-- Создание базы данных "Кафе быстрого питания"

CREATE DATABASE КафеБыстрогоПитания;

GO

-- Использование базы данных "Кафе быстрого питания"

USE КафеБыстрогоПитания;

GO

-- Создание таблицы "Меню"

CREATE TABLE Меню (

id\_блюда INT PRIMARY KEY,

Название VARCHAR(100) NOT NULL,

Описание VARCHAR(255),

Цена DECIMAL(10, 2) NOT NULL

);

GO

-- Создание таблицы "Сотрудники"

CREATE TABLE Сотрудники (

id\_сотрудника INT PRIMARY KEY,

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Должность VARCHAR(50) NOT NULL

);

GO

-- Создание таблицы "Смены"

CREATE TABLE Смены (

id\_смены INT PRIMARY KEY,

Начало\_смены TIME NOT NULL,

Окончание\_смены TIME NOT NULL

);

GO

-- Создание таблицы "Заказы"

CREATE TABLE Заказы (

id\_заказа INT PRIMARY KEY,

id\_сотрудника INT,

Дата DATETIME NOT NULL,

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

);

GO

-- Создание таблицы "Сотрудники\_Смены"

CREATE TABLE Сотрудники\_Смены (

id\_сотрудника INT,

id\_смены INT,

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

FOREIGN KEY (id\_смены) REFERENCES Смены(id\_смены)

);

GO

-- Создание таблицы "Заказы\_Блюда"

CREATE TABLE Заказы\_Блюда (

id\_заказа INT,

id\_блюда INT,

Количество INT NOT NULL,

FOREIGN KEY (id\_заказа) REFERENCES Заказы(id\_заказа),

FOREIGN KEY (id\_блюда) REFERENCES Меню(id\_блюда)

);

GO

-- Заполнение таблицы "Меню"

INSERT INTO Меню (id\_блюда, Название, Описание, Цена)

VALUES

(1, 'Гамбургер', 'Сочный гамбургер с говяжьей котлетой', 500),

(2, 'Картофель фри', 'Хрустящие палочки из картофеля', 60),

(3, 'Салат Цезарь', 'Классический салат с курицей и соусом Цезарь', 250);

-- Заполнение таблицы "Сотрудники"

INSERT INTO Сотрудники (id\_сотрудника, Имя, Фамилия, Должность)

VALUES

(1, 'Иван', 'Иванов', 'Повар'),

(2, 'Петр', 'Петров', 'Официант'),

(3, 'Анна', 'Сидорова', 'Бариста');

-- Заполнение таблицы "Смены"

INSERT INTO Смены (id\_смены, Начало\_смены, Окончание\_смены)

VALUES

(1, '09:00:00', '15:00:00'),

(2, '15:00:00', '21:00:00'),

(3, '09:00:00', '15:00:00');

-- Заполнение таблицы "Заказы"

INSERT INTO Заказы (id\_заказа, id\_сотрудника, Дата)

VALUES

(1, 2, '2023-05-01'),

(2, 3, '2023-05-01'),

(3, 1, '2023-05-02');

-- Заполнение таблицы "Сотрудники\_Смены"

INSERT INTO Сотрудники\_Смены (id\_сотрудника, id\_смены)

VALUES

(2, 1),

(3, 2),

(1, 3);

-- Заполнение таблицы "Заказы\_Блюда"

INSERT INTO Заказы\_Блюда (id\_заказа, id\_блюда, Количество)

VALUES

(1, 1, 2),

(2, 2, 1),

(3, 3, 1);

CREATE VIEW Сотрудники\_Заказы AS

SELECT Сотрудники.Имя, Сотрудники.Фамилия, Заказы.id\_заказа, Заказы.Дата

FROM Сотрудники

INNER JOIN Заказы ON Сотрудники.id\_сотрудника = Заказы.id\_сотрудника;

CREATE VIEW Заказы\_Состав AS

SELECT Заказы.id\_заказа, Заказы.Дата, Меню.Название AS Блюдо, Заказы\_Блюда.Количество

FROM Заказы

INNER JOIN Заказы\_Блюда ON Заказы.id\_заказа = Заказы\_Блюда.id\_заказа

INNER JOIN Меню ON Заказы\_Блюда.id\_блюда = Меню.id\_блюда;

CREATE VIEW Меню\_Цены AS

SELECT Название, Описание, Цена

FROM Меню;

select \* from Сотрудники\_Заказы

select \* from Заказы\_Состав

select \* from Меню\_Цены

CREATE PROCEDURE ДобавитьБлюдо

@Название VARCHAR(100),

@Описание VARCHAR(255),

@Цена DECIMAL(10, 2)

AS

BEGIN

INSERT INTO Меню (Название, Описание, Цена)

VALUES (@Название, @Описание, @Цена);

END;

CREATE PROCEDURE ИзменитьЦенуБлюда

@id\_блюда INT,

@НоваяЦена DECIMAL(10, 2)

AS

BEGIN

UPDATE Меню

SET Цена = @НоваяЦена

WHERE id\_блюда = @id\_блюда;

END;

CREATE PROCEDURE УдалитьБлюдоИзМеню

@id\_блюда INT

AS

BEGIN

DELETE FROM Меню

WHERE id\_блюда = @id\_блюда;

END;

CREATE TRIGGER ПроверкаЦеныПриДобавленииБлюда

ON Меню

FOR INSERT

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE Цена < 0)

BEGIN

RAISERROR('Цена не может быть отрицательной.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END;

END;

CREATE TRIGGER АвтоУдалениеЗаказаБлюдаПриУдаленииЗаказа

ON Заказы

AFTER DELETE

AS

BEGIN

DELETE FROM Заказы\_Блюда

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

END;

CREATE TRIGGER ПроверкаСотрудникаПриДобавленииЗаказа

ON Заказы

FOR INSERT

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE id\_сотрудника IS NULL)

BEGIN

RAISERROR('Не указан сотрудник для заказа.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END;

END;

CREATE FUNCTION ПолучитьСписокБлюдВМеню()

RETURNS TABLE

AS

RETURN

(

SELECT id\_блюда, Название, Описание, Цена

FROM Меню

);

CREATE FUNCTION ПолучитьСписокЗаказовПоСотруднику(@id\_сотрудника INT)

RETURNS TABLE

AS

RETURN

(

SELECT Заказы.id\_заказа, Заказы.Дата, Заказы\_Блюда.id\_блюда, Меню.Название, Меню.Цена

FROM Заказы

INNER JOIN Заказы\_Блюда ON Заказы.id\_заказа = Заказы\_Блюда.id\_заказа

INNER JOIN Меню ON Заказы\_Блюда.id\_блюда = Меню.id\_блюда

WHERE Заказы.id\_сотрудника = @id\_сотрудника

);

CREATE FUNCTION ПолучитьСписокСотрудниковПоСмене(@id\_смены INT)

RETURNS TABLE

AS

RETURN

(

SELECT Сотрудники.id\_сотрудника, Сотрудники.Имя, Сотрудники.Фамилия, Сотрудники.Должность

FROM Сотрудники

INNER JOIN Сотрудники\_Смены ON Сотрудники.id\_сотрудника = Сотрудники\_Смены.id\_сотрудника

WHERE Сотрудники\_Смены.id\_смены = @id\_смены

);

1. **Внедрение групповых политик**

Внедрение групповых политик (Group Policy) является важной практикой в области администрирования информационных систем и позволяет управлять конфигурацией и поведением компьютеров и пользователей в сети. Групповые политики используются в операционных системах семейства Windows, таких как Windows Server и Windows Desktop.

Основная цель внедрения групповых политик состоит в централизованном управлении параметрами безопасности, настройками приложений, реестра, сетевыми ресурсами и другими аспектами компьютерной конфигурации. Это позволяет администраторам легко применять и изменять настройки для группы компьютеров или пользователей, а также обеспечивает согласованность и стандартизацию параметров в сети.

Процесс внедрения групповых политик включает следующие шаги:

1. Создание групповых политик: Администратор определяет необходимые настройки и параметры, которые будут применяться к целевым компьютерам или пользователям.

2. Организация групповых политик: Групповые политики могут быть организованы в древовидную структуру, где высшие уровни представляют общие параметры для всех компьютеров или пользователей, а нижние уровни определяют более специфические настройки для определенных групп или отдельных объектов.

3. Применение групповых политик

4. Оценка и тестирование: перед окончательным внедрением групповых политик необходимо провести оценку и тестирование

5. Внедрение и применение: после успешного тестирования групповые политики могут быть окончательно внедрены в сеть и применены к целевым компьютерам или пользователям.

6. Мониторинг и обновление: После внедрения групповых политик необходимо регулярно мониторить и обновлять их для поддержания соответствия требованиям безопасности и бизнес-процессов.