результат проверки представления 1

Представление 2: Фильмы_по_жанру

Выводит список фильмов, сгруппированных по жанрам, с указанием их общего количества в каждом жанре. Проверка представления показана на рисунке 19.

```
-- Создание представления
CREATE VIEW Кинотеатр.Фильмы_по_жанру AS
SELECT
    жанр,
    COUNT (*) AS количество_фильмов
FROM
    Кинотеатр.ФИЛЬМ
GROUP BY
    жанр;

-- Проверка
SELECT * FROM Кинотеатр.Фильмы_по_жанру;
```

жанр	количество_фильмов
Боевик	1
Драма	3
Криминал	1
Фантастика	3
Фэнтези	1
Экшн	1

Рисунок 19 – Результат проверки представления 2

3.2. СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР

Процедура 1: Добавить_Бронирование

Эта процедура позволяет добавить новое бронирование, проверяя, что место не занято на выбранный сеанс. Проверка процедуры показана на рисунке 20.

```
-- Создание процедуры
CREATE PROCEDURE Кинотеатр.Добавить_Бронирование
    @номер_сеанса INT,
    @номер_клиента INT,
    @ряд INT,
    @место INT
AS
BEGIN
    -- Проверка на занятость места
    IF EXISTS (
        SELECT 1
        FROM Кинотеатр.БРОНИРОВАНИЕ
        WHERE номер_сеанса = @номер_сеанса AND ряд =
@ряд AND место = @место
    )
        BEGIN
            PRINT 'Ошибка: Место уже занято.';
            RETURN;
        END
    -- Добавление бронирования
    INSERT INTO Кинотеатр.БРОНИРОВАНИЕ (номер_сеанса,
номер_клиента, ряд, место)
        VALUES (@номер_сеанса, @номер_клиента, @ряд,
@место);
```

```

        PRINT 'Бронирование успешно добавлено.';
END;

-- Проверка процедуры
EXEC Кинотеатр.Добавить_Бронирование
    @номер_сеанса = 1,
    @номер_клиента = 2,
    @ряд = 2,
    @место = 3;

    (затронута одна строка)
    Бронирование успешно добавлено.

    Время выполнения: 2024-12-06T01:46:38.0566167+03:00

```

Рисунок 20 – Результат проверки процедуры 1

Процедура 2: Получить_Доступные_Места

Эта процедура возвращает список доступных мест для указанного сеанса.

Проверка процедуры показана на рисунке 21.

```

-- Создание процедуры
CREATE PROCEDURE Кинотеатр.Получить_Доступные_Места
    @номер_сеанса INT
AS
BEGIN
    SELECT
        МЕСТО.ряд,
        МЕСТО.место,
        МЕСТО.статус
    FROM
        Кинотеатр.МЕСТО
    LEFT JOIN
        Кинотеатр.БРОНИРОВАНИЕ
    ON

```

```

        МЕСТО.номер_зала = (SELECT номер_зала FROM
Кинотеатр.СЕАНС WHERE номер_сеанса = @номер_сеанса)
        AND МЕСТО.ряд = БРОНИРОВАНИЕ.ряд
        AND МЕСТО.место = БРОНИРОВАНИЕ.место
    WHERE
        БРОНИРОВАНИЕ.номер_сеанса IS NULL;
END;
-- Проверка процедуры
EXEC Кинотеатр.Получить_Доступные_Места @номер_сеанса
= 1;

```

ряд	место	статус
1	1	Свободно
1	2	Занято
2	1	Свободно
2	2	Свободно
2	3	Свободно

Рисунок 21 – Результат проверки процедуры 2

3.3. СОЗДАНИЕ UDF

Функция 1: Кинотеатр.Получить_Суммарную_Стоимость_Бронирований

Тип: Скалярная

Возвращает общую стоимость всех бронирований для указанного клиента.

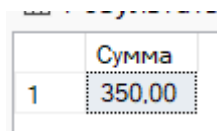
Результат проверки функции представлен на рисунке 22.

```
-- Создание функции

CREATE FUNCTION
Кинотеатр.Получить_Суммарную_Стоимость_Бронирований
( @номер_клиента INT
)
RETURNS MONEY
AS
BEGIN
    DECLARE @сумма MONEY;
    SELECT @сумма = SUM(СЕАНС.цена_билета)
    FROM Кинотеатр.БРОНИРОВАНИЕ Б
    JOIN Кинотеатр.СЕАНС СЕАНС ON Б.номер_сеанса =
СЕАНС.номер_сеанса
    WHERE Б.номер_клиента = @номер_клиента;
    RETURN ISNULL(@сумма, 0);
END;

-- Пример вызова функции

SELECT
Кинотеатр.Получить_Суммарную_Стоимость_Бронирований(1) AS
Сумма;
```



	Сумма
1	350,00

Рисунок 22 – Результат проверки функции 1

Функция 2: Кинотеатр.Доступные_Места_Для_Сеанса

Тип: Табличная (inline table-valued function)

Возвращает таблицу с доступными местами для указанного сеанса.

Результат проверки функции представлен на рисунке 23.

```
-- Создание функции

CREATE FUNCTION Кинотеатр.Доступные_Места_Для_Сеанса
(
    @номер_сеанса INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT
        МЕСТО.ряд,
        МЕСТО.место,
        МЕСТО.статус
    FROM
        Кинотеатр.МЕСТО
    LEFT JOIN
        Кинотеатр.БРОНИРОВАНИЕ
    ON
        МЕСТО.номер_зала = (SELECT номер_зала FROM
        Кинотеатр.СЕАНС WHERE номер_сеанса = @номер_сеанса)
        AND МЕСТО.ряд = БРОНИРОВАНИЕ.ряд
        AND МЕСТО.место = БРОНИРОВАНИЕ.место
    WHERE
        БРОНИРОВАНИЕ.номер_сеанса IS NULL
);

-- Пример вызова функции
```

```
SELECT * FROM
```

```
Кинотеатр.Доступные_Места_Для_Сеанса(1);
```

ряд	место	статус
1	1	Свободно
1	2	Занято
2	1	Свободно
2	2	Свободно
2	3	Свободно

Рисунок 23 – Результат проверки функции 2

Функция 3: Кинотеатр.Количество_Бронирований_Клиента

Тип: Скалярная

Возвращает количество бронирований, сделанных указанным клиентом.

Результат проверки функции представлен на рисунке 24.

```
-- Создание функции
```

```
CREATE FUNCTION
```

```
Кинотеатр.Количество_Бронирований_Клиента
```

```
(
```

```
    @номер_клиента INT
```

```
)
```

```
RETURNS INT
```

```
AS
```

```
BEGIN
```

```
    DECLARE @количество INT;
```

```
    SELECT @количество = COUNT(*)
```

```
    FROM Кинотеатр.БРОНИРОВАНИЕ
```

```
    WHERE номер_клиента = @номер_клиента;
```

```
    RETURN @количество;
```

```
END;
```

```
-- Пример вызова функции
```

```
SELECT Кинотеатр.Количество_Бронирований_Клиента (2)
AS Количество_Бронирований;
```

	Количество_Бронирований
1	2

Рисунок 24 – Результат проверки функции 3

Функция 4: Кинотеатр.Получить_Самый_Популярный_Фильм

Тип: Скалярная

Эта функция возвращает название самого популярного фильма, основываясь на количестве сеансов. Результат проверки показан на рисунке 25.

```
CREATE FUNCTION
Кинотеатр.Получить_Самый_Популярный_Фильм ()
RETURNS VARCHAR (255)
AS
BEGIN
    DECLARE @название VARCHAR (255) ;
    SELECT TOP 1 @название = ФИЛЬМ.название
    FROM Кинотеатр.ФИЛЬМ
    JOIN Кинотеатр.СЕАНС ON ФИЛЬМ.номер_фильма =
СЕАНС.номер_фильма
    GROUP BY ФИЛЬМ.название
    ORDER BY COUNT (*) DESC;
    RETURN @название;
END;
```

```
SELECT Кинотеатр.Получить_Самый_Популярный_Фильм ();
```

	(Отсутствует имя столбца)
1	Джокер

Рисунок 25 – Результат проверки функции 4

Функция 5: Кинотеатр.СредняяЦенаБилетаПоЖанру

Тип: Табличная

Эта функция возвращает среднюю цену билета для каждого жанра фильма.

Результат проверки показан на рисунке 26.

```
CREATE FUNCTION Кинотеатр.СредняяЦенаБилетаПоЖанру ()
RETURNS TABLE
AS
RETURN
(
    SELECT
        Ф.жанр,
        AVG(С.цена_билета) AS средняя_цена
    FROM Кинотеатр.ФИЛЬМ Ф
    JOIN Кинотеатр.СЕАНС С ON Ф.номер_фильма =
        С.номер_фильма
    GROUP BY Ф.жанр
);

SELECT * FROM Кинотеатр.СредняяЦенаБилетаПоЖанру ();
```

	жанр	средняя_цена
1	Боевик	450.000000
2	Драма	366.666666
3	Криминал	500.000000
4	Фантастика	366.666666
5	Фэнтези	350.000000
6	Экшн	300.000000

Рисунок 26 – Результат проверки функции 5

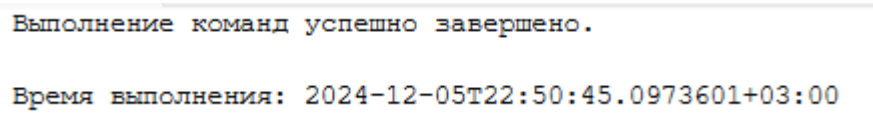
4. РЕЗЕРВНОЕ КОПИРОВАНИЕ

Для обеспечения постоянной сохранности данных и возможности восстановления базы данных после сбоя, необходимо регулярно создавать резервные копии журнала транзакций. Это позволяет отслеживать все изменения, происходящие между полными резервными копированиями базы данных.

Для этого следует выбрать полную модель восстановления базы данных, чтобы обеспечить сохранность всех данных и возможность точного восстановления:

```
ALTER DATABASE GardenWarehouseDB  
SET RECOVERY FULL;
```

Результат выполнения показан на рисунке 25.



Выполнение команд успешно завершено.

Время выполнения: 2024-12-05T22:50:45.0973601+03:00

Рисунок 25 – Результат выполнения команды

Теперь можно выполнять полные резервные копии базы данных, а также создавать резервные копии журнала транзакций для сохранения всех изменений, внесенных в базу данных с момента последнего полного резервного копирования.

Создание резервного копирования:

```
BACKUP DATABASE КиноТеатр  
TO DISK = 'C:\Program Files\Microsoft SQL  
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\КиноТеатр_full.bak  
';  
  
BACKUP LOG КиноТеатр  
TO DISK = 'C:\Program Files\Microsoft SQL  
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\КиноТеатр_logs.bak  
';
```

```

BACKUP DATABASE Кинотеатр
TO DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\Кинотеатр_diff.bak
'

WITH DIFFERENTIAL;

BACKUP LOG Кинотеатр
TO DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\Кинотеатр_logs.bak
'

WITH NOINIT;

```

На рисунке 26 представлен результат выполнения запроса. На рисунке 27 представлены созданные файлы.

```

Обработано 664 страниц для базы данных "Кинотеатр", файл "Кинотеатр" для файла 1.
Обработано 2 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
BACKUP DATABASE успешно обработал 666 страниц за 0.031 секунд (167.716 МБ/сек).
Обработано 3 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
BACKUP LOG успешно обработал 3 страниц за 0.003 секунд (7.812 МБ/сек).
Обработано 112 страниц для базы данных "Кинотеатр", файл "Кинотеатр" для файла 1.
Обработано 2 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
BACKUP DATABASE WITH DIFFERENTIAL успешно обработал 114 страниц за 0.015 секунд (59.114 МБ/сек).
Обработано 7 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 2.
BACKUP LOG успешно обработал 7 страниц за 0.003 секунд (16.927 МБ/сек).

Время выполнения: 2024-12-06T02:07:46.8446293+03:00

```

Рисунок 26 – Резервное копирование БД, результат выполнения запроса




 Кинотеатр_full.bak	06.12.2024 2:07	Файл "BAK"	5 428 КБ
 Кинотеатр_logs.bak	06.12.2024 2:07	Файл "BAK"	208 КБ
 Кинотеатр_diff.bak	06.12.2024 2:07	Файл "BAK"	1 012 КБ

Рисунок 27 – Резервное копирование БД созданные файлы

5. ПРОЦЕДУРА ВОССТАНОВЛЕНИЯ БД

```
RESTORE DATABASE Кинотеатр
FROM DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\Кинотеатр_full.bak
'
WITH NORECOVERY, REPLACE;
RESTORE DATABASE Кинотеатр
FROM DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\Кинотеатр_full.bak
'
WITH NORECOVERY;
RESTORE LOG Кинотеатр
FROM DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL16.SQLEXPRESS\MSSQL\Backup\Кинотеатр_logs.bak
'
WITH RECOVERY;
```

Результат выполнения запроса представлен на рисунке 28.

```
Обработано 664 страниц для базы данных "Кинотеатр", файл "Кинотеатр" для файла 1.
Обработано 2 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
RESTORE DATABASE успешно обработал 666 страниц за 0.014 секунд (371.372 МБ/сек).
Обработано 664 страниц для базы данных "Кинотеатр", файл "Кинотеатр" для файла 1.
Обработано 2 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
RESTORE DATABASE успешно обработал 666 страниц за 0.014 секунд (371.372 МБ/сек).
Обработано 0 страниц для базы данных "Кинотеатр", файл "Кинотеатр" для файла 1.
Обработано 3 страниц для базы данных "Кинотеатр", файл "Кинотеатр_log" для файла 1.
RESTORE LOG успешно обработал 3 страниц за 0.003 секунд (7.812 МБ/сек).
```

Время выполнения: 2024-12-06T02:11:18.2503606+03:00

Рисунок 28 – Результат восстановления БД из резервной копии

ЗАКЛЮЧЕНИЕ

В ходе выполнения индивидуального задания была спроектирована и реализована база данных «**Кинотеатр**» с использованием метода ER-диаграмм. База данных была разработана для управления информацией о фильмах, залах, сеансах, клиентах, местах и бронировании, что охватывает все основные аспекты функционирования современного кинотеатра.

При проектировании были учтены важные принципы нормализации данных для минимизации избыточности и обеспечения целостности информации. Созданные ER-диаграммы позволили наглядно визуализировать связи между сущностями и определить структуру базы данных. Основные сущности, такие как **ФИЛЬМ**, **ЗАЛ**, **МЕСТО**, **СЕАНС**, **КЛИЕНТ** и **БРОНИРОВАНИЕ**, были проработаны с учётом бизнес-логики, обеспечивая корректное хранение и обработку данных.

В процессе разработки были использованы знания, полученные в ходе выполнения лабораторных работ, включая создание таблиц, добавление ограничений и использование языка T-SQL. Для базы данных были реализованы различные объекты промежуточного слоя: представления, хранимые процедуры и функции, которые демонстрируют практическое применение SQL-запросов для обработки данных. Каждая из этих реализаций была проверена на корректность с использованием тестовых данных.

Отдельное внимание было уделено резервному копированию и восстановлению базы данных. Создана резервная копия, включающая полное, дифференциальное копирование и журнал транзакций. Это обеспечивает надёжность системы в случае сбоев или потери данных. Кроме того, были разработаны сценарии восстановления базы данных, что особенно важно для поддержания работоспособности информационной системы.

Стоит отметить, что процесс разработки базы данных для кинотеатра выявил некоторые интересные аспекты проектирования. Например, реализация таблицы **МЕСТО** с составным первичным ключом и учётом статуса мест продемонстрировала важность грамотного подхода к структурированию данных.

Также, благодаря разделению обязанностей между таблицами, удалось эффективно организовать информацию, связанную с бронированием мест, минимизировав вероятность возникновения конфликтов или ошибок.

Итоговая база данных является многофункциональной и гибкой системой, состоящей из 6 основных таблиц, нескольких представлений, хранимых процедур и функций. Этот проект стал отличной практической иллюстрацией применения теоретических знаний в реальных задачах, а также доказал, что правильная структура и организация базы данных значительно упрощают её использование и дальнейшее обслуживание.

Таким образом, выполненная работа продемонстрировала не только возможность создания надёжной и эффективной базы данных, но и важность подхода к её проектированию, что является неотъемлемой частью любой информационной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Горячев А.В., Новакова Н.Е. Распределенные базы данных. Мет. указания к лаб. работам., СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2008.
2. Горячев А.В., Новакова Н.Е. Особенности разработки и администрирования приложений баз данных: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016. 68с.
3. Эльмасри Р., Наватхе Ш.Б. Основы систем баз данных. 7-е изд. М.: Вильямс, 2016. 1216 с.
4. Сильбершатц А., Корф Г., Сударшан Ш. Концепции систем баз данных. 7-е изд. М.: Диалектика, 2020. 1376 с.
5. ISO/IEC 9075:2016. Информационные технологии – Языки баз данных – Стандарт SQL. Женева: Международная организация по стандартизации, 2016. 450 с.
6. Документация Microsoft SQL Server. Справочник по языку T-SQL // Microsoft Learn. URL: <https://learn.microsoft.com/ru-ru/sql/t-sql/> (дата обращения: 03.12.2024).