**МІНІСТЕРСТВО ОСВІТИ 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»

**на тему:** **Засоби оптимізації роботи СУБД PostgreSQL**

**Виконав**

студент II курсу

групи КП-91

Бабак Артем Андрiйович

**Перевірив**

викладач

Петрашенко Андрій Васильович

Київ – 2020

**Завдання:**

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.

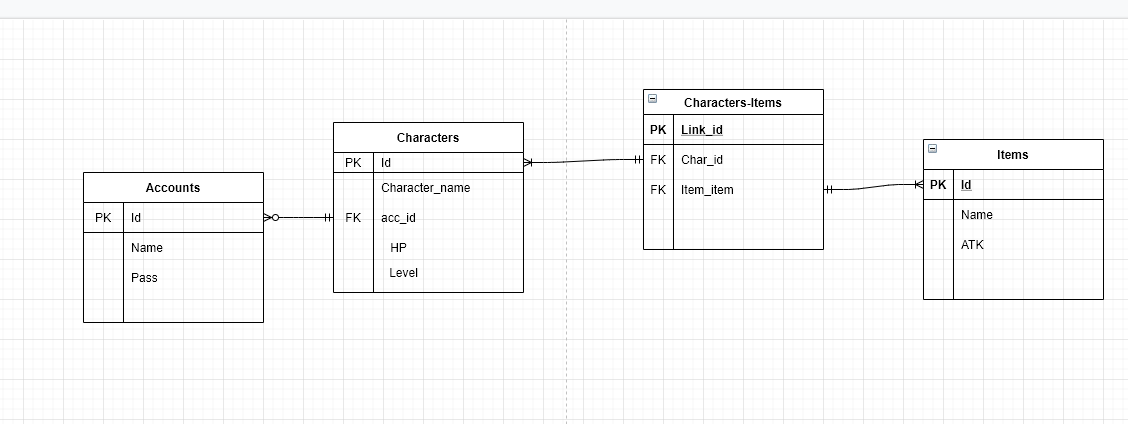
**Репозиторій:**

<https://github.com/tiwatit/Database/tree/main/DB_Lab3/DB_3>   
Варiант



**Пункт №1**

Було використано бібліотеку Entity Framework Core - об'єктно-орієнтовану технологію від компанії Microsoft для доступу до даних, яку можно вважати аналогічною до sqlAlchemy, так як мовою програмування була обрана C#.



Автоматично було створено клас контексту даних:

|  |
| --- |
| postgresContext.cs |
| using System;  using Microsoft.EntityFrameworkCore;  using Microsoft.EntityFrameworkCore.Metadata;  #nullable disable  namespace DB\_3  {  public partial class postgresContext : DbContext  {  public postgresContext()  {  }  public postgresContext(DbContextOptions<postgresContext> options)  : base(options)  {  }  public virtual DbSet<Account> Accounts { get; set; }  public virtual DbSet<Character> Characters { get; set; }  public virtual DbSet<CharactersItem> CharactersItems { get; set; }  public virtual DbSet<Item> Items { 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;Database=postgres;Username=postgres;Password=1111");  }  }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  modelBuilder.HasPostgresExtension("adminpack")  .HasAnnotation("Relational:Collation", "Ukrainian\_Ukraine.1251");  modelBuilder.Entity<Account>(entity =>  {  entity.ToTable("accounts");  entity.Property(e => e.Id)  .HasColumnName("id")  .HasIdentityOptions(2L, null, null, null, null, null);  entity.Property(e => e.Name)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("name");  entity.Property(e => e.Pword)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("pword");  });  modelBuilder.Entity<Character>(entity =>  {  entity.Property(e => e.Id)  .HasColumnName("id")  .HasIdentityOptions(2L, null, null, null, null, null);  entity.Property(e => e.AccId).HasColumnName("acc\_id");  entity.Property(e => e.CharacterName)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("Character\_name");  entity.Property(e => e.Hp).HasColumnName("HP");  entity.HasOne(d => d.Acc)  .WithMany(p => p.Characters)  .HasForeignKey(d => d.AccId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("fk\_char");  });  modelBuilder.Entity<CharactersItem>(entity =>  {  entity.HasKey(e => e.LinkId)  .HasName("Characters-Items\_pkey");  entity.ToTable("Characters-Items");  entity.Property(e => e.LinkId)  .ValueGeneratedOnAdd()  .HasColumnName("link\_id")  .HasIdentityOptions(2L, null, null, null, null, null);  entity.Property(e => e.CharId).HasColumnName("char\_id");  entity.Property(e => e.ItemId).HasColumnName("item\_id");  entity.HasOne(d => d.Char)  .WithMany(p => p.CharactersItems)  .HasForeignKey(d => d.CharId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("fk\_char");  entity.HasOne(d => d.Link)  .WithOne(p => p.CharactersItem)  .HasForeignKey<CharactersItem>(d => d.LinkId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("fk\_item");  });  modelBuilder.Entity<Item>(entity =>  {  entity.Property(e => e.Id)  .HasColumnName("id")  .HasIdentityOptions(2L, null, null, null, null, null);  entity.Property(e => e.Atk).HasColumnName("ATK");  entity.Property(e => e.Name)  .IsRequired()  .HasMaxLength(30)  .HasColumnName("name");  });  OnModelCreatingPartial(modelBuilder);  }  partial void OnModelCreatingPartial(ModelBuilder modelBuilder);  }  } |

А також автоматично були створені класи сутностей.

|  |
| --- |
| Characters |
| using System;  using System.Collections.Generic;  #nullable disable  namespace DB\_3  {  public partial class Character  {  public Character()  {  CharactersItems = new HashSet<CharactersItem>();  }  public int Id { get; set; }  public string CharacterName { get; set; }  public int Hp { get; set; }  public int Level { get; set; }  public int AccId { get; set; }  public virtual Account Acc { get; set; }  public virtual ICollection<CharactersItem> CharactersItems { get; set; }  }  } |

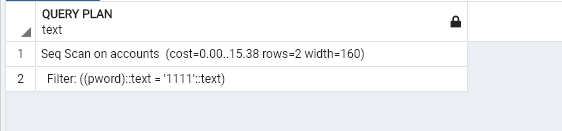
|  |
| --- |
| Items |
| using System;  using System.Collections.Generic;  #nullable disable  namespace DB\_3  {  public partial class Item  {  public int Id { get; set; }  public string Name { get; set; }  public int Atk { get; set; }  public virtual CharactersItem CharactersItem { get; set; }  }  } |

|  |
| --- |
| Accounts |
| using System;  using System.Collections.Generic;  #nullable disable  namespace DB\_3  {  public partial class Account  {  public Account()  {  Characters = new HashSet<Character>();  }  public int Id { get; set; }  public string Name { get; set; }  public string Pword { get; set; }  public virtual ICollection<Character> Characters { get; set; }  }  } |

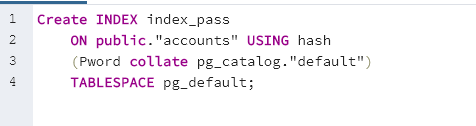
|  |
| --- |
| Characters-Items |
| using System;  using System.Collections.Generic;  #nullable disable  namespace DB\_3  {  public partial class CharactersItem  {  public int LinkId { get; set; }  public int CharId { get; set; }  public int ItemId { get; set; }  public virtual Character Char { get; set; }  public virtual Item Link { get; set; }  }  } |

**Пункт №2**

* **HASH**До індексації:

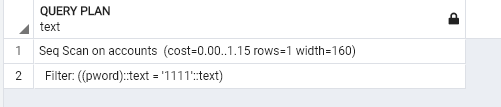


Iндекс:



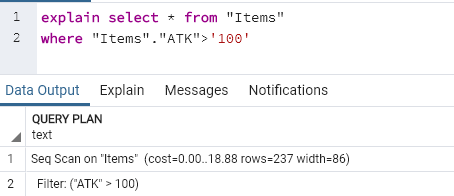
Після індексації:

Прискорення відбувається, бо використовується оператор “=”, саме цю операцію прискорює індексація типу HASH, по порівнюється не саме поле, а значення хєшу цього поля.

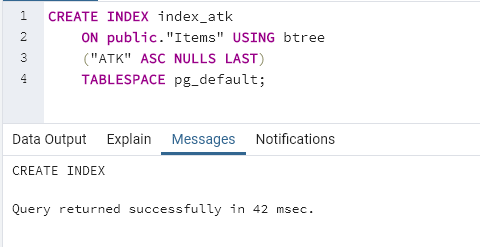


* **Btree**

До індексації:

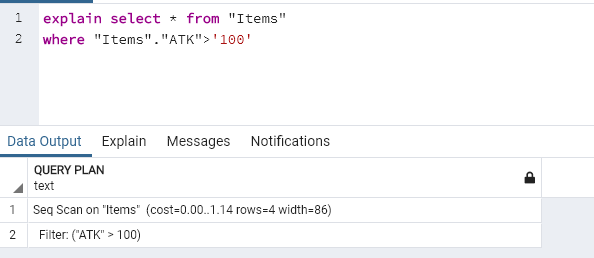


Iндекс:



Після індексації:

Універсальний тип індексації. Прискорення відбувається через використання збалансованого дерева, що дозволяє зменшити час виконання запиту.



**Завдання 3**

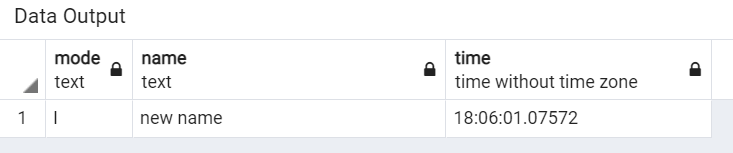
**Тригер**

|  |
| --- |
| Trigger |
| create or replace function tfunc\_flm() returns trigger as $$  begin  if new.name = '' then  raise exception 'name cannot be null';  end if;  if not (new.duration like '%:%') then  raise exception '% incorrect duration input', new.duration;  end if;  if(TG\_OP = 'UPDATE')then  insert into cin\_audit values('U', new.name, now());  return new;  elsif(TG\_OP = 'INSERT') then  insert into cin\_audit values('I', new.name, now()); return new;  end if;  end;  $$ language plpgsql;  create trigger film\_trigger  after insert or update on films for each row  execute procedure tfunc\_flm(); |

**Приклад**

Для INSERT





Для UPDATE

