**..МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

«Разработка серверной части информационной системы

в СУБД *MS* *SQL Server*»

ВАРИАНТ № 28

Выполнил:

студент гр. ИТП-31, Трацевский И. С.

Принял:

доцент Асенчик О. Д.

Гомель 2023

**Цель работы:** разработать серверную часть клиент-серверной информационной системы, основанной на базе данных в заданной предметной области средствами СУБД *MS SQL Server*.

**Задание:** 1. Разработать логическую модель реляционной базы данных, моделирующую предметную область согласно своему варианту задания. Структура БД должна быть нормализована – таблицы должны удовлетворять требованиям третьей нормальной формы.

2. Создать базу данных и таблицы в СУБД *MS SQL Server* и заполнить их тестовым набором данных, для этого написать *Transact SQL* скрипт:

2.1. Создания базы данных и ее таблиц.

При создании таблиц должны быть назначены первичные и внешние ключи и установлены необходимые ограничения целостности данных. Наименования таблиц и полей давать в соответствии с соглашением об именовании этих объектов.

2.2. Заполнения не менее чем трех таблиц БД данными (см. пример внутри этого электронного курса).

При выборе таблиц для заполнения тестовыми наборами руководствоваться следующим:

выбранные таблицы должны содержать основную информацию, касающиеся предметной области приложения;

таблицы должны быть связаны непосредственно;

следует воздерживаться от выбора таблиц, характеризующих кадровую подсистему приложения (данные о сотрудниках и их должностях);

не менее, чем одна таблица должна находиться в схеме базы данных на стороне отношения «многие».

При этом заполнение осуществлять в следующем порядке: сначала генерируются данные для таблиц на стороне отношения «один» (таблицы- «справочники»), потом – в таблицы на стороне отношения «многие» («оперативные таблицы»).

БД заполнить записями в количестве, необходимом для отладки и демонстрации возможностей приложения. Таблицы на стороне отношения один должны содержать не менее 500 записей, таблице на стороне отношения многие должны содержать не менее 20000 записей.

3. Создать с использованием средств *Transact SQL* представления, позволяющие отображать данные в удобном для пользователя виде, и заменяющие часто используемые запросы на выборку из двух и более связанных таблиц.

4. Написать не менее трех хранимых процедур с параметрами для вставки и (или) обновление данных в таблицы базы данных.

**Ход работы:** выполнение лабораторной работы заключается в создании *SQL-*скрипта, который выполнит по порядку следующие действия:

1. Создание базы данных;
2. Создание таблиц в базе данных;
3. Создание связей между таблицами;
4. Генерация случайных данных и заполнение ими таблиц;
5. Создание представлений и процедур.

Предметная область и задание варианта показано на рисунке 1.

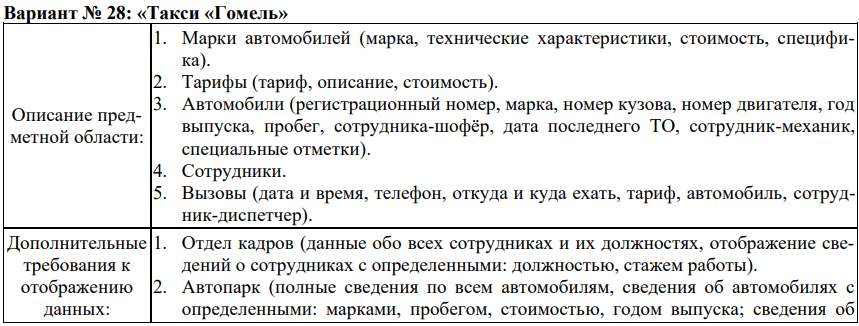


Рисунок 1 – Задание варианта

Для создания базы данных и таблиц использовались стандартные команды *CREATE DATABASE* и *CREATE TABLE*. В разработанном скрипте создается 5 таблиц: *CarModels, Rates, Cars, Positions, Employees, Calls.* После генерации таблиц были настроены связи между ними. На рисунке 2 показана диаграмма получившейся базы данных.

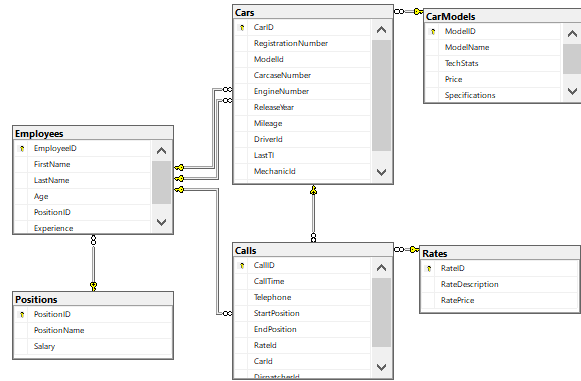
****

Рисунок 2 – Диаграмма базы данных

Заполнение таблиц выполнялось при помощи циклов и случайной генерации чисел и комбинаций символов. На рисунке 3 приведен пример одной из заполненных таблиц.

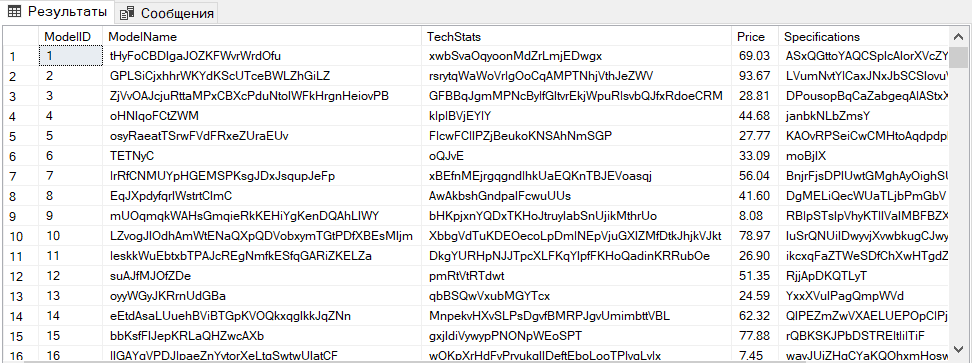


Рисунок 3 – Заполненная таблица *CarModels*

Заключительной частью работы стало написание нескольких процедур и представлений. Были разработаны два представления. Представление, показывающее для каждой машины ее водителя, а также представление, предоставляющее информацию о занимаемой должности каждого сотрудника. Процедуры же предоставляют функции по вставке и обновлению данных в таблицах.

Листинг разработанного скрипта приведен в приложении А.

**Вывод:** результатом выполнения лабораторной работы является *SQL-*скрипт. Данный скрипт выполняет задачу по созданию базы данный и таблиц в ней, а также заполняет таблицы случайными значениями. Помимо таблицы были созданы представления и процедуры. Разработанный скрипт был загружен в репозиторий *GitHub.*

***GitHub:*** https://github.com/IvanTratsevski/DB-LABS/tree/main/lab1

**ПРИЛОЖЕНИЕ А**

(обязательное)

***SQL*-скрипт**

USE master

CREATE DATABASE TaxiGomel

GO

ALTER DATABASE TaxiGomel SET RECOVERY SIMPLE

GO

USE TaxiGomel

CREATE TABLE dbo.CarModels (ModelID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

ModelName nvarchar(50),

TechStats nvarchar(50),

Price decimal(18,2),

Specifications nvarchar(50))

CREATE TABLE dbo.Rates (RateID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

RateDescription nvarchar(50),

RatePrice decimal(18,2))

CREATE TABLE dbo.Cars (CarID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

RegistrationNumber nvarchar(20),

ModelId int,

CarcaseNumber nvarchar(17),

EngineNumber nvarchar(17),

ReleaseYear date,

Mileage int,

DriverId int,

LastTI date,

MechanicId int,

SpecialMarks nvarchar(50))

CREATE TABLE dbo.Positions (PositionID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

PositionName nvarchar(50),

Salary decimal(18,2))

CREATE TABLE dbo.Employees (EmployeeID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

FirstName nvarchar(50),

LastName nvarchar(50),

Age int,

PositionID int,

Experience int

CREATE TABLE dbo.Calls (CallID int IDENTITY(1,1) NOT NULL PRIMARY KEY,

CallTime smalldatetime,

Telephone varchar(12),

StartPosition nvarchar(20),

EndPosition nvarchar(20),

RateId int,

CarId int,

DispatcherId int)

ALTER TABLE dbo.Cars WITH CHECK ADD CONSTRAINT FK\_Cars\_CarModels FOREIGN KEY(ModelID)

REFERENCES dbo.CarModels (ModelID) ON DELETE CASCADE

GO

ALTER TABLE dbo.Cars WITH CHECK ADD CONSTRAINT FK\_Cars\_EmployeesD FOREIGN KEY(DriverID)

REFERENCES dbo.Employees (EmployeeID) ON DELETE CASCADE

GO

ALTER TABLE dbo.Cars WITH CHECK ADD CONSTRAINT FK\_Cars\_EmployeesM FOREIGN KEY(MechanicID)

REFERENCES dbo.Employees (EmployeeID) ON DELETE NO ACTION

GO

ALTER TABLE dbo.Employees WITH CHECK ADD CONSTRAINT FK\_Employees\_Positions FOREIGN KEY(PositionID)

REFERENCES dbo.Positions (PositionID) ON DELETE CASCADE

GO

ALTER TABLE dbo.Calls WITH CHECK ADD CONSTRAINT FK\_Calls\_Cars FOREIGN KEY(CarID)

REFERENCES dbo.Cars (CarID) ON DELETE CASCADE

GO

ALTER TABLE dbo.Calls WITH CHECK ADD CONSTRAINT FK\_Calls\_Employees FOREIGN KEY(DispatcherID)

REFERENCES dbo.Employees (EmployeeID) ON DELETE NO ACTION

GO

ALTER TABLE dbo.Calls WITH CHECK ADD CONSTRAINT FK\_Calls\_Rates FOREIGN KEY(RateID)

REFERENCES dbo.Rates (RateID) ON DELETE NO ACTION

GO

SET NOCOUNT ON

DECLARE @Symbol CHAR(52)= 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz',

@Position int,

@PositionName nvarchar(50),

@Salary decimal(18,2),

@FirstName nvarchar(50),

@LastName nvarchar(50),

@Age int,

@PositionID int,

@Experience int,

@ModelName nvarchar(50),

@TechStats nvarchar(50),

@Price decimal(18,2),

@Specifications nvarchar(50),

@RegistrationNumber nvarchar(20),

@ModelId int,

@CarcaseNumber nvarchar(17),

@EngineNumber nvarchar(17),

@ReleaseYear date,

@Mileage int,

@DriverId int,

@LastTI date,

@MechanicId int,

@SpecialMarks nvarchar(50),

@i int,

@NameLimit int,

@odate date,

@Inc\_Exp real,

@RowCount INT,

@NumberPositions int,

@NumberEmployees int,

@NumberCarModels int,

@NumberCars int,

@MinNumberSymbols int,

@MaxNumberSymbols int

SET @NumberPositions = 500

SET @NumberCarModels = 500

SET @NumberCars = 20000

SET @NumberEmployees = 20000

BEGIN TRAN

SELECT @i=0 FROM dbo.Positions WITH (TABLOCKX) WHERE 1=0

SET @RowCount=1

SET @MinNumberSymbols=5

SET @MaxNumberSymbols=50

WHILE @RowCount<=@NumberPositions

BEGIN

SET @NameLimit=@MinNumberSymbols+RAND()\*(@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

SET @i=1

SET @PositionName=''

WHILE @i<=@NameLimit

BEGIN

SET @Position=RAND()\*52

SET @PositionName = @PositionName + SUBSTRING(@Symbol, @Position, 1)

SET @Salary = ROUND(RAND(CHECKSUM(NEWID())) \* (100), 2)

SET @i=@i+1

END

INSERT INTO dbo.Positions (PositionName, Salary) SELECT @PositionName, @Salary

SET @RowCount +=1

END

SELECT @i=0 FROM dbo.Employees WITH (TABLOCKX) WHERE 1=0

SET @RowCount=1

SET @MinNumberSymbols=5

SET @MaxNumberSymbols=50

WHILE @RowCount<=@NumberEmployees

BEGIN

SET @NameLimit=@MinNumberSymbols+RAND()\*(@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

SET @i=1

SET @FirstName = ''

SET @LastName = ''

WHILE @i<=@NameLimit

BEGIN

SET @Position=RAND()\*52

SET @FirstName = @FirstName + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @LastName = @LastName + SUBSTRING(@Symbol, @Position, 1)

SET @Age = CAST(RAND()\*52 as int)

SET @PositionID = CAST( (1+RAND()\*(@NumberPositions-1)) as int)

SET @Experience = CAST(RAND()\*52 as int)

SET @i=@i+1

END

INSERT INTO dbo.Employees (FirstName, LastName, Age, PositionID, Experience) SELECT @FirstName, @LastName, @Age, @PositionID, @Experience

SET @RowCount +=1

END

SELECT @i=0 FROM dbo.CarModels WITH (TABLOCKX) WHERE 1=0

SET @RowCount=1

SET @MinNumberSymbols=5

SET @MaxNumberSymbols=50

WHILE @RowCount<=@NumberCarModels

BEGIN

SET @NameLimit=@MinNumberSymbols+RAND()\*(@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

SET @i=1

SET @ModelName = ''

SET @TechStats = ''

SET @Specifications = ''

WHILE @i<=@NameLimit

BEGIN

SET @Position=RAND()\*52

SET @ModelName = @ModelName + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @TechStats = @TechStats + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @Specifications = @Specifications + SUBSTRING(@Symbol, @Position, 1)

SET @Price = ROUND(RAND(CHECKSUM(NEWID())) \* (100), 2)

SET @i=@i+1

END

INSERT INTO dbo.CarModels (ModelName, TechStats, Price, Specifications) SELECT @ModelName, @TechStats, @Price, @Specifications

SET @RowCount +=1

END

SELECT @i=0 FROM dbo.Cars WITH (TABLOCKX) WHERE 1=0

SET @RowCount=1

SET @MinNumberSymbols=5

SET @MaxNumberSymbols=50

WHILE @RowCount<=@NumberCars

BEGIN

SET @NameLimit=@MinNumberSymbols+RAND()\*(@MaxNumberSymbols-@MinNumberSymbols) -- имя от 5 до 50 символов

SET @i=1

SET @RegistrationNumber = ''

SET @CarcaseNumber = ''

SET @EngineNumber = ''

SET @SpecialMarks = ''

WHILE @i<=@NameLimit

BEGIN

SET @Position=RAND()\*52

SET @RegistrationNumber = @RegistrationNumber + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @CarcaseNumber = @CarcaseNumber + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @EngineNumber = @EngineNumber + SUBSTRING(@Symbol, @Position, 1)

SET @Position=RAND()\*52

SET @SpecialMarks = @SpecialMarks + SUBSTRING(@Symbol, @Position, 1)

SET @ModelId = CAST( (1+RAND()\*(@NumberCarModels-1)) as int)

SET @DriverId = CAST( (1+RAND()\*(@NumberEmployees-1)) as int)

SET @MechanicId = CAST( (1+RAND()\*(@NumberEmployees-1)) as int)

SET @Mileage=RAND()\*52

SET @LastTI=dateadd(day,-RAND()\*15000,GETDATE())

SET @ReleaseYear=dateadd(day,-RAND()\*15000,GETDATE())

SET @i=@i+1

END

INSERT INTO dbo.Cars (RegistrationNumber, ModelId, CarcaseNumber, EngineNumber, ReleaseYear, Mileage, DriverId, LastTI, MechanicId, SpecialMarks) SELECT @RegistrationNumber, @ModelId, @CarcaseNumber, @EngineNumber, @ReleaseYear, @Mileage, @DriverId, @LastTI, @MechanicId, @SpecialMarks

SET @RowCount +=1

END

COMMIT TRAN

GO

CREATE VIEW [dbo].[View\_EmployeeAndPositions]

AS

SELECT dbo.Employees.EmployeeID, dbo.Employees.FirstName, dbo.Employees.LastName, dbo.Employees.Age, dbo.Employees.Experience, dbo.Employees.PositionID,

dbo.Positions.PositionName, dbo.Positions.Salary

FROM dbo.Positions INNER JOIN

dbo.Employees ON dbo.Positions.PositionID = dbo.Employees.PositionID

GO

CREATE VIEW [dbo].[View\_CarsAndDrivers]

AS

SELECT dbo.Cars.CarID, dbo.Cars.RegistrationNumber, dbo.Cars.ModelID, dbo.Cars.DriverId, (dbo.Employees.FirstName + ' ' + dbo.Employees.LastName) AS Driver, dbo.Cars.SpecialMarks

FROM dbo.Employees INNER JOIN

dbo.Cars ON dbo.Employees.EmployeeID = dbo.Cars.DriverId

GO

CREATE PROCEDURE ChangeLastTI (@CarId int, @NewTI date)

AS UPDATE dbo.Cars

SET LastTi = @NewTI

WHERE(

dbo.Cars.CarID = @CarId

);

GO

CREATE PROCEDURE AddCar (@RegistrationNumber nvarchar(20), @ModelId int, @CarcaseNumber nvarchar(17), @EngineNumber nvarchar(17), @ReleaseYear date, @Mileage int, @DriverId int, @LastTI date, @MechanicId int, @SpecialMarks nvarchar(50))

AS INSERT INTO dbo.Cars(

RegistrationNumber,

ModelId,

CarcaseNumber,

EngineNumber,

ReleaseYear,

Mileage,

DriverId,

LastTI,

MechanicId,

SpecialMarks)

VALUES

(@RegistrationNumber,

@ModelId ,

@CarcaseNumber,

@EngineNumber,

@ReleaseYear,

@Mileage,

@DriverId,

@LastTI,

@MechanicId,

@SpecialMarks)

GO

CREATE PROCEDURE AddEmployee (@FirstName nvarchar(50), @LastName nvarchar(50), @Age int, @PositionID int, @Experience int)

AS INSERT INTO dbo.Employees(

FirstName,

LastName,

Age,

PositionID,

Experience)

VALUES

(@FirstName,

@LastName ,

@Age,

@PositionID,

@Experience)