**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: **«Разработка приложений баз данных для информационных систем»**

на тему: **Разработка серверной части информационной системы в СУБД *MS SQL Server***

Выполнил: студент группы ИТП-31

Пронуза М.Ю.

Проверил:

Малиновский И.Л.

Гомель 2024

**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* на локальном сервере и заполнить их тестовым набором данных, для этого написать *Transact SQL* скрипт (скрипты):

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 (см. советующие материалы).

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.

5. Разместить разработанную базу данных на удаленном сервере, для этого:

• выбрать подходящий хостинг для СУБД *MS SQL Server*;

• получить учетную запись путем выполнения рекомендаций хостинга;

• создать базу данных;

• запустить разработанный *Transact SQL* скрипт (скрипты) создания объектов базы данных и заполнения таблиц тестовым набором данных.

**Ход выполнения результаты работы:**

Текстовый вариант задания для варианта 17 представлен на рисунке 1.

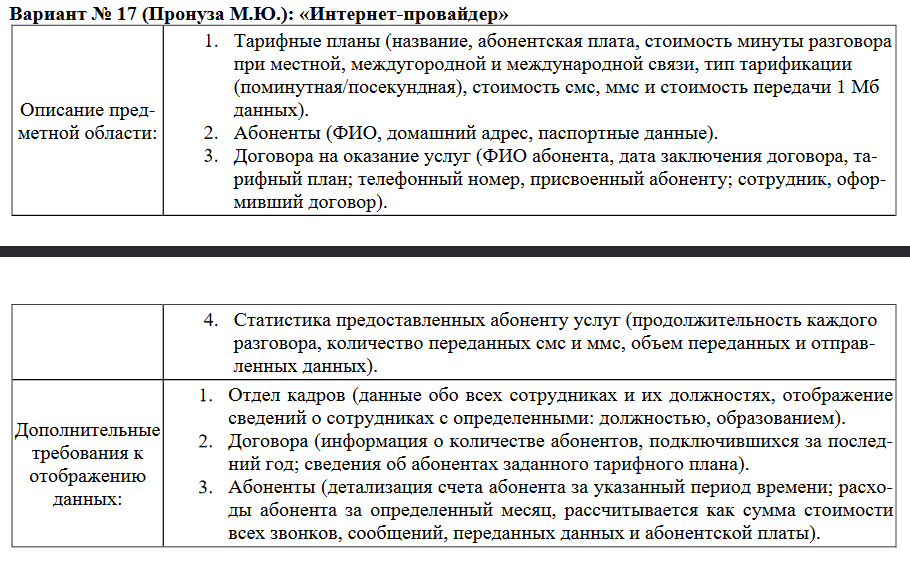


Рисунок 1 – Задание варианта 17

На рисунке 2 представлены созданные таблицы базы данных.

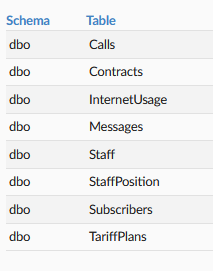


Рисунок 2 – Таблицы базы данных

На рисунке 3 представлено количество данных в каждой из таблиц.

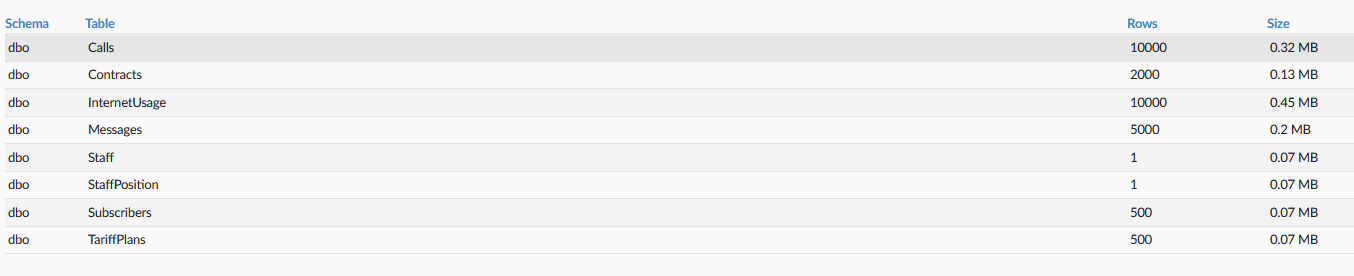


Рисунок 3 – Количество данных в таблицах

На рисунке 4 представлены созданные представления и хранимые процедуры.

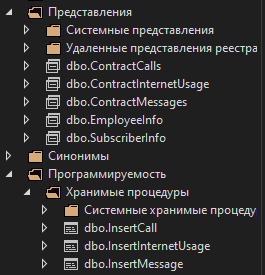


Рисунок 4 – Представления и хранимые процедуры

На рисунке 5 представлена схема базы данных.

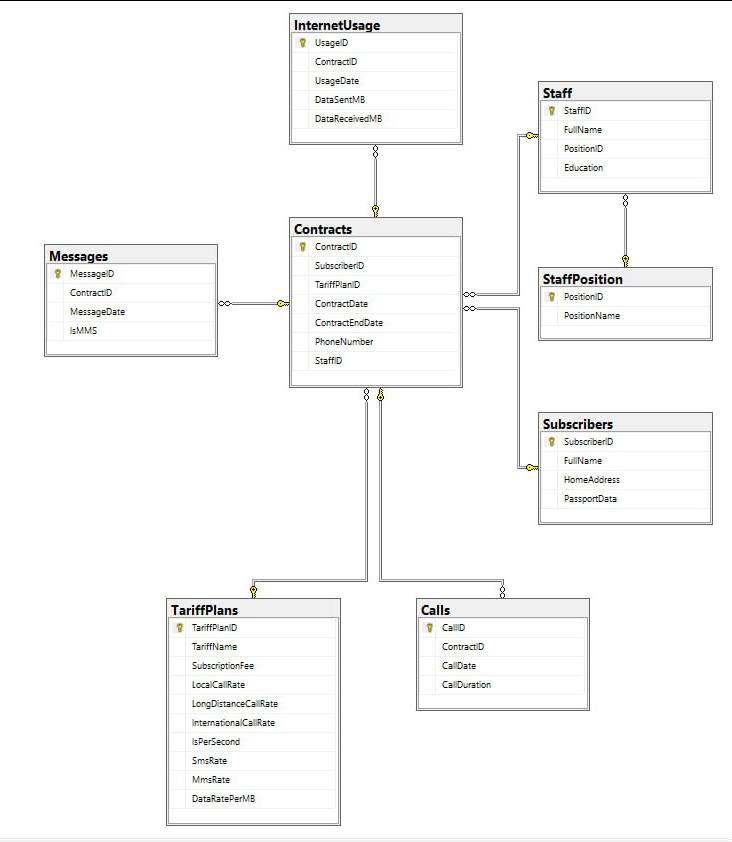


Рисунок 5 – Схема базы данных

Строка подключения к удаленному серверу имеет вид:

Server=db8328.databaseasp.net; Database=db8328; User Id=db8328; Password=5z?X=4Mb3m-Q; Encrypt=False; MultipleActiveResultSets=True;

Текст для создания базы данных и всех ее таблицпредставлен в приложении А.

Текст для генерации тестового набора данныхпредставлен в приложении Б.

Текст для не менее трех представлений представлен в приложении В.

Текст для не менее трех хранимых процедур представлен в приложении Г.

**Вывод:** в ходе выполнения лабораторной работы изучены основы серверной части клиент-серверной информационной системы. В результате выполнения лабораторной работы разработана база данных, заполнена данными и созданы представления и хранимые процедуры для работы с ней.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст создания базы данных и таблиц**

-- Таблица для хранения должностей сотрудников

CREATE TABLE StaffPosition (

PositionID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор должности

PositionName NVARCHAR(100) NOT NULL -- Название должности (например, "Менеджер", "Оператор")

);

-- Таблица для хранения информации о сотрудниках

CREATE TABLE Staff (

StaffID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор сотрудника

FullName NVARCHAR(150) NOT NULL, -- ФИО сотрудника

PositionID INT NOT NULL FOREIGN KEY REFERENCES StaffPosition(PositionID), -- Ссылка на должность сотрудника

Education NVARCHAR(100) -- Образование сотрудника

);

--Таблица для хранения информации о тарифных планах

CREATE TABLE TariffPlans (

TariffPlanID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор тарифного плана

TariffName NVARCHAR(100) NOT NULL, -- Название тарифного плана

SubscriptionFee DECIMAL(10, 2) NOT NULL, -- Абонентская плата

LocalCallRate DECIMAL(10, 2) NOT NULL, -- Стоимость минуты разговора при местной связи

LongDistanceCallRate DECIMAL(10, 2) NOT NULL,-- Стоимость минуты разговора при междугородной связи

InternationalCallRate DECIMAL(10, 2) NOT NULL,-- Стоимость минуты разговора при международной связи

IsPerSecond BIT NOT NULL, -- Если FALSE, то тарификация поминутная; если TRUE, то посекундная

SmsRate DECIMAL(10, 2) NOT NULL, -- Стоимость SMS

MmsRate DECIMAL(10, 2) NOT NULL, -- Стоимость MMS

DataRatePerMB DECIMAL(10, 2) NOT NULL -- Стоимость передачи 1 МБ данных

);

-- Таблица для хранения информации об абонентах

CREATE TABLE Subscribers (

SubscriberID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор абонента

FullName NVARCHAR(150) NOT NULL, -- ФИО абонента

HomeAddress NVARCHAR(255), -- Домашний адрес абонента

PassportData NVARCHAR(100) -- Паспортные данные абонента

);

-- Таблица для хранения договоров на оказание услуг

CREATE TABLE Contracts (

ContractID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор договора

SubscriberID INT NOT NULL FOREIGN KEY REFERENCES Subscribers(SubscriberID), -- Идентификатор абонента

TariffPlanID INT NOT NULL FOREIGN KEY REFERENCES TariffPlans(TariffPlanID), -- Идентификатор тарифного плана

ContractDate DATE NOT NULL, -- Дата заключения договора

ContractEndDate DATE NULL, -- Дата окончания договора (может быть NULL)

PhoneNumber NVARCHAR(20) NOT NULL, -- Телефонный номер, присвоенный абоненту

StaffID INT NOT NULL FOREIGN KEY REFERENCES Staff(StaffID) -- Ссылка на сотрудника, оформившего договор

);

-- Таблица для хранения информации о звонках

CREATE TABLE Calls (

CallID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор звонка

ContractID INT NOT NULL FOREIGN KEY REFERENCES Contracts(ContractID), -- Ссылка на договор

CallDate DATETIME NOT NULL, -- Дата и время звонка

CallDuration INT NOT NULL -- Продолжительность разговора (в секундах)

);

-- Таблица для хранения информации о смс и ммс

CREATE TABLE Messages (

MessageID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор сообщения

ContractID INT NOT NULL FOREIGN KEY REFERENCES Contracts(ContractID), -- Ссылка на договор

MessageDate DATETIME NOT NULL, -- Дата и время отправки сообщения (SMS или MMS)

IsMMS BIT NOT NULL -- Если значение FALSE, то это SMS; если TRUE, то MMS

);

-- Таблица для хранения информации о трафике интернета

CREATE TABLE InternetUsage (

UsageID INT PRIMARY KEY IDENTITY(1,1), -- Идентификатор записи использования интернета

ContractID INT NOT NULL FOREIGN KEY REFERENCES Contracts(ContractID), -- Ссылка на договор

UsageDate DATETIME NOT NULL, -- Дата и время использования интернета

DataSentMB DECIMAL(10, 2) NOT NULL, -- Объем отправленных данных (в МБ)

DataReceivedMB DECIMAL(10, 2) NOT NULL -- Объем полученных данных (в МБ)

);

**ПРИЛОЖЕНИЕ Б**

(обязательное)

**Текст заполнения таблиц данными**

-- Настройки для вставки данных

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

@Position INT,

@i INT,

@NameLimit INT,

@FullName NVARCHAR(150),

@HomeAddress NVARCHAR(255),

@PassportData NVARCHAR(100),

@TariffName NVARCHAR(100),

@TariffPlanID INT,

@SubscriberID INT,

@StaffID INT = 1, -- один сотрудник

@TariffPlanCount INT = 500,

@SubscriberCount INT = 500,

@ContractCount INT = 2000,

@PhoneNumber NVARCHAR(20),

@ContractDate DATE,

@ContractEndDate DATE,

@MinSymbols INT,

@MaxSymbols INT;

-- Добавление должности для сотрудника

INSERT INTO StaffPosition (PositionName)

VALUES ('Оператор');

-- Добавление сотрудников

INSERT INTO Staff (FullName, PositionID, Education)

VALUES

('Сергей Петров', 1, 'Среднее'),

('Александр Смирнов', 1, 'Среднее'),

('Дмитрий Соколов', 1, 'Высшее'),

('Андрей Кузнецов', 1, 'Среднее'),

('Михаил Попов', 1, 'Высшее'),

('Алексей Волков', 1, 'Высшее'),

('Николай Лебедев', 1, 'Среднее'),

('Олег Семёнов', 1, 'Среднее'),

('Владимир Павлов', 1, 'Высшее'),

('Игорь Козлов', 1, 'Среднее'),

('Максим Морозов', 1, 'Высшее'),

('Артём Новиков', 1, 'Среднее'),

('Евгений Беляев', 1, 'Высшее'),

('Антон Ершов', 1, 'Среднее'),

('Роман Фролов', 1, 'Высшее'),

('Григорий Сафонов', 1, 'Среднее'),

('Павел Гусев', 1, 'Среднее'),

('Виктор Киселёв', 1, 'Высшее'),

('Константин Фёдоров', 1, 'Среднее'),

('Станислав Орлов', 1, 'Высшее');

-- Заполнение таблицы тарифных планов

SET @i = 1;

SET @MinSymbols = 5;

SET @MaxSymbols = 15;

WHILE @i <= @TariffPlanCount

BEGIN

-- Генерация случайного названия тарифного плана

SET @NameLimit = @MinSymbols + RAND() \* (@MaxSymbols - @MinSymbols);

SET @TariffName = '';

WHILE LEN(@TariffName) < @NameLimit

BEGIN

SET @Position = RAND() \* 52;

SET @TariffName = @TariffName + SUBSTRING(@Symbol, @Position, 1);

END;

-- Вставка записи в таблицу тарифов

INSERT INTO TariffPlans (TariffName, SubscriptionFee, LocalCallRate, LongDistanceCallRate, InternationalCallRate, IsPerSecond, SmsRate, MmsRate, DataRatePerMB)

VALUES (@TariffName, RAND() \* 100, RAND() \* 2, RAND() \* 5, RAND() \* 10, CAST(ROUND(RAND(), 0) AS BIT), RAND() \* 2, RAND() \* 5, RAND() \* 10);

SET @i = @i + 1;

END;

-- Заполнение таблицы абонентов

SET @i = 1;

SET @MinSymbols = 5;

SET @MaxSymbols = 15;

WHILE @i <= @SubscriberCount

BEGIN

-- Генерация случайных данных абонента

SET @NameLimit = @MinSymbols + RAND() \* (@MaxSymbols - @MinSymbols);

SET @FullName = '';

SET @HomeAddress = '';

SET @PassportData = '';

-- Генерация ФИО

WHILE LEN(@FullName) < @NameLimit

BEGIN

SET @Position = RAND() \* 52;

SET @FullName = @FullName + SUBSTRING(@Symbol, @Position, 1);

END;

-- Генерация домашнего адреса

WHILE LEN(@HomeAddress) < @NameLimit + 5

BEGIN

SET @Position = RAND() \* 52;

SET @HomeAddress = @HomeAddress + SUBSTRING(@Symbol, @Position, 1);

END;

-- Генерация паспортных данных

WHILE LEN(@PassportData) < 10

BEGIN

SET @Position = RAND() \* 52;

SET @PassportData = @PassportData + SUBSTRING(@Symbol, @Position, 1);

END;

-- Вставка записи в таблицу абонентов

INSERT INTO Subscribers (FullName, HomeAddress, PassportData)

VALUES (@FullName, @HomeAddress, @PassportData);

SET @i = @i + 1;

END;

-- Заполнение таблицы договоров

SET @i = 1;

WHILE @i <= @ContractCount

BEGIN

-- Выбор случайного абонента и тарифного плана

SET @SubscriberID = CAST(1 + RAND() \* (@SubscriberCount - 1) AS INT);

SET @TariffPlanID = CAST(1 + RAND() \* (@TariffPlanCount - 1) AS INT);

-- Генерация уникального номера телефона

SET @PhoneNumber = '89' + CAST(CAST(RAND() \* 1000000000 AS INT) AS NVARCHAR(10));

-- Генерация дат контракта

SET @ContractDate = DATEADD(DAY, -RAND() \* 3650, GETDATE()); -- дата начала

SET @ContractEndDate = DATEADD(YEAR, 1 + RAND() \* 5, @ContractDate); -- дата окончания

-- Проверка на перекрытие контрактов для абонента

IF NOT EXISTS (

SELECT 1

FROM Contracts

WHERE SubscriberID = @SubscriberID

AND @ContractDate BETWEEN ContractDate AND ISNULL(ContractEndDate, GETDATE())

)

BEGIN

-- Вставка записи в таблицу договоров

INSERT INTO Contracts (SubscriberID, TariffPlanID, ContractDate, ContractEndDate, PhoneNumber, StaffID)

VALUES (@SubscriberID, @TariffPlanID, @ContractDate, @ContractEndDate, @PhoneNumber, @StaffID);

SET @i = @i + 1;

END;

END;

-- Заполнение таблицы звонков

DECLARE @CallCount INT = 10000; -- количество записей о звонках

DECLARE @CallDate DATETIME;

DECLARE @CallDuration INT;

SET @i = 1;

WHILE @i <= @CallCount

BEGIN

-- Выбор случайного контракта

DECLARE @ContractID INT = CAST(1 + RAND() \* (SELECT COUNT(\*) FROM Contracts) AS INT);

-- Получение даты начала и окончания контракта

SELECT @ContractDate = ContractDate, @ContractEndDate = ContractEndDate

FROM Contracts

WHERE ContractID = @ContractID;

-- Генерация случайной даты звонка в пределах действия контракта

SET @CallDate = DATEADD(DAY, CAST(RAND() \* DATEDIFF(DAY, @ContractDate, ISNULL(@ContractEndDate, GETDATE())) AS INT), @ContractDate);

-- Генерация случайной длительности звонка (в секундах)

SET @CallDuration = CAST(1 + RAND() \* 3600 AS INT); -- до 1 часа

-- Вставка записи в таблицу звонков

INSERT INTO Calls (ContractID, CallDate, CallDuration)

VALUES (@ContractID, @CallDate, @CallDuration);

SET @i = @i + 1;

END;

-- Заполнение таблицы сообщений

DECLARE @MessageCount INT = 5000; -- количество записей о сообщениях

DECLARE @MessageDate DATETIME;

DECLARE @IsMMS BIT;

SET @i = 1;

WHILE @i <= @MessageCount

BEGIN

-- Выбор случайного контракта

SET @ContractID = CAST(1 + RAND() \* (SELECT COUNT(\*) FROM Contracts) AS INT);

-- Получение даты начала и окончания контракта

SELECT @ContractDate = ContractDate, @ContractEndDate = ContractEndDate

FROM Contracts

WHERE ContractID = @ContractID;

-- Генерация случайной даты сообщения в пределах действия контракта

SET @MessageDate = DATEADD(DAY, CAST(RAND() \* DATEDIFF(DAY, @ContractDate, ISNULL(@ContractEndDate, GETDATE())) AS INT), @ContractDate);

-- Генерация типа сообщения (SMS или MMS)

SET @IsMMS = CAST(ROUND(RAND(), 0) AS BIT);

-- Вставка записи в таблицу сообщений

INSERT INTO Messages (ContractID, MessageDate, IsMMS)

VALUES (@ContractID, @MessageDate, @IsMMS);

SET @i = @i + 1;

END;

-- Заполнение таблицы использования интернета

DECLARE @UsageCount INT = 10000; -- количество записей о трафике

DECLARE @UsageDate DATETIME;

DECLARE @DataSentMB DECIMAL(10, 2);

DECLARE @DataReceivedMB DECIMAL(10, 2);

SET @i = 1;

WHILE @i <= @UsageCount

BEGIN

-- Выбор случайного контракта

SET @ContractID = CAST(1 + RAND() \* (SELECT COUNT(\*) FROM Contracts) AS INT);

-- Получение даты начала и окончания контракта

SELECT @ContractDate = ContractDate, @ContractEndDate = ContractEndDate

FROM Contracts

WHERE ContractID = @ContractID;

-- Генерация случайной даты использования интернета в пределах действия контракта

SET @UsageDate = DATEADD(DAY, CAST(RAND() \* DATEDIFF(DAY, @ContractDate, ISNULL(@ContractEndDate, GETDATE())) AS INT), @ContractDate);

-- Генерация случайного объема данных

SET @DataSentMB = ROUND(RAND() \* 500, 2); -- до 500 MB

SET @DataReceivedMB = ROUND(RAND() \* 500, 2); -- до 500 MB

-- Вставка записи в таблицу использования интернета

INSERT INTO InternetUsage (ContractID, UsageDate, DataSentMB, DataReceivedMB)

VALUES (@ContractID, @UsageDate, @DataSentMB, @DataReceivedMB);

SET @i = @i + 1;

END;

-- Количество записей в таблице StaffPosition

SELECT 'StaffPosition' AS TableName, COUNT(\*) AS RecordCount

FROM StaffPosition;

-- Количество записей в таблице Staff

SELECT 'Staff' AS TableName, COUNT(\*) AS RecordCount

FROM Staff;

-- Количество записей в таблице TariffPlans

SELECT 'TariffPlans' AS TableName, COUNT(\*) AS RecordCount

FROM TariffPlans;

-- Количество записей в таблице Subscribers

SELECT 'Subscribers' AS TableName, COUNT(\*) AS RecordCount

FROM Subscribers;

-- Количество записей в таблице Contracts

SELECT 'Contracts' AS TableName, COUNT(\*) AS RecordCount

FROM Contracts;

-- Количество записей в таблице Calls

SELECT 'Calls' AS TableName, COUNT(\*) AS RecordCount

FROM Calls;

-- Количество записей в таблице Messages

SELECT 'Messages' AS TableName, COUNT(\*) AS RecordCount

FROM Messages;

-- Количество записей в таблице InternetUsage

SELECT 'InternetUsage' AS TableName, COUNT(\*) AS RecordCount

FROM InternetUsage;

**ПРИЛОЖЕНИЕ В**

(обязательное)

**Текст создания представлений**

-- Представления

GO

CREATE VIEW EmployeeInfo AS

SELECT

s.StaffID,

s.FullName,

sp.PositionName,

s.Education

FROM

Staff s

JOIN

StaffPosition sp ON s.PositionID = sp.PositionID

GO

CREATE VIEW SubscriberInfo AS

SELECT

s.SubscriberID,

s.FullName AS SubscriberFullName,

s.HomeAddress,

s.PassportData,

c.ContractID,

c.ContractDate,

c.ContractEndDate,

c.PhoneNumber,

tp.TariffName

FROM

Subscribers s

JOIN

Contracts c ON s.SubscriberID = c.SubscriberID

JOIN

TariffPlans tp ON c.TariffPlanID = tp.TariffPlanID;

GO

CREATE VIEW ContractCalls AS

SELECT

c.ContractID,

c.PhoneNumber,

cl.CallID,

cl.CallDate,

cl.CallDuration

FROM

Contracts c

JOIN

Calls cl ON c.ContractID = cl.ContractID;

GO

CREATE VIEW ContractMessages AS

SELECT

c.ContractID,

c.PhoneNumber,

m.MessageID,

m.MessageDate,

m.IsMMS

FROM

Contracts c

JOIN

Messages m ON c.ContractID = m.ContractID;

GO

CREATE VIEW ContractInternetUsage AS

SELECT

c.ContractID,

c.PhoneNumber,

iu.UsageID,

iu.UsageDate,

iu.DataSentMB,

iu.DataReceivedMB

FROM

Contracts c

JOIN

InternetUsage iu ON c.ContractID = iu.ContractID;

GO

**ПРИЛОЖЕНИЕ Г**

(обязательное)

**Текст создания хранимых процедур**

-- Хранимые процедуры

CREATE PROCEDURE InsertCall

@ContractID INT,

@CallDate DATETIME,

@CallDuration INT

AS

BEGIN

INSERT INTO Calls (ContractID, CallDate, CallDuration)

VALUES (@ContractID, @CallDate, @CallDuration);

END;

GO

CREATE PROCEDURE InsertInternetUsage

@ContractID INT,

@UsageDate DATETIME,

@DataSentMB DECIMAL(10, 2),

@DataReceivedMB DECIMAL(10, 2)

AS

BEGIN

INSERT INTO InternetUsage (ContractID, UsageDate, DataSentMB, DataReceivedMB)

VALUES (@ContractID, @UsageDate, @DataSentMB, @DataReceivedMB);

END;

GO

CREATE PROCEDURE InsertMessage

@ContractID INT,

@MessageDate DATETIME,

@IsMMS BIT

AS

BEGIN

INSERT INTO Messages (ContractID, MessageDate, IsMMS)

VALUES (@ContractID, @MessageDate, @IsMMS);

END;