**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: Система керування базою даних комп’ютерних ігр

|  |  |  |
| --- | --- | --- |
| **Студент групи** КП-91 | **Бабак А.А**  (ПІБ) | \_\_\_\_\_\_\_\_\_  (підпис) |
| **Викладач**  **к.т.н, доцент кафедри СПіСКС** | **Петрашенко А.В.** | \_\_\_\_\_\_\_\_\_  (підпис) |

Захищено з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ – 2020

# 

**Змiст**

[**Анотацiя** 2](#_Toc61793548)

[**Аналіз інструментарію для виконання курсової роботи** 4](#_Toc61793549)

[**Структура БД** 5](#_Toc61793550)

[**Аналіз засобів резервування/відновлення** 8](#_Toc61793551)

[**Засоби підвищення швидкодії** 9](#_Toc61793552)

[**Аналіз експорту** 10](#_Toc61793553)

[**Опис результатів аналізу** 11](#_Toc61793554)

[**Висновки** 12](#_Toc61793555)

[**Література** 13](#_Toc61793556)

[**Додатки** 14](#_Toc61793557)

# **Анотацiя**

Курсова робота є базою даних для зберiгання iнформацiї про ігри, їх рейтинги та гравців, з консольним інтерфейсом для керування нею. Також у консольному інтерфейсі передбачені функції генерації, аналізу та backup даних.

У результаті виконання курсової роботи я використав всі навички та вміння набуті за курс Баз Даних, щоб розробити програмне забезпечення для реляційних баз даних. Оволодів основами використання СУБД, а також інструментальними засобами підтримки розробки додатків для подібних баз даних.

**Вступ**

Кожного року ігрова індустрія розвивається все більше. Багато людей проникаються культурою геймінгу. На сьогодняшній день вже вишла дуже велика кількість комп’ютерних ігр і людям, що тільки зацікавились цією культурою дуже складно увійти до неї. Саме тому я створив додаток для відслідковування ігр, їх рейтингів та кількості людей, що в них грають.

У додатку можна додавати дані про:

* Ігри
* Рейтинги ігр
* Гравців
* Зв’язки між іграми та рейтингами

Також є можливість аналізу даних для відслідковування самих популярних жанрів ігр та розробників, що зробили найбільшу кількість ігр.

# **Аналіз інструментарію для виконання курсової роботи**

СКБД(PostgreSQL):

Переваги:

* Open-source
* Підтримує ACID
* Велика кількість додатків та доповнень
* Існує можливість розширення функціоналу за рахунок власних функцій
* Використовуються збережені процедури

Недоліки:

* Поступається в швидкодії деяким іншим СКБД
* Складна реалізація

Microsoft Entity Framework version 5.0.0-фреймворк для роботи з базами даних. Дозволяє використовувати ORM та взаємодіяти з об’єктами класів, а не читати дані запитами. Додаток економить велику кількість часу, бо сам генерує об’єкти класів та велику кількість коду для опису функціоналу. Але для впровадження важкої логіки нам все ще потрібно використовувати RAW запити.

# **Структура БД**

Таблиці баз даних:

* Games – містить інформацію про ігри
* Players – містить інформацію про кількість гравців
* Ratings – містить інформацію про рейтинги ігр.
* Game-Rating – таблиця лінків для Games і Ratings

**Опис програмного забезпечення**

Загальна структура:

* Засоби фільтрації та валідації
* Засоби генерації
* Засоби резервування/відновлення
* Засоби аналізу даних
* Засоби візуалізації даних

Опис модулів

1. Model – модуль для взаємодії з базою даних(запити)
2. View – модуль для взаємодії з користувачем(результати запитів)
3. Controller – об’єднує model та view. Слугує як меню.

**Опис основних алгоритмів роботи**

Створити датасет з різних csv таблиць. Створити базовий функціонал програми, сповіщення про результати, методи валідації, методи рандомного створення даних.

# **Аналіз засобів резервування/відновлення**

Користувач може здійснити резервування даних прямо під час роботи програми в будь-який момент. Для цього я використовую утиліту pg\_dump. Вона зручна у використанні та не перешкоджає доступ до бази даних іншим користувачам.

# **Засоби підвищення швидкодії**

Були використані індекси(додатки). Я використовував btree індекс, бо він універсальний. Я створив індекс для розробликів ігр.

1. Explain analyze select developer from games where name=’nintendo’
2. Explain analyze select developer from games where name=’ubisoft’

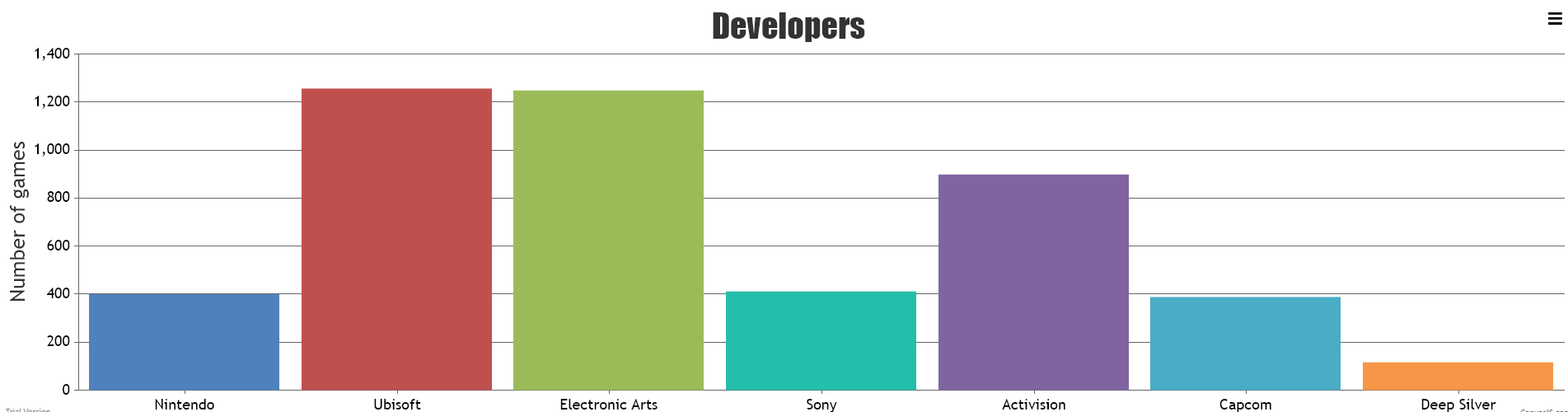
# **Аналіз експорту**

Користувач може експортувати БД у вигляді csv таблиці у певну директорію.

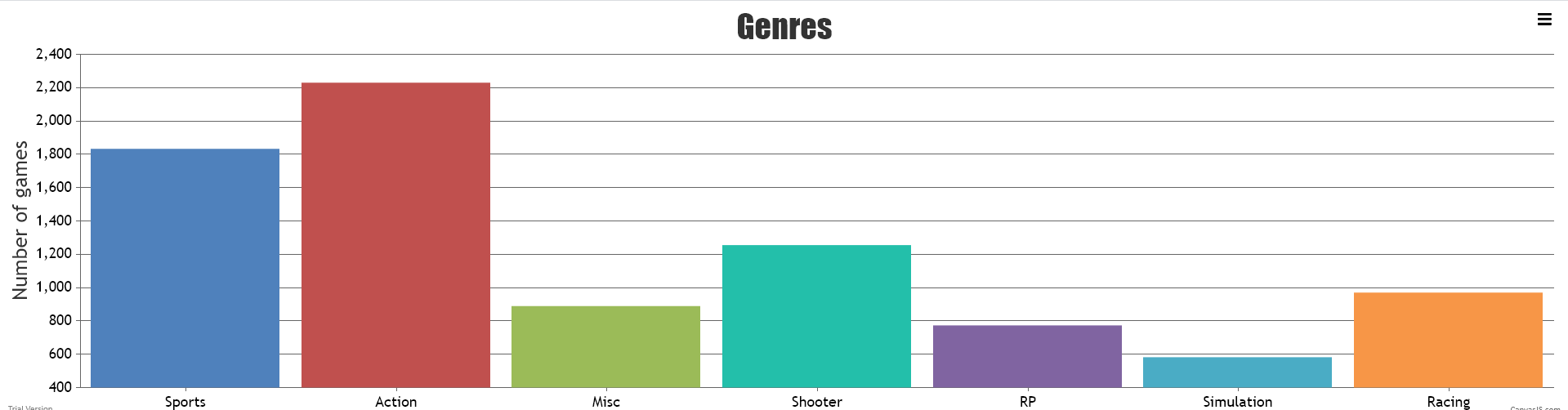
# **Опис результатів аналізу**

Дана програма аналізує:

1. Топ 7 самих відомих розробників з найбільшою кількістю ігр.
2. Топ 7 самих відомих жанрів ігр.
3. Топ 7 самих відомих розробників з найбільшою кількістю ігр:



1. Топ 7 самих відомих жанрів ігр



# **Висновки**

у результаті роботи я проаналізував та познайомився з багатьма фреймворками та сервісами. На основі цього аналізу прийняв максимально ефективні рішення для розробки програми.

Спочатку були реалізовані базові функції для взаємодії та ручного створення та видалення даних.

Були створені засоби генерації даних.

Були реалізовані засоби фільтрації та валідації даних.

Були реалізовані засоби аналізу даних на основі яких складаються діаграми для наглядної демонстрації статистики.

Були реалізовані засоби резервування та відновлення даних.

Були реалізовані засобі підвищення швидкодії запитів, а саме індекс btree.

У результаті роботи я навчився розробляти програмне забезпечення для роботи з СКБД. Були здобуті навички оформлення даних у формі проектної документації.

Також були здобуді навички володіння СКБД та засобами розробки додатків для подібних баз даних.

# 

# **Література**

1. Індекс btree: https://ru.haru-atari.com/blog/6-indexes-in-postgresql-full-manual

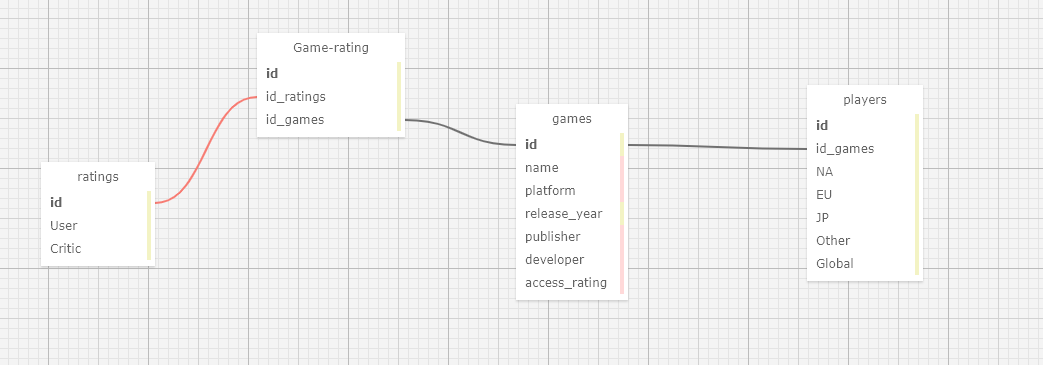
2. Advantages of PostgreSQL: https://bitnine.net/blog-postgresql/advantages-of-postgresql/?ckattempt=2

3. pg\_dump documentation: https://www.postgresql.org/docs/9.3/app-pgdump.html

4. Entity framework: docs.microsoft.com/ru-ru/ef/

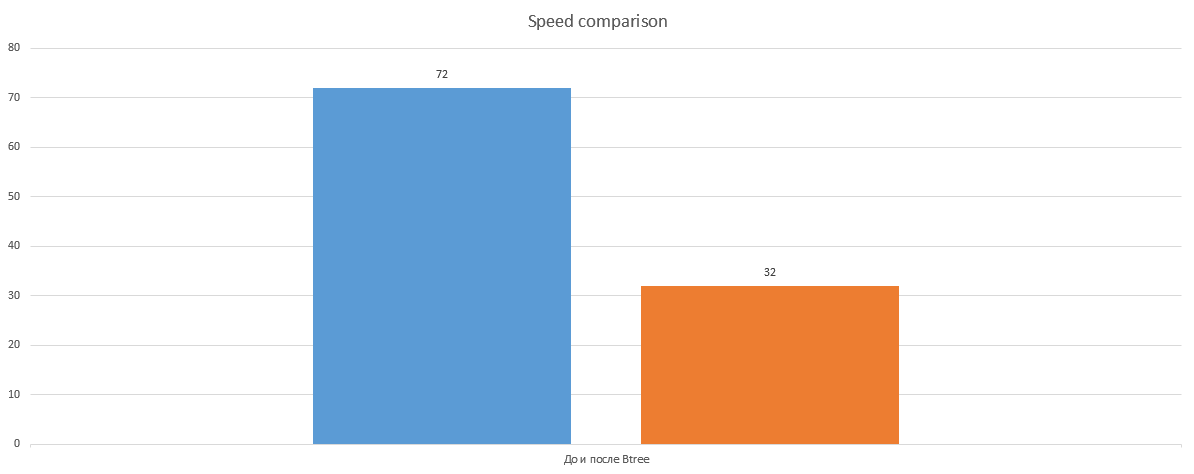
# **Додатки**

Структура БД:



Структура програмних засобів:

Діаграма швидкодії



Функція генерації ігр:

|  |
| --- |
| game\_generation |
| using var cmd = new NpgsqlCommand("INSERT INTO \"games\" (\"name\", \"platform\", \"release\_year\" ,\"genre\", \"publisher\",\"developer\",\"access\_rating\") SELECT chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) , chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) , trunc(random() \* 500 + 20),chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) ,chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) ,chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) ,chr(trunc(65 + random()\*25)::int) || chr(trunc(97 + random()\*25)::int) FROM generate\_series(1, @num)", db);  cmd.Parameters.AddWithValue("@num", num);  cmd.ExecuteNonQuery(); |

Генерування файлів для статистики:

|  |
| --- |
| games\_export\_csv |
| using (Course\_workContext db = new Course\_workContext())  {  List<Game> games = db.Games.ToList();  Collection a = new Collection();  foreach (Game b in games)  {  collection\_element c = new collection\_element(b.Genre);  a.add(c);  }  for (int i = 0; i < a.genres.Count() - 1; i++)  {  for (int j = 0; j < a.genres.Count() - 1; j++)  {  if (a.genres[j].k < a.genres[i].k)  {  collection\_element c = new collection\_element(a.genres[i].key);  int d = a.genres[i].k;  a.genres[i] = a.genres[j];  a.genres[j] = c;  a.genres[j].k = d;  }  }  }  var sw = new StreamWriter("D:\\DB\\course\_work\\genre\\genre.csv", false);  for (int i = 0; i < 5; i++)  {  var first = a.genres[i].key;  var second = a.genres[i].k;  var line = string.Format("{0},{1}", first, second);  sw.WriteLine(line);  sw.Flush();  }  } |

|  |
| --- |
| dev\_export\_csv |
| using (Course\_workContext db = new Course\_workContext())  {  List<Game> games = db.Games.ToList();  Collection a = new Collection();  foreach (Game b in games)  {  collection\_element c = new collection\_element(b.Developer);  a.add(c);  }  for (int i=0;i<a.genres.Count()-1;i++)  {  for (int j = 0; j < a.genres.Count() - 1; j++)  {  if (a.genres[j].k<a.genres[i].k)  {  collection\_element c = new collection\_element(a.genres[i].key);  int d = a.genres[i].k;  a.genres[i] = a.genres[j];  a.genres[j] = c;  a.genres[j].k = d;  }  }  }  var sw = new StreamWriter("D:\\DB\\course\_work\\developer\\developer.csv", false);  for (int i = 0; i < 10; i++)  {  var first = a.genres[i].key;  var second = a.genres[i].k;  var line = string.Format("{0},{1}", first, second);  sw.WriteLine(line);  sw.Flush();  }  }  } |

Створення статистики:

|  |
| --- |
| analysis\_game\_menu |
| while (1 == 1)  {  int game = 0;  Int32.TryParse(view.game\_analysis(), out game);  if (game == 1)  {  model.dev\_export\_csv();  var proc = Process.Start(@"cmd.exe ", @"/c D:\DB\course\_work\developer\developer.html");  }  else if (game == 2)  {  model.games\_export\_csv();  var proc = Process.Start(@"cmd.exe ", @"/c D:\DB\course\_work\genre\genre.html");  }  else if (game == 3)  {  break;  }  else  {  view.error\_option();  }  } |

Функція backup:

|  |
| --- |
| backup |
| String dumpCommand = "\"" + "C:\\Program Files\\PostgreSQL\\12\\bin\\pg\_dump.exe" + "\"" + " -Fc" + " -h " + "localhost" + " -p " + 5432 + " -d " + "Course\_work" + " -U " + "postgres" + "";  String passFileContent = "" + "localhost" + ":" + 5432 + ":" + "Course\_work" + ":" + "postgres" + ":" + 1111 + "";  String batFilePath = Path.Combine(  Path.GetTempPath(),  Guid.NewGuid().ToString() + ".bat");  String passFilePath = Path.Combine(  Path.GetTempPath(),  Guid.NewGuid().ToString() + ".conf");  try  {  String batchContent = "";  batchContent += "@" + "set PGPASSFILE=" + passFilePath + "\n";  batchContent += "@" + dumpCommand + " > " + "\"" + "D:\\DB\\coursework.sql" + "\"" + "\n";  File.WriteAllText(  batFilePath,  batchContent,  Encoding.ASCII);  File.WriteAllText(  passFilePath,  passFileContent,  Encoding.ASCII);  if (File.Exists("D:\\DB\\coursework.sql"))  File.Delete("D:\\DB\\coursework.sql");  ProcessStartInfo oInfo = new ProcessStartInfo(batFilePath);  oInfo.UseShellExecute = false;  oInfo.CreateNoWindow = true;  using (Process proc = System.Diagnostics.Process.Start(oInfo))  {  proc.WaitForExit();  proc.Close();  }  }  finally  {  if (File.Exists(batFilePath))  File.Delete(batFilePath);  if (File.Exists(passFilePath))  File.Delete(passFilePath);  } |

|  |
| --- |
| Course\_workContext |
| using System;  using Microsoft.EntityFrameworkCore;  using Microsoft.EntityFrameworkCore.Metadata;  #nullable disable  namespace course\_work  {  public partial class Course\_workContext : DbContext  {  public Course\_workContext()  {  }  public Course\_workContext(DbContextOptions<Course\_workContext> options)  : base(options)  {  }  public virtual DbSet<Game> Games { get; set; }  public virtual DbSet<GameRating> GameRatings { get; set; }  public virtual DbSet<Player> Players { get; set; }  public virtual DbSet<Rating> Ratings { 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=Course\_work;Username=postgres;Password=1111");  }  }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  modelBuilder.HasAnnotation("Relational:Collation", "Ukrainian\_Ukraine.1251");  modelBuilder.Entity<Game>(entity =>  {  entity.ToTable("games");  entity.Property(e => e.Id)  .HasColumnName("id")  .UseIdentityAlwaysColumn()  .HasIdentityOptions(10016L, null, null, null, null, null);  entity.Property(e => e.AccessRating)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("access\_rating");  entity.Property(e => e.Developer)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("developer");  entity.Property(e => e.Genre)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("genre");  entity.Property(e => e.Name)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("name");  entity.Property(e => e.Platform)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("platform");  entity.Property(e => e.Publisher)  .IsRequired()  .HasColumnType("character varying")  .HasColumnName("publisher");  entity.Property(e => e.ReleaseYear).HasColumnName("release\_year");  });  modelBuilder.Entity<GameRating>(entity =>  {  entity.ToTable("game-rating");  entity.Property(e => e.Id)  .HasColumnName("id")  .HasIdentityOptions(10015L, null, null, null, null, null);  entity.Property(e => e.GameId).HasColumnName("game\_id");  entity.Property(e => e.RatingId).HasColumnName("rating\_id");  entity.HasOne(d => d.Game)  .WithMany(p => p.GameRatings)  .HasForeignKey(d => d.GameId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("game\_fkey");  entity.HasOne(d => d.Rating)  .WithMany(p => p.GameRatings)  .HasForeignKey(d => d.RatingId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("rating\_fkey");  });  modelBuilder.Entity<Player>(entity =>  {  entity.ToTable("players");  entity.Property(e => e.Id)  .ValueGeneratedNever()  .HasColumnName("id");  entity.Property(e => e.Eu).HasColumnName("EU");  entity.Property(e => e.GameId).HasColumnName("game\_id");  entity.Property(e => e.Jp).HasColumnName("JP");  entity.Property(e => e.Na).HasColumnName("NA");  entity.HasOne(d => d.Game)  .WithMany(p => p.Players)  .HasForeignKey(d => d.GameId)  .OnDelete(DeleteBehavior.ClientSetNull)  .HasConstraintName("game\_fkey");  });  modelBuilder.Entity<Rating>(entity =>  {  entity.ToTable("rating");  entity.Property(e => e.Id)  .HasColumnName("id")  .UseIdentityAlwaysColumn()  .HasIdentityOptions(10016L, null, null, null, null, null);  entity.Property(e => e.Critic).HasColumnName("critic");  entity.Property(e => e.User).HasColumnName("user");  });  OnModelCreatingPartial(modelBuilder);  }  partial void OnModelCreatingPartial(ModelBuilder modelBuilder);  }  } |