-- Создание базы данных "ЦентрЗанятости"

CREATE DATABASE ЦентрЗанятости;

GO

-- Использование базы данных "ЦентрЗанятости"

USE ЦентрЗанятости;

GO

-- Создание таблицы "Соискатели"

CREATE TABLE Соискатели (

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

Имя VARCHAR(100) NOT NULL,

Фамилия VARCHAR(100) NOT NULL,

Должность VARCHAR(50) NOT NULL,

Телефон VARCHAR(20) NOT NULL,

Email VARCHAR(100) NOT NULL

);

GO

-- Создание таблицы "Работодатели"

CREATE TABLE Работодатели (

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

Название\_компании VARCHAR(100) NOT NULL,

Адрес VARCHAR(100) NOT NULL,

Телефон VARCHAR(20) NOT NULL,

Email VARCHAR(100) NOT NULL

);

GO

-- Создание таблицы "Вакансии"

CREATE TABLE Вакансии (

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

Название VARCHAR(100) NOT NULL,

Описание TEXT,

Должность VARCHAR(50) NOT NULL,

Зарплата INT,

id\_работодателя INT,

FOREIGN KEY (id\_работодателя) REFERENCES Работодатели(id\_работодателя)

);

GO

-- Создание таблицы "Отклики"

CREATE TABLE Отклики (

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

id\_соискателя INT,

id\_вакансии INT,

Дата\_отклика DATE,

FOREIGN KEY (id\_соискателя) REFERENCES Соискатели(id\_соискателя),

FOREIGN KEY (id\_вакансии) REFERENCES Вакансии(id\_вакансии)

);

GO

-- Вставка данных в таблицу "Соискатели"

INSERT INTO Соискатели (Имя, Фамилия, Должность, Телефон, Email)

VALUES

('Иван', 'Иванов', 'Программист', '123456789', 'ivan@example.com'),

('Петр', 'Петров', 'Дизайнер', '987654321', 'petr@example.com'),

('Анна', 'Сидорова', 'Менеджер', '555555555', 'anna@example.com'),

('Мария', 'Смирнова', 'Администратор', '777777777', 'maria@example.com'),

('Алексей', 'Козлов', 'Тестировщик', '999999999', 'alexey@example.com');

-- Вставка данных в таблицу "Работодатели"

INSERT INTO Работодатели (Название\_компании, Адрес, Телефон, Email)

VALUES

('Компания1', 'Адрес1', '111111111', 'company1@example.com'),

('Компания2', 'Адрес2', '222222222', 'company2@example.com'),

('Компания3', 'Адрес3', '333333333', 'company3@example.com'),

('Компания4', 'Адрес4', '444444444', 'company4@example.com'),

('Компания5', 'Адрес5', '555555555', 'company5@example.com');

-- Вставка данных в таблицу "Вакансии"

INSERT INTO Вакансии (Название, Описание, Должность, Зарплата, id\_работодателя)

VALUES

('Вакансия1', 'Описание1', 'Должность1', 50000, 1),

('Вакансия2', 'Описание2', 'Должность2', 60000, 1),

('Вакансия3', 'Описание3', 'Должность3', 70000, 2),

('Вакансия4', 'Описание4', 'Должность4', 80000, 3),

('Вакансия5', 'Описание5', 'Должность5', 90000, 3);

-- Вставка данных в таблицу "Отклики"

INSERT INTO Отклики (id\_соискателя, id\_вакансии, Дата\_отклика)

VALUES

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

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

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

(4, 3, '2023-05-04'),

(5, 3, '2023-05-05');

CREATE VIEW Соискатели\_Вакансии AS

SELECT

Соискатели.id\_соискателя,

Соискатели.Имя,

Соискатели.Фамилия,

Вакансии.id\_вакансии,

Вакансии.Название AS Название\_вакансии,

Вакансии.Должность AS Должность\_вакансии

FROM

Соискатели

JOIN

Отклики ON Соискатели.id\_соискателя = Отклики.id\_соискателя

JOIN

Вакансии ON Отклики.id\_вакансии = Вакансии.id\_вакансии;

CREATE VIEW Работодатели\_Вакансии AS

SELECT

Работодатели.id\_работодателя,

Работодатели.Название\_компании,

Вакансии.id\_вакансии,

Вакансии.Название AS Название\_вакансии,

Вакансии.Должность AS Должность\_вакансии

FROM

Работодатели

JOIN

Вакансии ON Работодатели.id\_работодателя = Вакансии.id\_работодателя;

CREATE VIEW Соискатели\_Контакты AS

SELECT

Соискатели.id\_соискателя,

Соискатели.Имя,

Соискатели.Фамилия,

Соискатели.Телефон,

Соискатели.Email

FROM

Соискатели;

CREATE PROCEDURE ДобавитьСоискателя

@Имя VARCHAR(100),

@Фамилия VARCHAR(100),

@Должность VARCHAR(50),

@Телефон VARCHAR(20),

@Email VARCHAR(100)

AS

BEGIN

INSERT INTO Соискатели (Имя, Фамилия, Должность, Телефон, Email)

VALUES (@Имя, @Фамилия, @Должность, @Телефон, @Email);

END;

CREATE PROCEDURE ДобавитьВакансию

@Название VARCHAR(100),

@Описание TEXT,

@Должность VARCHAR(50),

@Зарплата INT,

@id\_работодателя INT

AS

BEGIN

INSERT INTO Вакансии (Название, Описание, Должность, Зарплата, id\_работодателя)

VALUES (@Название, @Описание, @Должность, @Зарплата, @id\_работодателя);

END;

CREATE PROCEDURE ПолучитьВсеВакансии

AS

BEGIN

SELECT \* FROM Вакансии;

END;

CREATE TRIGGER ПроверкаЗарплаты

ON Вакансии

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Зарплата < 0)

BEGIN

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

ROLLBACK TRANSACTION;

END;

END;

CREATE TRIGGER ОбновлениеАдресаРаботодателя

ON Работодатели

AFTER UPDATE

AS

BEGIN

IF UPDATE(Адрес)

BEGIN

PRINT 'Адрес работодателя был изменен.';

END;

END;

CREATE TRIGGER УдалениеСоискателя

ON Соискатели

INSTEAD OF DELETE

AS

BEGIN

DELETE FROM Отклики WHERE id\_соискателя IN (SELECT id\_соискателя FROM deleted);

DELETE FROM Соискатели WHERE id\_соискателя IN (SELECT id\_соискателя FROM deleted);

END;

CREATE FUNCTION ПолучитьВакансииПоДолжности(@должность VARCHAR(50))

RETURNS TABLE

AS

RETURN (

SELECT Вакансии.\*

FROM Вакансии

WHERE Вакансии.Должность = @должность

);

CREATE FUNCTION ПолучитьОткликиСоискателя(@id\_соискателя INT)

RETURNS TABLE

AS

RETURN (

SELECT Отклики.\*

FROM Отклики

WHERE Отклики.id\_соискателя = @id\_соискателя

);

CREATE FUNCTION ПолучитьОткликиНаВакансию(@id\_вакансии INT)

RETURNS TABLE

AS

RETURN (

SELECT Отклики.\*

FROM Отклики

WHERE Отклики.id\_вакансии = @id\_вакансии

);

1. **Модели восстановления SQL-сервера.**

SQL-сервер обеспечивает несколько моделей восстановления данных в случае сбоев или потери данных. Вот некоторые из наиболее распространенных моделей восстановления SQL-сервера:

1. Полная модель восстановления (Full Recovery Model): В этой модели все данные и транзакции полностью сохраняются в журнале транзакций. Это позволяет проводить резервное копирование журнала транзакций и восстанавливать базу данных к любой точке во времени, включая полное восстановление после сбоя.

2. Модель восстановления с использованием регистрации изменений (Bulk-Logged Recovery Model): В этой модели данные сохраняются в журнале транзакций, но некоторые операции могут быть зарегистрированы в сжатом виде. Это позволяет ускорить операции, связанные с массовыми операциями, такими как загрузка данных в базу данных. Однако, восстановление базы данных может быть ограничено в случае использования этой модели.

3. Простая модель восстановления (Simple Recovery Model): В этой модели данные сохраняются только до завершения каждой транзакции. Журнал транзакций очищается после успешного завершения каждой операции. Это обеспечивает простоту управления журналом транзакций и экономию пространства на диске, но ограничивает возможности восстановления данных после сбоя.

Каждая из этих моделей восстановления имеет свои преимущества и подходит для разных сценариев использования. Выбор модели восстановления зависит от требований к сохранности данных, времени восстановления и доступности системы