Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

ЗВІТ

Дисципліна «Фреймворки розробки програмного забезпечення»

Робота №3

Тема «Реалізація зберігання, видобування та обробки даних»

Виконав варіант 19

Студент КНТ-122 Онищенко О. А.

Прийняли

Викладач Зелік О. В.

2024

МЕТА

Ознайомитися з сучасниим системами керування базами даних, можливостями створення з’єднань з базами даних через програмні додатки, мовою XML та навчитися на практиці використовувати бази даних в якості сховищ даних.

ЗАВДАННЯ

Виконати аналіз ТЗ та розробленої архітектури системи щодо вимог до організації даних.

Прийняти рішення про систему збереження даних та обґрунтувати її.

Розробити структуру бази даних, необхідної для збереження даних системи. Додатково (не є обов’язковою вимогою) для зберігання деяких даних можуть використовуватися XML-файли або вони можуть бути створені як паралельне сховище.

Реалізувати функціональність програмного забезпечення, пов’язану зі взаємодією з даними. Під час реалізації взаємодії має обов’язково враховуватися можливість роботи з даними з віддалених пристроїв, що мінімально має підтримуватися на рівні відповідної системи керування базами даних.

Виконати тестування розробленого програмного забезпечення. У процесі тестування має обов’язково застосовуватись модульне тестування.

Тестування має виконуватися шляхом взаємодії з різними файлами (в яких зберігаються дані) визначеної структури на пристроях з різними апаратними характеристиками під керуванням різних операційних систем або версій операційних систем.

Виконати аналіз отриманих результатів тестування. У процесі аналізу отриманих результатів має бути порівняно результати, отримані під керуванням різних операційних систем (або їх версій) та на різних пристроях.

ВИКОНАННЯ

Індивідуальне завдання розбито на кроки:

1. Вибір бази даних
2. Структура бази даних
3. Програма взаємодії з базою
4. Тестування програми
5. Аналіз результатів

Деталі кожного кроку наведено у відповідному пункті.

1 Вибір бази даних

З урахуванням попередніх вимог у Технічному завданні та розробленої архітектури програмного забезпечення прийнято рішення використовувати систему керування базами даних MySQL, оскільки можливість її використання була зазначення у вимогах та розробник має досвід користування нею.

2 Структура бази даних

Процес розробки структури бази даних покроково описано нижче:

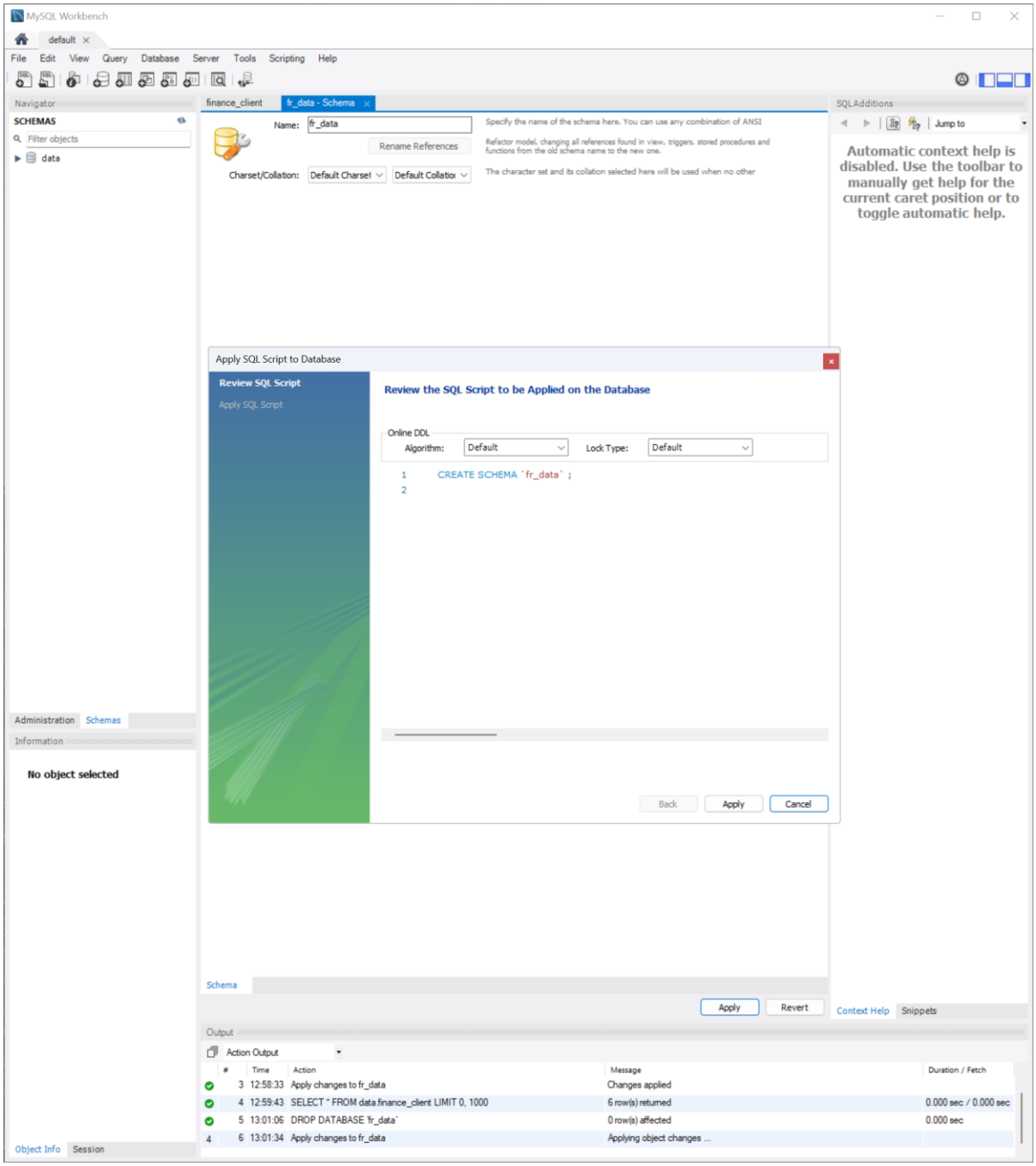

Рисунок 2.1 – Створення нової схеми бази даних для відокремлення відповідальностей між різними проєктами

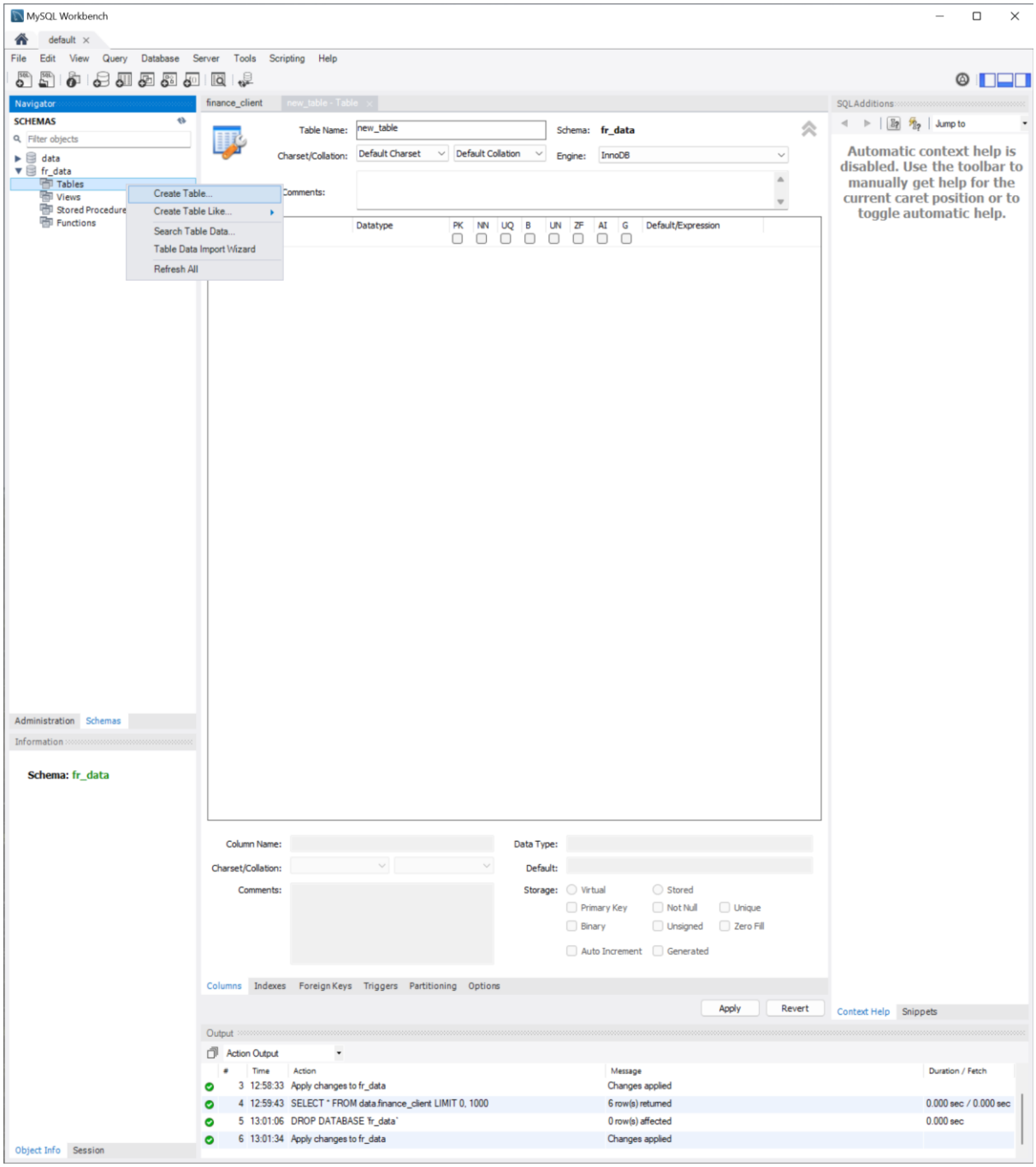

Рисунок 2.2 – Створення нової таблиці

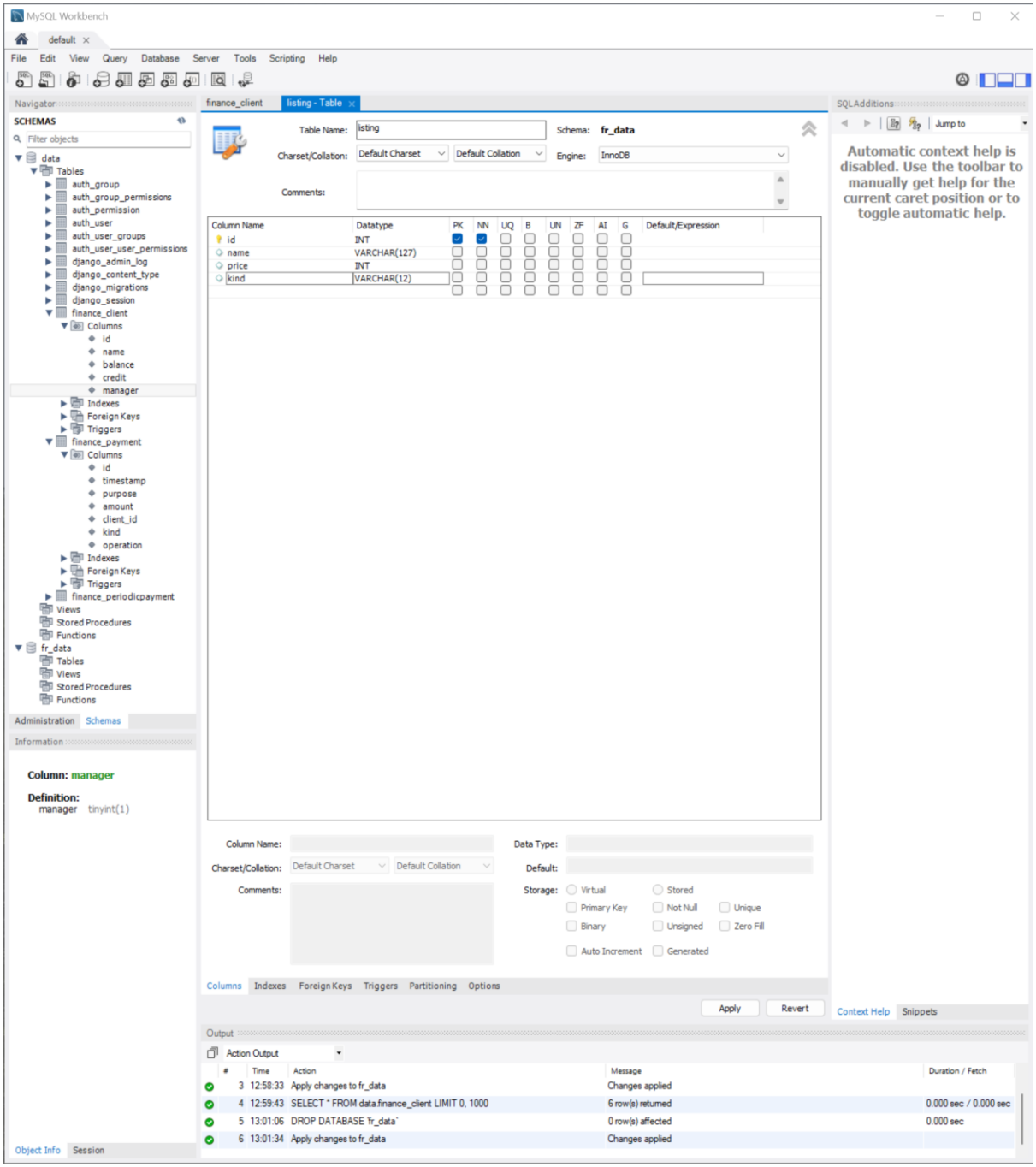

Рисунок 2.3 – Таблиця оголошення із заповненими полями

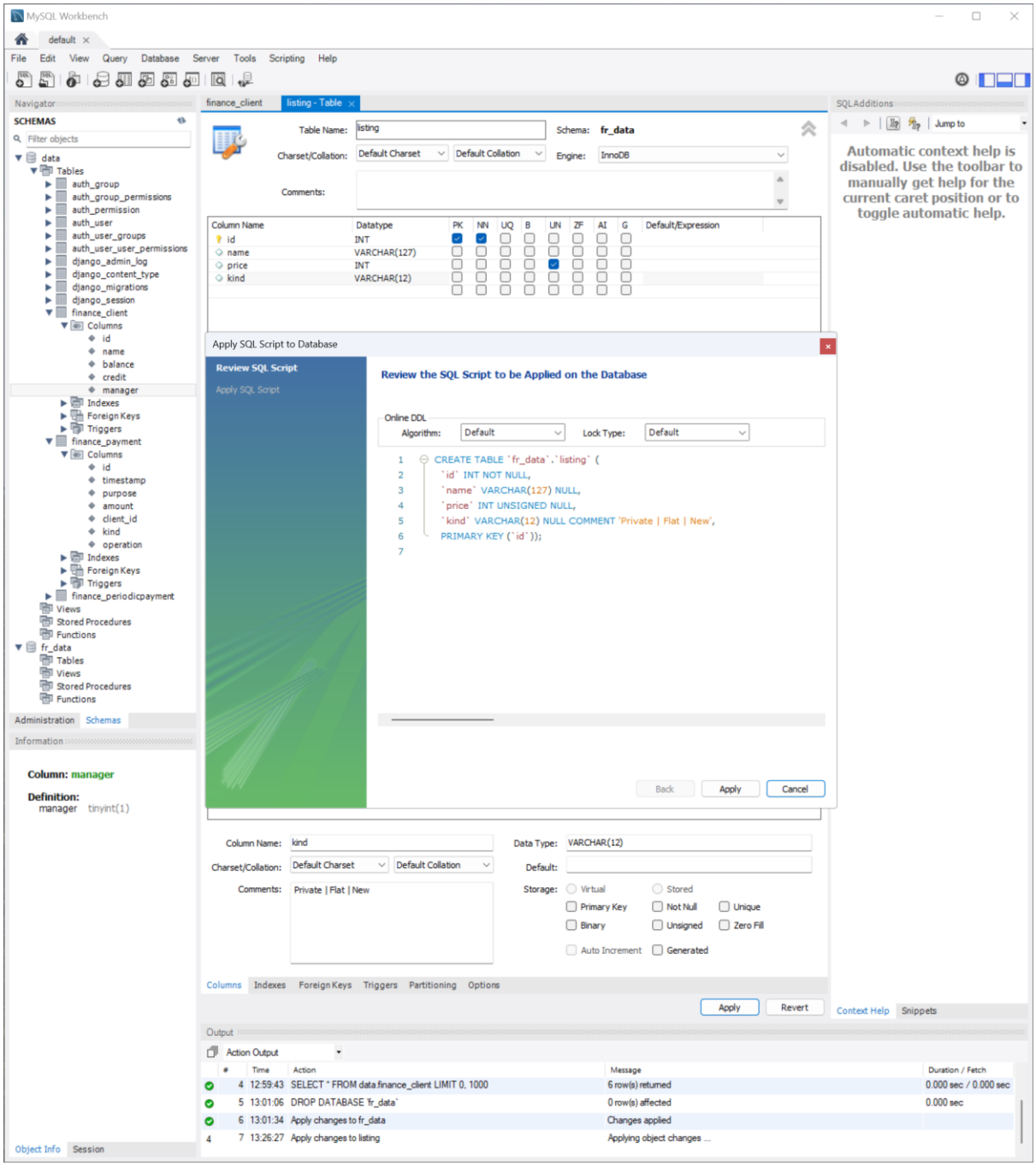

Рисунок 2.4 – Таблиця оголошень зі змінами та SQL запитом на створення

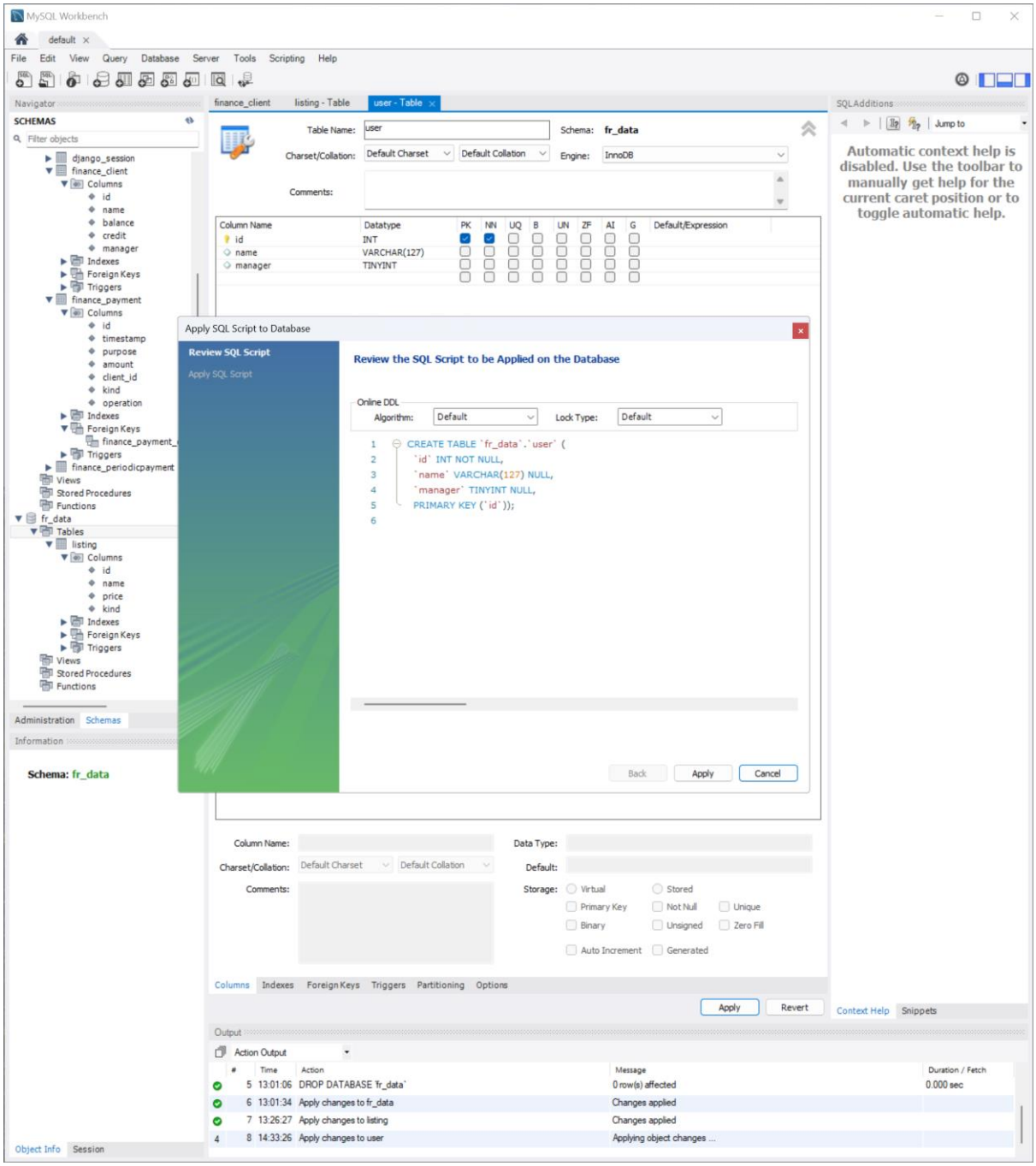

Рисунок 2.5 – Таблиця користувача із запитом

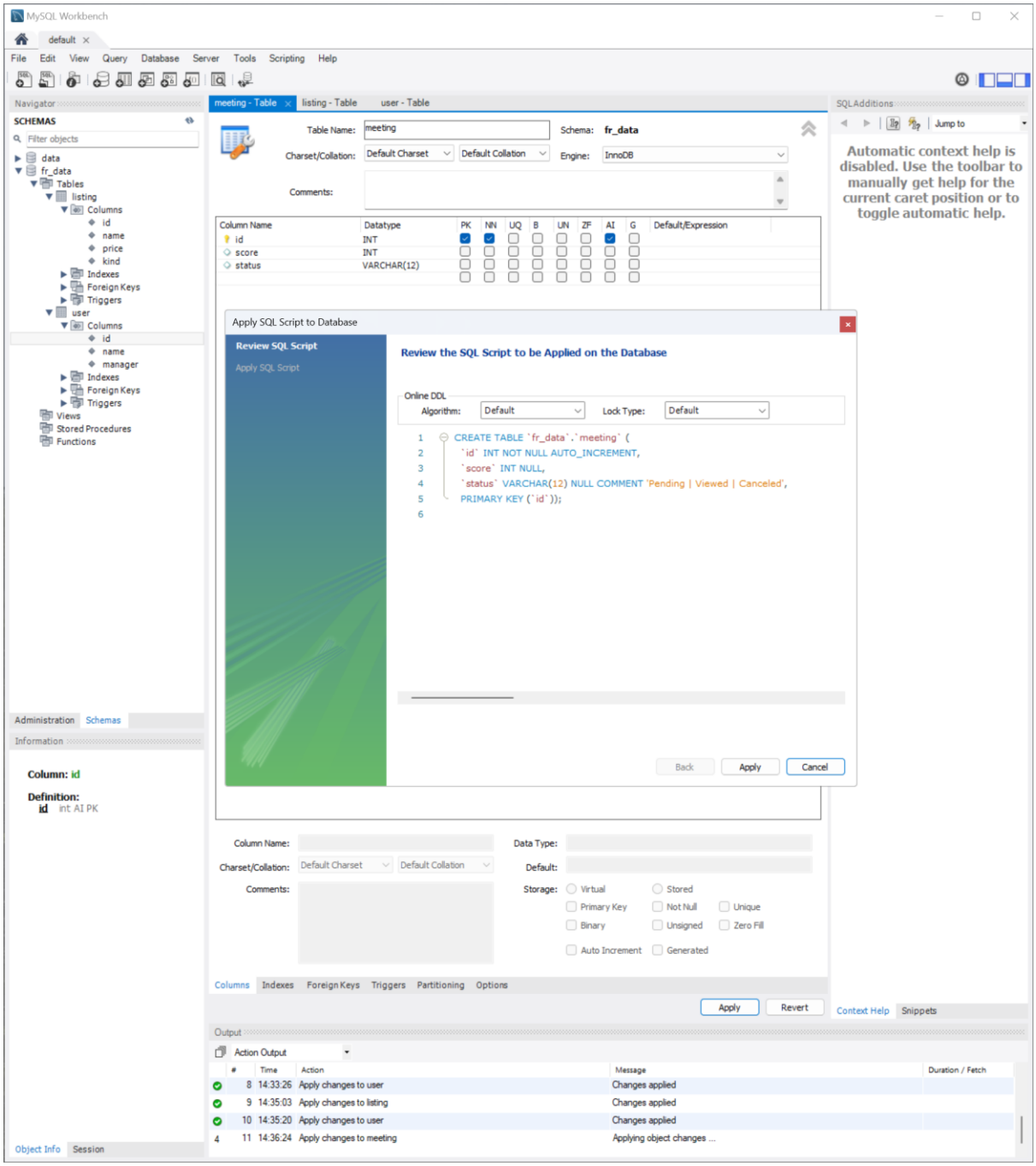

Рисунок 2.6 – Таблиця зустрічі з запитом

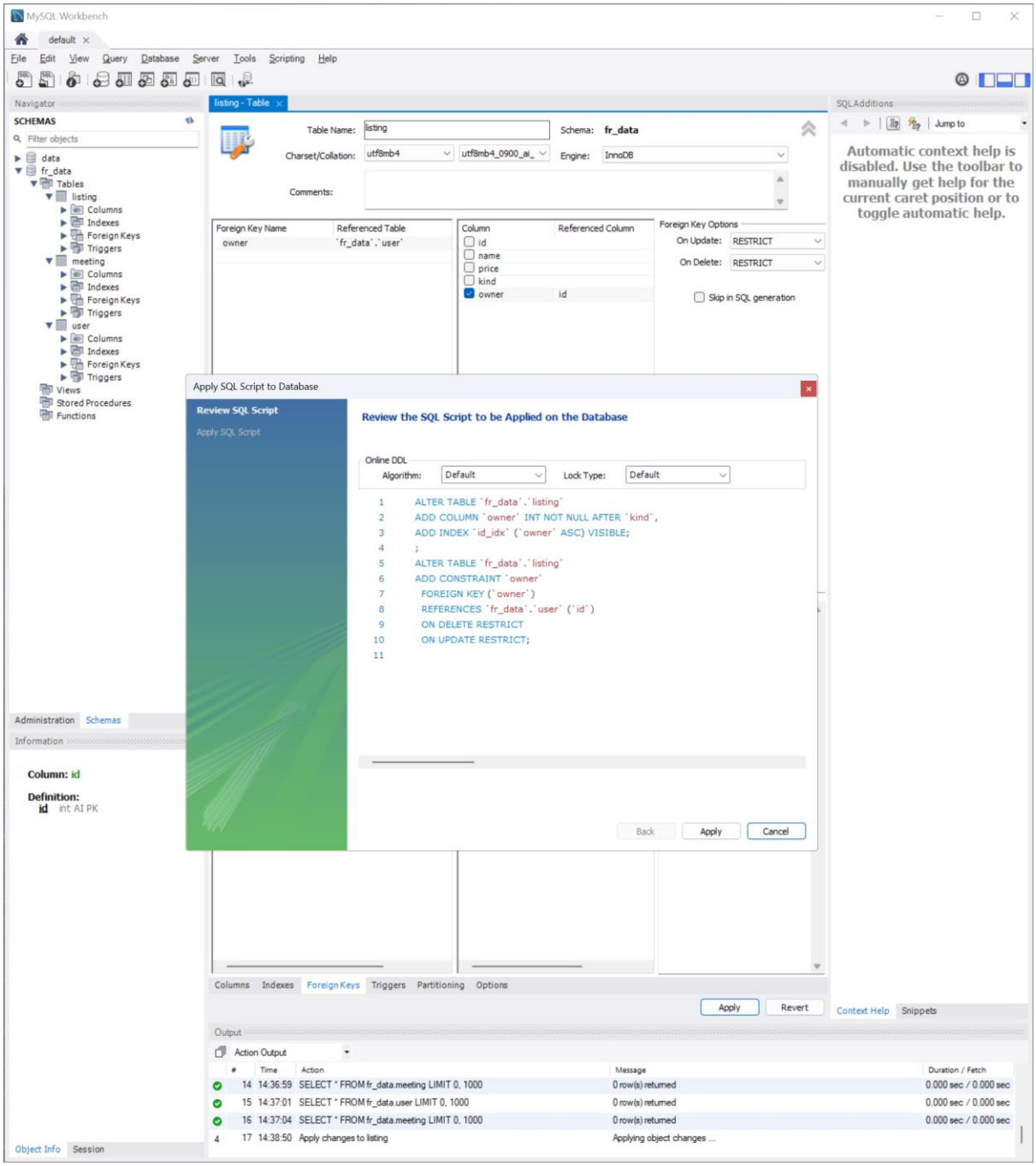

Рисунок 2.7 – Додавання посилання на власника для оголошення

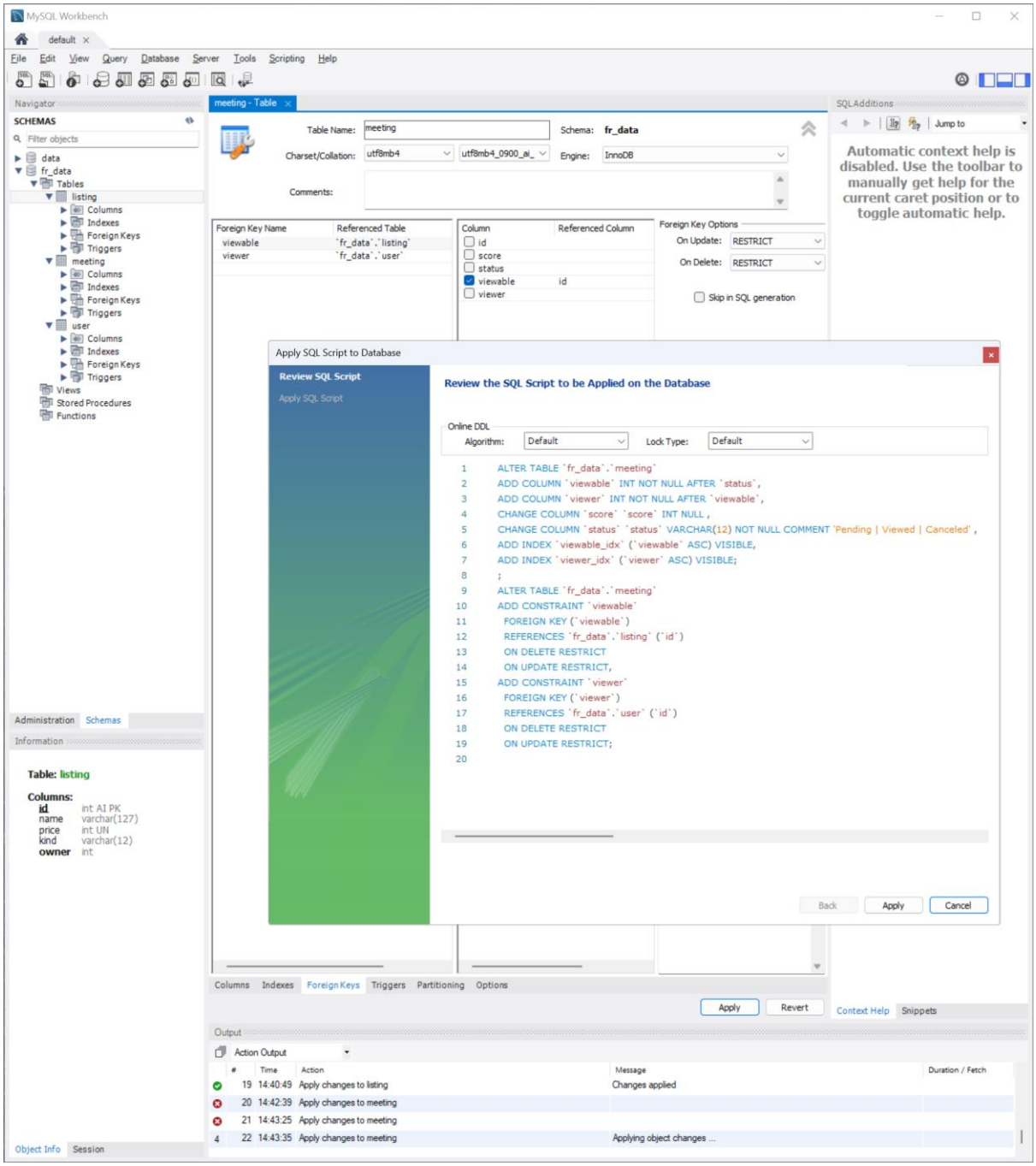

Рисунок 2.8 – Додавання посилань на переглядаємий об’єкт та користувача переглядача для таблиці зустрічі

По виконанню кроку отримана наступна структура:

Фрагмент коду 2.1 – Структурна схема бази даних

```
User
  id INT PrimaryKey AutoIncrement
  name VARCHAR(127) NotNull
```

```
  manager BOOLEAN NotNull

Listing
  id INT PrimaryKey AutoIncrement
  name VARCHAR(127) NotNull
  price INT Unsigned NotNull
  kind VARCHAR(12) NotNull <Private|Flat|New>
  INT owner ForeignKey NotNull <User.id>

Meeting
  id INT PrimaryKey AutoIncrement
  score INT
  status VARCHAR(12) default 'Pending' <Pending|Viewed|Canceled>
  viewable INT ForeignKey NotNull <Listing.id>
  viewer INT ForeignKey NotNull <User.id>
```

3 Програма взаємодії з базою

Код програми взаємодії з базою даних мовою C# нижче.

Фрагмент коду 3.1 – Взаємодія з базою даних

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;

namespace three_source
{
    public class User
    {
        public int id { get; set; }
        public string name { get; set; }
        public int manager { get; set; } = 0;
    }
    public class Listing
    {
        public int id { get; set; }
        public string name { get; set; }
        public int price { get; set; }
        public string kind { get; set; }
        public User owner { get; set; }
    }
    public class Meeting
    {
        public int id { get; set; }
        public int score { get; set; } = 0;
        public string status { get; set; } = "Pending";
        public Listing viewable { get; set; }
        public User viewer { get; set; }
```

```
    }
    public class UserHandler
    {
        string query;
        MySqlCommand command;
        MySqlDataReader reader;
        MySqlConnection connection;
        public UserHandler(MySqlConnection connection) {
this.connection = connection; }
        public User create(string name, int manager = 0)
        {
            query = $"INSERT INTO user (name,manager) VALUES
('{name}',{manager});";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();

            query = "SELECT LAST_INSERT_ID();";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var user = new User();
            while (reader.Read())
            {
                user.id = reader.GetInt32(0);
                user.name = name;
                user.manager = manager;
            }
            reader.Close();
            return user;
        }
        public User read(int id)
        {
            query = $"SELECT id,name,manager FROM user WHERE id={id};";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                var user = new User();
                user.id = reader.GetInt32(0);
                user.name = reader.GetString(1);
                user.manager = reader.GetInt32(2);
                reader.Close();
                return user;
            }
            reader.Close();
            return null;
        }
        public List<User> readAll()
        {
            query = "SELECT id,name,manager FROM user;";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var users = new List<User>();
            while (reader.Read())
            {
                var user = new User();
                user.id = reader.GetInt32(0);
                user.name = reader.GetString(1);
                user.manager = reader.GetInt32(2);
```

```
                users.Add(user);
            }
            reader.Close();
            return users;
        }
        public void update(User user)
        {
            query = $"UPDATE user SET
name='{user.name}',manager={user.manager} WHERE id={user.id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
        public void delete(int id)
        {
            query = $"DELETE FROM user WHERE id={id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
    }
    public class ListingHandler
    {
        string query;
        MySqlCommand command;
        MySqlDataReader reader;
        MySqlConnection connection;
        List<string> kinds = new List<string>{
            "House",
            "Flat",
            "New",
        };
        public ListingHandler(MySqlConnection connection) {
this.connection = connection; }
        public Listing create(string name, int price, string kind, User
owner)
        {
            var correctKind = kinds.Contains(kind);
            if (!correctKind)
            {
                return null;
            }

            query = $"INSERT INTO listing (name,price,kind,owner)
VALUES ('{name}','{price}','{kind}','{owner.id}');";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();

            query = "SELECT LAST_INSERT_ID();";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var listing = new Listing();
            while (reader.Read())
            {
                listing.id = reader.GetInt32(0);
                listing.name = name;
                listing.price = price;
                listing.kind = kind;
                listing.owner = owner;
            }
```

```
            reader.Close();
            return listing;
        }
        public Listing read(int id)
        {
            query = $"SELECT id,name,price,kind,owner FROM listing
WHERE id={id};";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            int tempOwner = 0;
            var listing = new Listing();
            while (reader.Read())
            {
                listing.id = reader.GetInt32(0);
                listing.name = reader.GetString(1);
                listing.price = reader.GetInt32(2);
                listing.kind = reader.GetString(3);
                tempOwner = reader.GetInt32(4);
            }
            reader.Close();
            listing.owner = new
UserHandler(connection).read(tempOwner);
            return listing;
        }
        public List<Listing> readAll()
        {
            query = "SELECT id,name,price,kind,owner FROM listing;";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var listings = new List<Listing>();
            var owners = new List<int>();
            while (reader.Read())
            {
                var listing = new Listing();
                listing.id = reader.GetInt32(0);
                listing.name = reader.GetString(1);
                listing.price = reader.GetInt32(2);
                listing.kind = reader.GetString(3);
                owners.Add(reader.GetInt32(4));
                listings.Add(listing);
            }
            reader.Close();
            for (int i = 0; i < listings.Count; i++)
            {
                listings[i].owner = new
UserHandler(connection).read(owners[i]);
            }
            return listings;
        }
        public void update(Listing listing)
        {
            query = $"UPDATE listing SET
name='{listing.name}',price={listing.price},kind='{listing.kind}',owner
={listing.owner.id} WHERE id={listing.id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
        public void delete(int id)
```

```
        {
            query = $"DELETE FROM listing WHERE id={id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
    }
    public class MeetingHandler
    {
        string query;
        MySqlCommand command;
        MySqlDataReader reader;
        MySqlConnection connection;
        List<string> statuses = new List<string>{
            "Pending",
            "Viewed",
            "Canceled",
        };
        public MeetingHandler(MySqlConnection connection) {
this.connection = connection; }
        public Meeting create(Listing viewable, User viewer, int score
= 0, string status = "Pending")
        {
            var correctStatus = statuses.Contains(status);
            if (!correctStatus)
            {
                return null;
            }

            query = $"INSERT INTO meeting
(score,status,viewable,viewer) VALUES
({score},'{status}',{viewable.id},{viewer.id});";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();

            query = "SELECT LAST_INSERT_ID();";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var meeting = new Meeting();
            while (reader.Read())
            {
                meeting.id = reader.GetInt32(0);
                meeting.score = score;
                meeting.status = status;
                meeting.viewable = viewable;
                meeting.viewer = viewer;
            }
            reader.Close();
            return meeting;
        }
        public Meeting read(int id)
        {
            query = $"SELECT id,score,status,viewable,viewer FROM
meeting WHERE id={id};";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var meeting = new Meeting();
            var tempViewable = 0;
            var tempViewer = 0;
```

```
            while (reader.Read())
            {
                meeting.id = reader.GetInt32(0);
                meeting.score = reader.GetInt32(1);
                meeting.status = reader.GetString(2);
                tempViewable = reader.GetInt32(3);
                tempViewer = reader.GetInt32(4);
            }
            reader.Close();
            meeting.viewable = new
ListingHandler(connection).read(tempViewable);
            meeting.viewer = new
UserHandler(connection).read(tempViewer);
            return meeting;
        }
        public List<Meeting> readAll()
        {
            query = "SELECT id,score,status,viewable,viewer FROM
meeting;";
            command = new MySqlCommand(query, connection);
            reader = command.ExecuteReader();
            var meetings = new List<Meeting>();
            var viewables = new List<int>();
            var viewers = new List<int>();
            while (reader.Read())
            {
                var meeting = new Meeting();
                meeting.id = reader.GetInt32(0);
                meeting.score = reader.GetInt32(1);
                meeting.status = reader.GetString(2);
                viewables.Add(reader.GetInt32(3));
                viewers.Add(reader.GetInt32(4));
                meetings.Add(meeting);
            }
            reader.Close();
            for (int i = 0; i < meetings.Count; i++)
            {
                meetings[i].viewable = new
ListingHandler(connection).read(viewables[i]);
                meetings[i].viewer = new
UserHandler(connection).read(viewers[i]);
            }
            return meetings;
        }
        public void update(Meeting meeting)
        {
            query = $"UPDATE meeting SET
score={meeting.score},status='{meeting.status}',viewable={meeting.viewa
ble.id},viewer={meeting.viewer.id} WHERE id={meeting.id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
        public void delete(int id)
        {
            query = $"DELETE FROM meeting WHERE id={id};";
            command = new MySqlCommand(query, connection);
            command.ExecuteNonQuery();
        }
```

```
    }
    public class Test
    {
        MySqlConnection connection;
        public Test(MySqlConnection connection) { this.connection =
connection; }
        public void testUser(string userName = "Luke")
        {
            // Create
            var handler = new UserHandler(connection);
            var user = handler.create(userName);
            Console.WriteLine($"Created user {user.id} name {user.name}
manager {Convert.ToBoolean(user.manager)}");

            // Read
            user = handler.read(user.id);
            Console.WriteLine($"Read user {user.id} name {user.name}
manager {Convert.ToBoolean(user.manager)}");

            // Update
            user.manager = 1;
            handler.update(user);

            user = handler.read(user.id);
            Console.WriteLine($"Updated user {user.id} name {user.name}
manager {Convert.ToBoolean(user.manager)}");

            // Delete
            handler.delete(user.id);
            Console.WriteLine($"Deleted user {user.id}");

            // Show all
            var users = handler.readAll();
            Console.WriteLine($"All users ({users.Count}):");
            foreach (var u in users)
            {
                Console.WriteLine($"- User {u.id} name {u.name} manager
{Convert.ToBoolean(u.manager)}");
            }
        }
        public void testListing(string listingName = "Quiet House in
the Countryside", int price = 120, string kind = "House", User owner =
null)
        {
            // Create
            var handler = new ListingHandler(connection);
            var listing = handler.create(listingName, price, kind,
owner);
            Console.WriteLine($"Created listing {listing.id} name
{listing.name} price {listing.price} kind {listing.kind} owner
{listing.owner.name}");

            // Read
            listing = handler.read(listing.id);
            Console.WriteLine($"Read listing {listing.id} name
{listing.name} price {listing.price} kind {listing.kind} owner
{listing.owner.name}");
```

```
            // Update
            listing.name = "Modern Appartament near Court";
            listing.kind = "New";
            handler.update(listing);

            listing = handler.read(listing.id);
            Console.WriteLine($"Updated listing {listing.id} name
{listing.name} price {listing.price} kind {listing.kind} owner
{listing.owner.name}");

            // Delete
            handler.delete(listing.id);
            Console.WriteLine($"Deleted listing {listing.id}");

            // Show all
            var listings = handler.readAll();
            Console.WriteLine($"All listings ({listings.Count}):");
            foreach (var l in listings)
            {
                Console.WriteLine($"- Listing {l.id} name {l.name}
price {l.price} kind {l.kind} owner {l.owner.name}");
            }
        }
        public void testMeeting(Listing viewable = null, User viewer =
null)
        {
            // Create
            var handler = new MeetingHandler(connection);
            var meeting = handler.create(viewable, viewer);
            Console.WriteLine($"Created meeting {meeting.id} score
{meeting.score} status {meeting.status} viewable
{meeting.viewable.name} viewer {meeting.viewer.name}");

            // Read
            meeting = handler.read(meeting.id);
            Console.WriteLine($"Read meeting {meeting.id} score
{meeting.score} status {meeting.status} viewable
{meeting.viewable.name} viewer {meeting.viewer.name}");

            // Update
            meeting.score = 7;
            meeting.status = "Viewed";
            handler.update(meeting);

            meeting = handler.read(meeting.id);
            Console.WriteLine($"Updated meeting {meeting.id} score
{meeting.score} status {meeting.status} viewable
{meeting.viewable.name} viewer {meeting.viewer.name}");

            // Delete
            handler.delete(meeting.id);
            Console.WriteLine($"Deleted meeting {meeting.id}");

            // Show all
            var meetings = handler.readAll();
            Console.WriteLine($"All meetings ({meetings.Count}):");
            foreach (var m in meetings)
            {
```

```
                Console.WriteLine($"- Meeting {m.id} score {m.score}
status {m.status} viewable {m.viewable.name} viewer {m.viewer.name}");
            }
        }
    }
    public class Program
    {
        static void Main(string[] args)
        {
            const string conStr =
"uid=root;pwd=1313;host=localhost;port=3306;database=fr_data";
            var connection = new MySqlConnection(conStr);
            connection.Open();

            var tester = new Test(connection);
            tester.testUser();
            Console.WriteLine();
            tester.testListing(owner: new
UserHandler(connection).read(1));
            Console.WriteLine();
            tester.testMeeting(viewable: new
ListingHandler(connection).read(1), viewer: new
UserHandler(connection).read(1));
            Console.WriteLine();

            connection.Close();
        }
    }
}
```

Структура організації даних програми взаємодії з базою даних наведено нижче:

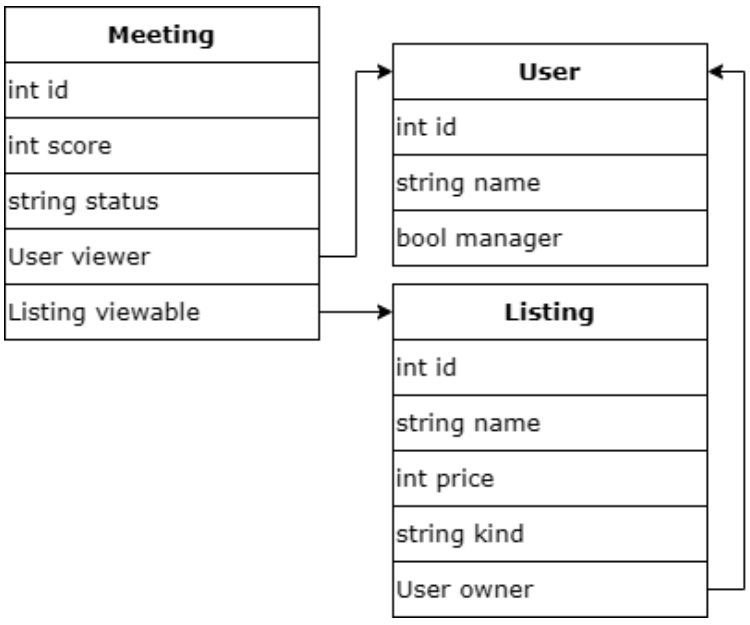

Рисунок 3.1 – Структура організації даних у програмі

4 Тестування програми

Після виконання програма дає такий вихід у консоль:

```
Created listing 30 name Quiet House in the Countryside price 120 kind House owner admin
Read listing 30 name Quiet House in the Countryside price 120 kind House owner admin
Updated listing 30 name Modern Appartament near Court price 120 kind New owner admin
Deleted listing 30
All listings (6):
- Listing 1 name test listing price 120 kind Flat owner admin
- Listing 2 name Test Listing price 120 kind New owner admin
- Listing 3 name Test Listing price 120 kind New owner admin
- Listing 4 name Quiet House in the Countryside price 120 kind House owner admin
- Listing 5 name Quiet House in the Countryside price 120 kind House owner admin
- Listing 6 name Quiet House in the Countryside price 120 kind House owner admin

Created meeting 16 score 0 status Pending viewable test listing viewer admin
Read meeting 16 score 0 status Pending viewable test listing viewer admin
Updated meeting 16 score 7 status Viewed viewable test listing viewer admin
Deleted meeting 16
All meetings (5):
- Meeting 4 score 7 status Canceled viewable test listing viewer admin
- Meeting 6 score 7 status Canceled viewable test listing viewer admin
- Meeting 8 score 7 status Canceled viewable test listing viewer admin
- Meeting 12 score 8 status Viewed viewable test listing viewer admin
- Meeting 14 score 10 status Pending viewable test listing viewer admin
```

Рисунок 4.1 – Результати роботи програми у консолі

5 Аналіз результатів

Система працює коректно, всі функції виконує.