МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 3

з дисципліни “ Бази даних”

Виконав

Студент II курсу

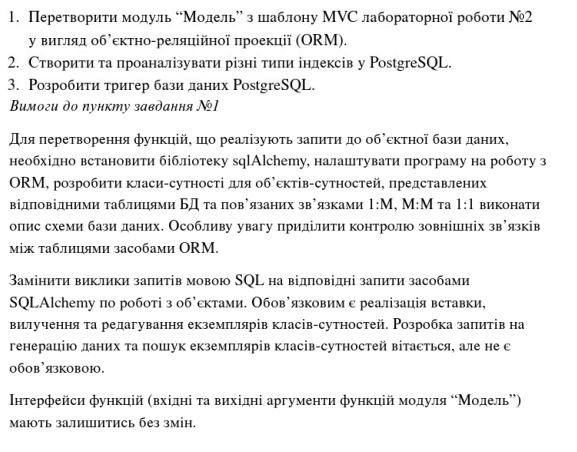
Групи КП-91

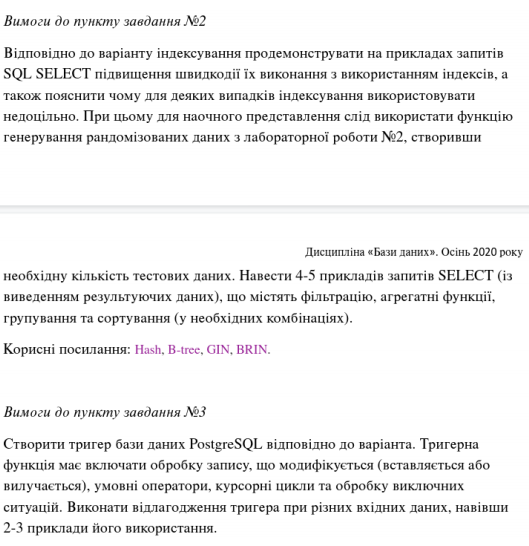
Маховой Олександр Вікторович

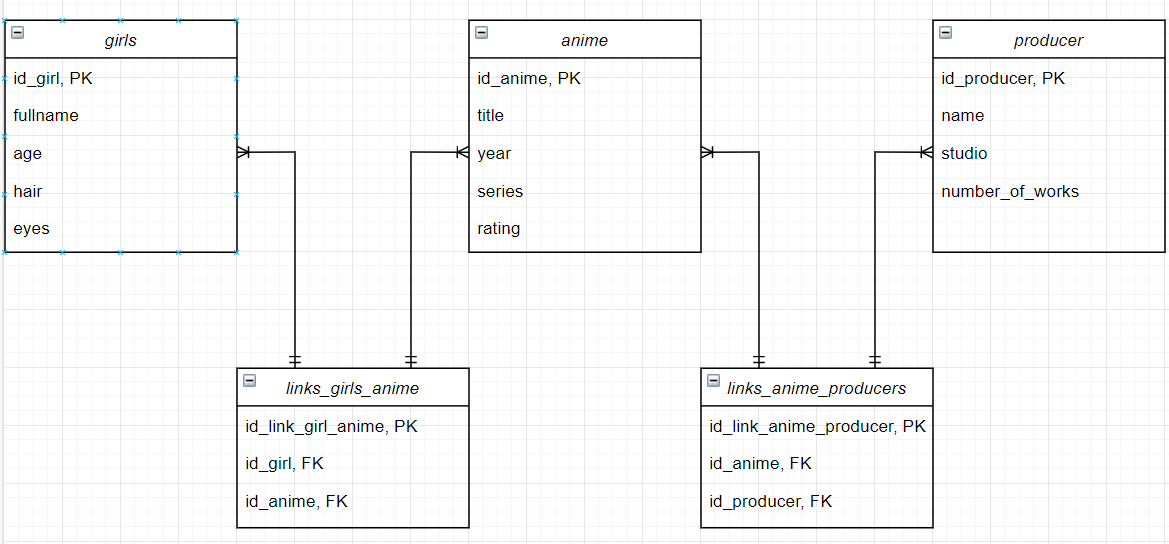
Київ 2020

Варіант №18



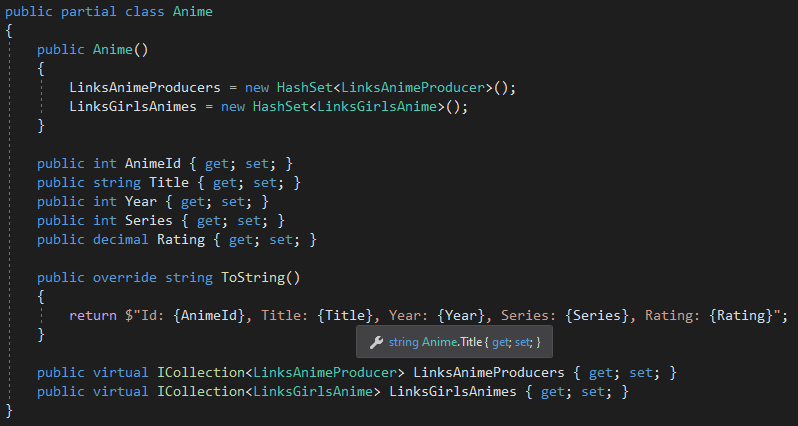
**

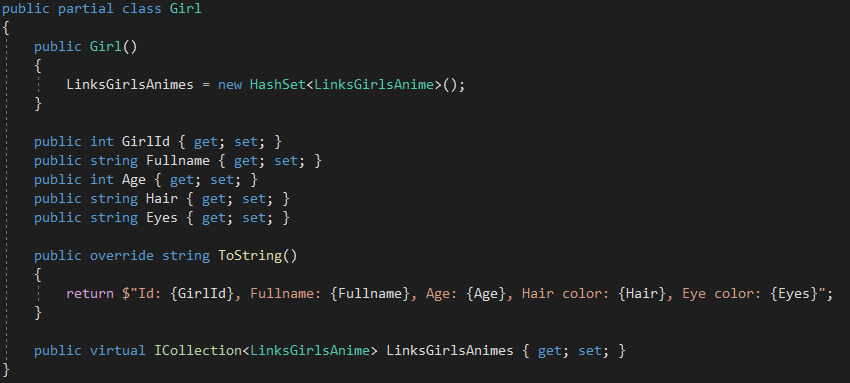
**

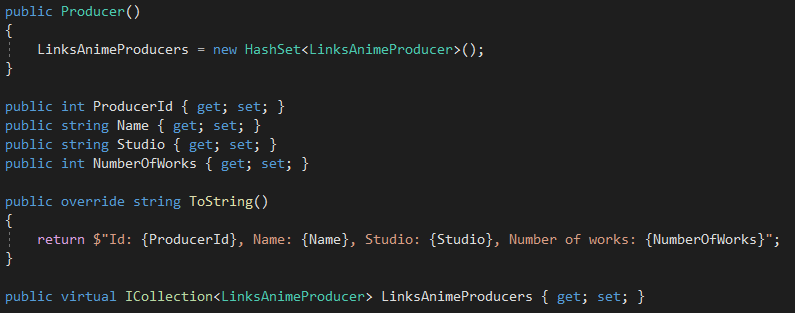


**Завдання 1**

Класи ORM

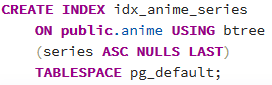






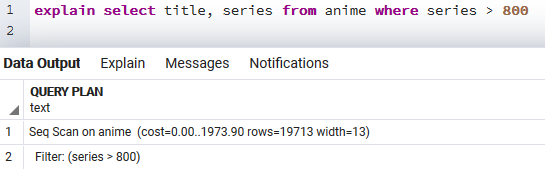
|  |
| --- |
| **DatabaseContext** |
| public partial class DatabaseContext : DbContext  {  public DatabaseContext()  {  }  public DatabaseContext(DbContextOptions<DatabaseContext> options)  : base(options)  {  }  public virtual DbSet<Anime> Animes { get; set; }  public virtual DbSet<Girl> Girls { get; set; }  public virtual DbSet<LinksAnimeProducer> LinksAnimeProducers { get; set; }  public virtual DbSet<LinksGirlsAnime> LinksGirlsAnimes { get; set; }  public virtual DbSet<Producer> Producers { get; set; }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  if (!optionsBuilder.IsConfigured)  {  #warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.  optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=anime\_girls;Username=postgres;Password=asd");  }  }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  modelBuilder.Entity<Anime>(entity =>  {  entity.ToTable("anime");  entity.Property(e => e.AnimeId)  .HasColumnName("anime\_id")  .HasDefaultValueSql("nextval('anime\_id\_anime\_seq'::regclass)");  entity.Property(e => e.Rating)  .HasPrecision(4, 2)  .HasColumnName("rating");  entity.Property(e => e.Series).HasColumnName("series");  entity.Property(e => e.Title)  .IsRequired()  .HasMaxLength(100)  .HasColumnName("title");  entity.Property(e => e.Year).HasColumnName("year");  });  modelBuilder.Entity<Girl>(entity =>  {  entity.ToTable("girls");  entity.Property(e => e.GirlId)  .HasColumnName("girl\_id")  .HasDefaultValueSql("nextval('girls\_id\_girl\_seq'::regclass)");  entity.Property(e => e.Age).HasColumnName("age");  entity.Property(e => e.Eyes)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("eyes");  entity.Property(e => e.Fullname)  .IsRequired()  .HasMaxLength(50)  .HasColumnName("fullname");  entity.Property(e => e.Hair)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("hair");  });  modelBuilder.Entity<LinksAnimeProducer>(entity =>  {  entity.ToTable("links\_anime\_producers");  entity.Property(e => e.Id)  .HasColumnName("id")  .HasDefaultValueSql("nextval('links2\_id\_link1\_seq'::regclass)");  entity.Property(e => e.AnimeId).HasColumnName("anime\_id");  entity.Property(e => e.ProducerId).HasColumnName("producer\_id");  entity.HasOne(d => d.Anime)  .WithMany(p => p.LinksAnimeProducers)  .HasForeignKey(d => d.AnimeId)  .HasConstraintName("anime\_fkey");  entity.HasOne(d => d.Producer)  .WithMany(p => p.LinksAnimeProducers)  .HasForeignKey(d => d.ProducerId)  .HasConstraintName("producer\_fkey");  });  modelBuilder.Entity<LinksGirlsAnime>(entity =>  {  entity.ToTable("links\_girls\_anime");  entity.Property(e => e.Id)  .HasColumnName("id")  .HasDefaultValueSql("nextval('links1\_id\_link1\_seq'::regclass)");  entity.Property(e => e.AnimeId).HasColumnName("anime\_id");  entity.Property(e => e.GirlId).HasColumnName("girl\_id");  entity.HasOne(d => d.Anime)  .WithMany(p => p.LinksGirlsAnimes)  .HasForeignKey(d => d.AnimeId)  .HasConstraintName("anime\_fkey");  entity.HasOne(d => d.Girl)  .WithMany(p => p.LinksGirlsAnimes)  .HasForeignKey(d => d.GirlId)  .HasConstraintName("girl\_fkey");  });  modelBuilder.Entity<Producer>(entity =>  {  entity.ToTable("producers");  entity.Property(e => e.ProducerId)  .HasColumnName("producer\_id")  .HasDefaultValueSql("nextval('producers\_id\_producer\_seq'::regclass)");  entity.Property(e => e.Name)  .IsRequired()  .HasMaxLength(50)  .HasColumnName("name");  entity.Property(e => e.NumberOfWorks).HasColumnName("number\_of\_works");  entity.Property(e => e.Studio)  .IsRequired()  .HasMaxLength(50)  .HasColumnName("studio");  });  OnModelCreatingPartial(modelBuilder);  }  partial void OnModelCreatingPartial(ModelBuilder modelBuilder);  } |

**Завдання 2**

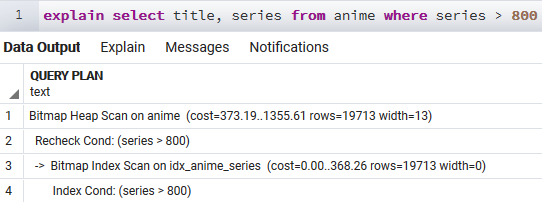
****

****

Before using BTree:

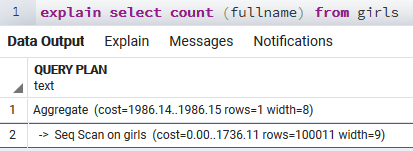


After using BTree:

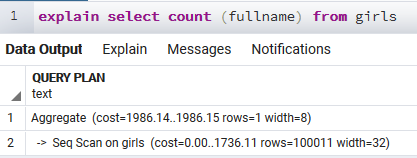


Для ефективного використання GIN у згенерованій таблиці не було типів даних, для яких це було б доцільно, тож час виконання запитів майже ідентичний.

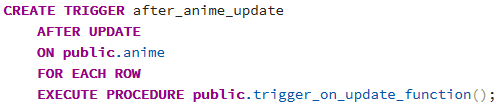
Before using GIN:

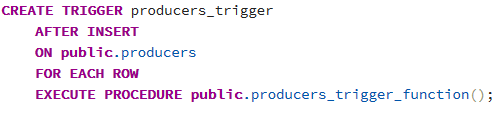


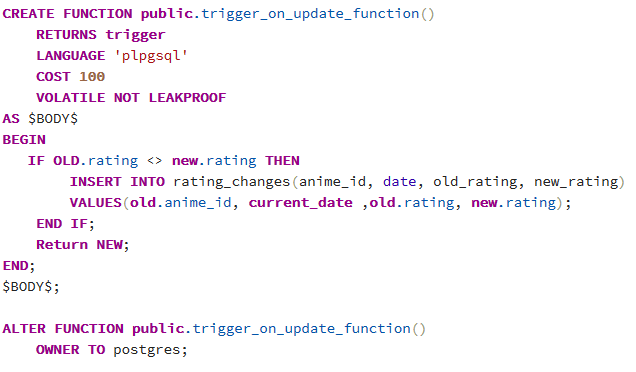
After using GIN:



**Завдання 3**

****



****



