-- Создание базы данных "Прокат спортивного инвентаря"

CREATE DATABASE ПрокатСпортивногоИнвентаря;

GO

-- Использование базы данных "Прокат спортивного инвентаря"

USE ПрокатСпортивногоИнвентаря;

GO

-- Создание таблицы "Инвентарь"

CREATE TABLE Инвентарь (

id\_инвентаря INT PRIMARY KEY IDENTITY(1, 1),

Название VARCHAR(100) NOT NULL,

Тип VARCHAR(50) NOT NULL,

Размер VARCHAR(10),

СтоимостьПроката INT

);

GO

-- Создание таблицы "Клиенты"

CREATE TABLE Клиенты (

id\_клиента INT PRIMARY KEY IDENTITY(1, 1),

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Адрес VARCHAR(100),

Телефон VARCHAR(15),

Email VARCHAR(50)

);

GO

-- Создание таблицы "Сотрудники"

CREATE TABLE Сотрудники (

id\_сотрудника INT PRIMARY KEY IDENTITY(1, 1),

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Должность VARCHAR(50) NOT NULL,

Зарплата INT

);

GO

-- Создание таблицы "Прокаты"

CREATE TABLE Прокаты (

id\_проката INT PRIMARY KEY,

id\_инвентаря INT,

id\_клиента INT,

id\_сотрудника INT,

ДатаНачала DATE,

ДатаОкончания DATE,

FOREIGN KEY (id\_инвентаря) REFERENCES Инвентарь(id\_инвентаря),

FOREIGN KEY (id\_клиента) REFERENCES Клиенты(id\_клиента),

FOREIGN KEY (id\_сотрудника) REFERENCES Сотрудники(id\_сотрудника),

);

GO

-- Заполнение таблицы "Инвентарь"

INSERT INTO Инвентарь (Название, Тип, Размер, СтоимостьПроката)

VALUES ('Теннисная ракетка', 'Теннис', 'M', 200),

('Велосипед', 'Велоспорт', 'L', 300),

('Лыжи', 'Горные лыжи', 'XL', 250);

-- Заполнение таблицы "Клиенты"

INSERT INTO Клиенты (Имя, Фамилия, Адрес, Телефон, Email)

VALUES ('Иван', 'Иванов', 'ул. Ленина, 10', '+79991234567', 'ivanov@example.com'),

('Петр', 'Петров', 'ул. Гагарина, 5', '+79999876543', 'petrov@example.com'),

('Анна', 'Сидорова', 'ул. Пушкина, 15', '+79995555555', 'sidorova@example.com');

-- Заполнение таблицы "Сотрудники"

INSERT INTO Сотрудники (Имя, Фамилия, Должность, Зарплата)

VALUES ('Алексей', 'Смирнов', 'Менеджер', 50000),

('Мария', 'Иванова', 'Администратор', 45000),

('Дмитрий', 'Петров', 'Консультант', 40000);

-- Заполнение таблицы "Прокаты"

INSERT INTO Прокаты (id\_проката, id\_инвентаря, id\_клиента, id\_сотрудника, ДатаНачала, ДатаОкончания)

VALUES (1, 1, 2, 3, '2023-05-01', '2023-05-03'),

(2, 3, 1, 2, '2023-05-02', '2023-05-04'),

(3, 2, 3, 1, '2023-05-03', '2023-05-05');

CREATE VIEW СписокПрокатов AS

SELECT Прокаты.id\_проката, Инвентарь.Название AS НазваниеИнвентаря, Клиенты.Имя AS ИмяКлиента, Клиенты.Фамилия AS ФамилияКлиента,

Сотрудники.Имя AS ИмяСотрудника, Сотрудники.Фамилия AS ФамилияСотрудника, Прокаты.ДатаНачала, Прокаты.ДатаОкончания

FROM Прокаты

JOIN Инвентарь ON Прокаты.id\_инвентаря = Инвентарь.id\_инвентаря

JOIN Клиенты ON Прокаты.id\_клиента = Клиенты.id\_клиента

JOIN Сотрудники ON Прокаты.id\_сотрудника = Сотрудники.id\_сотрудника;

CREATE VIEW СписокИнвентарей AS

SELECT Название, Тип, Размер, СтоимостьПроката

FROM Инвентарь;

CREATE VIEW ОбщаяВыручка AS

SELECT SUM(Инвентарь.СтоимостьПроката) AS Выручка

FROM Прокаты

JOIN Инвентарь ON Прокаты.id\_инвентаря = Инвентарь.id\_инвентаря;

CREATE PROCEDURE ДобавитьИнвентарь

@Название VARCHAR(100),

@Тип VARCHAR(50),

@Размер VARCHAR(10),

@СтоимостьПроката INT

AS

BEGIN

INSERT INTO Инвентарь (Название, Тип, Размер, СтоимостьПроката)

VALUES (@Название, @Тип, @Размер, @СтоимостьПроката);

END;

CREATE PROCEDURE АрендоватьИнвентарь

@id\_инвентаря INT,

@id\_клиента INT,

@id\_сотрудника INT,

@ДатаНачала DATE,

@ДатаОкончания DATE

AS

BEGIN

INSERT INTO Прокаты (id\_инвентаря, id\_клиента, id\_сотрудника, ДатаНачала, ДатаОкончания)

VALUES (@id\_инвентаря, @id\_клиента, @id\_сотрудника, @ДатаНачала, @ДатаОкончания);

END;

CREATE PROCEDURE УдалитьИнвентарь

@id\_инвентаря INT

AS

BEGIN

DELETE FROM Инвентарь

WHERE id\_инвентаря = @id\_инвентаря;

END;

CREATE TRIGGER КонтрольМаксимальногоКоличестваПрокатов

ON Прокаты

AFTER INSERT

AS

BEGIN

DECLARE @MaxПрокаты INT = 3; -- Максимальное количество прокатов для клиента

IF EXISTS (

SELECT Клиенты.id\_клиента

FROM inserted ins

INNER JOIN Клиенты ON ins.id\_клиента = Клиенты.id\_клиента

GROUP BY Клиенты.id\_клиента

HAVING COUNT(\*) > @MaxПрокаты

)

BEGIN

RAISERROR ('Достигнуто максимальное количество прокатов для клиента.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

CREATE TRIGGER ПослеУдаленияКлиента

ON Клиенты

AFTER DELETE

AS

BEGIN

-- Удаление связанных прокатов после удаления клиента

DELETE FROM Прокаты

WHERE id\_клиента IN (SELECT id\_клиента FROM deleted);

END;

CREATE TRIGGER УстановкаДатыПроката

ON Прокаты

AFTER INSERT

AS

BEGIN

UPDATE Прокаты

SET ДатаНачала = GETDATE()

WHERE id\_проката IN (SELECT id\_проката FROM inserted);

END;

CREATE FUNCTION ПолучитьИнвентарьПоТипу(@тип VARCHAR(50))

RETURNS TABLE

AS

RETURN (

SELECT \*

FROM Инвентарь

WHERE Тип = @тип

);

CREATE FUNCTION ПолучитьПрокатыПоКлиенту(@id\_клиента INT)

RETURNS TABLE

AS

RETURN (

SELECT \*

FROM Прокаты

WHERE id\_клиента = @id\_клиента

);

CREATE FUNCTION ПолучитьСотрудниковПоЗарплате(@зарплата INT)

RETURNS TABLE

AS

RETURN (

SELECT \*

FROM Сотрудники

WHERE Зарплата = @зарплата

);

1. **Обеспечение безопасного доступа к общим файлам**

Обеспечение безопасного доступа к общим файлам является важной задачей для защиты конфиденциальности и целостности данных. Для этого можно использовать следующие методы и меры:

1. Установка соответствующих разрешений доступа: На уровне файловой системы или сетевого хранилища можно определить разрешения доступа для каждого файла или папки.

2. Использование групповых политик: Групповые политики в операционной системе позволяют определить правила доступа к общим файлам для определенных групп пользователей.

3. Шифрование файлов: Шифрование файлов может быть использовано для обеспечения конфиденциальности данных. При шифровании файлов доступ к ним будет иметь только тот, кто обладает правильным ключом или паролем.

4. Аудит доступа: Вести аудит доступа к общим файлам позволяет отслеживать, кто и когда получал доступ к файлам. Это помогает выявить нежелательные активности и своевременно реагировать на них.

5. Регулярное обновление паролей и контроль доступа: Регулярное изменение паролей пользователей и контроль доступа помогает предотвратить несанкционированный доступ к общим файлам.

6. Обучение пользователей: Обучение пользователей о безопасном использовании общих файлов и соблюдении политик безопасности помогает повысить осведомленность и предотвращает небрежное или нежелательное обращение с данными.