МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный технический университет

имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №1**

по дисциплине: «Базы данных»

на тему: **«Выбор и описание предметной области исходя из назначенного студенту нереляционного СУБД»**

Выполнил: студент гр. ИТП-21

Иванов И.И.

Принял: преподаватель

Ястребов А.А.

Гомель 2024

**Задание:** для конкретного варианта работы изучить предметную область, разработать реляционную схему данных, схему запросов, заполнить таблицы данными.

Это берем из файла с л.р. (задание)

Вариант 4. Журнал инспектора энергонадзора.

Основные функции: просмотр сведений о посещениях предприятия инспекторами в течение заданного периода. Сумма штрафов за заданный период для заданного предприятия. Вид нерационального использования энергоресурсов, по которому в прошлом году было выплачено больше всего штрафов.

Атрибуты: инспектор, предприятие, дата посещения предприятия, вид нерационального использования энергоресурсов, сумма штрафных санкций, дата оплаты штрафа.

**Ход работы и результаты выполнения**

Для заданной предметной области «Журнал инспектора энергонадзора» разработана схема базы данных, представленная на рисунке 1.

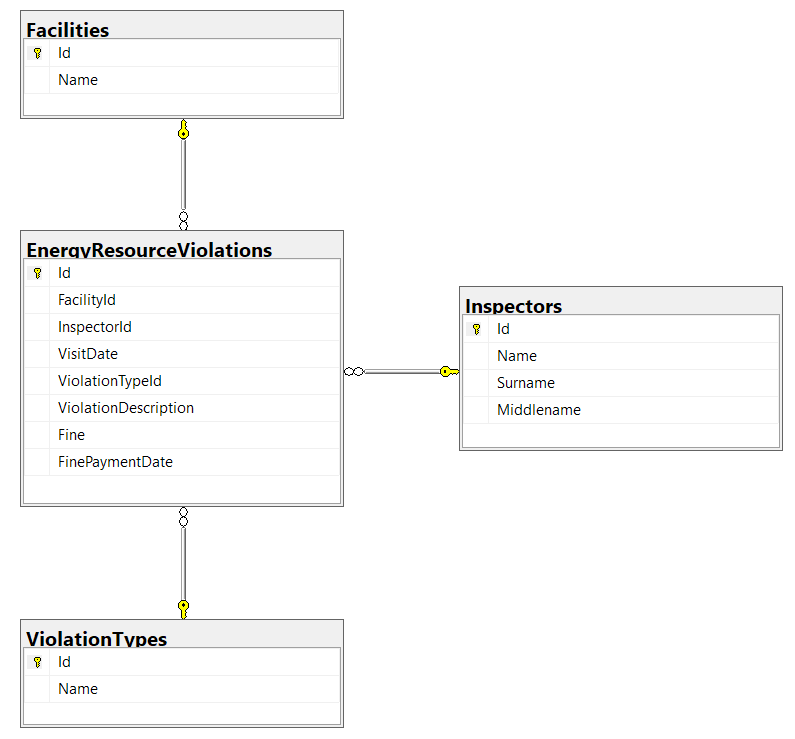


Рисунок 1 – Схема базы данных «Журнал инспектора энергонадзора»

Для создания таблиц базы данных и заполнения их тестовыми данными разработан *T-SQL* скрипт.

База данных содержит следующие таблицы:

– «*Facilities*» – содержит информацию о предприятиях;

– «*Inspectors*» – содержит информацию об инспекторах;

– «*ViolationTypes*» – содержит информацию о типах нарушений;

– «*EnergyResourceViolations*» – содержит отчеты о посещениях предприятия инспекторами.

Для вывода данных в удобном виде разработаны соответствующие процедуры. Результат работы процедуры, которая выводит отчеты о посещениях предприятия инспекторами представлена на рисунке 2.

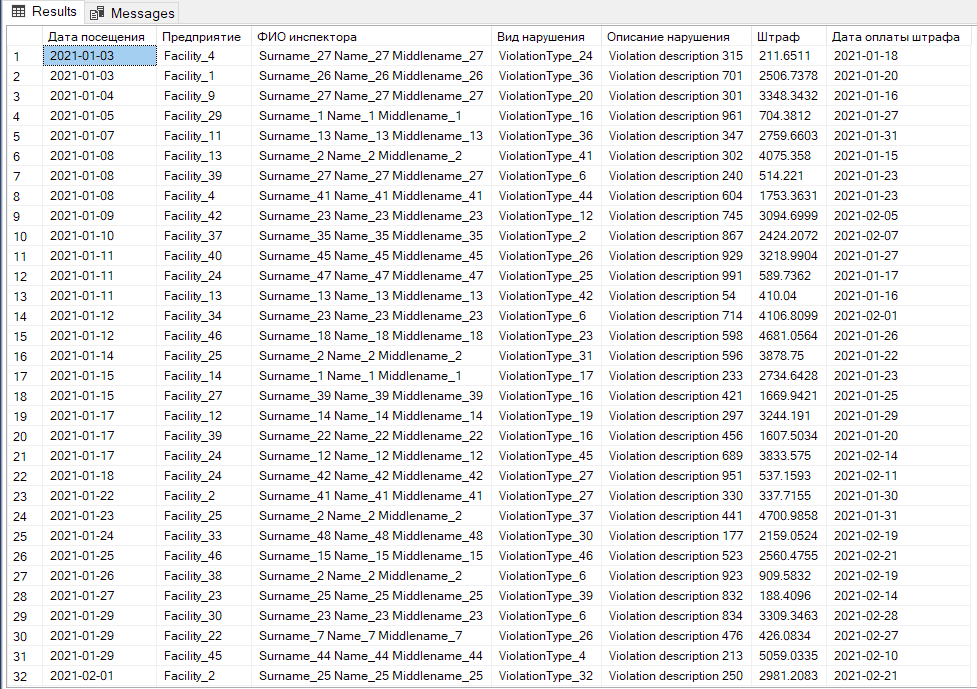


Рисунок 2 – Список отчетов о посещениях предприятия инспекторами (тестовые данные)

Также разработаны процедуры для поиска конкретной информации по заданным пользователем критериям. Список процедур и выполняемые ими задачи следующий:

– процедура «*usp\_VisitsPerPeriod*» – возвращает сведения о посещениях предприятия инспекторами в течение заданного пользователем периода;

– процедура «*usp\_FacilityFinesPerPeriod*» – возвращает сумму штрафов за заданный пользователем период для заданного пользователем предприятия;

– процедура «*usp\_MajorViolationInYear*» – возвращает вид нерационального использования энергоресурсов, по которому в прошлом году было выплачено больше всего штрафов.

На рисунке 3 представлен результат работы процедуры «*usp\_VisitsPerPeriod*».

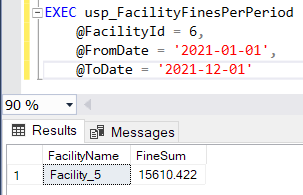


Рисунок 3 – Результат работы процедуры «*usp\_VisitsPerPeriod*»

На рисунке 4 представлен результат работы процедуры «*usp\_FacilityFinesPerPeriod*».

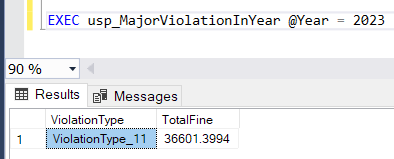


Рисунок 4 – Результат работы процедуры «*usp\_FacilityFinesPerPeriod*»

На рисунке 5 представлен результат работы процедуры «*usp\_MajorViolationInYear*».

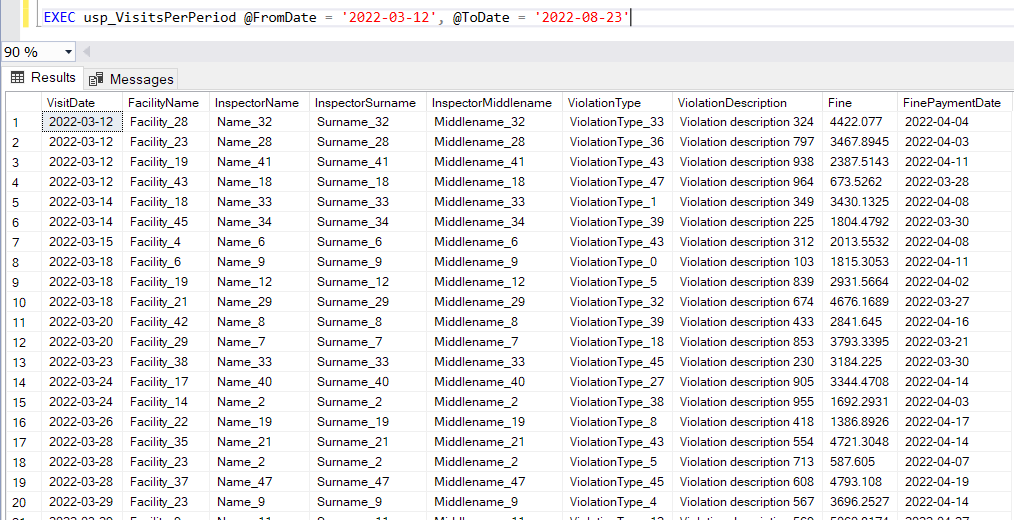


Рисунок 5 – результат работы процедуры «*usp\_MajorViolationInYear*»

Текст разработанных *T-SQL* скриптов представлен в приложении А.

**Вывод:** в результате выполнения лабораторной работы разработана схема базы данных «Журнал инспектора энергонадзора». Разработаны *T-SQL* скрипты для создания таблиц базы данных и генерации тестовых данных. Разработаны представления для удобного просмотра данных, а также процедуры для вывода конкретной информации по заданным пользователем критериям.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст *T-SQL* скриптов**

***T-SQL* скрипт для создания базы данных и таблиц:**

IF NOT EXISTS (SELECT 1 FROM sys.databases WHERE name = 'FACILITY\_INSPECTION')

BEGIN

CREATE DATABASE FACILITY\_INSPECTION;

END;

GO

USE FACILITY\_INSPECTION;

CREATE TABLE Facilities(

Id INT PRIMARY KEY IDENTITY,

Name NVARCHAR(30) NOT NULL UNIQUE

);

CREATE TABLE Inspectors(

Id INT PRIMARY KEY IDENTITY,

Name NVARCHAR(30) NOT NULL,

Surname NVARCHAR(30) NOT NULL,

Middlename NVARCHAR(30) NOT NULL

);

CREATE TABLE ViolationTypes(

Id INT PRIMARY KEY IDENTITY,

Name NVARCHAR(30) NOT NULL UNIQUE

);

CREATE TABLE EnergyResourceViolations(

Id INT PRIMARY KEY IDENTITY,

FacilityId INT NOT NULL,

InspectorId INT NOT NULL,

VisitDate DATE NOT NULL,

ViolationTypeId INT NOT NULL,

ViolationDescription NVARCHAR(300) NOT NULL,

Fine SMALLMONEY NOT NULL,

FinePaymentDate DATE NOT NULL,

CONSTRAINT FK\_EnergyResourceViolations\_To\_Facilities

FOREIGN KEY (FacilityId)

REFERENCES Facilities(Id)

ON DELETE CASCADE,

CONSTRAINT FK\_EnergyResourceViolations\_To\_Inspectors

FOREIGN KEY (InspectorId)

REFERENCES Inspectors(Id)

ON DELETE CASCADE,

CONSTRAINT FK\_EnergyResourceViolations\_To\_ViolationTypes

FOREIGN KEY (ViolationTypeId)

REFERENCES ViolationTypes(Id)

ON DELETE CASCADE,

);

***T-SQL* скрипт для генерации тестовых данных:**

USE FACILITY\_INSPECTION;

DECLARE @Counter INT = 0;

DECLARE @EntriesTier0 INT = 50;

DECLARE @EntriesTier1 INT = 1000;

WHILE @Counter < @EntriesTier0

BEGIN

INSERT INTO Facilities (Name)

VALUES ('Facility\_' + CAST(@Counter AS NVARCHAR));

SET @Counter = @Counter + 1;

END;

SET @Counter = 0;

WHILE @Counter < @EntriesTier0

BEGIN

INSERT INTO Inspectors (Name, Surname, Middlename)

VALUES (

'Name\_' + CAST(@Counter AS NVARCHAR),

'Surname\_' + CAST(@Counter AS NVARCHAR),

'Middlename\_' + CAST(@Counter AS NVARCHAR)

);

SET @Counter = @Counter + 1;

END;

SET @Counter = 0;

WHILE @Counter < @EntriesTier0

BEGIN

INSERT INTO ViolationTypes (Name)

VALUES ('ViolationType\_' + CAST(@Counter AS NVARCHAR));

SET @Counter = @Counter + 1;

END;

SET @Counter = 0;

DECLARE @StartDate DATE = '2021-01-01';

DECLARE @EndDate DATE = '2023-12-01';

DECLARE @DaysInRangeBig INT = DATEDIFF(DAY, @StartDate, @EndDate);

DECLARE @DaysInRangeSmall INT = 30;

DECLARE @RandomDays1 INT = 0;

DECLARE @RandomDays2 INT = 0;

DECLARE @RandomDate1 DATE = @StartDate;

DECLARE @RandomDate2 DATE = @StartDate

DECLARE @RandomInt1 INT = 0;

DECLARE @RandomInt2 INT = 0;

DECLARE @RandomInt3 INT = 0;

DECLARE @RandomMoney SMALLMONEY = 0;

DECLARE @MoneyRange INT = 5000;

DECLARE @MoneyMin INT = 100;

WHILE @Counter < @EntriesTier1

BEGIN

-- Generate random Id Tier 0

SET @RandomInt1 = ROUND(RAND() \* (@EntriesTier0 - 1) + 1, 0);

SET @RandomInt2 = ROUND(RAND() \* (@EntriesTier0 - 1) + 1, 0);

SET @RandomInt3 = ROUND(RAND() \* (@EntriesTier0 - 1) + 1, 0);

-- Generate random money value

SET @RandomMoney = CAST((RAND() \* @MoneyRange + @MoneyMin) AS SMALLMONEY);

-- Generate a random number of days to add to date

SET @RandomDays1 = ROUND(RAND() \* @DaysInRangeBig, 0);

SET @RandomDays2 = ROUND(RAND() \* @DaysInRangeSmall, 0);

-- Calculate the random date within the range

SET @RandomDate1 = DATEADD(DAY, @RandomDays1, @StartDate);

SET @RandomDate2 = DATEADD(DAY, @RandomDays2, @RandomDate1);

INSERT INTO EnergyResourceViolations (FacilityId, InspectorId, VisitDate, ViolationTypeId, ViolationDescription, Fine, FinePaymentDate)

VALUES (@RandomInt1, @RandomInt2, @RandomDate1, @RandomInt3, 'Violation description ' + CAST(@Counter AS NVARCHAR), @RandomMoney, @RandomDate2);

SET @Counter = @Counter + 1;

END;

***T-SQL* скрипт для создания представления:**

USE FACILITY\_INSPECTION;

GO

CREATE VIEW VI\_EnergyResourceViolations AS

SELECT

ERV.VisitDate AS 'Дата посещения',

F.Name AS 'Предприятие',

I.Surname + ' ' + I.Name + ' ' + I.Middlename AS 'ФИО инспектора',

VT.Name AS 'Вид нарушения',

ERV.ViolationDescription AS 'Описание нарушения',

ERV.Fine AS 'Штраф',

ERV.FinePaymentDate AS 'Дата оплаты штрафа'

FROM

EnergyResourceViolations AS ERV

INNER JOIN Facilities AS F

ON F.Id = ERV.FacilityId

INNER JOIN Inspectors AS I

ON I.Id = ERV.InspectorId

INNER JOIN ViolationTypes AS VT

ON VT.Id = ERV.ViolationTypeId

***T-SQL* скрипт для создания процедур:**

USE FACILITY\_INSPECTION;

GO

CREATE PROCEDURE usp\_VisitsPerPeriod

@FromDate DATE,

@ToDate DATE

AS

BEGIN

SELECT

ERV.VisitDate,

F.Name AS FacilityName,

I.Name AS InspectorName,

I.Surname AS InspectorSurname,

I.Middlename AS InspectorMiddlename,

VT.Name AS ViolationType,

ERV.ViolationDescription AS ViolationDescription,

ERV.Fine,

ERV.FinePaymentDate

FROM

EnergyResourceViolations AS ERV

INNER JOIN Facilities AS F

ON ERV.FacilityId = F.Id

INNER JOIN Inspectors AS I

ON ERV.InspectorId = I.Id

INNER JOIN ViolationTypes AS VT

ON ERV.ViolationTypeId = VT.Id

WHERE ERV.VisitDate BETWEEN @FromDate AND @ToDate

ORDER BY ERV.VisitDate

END;

GO

CREATE PROCEDURE usp\_FacilityFinesPerPeriod

@FacilityId INT,

@FromDate DATE,

@ToDate DATE

AS

BEGIN

SELECT

F.Name AS FacilityName,

SUM(ERV.Fine) AS FineSum

FROM

EnergyResourceViolations AS ERV

INNER JOIN Facilities AS F

ON ERV.FacilityId = F.Id

WHERE F.Id = @FacilityId

AND ERV.VisitDate BETWEEN @FromDate AND @ToDate

GROUP BY F.Name

END;

GO

CREATE PROCEDURE usp\_MajorViolationInYear

@Year INT

AS

BEGIN

DECLARE @MaxFine SMALLMONEY;

DECLARE @ViolationSummary TABLE (

ViolationType NVARCHAR(30),

TotalFine SMALLMONEY

);

-- Calculate Total Fine

INSERT INTO @ViolationSummary (ViolationType, TotalFine)

SELECT

VT.Name as ViolationType,

SUM(ERV.Fine) AS TotalFine

FROM

EnergyResourceViolations AS ERV

INNER JOIN ViolationTypes AS VT

ON ERV.ViolationTypeId = VT.Id

WHERE YEAR(ERV.VisitDate) = @Year

GROUP BY VT.Name

-- Get max fine

SELECT @MaxFine = MAX(TotalFine) FROM @ViolationSummary

-- Get Violation Type with max fine

SELECT \* FROM @ViolationSummary WHERE TotalFine = @MaxFine

END;