Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

Реализация связей и генерация данных

тема

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | А.К. Погребников |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ18-16б 031831229 |  | В.А. Прекель |
| номер группы, зачетной книжки | подпись, дата | инициалы, фамилия |

Красноярск 2020

# 1 Цель работы

Реализовать связи и сгенерировать данные.

# 2 Общая постановка задачи

В рамках данной практической работы необходимо реализовать связи между таблицами в

соответствии с разработанной моделью данных, а также сгенерировать релевантный набор

тестовых данных для дальнейших манипуляций.

0. Проанализируйте схему данных и установите, какой вид связи подходит в каждом

конкретном случае. Так к примеру для связи с таблицами справочниками больше подходят

односторонние объектные ссылки, в то время как в случае, когда один из объектов

является «контейнером» больше подойдут отношения 1-n (подробнее про Relationship см.

презентацию).

1. Реализуйте оставшиеся связи.

2. Проверьте корректность связывания используя SQL SELECT JOIN запросы.

3. Заполните таблицы справочники.

4. Добавьте к базовым таблицам наследование от класса %Populate.

5. Настройте параметры POPSPEC у полей базовых таблиц, таким образом, чтобы

сгенерированные данные выглядели реалистично (для выбранной предметной области).

Подробнее о настройках Populate можно почитать в документации (%Populate[EN] или же

с 23 страницы презентации).

6. Сгенерируйте не менее 200 строк данных для базовых таблиц.

7. Придумайте и составьте SQL запрос, включающий не менее 4х таблиц и результат которого

может быть полезен для дальнейших практических заданий

# 3 Исходный код

Листинг 1 – MyStore\MyStore.Data\Context.cs

using System;  
  
using Microsoft.EntityFrameworkCore;  
  
using MyStore.Data.Entity;  
using MyStore.Data.Entity.Support;  
  
namespace MyStore.Data  
{  
 public class Context : DbContext  
 {  
 public Context()  
 {  
 }  
  
 public Context(DbContextOptions<Context> options)  
 : base(options)  
 {  
 }  
  
 public DbSet<Cart> Carts { get; set; }  
 public DbSet<CartProduct> CartProducts { get; set; }  
 public DbSet<Customer> Customers { get; set; }  
 public DbSet<Order> Orders { get; set; }  
 public DbSet<Product> Products { get; set; }  
 public DbSet<OrderedProduct> OrderedProducts { get; set; }  
 public DbSet<Answer> SupportAnswers { get; set; }  
 public DbSet<Operator> SupportOperators { get; set; }  
 public DbSet<Question> SupportQuestions { get; set; }  
 public DbSet<Ticket> SupportTickets { get; set; }  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
 optionsBuilder  
 .LogTo(Console.WriteLine)  
 .UseNpgsql("Host=localhost;Database=postgres;Username=postgres;Password=qwerty123");  
 }  
  
 protected override void OnModelCreating(ModelBuilder modelBuilder)  
 {  
 modelBuilder.Entity<Customer>(  
 e =>  
 {  
 e.HasKey(entity => entity.CustomerId);  
 e.Property(entity => entity.FirstName)  
 .HasMaxLength(60)  
 .IsRequired();  
 e.Property(entity => entity.LastName)  
 .HasMaxLength(60);  
 e.Property(entity => entity.Honorific)  
 .HasMaxLength(30)  
 .HasDefaultValue("РЈРІ.");  
 e.Property(entity => entity.Email)  
 .HasMaxLength(60)  
 .IsRequired();  
 e.Property(entity => entity.PasswordHash)  
 .HasMaxLength(32)  
 .IsRequired();  
 e.Property(entity => entity.PasswordSalt)  
 .IsRequired();  
 e.HasOne(entity => entity.CurrentCart)  
 .WithMany(cart => cart.CurrentCustomers)  
 .HasForeignKey(customer => customer.CurrentCartId)  
 .IsRequired(false);  
 });  
  
 modelBuilder.Entity<Product>(  
 e =>  
 {  
 e.HasKey(cart => cart.ProductId);  
 e.Property(cart => cart.Name)  
 .HasMaxLength(100)  
 .IsRequired();  
 e.Property(cart => cart.Description)  
 .IsRequired();  
 e.Property(cart => cart.Price)  
 .HasColumnType("numeric(20, 2)")  
 .IsRequired();  
 });  
  
 modelBuilder.Entity<Cart>(  
 e =>  
 {  
 e.HasKey(cart => cart.CartId);  
 e.HasMany(cart => cart.Products)  
 .WithMany(product => product.Carts)  
 .UsingEntity<CartProduct>(  
 j => j  
 .HasOne(cp => cp.Product)  
 .WithMany(p => p.CartProducts)  
 .HasForeignKey(cp => cp.ProductId),  
 j => j  
 .HasOne(cp => cp.Cart)  
 .WithMany(c => c.CartProducts)  
 .HasForeignKey(cp => cp.CartId),  
 j => { j.HasKey(cp => new {cp.CartId, cp.ProductId}); });  
 e.HasOne(cart => cart.OwnerCustomer)  
 .WithMany(customer => customer.OwnedCarts)  
 .HasForeignKey(cart => cart.OwnerCustomerId);  
 });  
  
 modelBuilder.Entity<Order>(  
 e =>  
 {  
 e.HasKey(order => order.OrderId);  
 e.HasOne(order => order.Customer)  
 .WithMany(customer => customer.Orders)  
 .HasForeignKey(order => order.CustomerId);  
 e.Property(order => order.CreateTimeOffset)  
 .HasDefaultValueSql("current\_timestamp")  
 .IsRequired();  
 });  
  
 modelBuilder.Entity<OrderedProduct>(  
 e =>  
 {  
 e.HasKey(op => new {op.ProductId, op.OrderId});  
 e.HasOne(op => op.Product)  
 .WithMany(p => p.OrderedProducts)  
 .HasForeignKey(op => op.ProductId);  
 e.Property(op => op.OrderedPrice)  
 .HasColumnType("numeric(20, 2)")  
 .IsRequired();  
 e.HasOne(op => op.Order)  
 .WithMany(o => o.OrderedProducts)  
 .HasForeignKey(op => op.OrderId);  
 });  
  
 modelBuilder.Entity<Answer>(  
 b =>  
 {  
 b.HasKey(answer => answer.SupportAnswerId);  
 b.HasOne(answer => answer.SupportOperator)  
 .WithMany(op => op.SupportAnswers)  
 .HasForeignKey(answer => answer.SupportOperatorId);  
 b.HasOne(answer => answer.SupportTicket)  
 .WithMany(ticket => ticket.SupportAnswers)  
 .HasForeignKey(answer => answer.SupportTicketId);  
 b.Property(answer => answer.SendTimestamp)  
 .HasDefaultValueSql("current\_timestamp")  
 .IsRequired();  
 b.Property(answer => answer.Text)  
 .IsRequired();  
 });  
  
 modelBuilder.Entity<Ticket>(  
 b =>  
 {  
 b.HasKey(ticket => ticket.SupportTicketId);  
 b.HasOne(ticket => ticket.SupportOperator)  
 .WithMany(op => op.SupportTickets)  
 .HasForeignKey(ticket => ticket.SupportOperatorId);  
 b.HasOne(ticket => ticket.Customer)  
 .WithMany(customer => customer.SupportTickets)  
 .HasForeignKey(ticket => ticket.CustomerId);  
 b.Property(ticket => ticket.CreateTimestamp)  
 .HasDefaultValueSql("current\_timestamp")  
 .IsRequired();  
 b.HasOne(ticket => ticket.Order)  
 .WithOne(order => order.SupportTicket)  
 .HasForeignKey<Ticket>(ticket => ticket.OrderId)  
 .IsRequired(false);  
 });  
  
 modelBuilder.Entity<Operator>(  
 b =>  
 {  
 b.HasKey(op => op.SupportOperatorId);  
 b.Property(op => op.FirstName)  
 .HasMaxLength(60)  
 .IsRequired();  
 b.Property(op => op.LastName)  
 .HasMaxLength(60)  
 .IsRequired();  
 b.Property(op => op.Email)  
 .HasMaxLength(60)  
 .IsRequired();  
 b.Property(op => op.PasswordHash)  
 .IsRequired();  
 b.Property(op => op.PasswordSalt)  
 .IsRequired();  
 });  
  
 modelBuilder.Entity<Question>(b =>  
 {  
 b.HasKey(question => question.SupportQuestionId);  
 b.HasOne(question => question.SupportTicket)  
 .WithMany(ticket => ticket.SupportQuestions)  
 .HasForeignKey(question => question.SupportTicketId);  
 b.Property(question => question.SendTimestamp)  
 .HasDefaultValueSql("current\_timestamp")  
 .IsRequired();  
 b.Property(question => question.ReadTimestamp);  
 b.Property(question => question.Text)  
 .IsRequired();  
 });  
 }  
 }  
}

Листинг 2 – MyStore\MyStore.Data\Crypto.cs

using System;  
using System.Linq;  
using System.Security.Cryptography;  
using System.Text;  
  
namespace MyStore.Data  
{  
 public static class Crypto  
 {  
 public static int GenerateSaltForPassword()  
 {  
 var rng = new RNGCryptoServiceProvider();  
 var saltBytes = new byte[4];  
 rng.GetNonZeroBytes(saltBytes);  
 return (saltBytes[0] << 24) + (saltBytes[1] << 16) + (saltBytes[2] << 8) + saltBytes[3];  
 }  
  
 public static byte[] ComputePasswordHash(string password, int salt)  
 {  
 var saltBytes = new byte[4];  
 saltBytes[0] = (byte) (salt >> 24);  
 saltBytes[1] = (byte) (salt >> 16);  
 saltBytes[2] = (byte) (salt >> 8);  
 saltBytes[3] = (byte) salt;  
  
 var passwordBytes = Encoding.UTF8.GetBytes(password);  
  
 var preHashed = new byte[saltBytes.Length + passwordBytes.Length];  
 Buffer.BlockCopy(passwordBytes, 0, preHashed, 0, passwordBytes.Length);  
 Buffer.BlockCopy(saltBytes, 0, preHashed, passwordBytes.Length, saltBytes.Length);  
  
 var sha1 = SHA256.Create();  
 return sha1.ComputeHash(preHashed);  
 }  
  
 public static bool IsPasswordValid(string passwordToValidate, int salt, byte[] correctPasswordHash)  
 {  
 var hashedPassword = ComputePasswordHash(passwordToValidate, salt);  
  
 return hashedPassword.SequenceEqual(correctPasswordHash);  
 }  
 }  
}

Листинг 3 – MyStore\MyStore.Data\Entity\Cart.cs

using System.Collections.Generic;  
  
namespace MyStore.Data.Entity  
{  
 public record Cart  
 {  
 public int CartId { get; set; }  
  
 public bool IsPublic { get; set; }  
 public int? OwnerCustomerId { get; set; }  
 public Customer? OwnerCustomer { get; set; }  
  
 public ICollection<Product> Products { get; set; }  
  
 public ICollection<Customer> CurrentCustomers { get; set; }  
  
 public List<CartProduct> CartProducts { get; set; }  
 }  
}

Листинг 4 – MyStore\MyStore.Data\Entity\CartProduct.cs

namespace MyStore.Data.Entity  
{  
 public record CartProduct  
 {  
 public int CartId { get; set; }  
 public Cart Cart { get; set; }  
  
 public int ProductId { get; set; }  
 public Product Product { get; set; }  
 }  
}

Листинг 5 – MyStore\MyStore.Data\Entity\Customer.cs

using System.Collections.Generic;  
using System.ComponentModel.DataAnnotations;  
  
using MyStore.Data.Entity.Support;  
  
namespace MyStore.Data.Entity  
{  
 public record Customer  
 {  
 public int CustomerId { get; set; }  
 public string FirstName { get; set; }  
 public string? LastName { get; set; }  
 public string? Honorific { get; set; }  
  
 [EmailAddress]  
 public string Email { get; set; }  
  
 public byte[] PasswordHash { get; set; }  
 public int PasswordSalt { get; set; }  
  
 public int? CurrentCartId { get; set; }  
 public Cart? CurrentCart { get; set; }  
  
 public ICollection<Order> Orders { get; set; }  
  
 public ICollection<Cart> OwnedCarts { get; set; }  
  
 public ICollection<Ticket> SupportTickets { get; set; }  
 }  
}

Листинг 6 – MyStore\MyStore.Data\Entity\Order.cs

using System;  
using System.Collections.Generic;  
  
using MyStore.Data.Entity.Support;  
  
namespace MyStore.Data.Entity  
{  
 public record Order  
 {  
 public int OrderId { get; set; }  
  
 public int CustomerId { get; set; }  
 public Customer Customer { get; set; }  
  
 public DateTimeOffset CreateTimeOffset { get; set; }  
  
 public ICollection<OrderedProduct> OrderedProducts { get; set; }  
   
 public Ticket? SupportTicket { get; set; }  
 }  
}

Листинг 7 – MyStore\MyStore.Data\Entity\OrderedProduct.cs

namespace MyStore.Data.Entity  
{  
 public record OrderedProduct  
 {  
 public int ProductId { get; set; }  
 public Product Product { get; set; }  
  
 public int OrderId { get; set; }  
 public Order Order { get; set; }  
  
 public decimal OrderedPrice { get; set; }  
 }  
}

Листинг 8 – MyStore\MyStore.Data\Entity\Product.cs

using System.Collections.Generic;  
  
namespace MyStore.Data.Entity  
{  
 public record Product  
 {  
 public int ProductId { get; set; }  
  
 public string Name { get; set; }  
  
 public string Description { get; set; }  
  
 public decimal Price { get; set; }  
  
 public ICollection<Cart> Carts { get; set; }  
  
 public ICollection<OrderedProduct> OrderedProducts { get; set; }  
 public List<CartProduct> CartProducts { get; set; }  
 }  
}

Листинг 9 – MyStore\MyStore.Data\Entity\Support\Answer.cs

using System;  
  
namespace MyStore.Data.Entity.Support  
{  
 public record Answer  
 {  
 public int SupportAnswerId { get; set; }  
 public int SupportTicketId { get; set; }  
 public Ticket SupportTicket { get; set; }  
 public int SupportOperatorId { get; set; }  
 public Operator SupportOperator { get; set; }  
 public DateTimeOffset SendTimestamp { get; set; }  
 public string Text { get; set; }  
 }  
}

Листинг 10 – MyStore\MyStore.Data\Entity\Support\Operator.cs

using System.Collections.Generic;  
  
namespace MyStore.Data.Entity.Support  
{  
 public record Operator  
 {  
 public int SupportOperatorId { get; set; }  
 public string FirstName { get; set; }  
 public string LastName { get; set; }  
 public string Email { get; set; }  
 public byte[] PasswordHash { get; set; }  
 public int PasswordSalt { get; set; }  
  
 public ICollection<Answer> SupportAnswers { get; set; }  
 public ICollection<Ticket> SupportTickets { get; set; }  
 }  
}

Листинг 11 – MyStore\MyStore.Data\Entity\Support\Question.cs

using System;  
  
namespace MyStore.Data.Entity.Support  
{  
 public record Question  
 {  
 public int SupportQuestionId { get; set; }  
 public int SupportTicketId { get; set; }  
 public Ticket SupportTicket { get; set; }  
 public DateTimeOffset SendTimestamp { get; set; }  
 public DateTimeOffset? ReadTimestamp { get; set; }  
 public string Text { get; set; }  
 }  
}

Листинг 12 – MyStore\MyStore.Data\Entity\Support\Ticket.cs

using System;  
using System.Collections.Generic;  
  
namespace MyStore.Data.Entity.Support  
{  
 public record Ticket  
 {  
 public int SupportTicketId { get; set; }  
 public int CustomerId { get; set; }  
 public Customer Customer { get; set; }  
 public int SupportOperatorId { get; set; }  
 public Operator SupportOperator { get; set; }  
   
 public int? OrderId { get; set; }  
   
 public Order? Order { get; set; }  
   
 public DateTimeOffset CreateTimestamp { get; set; }  
  
 public ICollection<Answer> SupportAnswers { get; set; }  
 public ICollection<Question> SupportQuestions { get; set; }  
 }  
}

Листинг 13 – MyStore\MyStore.Data.Populater\Populater.cs

using System;  
using System.Linq;  
using System.Text.RegularExpressions;  
  
using Microsoft.EntityFrameworkCore;  
  
using MyStore.Data.Entity;  
using MyStore.Data.Entity.Support;  
  
using VkNet;  
using VkNet.Enums.Filters;  
using VkNet.Enums.SafetyEnums;  
  
namespace MyStore.Data.Populater  
{  
 public class Populater  
 {  
 public Populater(VkApi api) => Api = api;  
  
 private VkApi Api { get; }  
  
 public void PopulateCustomers(int n)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var cyrRegexp = new Regex("[Рђ-РЇР°-СЏРЃРµ]{3,30}");  
 var names = Api.Users.Get(  
 Enumerable.Range(1, n \* 5).Select(t => (long) r.Next(1, 620\_330\_243)),  
 ProfileFields.FirstName | ProfileFields.LastName,  
 NameCase.Nom  
 ).Select(user => new {user.FirstName, user.LastName})  
 .Where(usernames => cyrRegexp.IsMatch(usernames.FirstName) && cyrRegexp.IsMatch(usernames.LastName))  
 .ToList();  
  
 var emailDomains = new[] {"yandex.ru", "gmail.com", "mail.ru", "hotmail.com"};  
  
 for (var i = 0; i < n; i++)  
 {  
 var salt = Crypto.GenerateSaltForPassword();  
 var customer = new Customer  
 {  
 FirstName = names[r.Next(names.Count - 1)].FirstName,  
 LastName = r.NextDouble() < 0.7 ? names[r.Next(names.Count - 1)].LastName : null,  
 Honorific = r.NextDouble() < 0.1 ? "Р”РѕСЂ." : null,  
 Email =  
 $"{String.Join("", Enumerable.Range(0, 8).Select(t => (char) r.Next('a', 'z')))}{r.Next(100, 999)}@{emailDomains[r.Next(emailDomains.Length - 1)]}",  
 PasswordHash = Crypto.ComputePasswordHash("qwerty", salt),  
 PasswordSalt = salt  
 };  
 context.Customers.Add(customer);  
 }  
  
 context.SaveChanges();  
 }  
  
 public void PopulateProducts(int n)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 for (var i = 0; i < n; i++)  
 {  
 var name =  
 $"{(char) r.Next('Рђ', 'РЇ')}{String.Join("", Enumerable.Range(0, 8).Select(t => (char) r.Next('Р°', 'СЏ')))}";  
 var product = new Product  
 {  
 Name = name,  
 Description = $"РћРїРёСЃР°РЅРёРµ С‚РѕРІР°СЂР° {name}",  
 Price = r.Next(10, 10000) / (decimal) 10  
 };  
 context.Products.Add(product);  
 }  
  
 context.SaveChanges();  
 }  
  
 public void PopulateCarts(int n, int m, int k)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var customers = context.Customers.ToList();  
 var products = context.Products.ToList();  
  
 var customersCount = context.Customers.Count();  
 var productsCount = context.Products.Count();  
  
  
 for (var i = 0; i < n; i++)  
 {  
 var isPublic = r.NextDouble() > 0.7;  
 var cart = new Cart  
 {  
 IsPublic = isPublic,  
 OwnerCustomer = r.NextDouble() > 0.7 || !isPublic ? customers[r.Next(customersCount - 1)] : null  
 };  
 for (var j = 0; j < r.Next(m); j++)  
 {  
 context.CartProducts.Add(  
 new CartProduct  
 {  
 Cart = cart,  
 Product = products[r.Next(productsCount - 1)]  
 });  
 }  
  
 if (cart.IsPublic)  
 {  
 for (var j = 0; j < r.Next(k); j++)  
 {  
 customers[r.Next(customersCount - 1)].CurrentCart = cart;  
 }  
 }  
 else if (r.NextDouble() > 0.7)  
 {  
 cart.OwnerCustomer.CurrentCart = cart;  
 }  
  
 context.Carts.Add(cart);  
 }  
  
 context.SaveChanges();  
 }  
  
 public void PopulateOrdersOrderedProducts(int n, int m)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var customers = context.Customers.ToList();  
 var products = context.Products.ToList();  
  
 for (var i = 0; i < n; i++)  
 {  
 var order = new Order  
 {  
 Customer = customers[r.Next(customers.Count - 1)]  
 };  
 order.OrderedProducts = Enumerable.Range(0, m)  
 .Select(\_ => r.Next(products.Count - 1))  
 .Distinct()  
 .Select(ind => products[ind])  
 .Select(product => new OrderedProduct  
 {  
 Product = product,  
 Order = order,  
 OrderedPrice = r.NextDouble() > 0.8 ? product.Price \* 0.8m : product.Price  
 })  
 .ToList();  
  
 context.Orders.Add(order);  
 }  
  
 context.SaveChanges();  
 }  
  
 public void PopulateSupportOperators(int n)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var cyrRegexp = new Regex("[Рђ-РЇР°-СЏРЃРµ]{3,30}");  
 var names = Api.Users.Get(  
 Enumerable.Range(1, n \* 5).Select(t => (long) r.Next(1, 620\_330\_243)),  
 ProfileFields.FirstName | ProfileFields.LastName,  
 NameCase.Nom  
 ).Select(user => new {user.FirstName, user.LastName})  
 .Where(usernames => cyrRegexp.IsMatch(usernames.FirstName) && cyrRegexp.IsMatch(usernames.LastName))  
 .ToList();  
  
 var emailDomains = new[] {"yandex.ru", "gmail.com", "mail.ru", "hotmail.com"};  
  
 for (var i = 0; i < n; i++)  
 {  
 var salt = Crypto.GenerateSaltForPassword();  
 var op = new Operator  
 {  
 FirstName = names[r.Next(names.Count - 1)].FirstName,  
 LastName = names[r.Next(names.Count - 1)].LastName,  
 Email =  
 $"{String.Join("", Enumerable.Range(0, 8).Select(t => (char) r.Next('a', 'z')))}{r.Next(100, 999)}@{emailDomains[r.Next(emailDomains.Length - 1)]}",  
 PasswordHash = Crypto.ComputePasswordHash("qwerty", salt),  
 PasswordSalt = salt  
 };  
 context.SupportOperators.Add(op);  
 }  
  
 context.SaveChanges();  
 }  
  
 public void PopulateSupportTickets(int n)  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var customers = context.Customers.ToList();  
 var operators = context.SupportOperators.ToList();  
 var orders = context.Orders.ToList();  
  
 context.SupportTickets.AddRange(  
 Enumerable.Range(0, n)  
 .Select(\_ => new Ticket  
 {  
 Customer = customers[r.Next(customers.Count - 1)],  
 SupportOperator = operators[r.Next(operators.Count - 1)],  
 Order = r.NextDouble() < 0.4 ? orders[r.Next(orders.Count - 1)] : null  
 })  
 );  
  
 context.SaveChanges();  
 }  
  
 public void PopulateAnswersQuestions()  
 {  
 using var context = new Context();  
  
 var r = new Random();  
  
 var tickets = context.SupportTickets.ToList();  
 var ops = context.SupportOperators.ToList();  
  
 foreach (var ticket in tickets)  
 {  
 var randomstring = String.Join("", Enumerable.Range(0, 8).Select(t => (char) r.Next('Р°', 'СЏ')));  
 var question = new Question  
 {  
 SupportTicket = ticket,  
 ReadTimestamp = DateTimeOffset.Now + TimeSpan.FromSeconds(10),  
 Text = $"Р’РѕРїСЂРѕСЃ {randomstring}"  
 };  
 context.SupportQuestions.Add(question);  
 var answer = new Answer  
 {  
 SupportOperator = r.NextDouble() < 0.9 ? ticket.SupportOperator : ops[r.Next(ops.Count - 1)],  
 SupportTicket = ticket,  
 SendTimestamp = DateTimeOffset.Now + TimeSpan.FromSeconds(15),  
 Text = $"РћС‚РІРµС‚ {randomstring}"  
 };  
 context.SupportAnswers.Add(answer);  
 if (r.NextDouble() > 0.5)  
 {  
 var isRead = r.NextDouble() > 0.6;  
 var q = new Question  
 {  
 SupportTicket = ticket,  
 SendTimestamp = DateTimeOffset.Now + TimeSpan.FromSeconds(20),  
 Text = $"Р”РѕРїРѕР»РЅРёС‚РµР»СЊРЅС‹Р№ РІРѕРїСЂРѕСЃ {randomstring}"  
 };  
 if (isRead)  
 {  
 q.ReadTimestamp = DateTimeOffset.Now + TimeSpan.FromSeconds(30);  
 }  
  
 context.SupportQuestions.Add(q);  
  
 if (r.NextDouble() > 0.5 && isRead)  
 {  
 var ans2 = new Answer  
 {  
 SupportOperator =  
 r.NextDouble() < 0.9 ? ticket.SupportOperator : ops[r.Next(ops.Count - 1)],  
 SupportTicket = ticket,  
 SendTimestamp = DateTimeOffset.Now + TimeSpan.FromSeconds(35),  
 Text = $"РћС‚РІРµС‚ РЅР° РґРѕРїРѕР»РЅРёС‚РµР»СЊРЅС‹Р№ РІРѕРїСЂРѕСЃ {randomstring}"  
 };  
 context.SupportAnswers.Add(ans2);  
 }  
 }  
 }  
  
 context.SaveChanges();  
 }  
 }  
}

Листинг 14 – MyStore\MyStore.Data.Populater\Program.cs

п»їusing VkNet;  
using VkNet.Enums;  
using VkNet.Model;  
  
namespace MyStore.Data.Populater  
{  
 internal static class Program  
 {  
 private static void Main(string[] args)  
 {  
 using (var context = new Context())  
 {  
 context.Database.EnsureCreated();  
 }  
  
 var api = new VkApi();  
 api.Authorize(new ApiAuthParams  
 {  
 AccessToken = "1bb9ca221bb9ca221bb9ca22ad1bdfa76e11bb91bb9ca22441bbfc7d2cfe35c00c4a071"  
 });  
 api.SetLanguage(Language.Ru);  
  
 var populater = new Populater(api);  
 var c = 4000;  
 for (var i = 72000; i < 500000; i += c)  
 {  
 populater.PopulateCustomers(c);  
 }  
   
 populater.PopulateProducts(500000);  
 populater.PopulateCarts(500000, 2, 3);  
 populater.PopulateOrdersOrderedProducts(600000, 4);  
  
 for (var i = 0; i < 125000; i += c)  
 {  
 populater.PopulateSupportOperators(c);  
 }  
   
 populater.PopulateSupportTickets(150000);  
 populater.PopulateAnswersQuestions();  
 }  
 }  
}