Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №3

Создание ER-диаграммы. Описание технических требований к приложению BerkeleyDB

Студент: К.В. Горбачевский

Преподователь: А.И. Крюков

МИНСК 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc181438079)

[1 СОЗДАНИЕ UML-ДИАГРАММЫ 5](#_Toc181438080)

[1.1 Предметная область 5](#_Toc181438081)

[1.2 Типы объектов 5](#_Toc181438082)

[1.3 Атрибуты объектов 6](#_Toc181438083)

[1.4 Типы связей 6](#_Toc181438084)

[2 КОНВЕРТЕР 7](#_Toc181438085)

[ЗАКЛЮЧЕНИЕ 10](#_Toc181438086)

# ВВЕДЕНИЕ

В лабораторной работе выполняется концептуального проектирование БД с использованием UML-модели представления данных (модели «сущность- связь»). Требуется разработать UML-модель данных с учетом семантических ограничений заданной предметной области и представить модель в виде UML- диаграммы.

Темой данной лабораторной работы является разработка UML-диаграммы сущностей и связей в организации «Студия звукозаписи».

«Студия звукозаписи» - это стандартная модель, работающая по принципу клиент, услуга, продавец. В качестве продавца выступает владелец студии и напрямую с клиентом он не взаимодействует. В качестве клиента выступает музыкальный исполнитель или группа, которая хочет арендовать помещение на короткий срок (не более суток), в определенном жанре.

# СОЗДАНИЕ UML-ДИАГРАММЫ

Исходное задание: Создать концептуальную модель организации «Студия звукозаписи»и представить сущности и связи в виде ER-диаграммы.

Концептуальная UML-диаграмма представлена на рисунке 1.

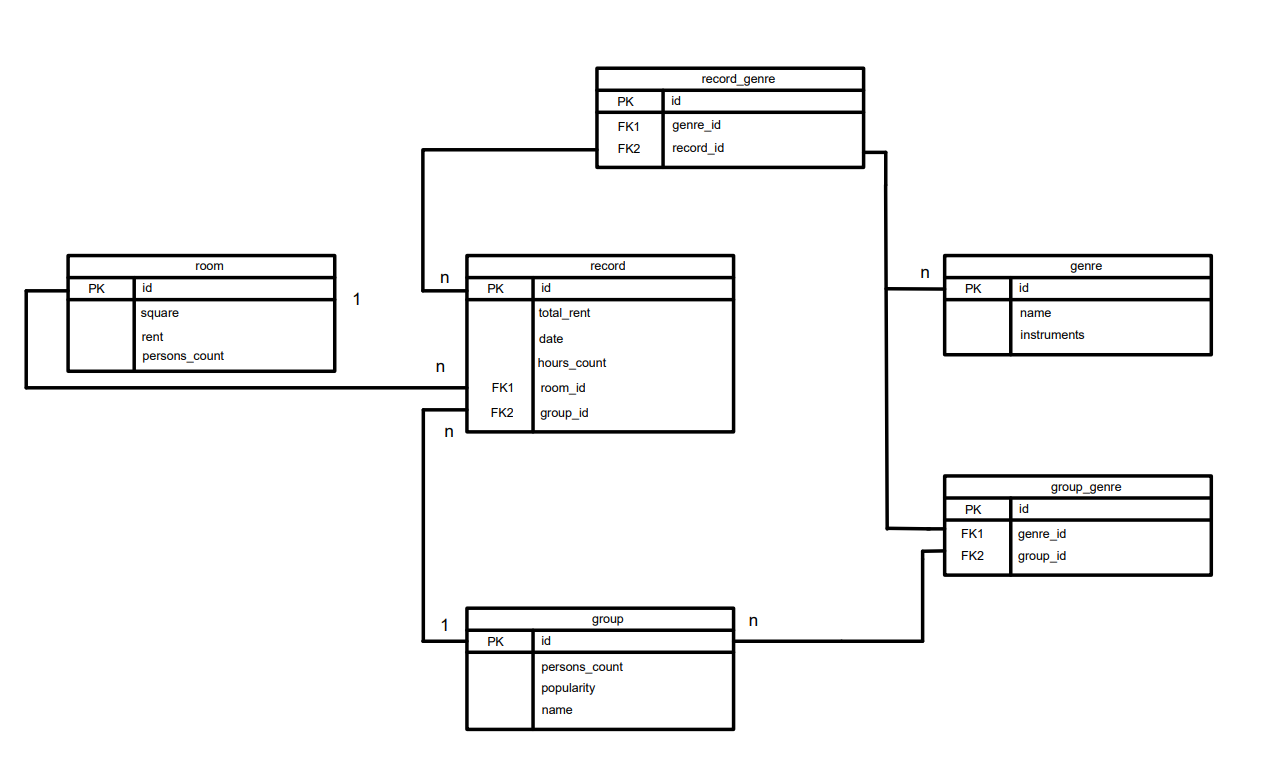


Рисунок 1 – UML-диаграмма

## Предметная область

Предметная область – «Студия звукозаписи». Модели по типу «клиент-продавец». Предоставляемая услуга – запись голоса или инструментов, аренда инструментов и комнат.

## Типы объектов

Для модели «Студия звукозаписи» было выделено 6 типов объектов

1) «Комната» – физическое пространство, в котором будет происходить запись.

2) «Группа» – представляет клиента студии.

3) «Жанр» – представляет справочную информацию о жанрах музыки для записи.

4) «Запись» - представляет исчерпывающую информацию о конкретной записи (дата, время, комната, стоимость и т.д.)

Созданы связи типа многие ко многим для таблиц жанры и группы. Так как одна и так же группа может играть сразу в нескольких жанрах. Например рок и блюз. Для данной связи была создана отдельная таблица. Также созданы связи многим для таблицы записи и жанры. Так как запись может быть одновремененно в нескольких жанрах.

## Атрибуты объектов

Атрибуты объекта «Комната» – «Площадь», «Стоимость аренды» и «Допустимое кол-во человек».

Для объекта «Группа» – «Название», «Популярность» и «Кол-во участников».

Для объекта «Жанр» – «Название», «Инструменты».

Атрибуты объекта «Запись» – собственные атрибуты «Дата», «Кол-во часов» и «Общая стоимость». Также у данного объекты имеются атрибуты, являющиеся ссылками на все другие объекты.

## Типы связей

Самым главным объектом в данной организации является «Запись». Этот объект имеет прямые связи с остальными пятью объектами, так объединяет всю информацию о используемый инструментах, сотруднике, который будет осуществлять запись, комнате, в которой будет проходить запись и другие аспекты записи.

Остальные объекты независимы, что в дальнейшем поможет разработать модульное ПО для данного типа организации.

# КОНВЕРТЕР

Разработан конвертер базы данных их PostgreSQL в BerkeleyDB.

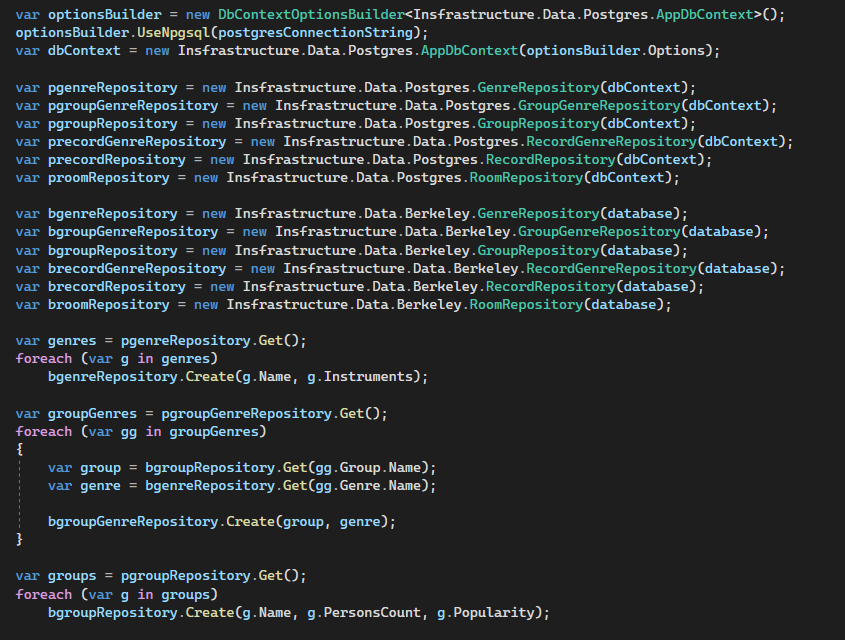


Рисунок 2.1 – Скриншот кода конвертера

Для работы конвертера были добавлены реализации интерфейсов для BerkeleyDB, суть работы конвертера заключается в получении данных из Postgres, биндинга их в модели и добавление в Berkeley через реализацию репозиторя. Полный код представлен ниже.

using BerkeleyDB;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

namespace DatabaseDataMigration

{

internal class Program

{

static void Main(string[] args)

{

var builder = new ConfigurationBuilder().SetBasePath(Directory.GetCurrentDirectory()).AddJsonFile("appsettings.json", optional: false).AddEnvironmentVariables();

IConfiguration configuration = builder.Build();

var berkeleyDbPath = configuration.GetConnectionString("Berkeley");

var postgresConnectionString = configuration.GetConnectionString("Postgres");

var database = Database.Open(berkeleyDbPath, new BTreeDatabaseConfig

{

Creation = CreatePolicy.IF\_NEEDED,

CacheSize = new CacheInfo(0, 64 \* 1024, 1),

PageSize = 8 \* 1024

});

var optionsBuilder = new DbContextOptionsBuilder<Insfrastructure.Data.Postgres.AppDbContext>();

optionsBuilder.UseNpgsql(postgresConnectionString);

var dbContext = new Insfrastructure.Data.Postgres.AppDbContext(optionsBuilder.Options);

var pgenreRepository = new Insfrastructure.Data.Postgres.GenreRepository(dbContext);

var pgroupGenreRepository = new Insfrastructure.Data.Postgres.GroupGenreRepository(dbContext);

var pgroupRepository = new Insfrastructure.Data.Postgres.GroupRepository(dbContext);

var precordGenreRepository = new Insfrastructure.Data.Postgres.RecordGenreRepository(dbContext);

var precordRepository = new Insfrastructure.Data.Postgres.RecordRepository(dbContext);

var proomRepository = new Insfrastructure.Data.Postgres.RoomRepository(dbContext);

var bgenreRepository = new Insfrastructure.Data.Berkeley.GenreRepository(database);

var bgroupGenreRepository = new Insfrastructure.Data.Berkeley.GroupGenreRepository(database);

var bgroupRepository = new Insfrastructure.Data.Berkeley.GroupRepository(database);

var brecordGenreRepository = new Insfrastructure.Data.Berkeley.RecordGenreRepository(database);

var brecordRepository = new Insfrastructure.Data.Berkeley.RecordRepository(database);

var broomRepository = new Insfrastructure.Data.Berkeley.RoomRepository(database);

var genres = pgenreRepository.Get();

foreach (var g in genres)

bgenreRepository.Create(g.Name, g.Instruments);

var groupGenres = pgroupGenreRepository.Get();

foreach (var gg in groupGenres)

{

var group = bgroupRepository.Get(gg.Group.Name);

var genre = bgenreRepository.Get(gg.Genre.Name);

bgroupGenreRepository.Create(group, genre);

}

var groups = pgroupRepository.Get();

foreach (var g in groups)

bgroupRepository.Create(g.Name, g.PersonsCount, g.Popularity);

var recordGenres = precordGenreRepository.Get();

foreach (var rg in recordGenres)

{

var record = brecordRepository.Get(rg.Record.Number);

var genre = bgenreRepository.Get(rg.Genre.Name);

brecordGenreRepository.Create(record, genre);

}

var records = precordRepository.Get();

foreach(var r in records)

{

var room = broomRepository.Get(r.Room.Number);

var group = bgroupRepository.Get(r.Group.Name);

brecordRepository.Create(r.Number, r.Date, r.Hours, room, group);

}

var rooms = proomRepository.Get();

foreach (var r in rooms)

broomRepository.Create(r.Number, r.Square, r.PersonsCount, r.Rent);

database.Dispose();

dbContext.Dispose();

}

}

}

# ЗАКЛЮЧЕНИЕ

В результате работы над лабораторной работой была построена ER-диаграмма организации «Студия звукозаписи». Были выделены основные объекты и представлены связи между ними.

Были описаны технические требования для серверного и клиентского приложения с учетом специфики разработки на языках высокого уровня.

Программа для работы с базами данных PostgreSQL была успешно установлена на ПК.