-- Создание базы данных "Отдел кадров"

CREATE DATABASE ОтделКадров;

GO

-- Использование базы данных "Отдел кадров"

USE ОтделКадров;

GO

-- Создание таблицы "Отделы"

CREATE TABLE Отделы (

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

Название VARCHAR(100) NOT NULL,

Местоположение VARCHAR(100),

);

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,

Дата\_приема DATE,

id\_отдела INT,

FOREIGN KEY (id\_отдела) REFERENCES Отделы(id\_отдела)

);

GO

-- Создание таблицы "Отпуска"

CREATE TABLE Отпуска (

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

id\_сотрудника INT,

Дата\_начала DATE,

Дата\_окончания DATE,

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

);

GO

-- Создание таблицы "Рекрутинг"

CREATE TABLE Рекрутинг (

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

Название\_вакансии VARCHAR(100) NOT NULL,

Описание\_вакансии VARCHAR(500),

Требования VARCHAR(500),

Дата\_размещения DATE,

id\_отдела INT,

FOREIGN KEY (id\_отдела) REFERENCES Отделы(id\_отдела)

);

GO

-- Создание таблицы "Образование"

CREATE TABLE Образование (

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

Название\_образования VARCHAR(100) NOT NULL

);

GO

-- Создание таблицы "Образование\_сотрудника"

CREATE TABLE Образование\_сотрудника (

id\_сотрудника INT,

id\_образования INT,

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

FOREIGN KEY (id\_образования) REFERENCES Образование(id\_образования)

);

GO

-- Вставка данных в таблицу "Отделы"

INSERT INTO Отделы (Название, Местоположение)

VALUES ('Отдел продаж', 'Москва'),

('Отдел разработки', 'Санкт-Петербург'),

('Отдел маркетинга', 'Киев');

-- Вставка данных в таблицу "Сотрудники"

INSERT INTO Сотрудники (Имя, Фамилия, Должность, Зарплата, Дата\_приема, id\_отдела)

VALUES ('Иван', 'Иванов', 'Менеджер', 50000, '2022-01-01', 1),

('Петр', 'Петров', 'Разработчик', 70000, '2021-05-15', 2),

('Анна', 'Сидорова', 'Маркетолог', 45000, '2023-02-20', 3);

-- Вставка данных в таблицу "Отпуска"

INSERT INTO Отпуска (id\_сотрудника, Дата\_начала, Дата\_окончания)

VALUES (1, '2023-06-15', '2023-06-30'),

(2, '2023-07-10', '2023-07-25');

-- Вставка данных в таблицу "Рекрутинг"

INSERT INTO Рекрутинг (Название\_вакансии, Описание\_вакансии, Требования, Дата\_размещения, id\_отдела)

VALUES ('Менеджер по продажам', 'Описание вакансии 1', 'Требования кандидата 1', '2023-05-20', 1),

('Веб-разработчик', 'Описание вакансии 2', 'Требования кандидата 2', '2023-05-25', 2);

-- Вставка данных в таблицу "Образование"

INSERT INTO Образование (Название\_образования)

VALUES ('Высшее'),

('Среднее специальное'),

('Неполное высшее');

-- Вставка данных в таблицу "Образование\_сотрудника"

INSERT INTO Образование\_сотрудника (id\_сотрудника, id\_образования)

VALUES (1, 1),

(2, 3),

(3, 2);

CREATE VIEW СотрудникиСОтделами AS

SELECT Сотрудники.\*, Отделы.Название AS Название\_отдела

FROM Сотрудники

JOIN Отделы ON Сотрудники.id\_отдела = Отделы.id\_отдела;

CREATE VIEW СотрудникиВОтпусках AS

SELECT Сотрудники.\*, Отпуска.Дата\_начала, Отпуска.Дата\_окончания

FROM Сотрудники

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

CREATE VIEW ВакансииСОтделами AS

SELECT Рекрутинг.\*, Отделы.Название AS Отдел

FROM Рекрутинг

LEFT JOIN Отделы ON Рекрутинг.id\_отдела = Отделы.id\_отдела;

CREATE PROCEDURE ДобавитьСотрудника

@Имя VARCHAR(100),

@Фамилия VARCHAR(100),

@Должность VARCHAR(50),

@Зарплата INT,

@Дата\_приема DATE,

@id\_отдела INT

AS

BEGIN

INSERT INTO Сотрудники (Имя, Фамилия, Должность, Зарплата, Дата\_приема, id\_отдела)

VALUES (@Имя, @Фамилия, @Должность, @Зарплата, @Дата\_приема, @id\_отдела);

END;

CREATE PROCEDURE УдалитьОтпуск

@id\_отпуска INT

AS

BEGIN

DELETE FROM Отпуска

WHERE id\_отпуска = @id\_отпуска;

END;

CREATE PROCEDURE ПолучитьСотрудниковВОтделе

@id\_отдела INT

AS

BEGIN

SELECT \*

FROM Сотрудники

WHERE id\_отдела = @id\_отдела;

END;

CREATE TRIGGER АвтоматическоеОбновлениеДатыПриема

ON Сотрудники

AFTER INSERT

AS

BEGIN

UPDATE Сотрудники

SET Дата\_приема = GETDATE()

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

END;

CREATE TRIGGER ОграничениеКоличестваСимволовВОписанииВакансии

ON Рекрутинг

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE Рекрутинг

SET Описание\_вакансии = LEFT(Описание\_вакансии, 500)

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

END;

CREATE TRIGGER ПроверкаДатыПриемаСотрудника

ON Сотрудники

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT 1 FROM inserted WHERE Дата\_приема > GETDATE())

BEGIN

RAISERROR ('Дата приема не может быть в будущем.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

CREATE FUNCTION ПолучитьСреднююЗарплатуОтдела(@id\_отдела INT)

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @средняяЗарплата DECIMAL(10, 2);

SELECT @средняяЗарплата = AVG(Сотрудники.Зарплата)

FROM Сотрудники

WHERE Сотрудники.id\_отдела = @id\_отдела;

RETURN @средняяЗарплата;

END;

CREATE FUNCTION ПолучитьКоличествоСотрудниковВОтпуске(@Дата\_начала DATE, @Дата\_окончания DATE)

RETURNS INT

AS

BEGIN

DECLARE @количество INT;

SELECT @количество = COUNT(\*)

FROM Отпуска

WHERE (@Дата\_начала <= Отпуска.Дата\_окончания) AND (@Дата\_окончания >= Отпуска.Дата\_начала);

RETURN @количество;

END;

CREATE FUNCTION ПолучитьСписокСотрудниковСОбразованием(@id\_образования INT)

RETURNS TABLE

AS

RETURN

(

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

FROM Сотрудники

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

WHERE Образование\_сотрудника.id\_образования = @id\_образования

);

1. **Способы контроля доступа к данным и управления привилегиями.**

Способы контроля доступа к данным и управления привилегиями в базах данных включают:

1. Ролевая модель доступа: Определение различных ролей или групп пользователей с определенными привилегиями. Администратор базы данных назначает роли пользователям, а затем определяет права доступа для каждой роли.

2. Аутентификация и авторизация: Использование учетных данных и механизма проверки подлинности для идентификации пользователей и предоставления им прав доступа на основе определенных правил.

3. Уровни доступа: Определение уровней доступа к данным на основе их конфиденциальности и важности. Различные пользователи или группы пользователей могут иметь различные уровни доступа, позволяющие им видеть и изменять только определенные данные.

4. Аудит доступа: Ведение журнала аудита, который записывает информацию о доступе пользователей к базе данных. Это помогает отслеживать и мониторить активность пользователей и выявлять любые нарушения безопасности.

5. Шифрование данных: Защита данных путем их шифрования при передаче или хранении. Шифрование обеспечивает конфиденциальность и защиту от несанкционированного доступа к данным.