УДК 004.655.3, 004.042

Бренчук Н. І., ст. магістратури факультету кібернетики, науковий керівник – к.пед.н., доцент Лотюк Ю. Г. (Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука, м. Рівне)

РОБОТА З БАЗОЮ ДАНИХ НА ПЛАТФОРМІ NET НА ПРИКЛАДІ ПОБУДОВИ ПРОЕКТУ

Анотація. *У статті досліджено можливості програмної технології Net для роботи з реляційною базою даних. Наведено реляційну модель та запропоновано концепцію посилань по ключу. Розкрито можливість використання інструментів Object Relational Mapping для відображення таблиць у реляційній базі даних. Розглянуто інструмент ORM Entity Framework для опису моделі даних будь-якою зручною мовою програмування, з подальшою генерацією таблиці у базі даних.*
Ключові слова: *реляційна модель даних, посилання по ключу, Object Relational Mapping, Entity Framework.*

Abstract. *The article explores the capabilities of Net software technology to work with a relational database. The relational model and the concept of the key-link are presented. The use of Object Relational Mapping tools to display tables in a relational database in application classes is disclosed. The ORM Entity Framework tool is described to describe the application data model in any convenient programming language, with subsequent table generation in the database.*
Key words: *relational data model, key link, Object Relational Mapping, Entity Framework.*

Характерною рисою переважної більшості програмних продуктів є використання даних, що зберігаються у відповідних базах. Існує багато типів баз даних: реляційні, нереляційні, об'єктно-орієнтовні, але найбільш популярним типом нині є реляційні бази даних, тому у подальшому у нашій статті будемо розглядати їх [1].

Реляційні бази даним базуються на реляційній моделі [2], яка була розроблена у 1969 році Едгаром Кодом [3]. Реляційна модель пропонує декларативний метод опису даних і запитів. Це досягається описанням домену через набір таблиць (відношень), які містять деякі атрибути. Значення в таблицях називаються записами або кортежами. Обов'язковим елементом таблиці є унікальний ключ, який однозначно ідентифікує запис (кортеж). Він складається з одного або декількох атрибутів таблиці [4].

Одна з найважливіших частин розробки застосування є перенесення домену в таблиці.

Питання актуальності використання баз даних на платформі NET досліджувалося різними науковцями, зокрема: К. Дж. Дейтом [5]: підхід до інтеграції реляційної та об'єктної технологій; М. Р. Когаловським [6]: моделювання даних, концептуальне та онтологічне моделювання, інтеграція інформаційних ресурсів.

Використання баз даних на платформі NET спрошує доступ до SQL бази, таблиці у базі генеруються відповідно до моделі dotnet ef database update а доступ до об'єктів бази здійснюється за допомогою класа, створеного програмістом.

Модель даних в процесі розробки може змінитися і перестане відповідати базі даних. Однак, завжди можна видалити базу даних і EF створить нову версію, яка в точності відповідатиме моделі. Але така процедура призводить до втрати поточних даних. Функція міграції в EF Core дозволяє послідовно застосовувати зміни схеми до бази даних, щоб синхронізувати її з моделлю в додатку без втрати існуючих даних [7].

Такі міграції включають засоби командного рядка і API-інтерфейси, які використовуються у вирішенні таких завдань: створення міграції, оновлення бази даних, налаштування коду міграції, видалення міграції, скасування міграції, створення скриптів SQL, застосування міграції під час виконання.

Розглянемо *приклад* бази даних, у якій зберігаються дані про відвідування певного об'єкту. У застосуванні для обліку відвідування, доменом додатку будуть такі елементи:

– Користувач;
– Адміністратор;
– Відвідування.

Переведемо цей домен у таблиці мовою SQL:

```
CREATE TABLE User (UserId INT,
Name VARCHAR(200),
CreatedOn DATETIMEOFFSET
Role VARCHAR(10)
CREATE TABLE Visit (VisitId Int,
Time DATETIMEOFFSET,
UserName VARCHAR(200)
```

Цей SQL описує дві таблиці – User і Visit. В таблиці Visit зберігається ім'я користувача, який здійснив відвідування. Проте, цей метод не є раціональним.

По-перше: в таблиці User роль визначена як поле VARCHAR, яке є рядком. При додаванні даних, отримуються багато кортежів із однаковим значенням «Admin» або «User». Такого підходу варто уникати, тому що це

ускладнює роботу з базою даних, і відкриває багато можливостей для помилок. Тому потрібно створити ще одну таблицю Role і зв'язати її із user.

По-друге: в таблиці Visit поле UserName теж є рядком. Аналізуючи модель даних, логічно було б зв'язати таблиці Visit і User.

Реляційна модель пропонує для цього концепцію посилань по ключу. SQL таблиця покращеної моделі даних:

CREATE TABLE ROLE (RoleId INT,
RoleName VARCHAR)
CREATE TABLE USERROLE (RoleId Foreign KEY (Role.RoleId),
UserId (FOREIGN KEY (User.UserId)),
CREATE TABLE User (UserId Int,
Name VARCHAR
CREATEDON DATETIMEOFFSET)
CREATE TABLE VISIT (Date DATETIMEOFFSET,
UserId FOREIGN KEY (User.UserId)

Таким чином можна описати модель із чотирьох таблиць:

– User;
– Role;
– UserRole – створює зв'язок «багато до багатьох» між User і Role;
– Visit – із посиланням на User, що створює зв'язок «один до багатьох».

Реляційна модель дуже потужна, але працювати з SQL-кодом не надто зручно, і це сповільнює розробку [8]. Для вирішення цієї проблеми більшість платформ для розробки моделі використання пропонують інструменти для спрощення роботи з SQL, щоб розробники могли сконцентруватись більше на бізнес-логіці і вимогах до використання, ніж на описанні таблиць в SQL. Ці інструменти називаються ORM (Object Relational Mapping) [9].

В. М. Терещенко та С. Г. Волоши запропонували узагальнену схема, в основі якої лежать метадані, що описують класи та їхні атрибути. Атрибути поділяються на два типи: статичні та динамічні. Від їх типу залежить спосіб зберігання та спосіб маніпуляції. Всі класи системи успадковані від єдиного кореня**.** Це дозволяє мати єдиний унікальний нумератор всіх об'єктів та винести всі спільні атрибути в одну таблицю. Для статичних полів створюється таблиця зі зв'язком один до одного між таблицею кореня та нащадками. Значення всіх динамічних атрибутів зберігаються в одній таблиці. Залежно від типу значення динамічного поля дані зберігаються у двох таблицях: таблиця для простих значень і таблиця для значень із посиланням. Це забезпечує цілісність даних на рівні схеми СУБД [6].

Завданням нашого дослідження є створення відображення таблиць у реляційній базі даних у класи використання.

Платформа NET пропонує інструмент ORM, який називається Entity Framework (EF). Entity Framework є продовженням технології Microsoft

ActiveX Data і надає можливість для роботи з базами даних через об'єктно-орієнтований код C#. Цей підхід надає ряд істотних переваг: не потрібно створювати код доступу до даних, не потрібно знати деталей роботи СУБД SQL Server і синтаксису мови запитів SQL, замість цього розробник працює з таблицями бази даних як з класами C#, з полями цих таблиць – як з властивостями класів. Entity Framework перетворює код C# в SQL – інструкції [10].

Працюючи з EF, описуємо модель даних застосування будь-якою зручною мовою програмування та генеруємо за EF таблиці у базі даних.

Для цього створимо новий проект і опишемо модель у класах на C#:

```
public class ModelBase
{ public string Id { get; set; }
public bool IsDeleted { get; set; }
public class Person : ModelBase
{ public string Name { get; set; }
public ICollection<Role> Roles { get; set; }
public class Role : ModelBase
{ public string Name { get; set; }
public virtual ICollection<Person> People { get; set; }
public class Visit
{ public DateTimeOffset VisitedOn { get; set; }
```

Створимо контекст бази даних для моделі даних:

```
public class Context : DbContext
{public Context (DbContextOptions options) : base(options)
{ public Context (string connectionString): base(connectionString) string}
 public DbSet< Person > Accounts { get; set; }
 public DbSet< Role > Bodies { get; set; }
 public DbSet< Visit > Headers { get; set; }
protected override void OnModelCreating(ModelBuilder modelBuilder)
```

Після цього запустимо EF щоб згенерувати таблиці у базі відповідно до моделі dotnet ef database update.

Таким чином був отриманий простий і зручний спосіб доступу до бази даних. Всі запити будуть виконуватися з допомогою класу Context.

Перед початком користування контекстом, його потрібно ініціалізувати:

Var context = new Context(connectionString); де connectionString це рядок опису під'єднання до бази даних. Після цього доступ до об'єктів бази здійснюється за допомогою цього об'єкта наступним чином:

```
context.People.Add(new Person{Name = «Giga, Stepan»});
context.SaveChanges();
var people = context.People.ToList();
```

1. Реляційна база даних. URL: https://en.wikipedia.org/wiki/Relational_database (дата звернення: 07.09.2019). **2.** Реляційна модель для бази даних. URL: https://en.wikipedia.

org/wiki/Relational_model (дата звернення: 07.09.2019). **3.** Кодда Е. Ф. Теорія реляційних баз даних відношення URL: https://en.wikipedia.org/wiki/ Relation_ (database) (дата звернення: 09.09.2019). **4.** Третя нормальна форма в нормалізації в базі даних. URL: https://en.wikipedia.org/wiki/Third_normal_form (дата звернення: 10.09.2019). **5.** Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. 8-е изд. М.: «Вильямс», 2006. С. 1328. **6.** Терещенко В. М., Волошин С. Г. Підхід щодо побудови об'єктно-реляційної бази даних. *Наукові записки НаУКМА. Комп'ютерні науки*. 2010. Т. 112. С. 78–84. **7.** Entity Framework Core. Управление схемами баз даннях. URL: https://docs.microsoft.com/ru-ru/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli (дата звернення: 0.09.2019). **8.** Евсеева О. Н., Шамшев А. Б. Работа с базами данных на языке C#. Технология ADO.NET. учеб. пособие. Ульяновск: УлГТУ, 2017. 170 с. **9.** Начало работы с EF Database First с помощью MVC. URL: https://docs.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/database-first-development/setting-up-database. (дата звернення: 07.09.2019). **10.** Уроки по C# и платформе NET Framework. URL: https://professorweb.ru/ (дата звернення: 07.09.2019).