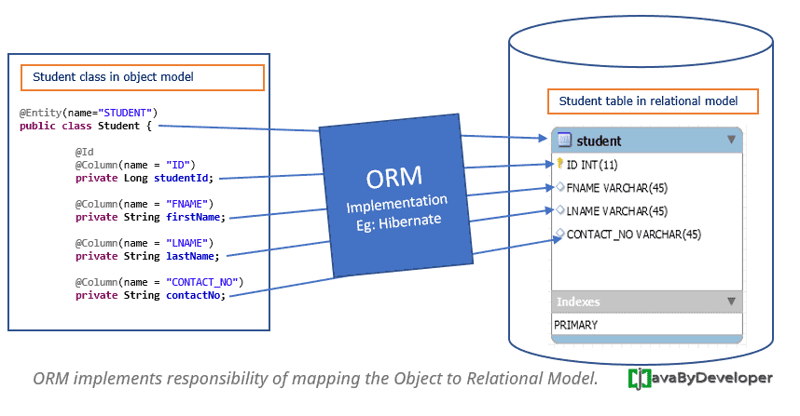
**Mapowanie obiektowo-relacyjne** – technika wykorzystywana podczas programowania, która umożliwia przenoszenie danych z relacyjnych baz danych na obiekty w kodzie programu. Dzięki wykorzystaniu obiektowego programowania można łatwo pracować na relacyjnej bazie danych. Mapowane są obiekty w kodzie na wierszy w tabelach w bazie danych, co umożliwia wykonywanie operacji bez potrzeby pisania zapytań w języku SQL. Dzięki mapowaniu obiektowo-relacyjnym obsługa bazy danych staje się łatwiejsza i łatwiejsza jest integracja zapytań z kodem programu.

Do najważniejszych funkcji ORM można zaliczyć:

* Tworzenie zapytań SQL.
* Tworzenie modeli danych w kodzie, przedstawiających tabele z bazy danych.
* Obsługa relacji między tabelami.



Źródło: <https://media2.dev.to/dynamic/image/width=800%2Cheight=%2Cfit=scale-down%2Cgravity=auto%2Cformat=auto/https%3A%2F%2Fdev-to-uploads.s3.amazonaws.com%2Fuploads%2Farticles%2Fdma6gfcqhfospng9scqe.png>

**Zasady pracy z narzędziami ORM to:**

1. Utworzenie modelu danych w obiektowym języku programowania
2. Utworzenie schematu bazy danych, który odpowiada temu modelowi
3. Zdefiniowanie odwzorowania bazy danych na model relacyjny
4. Stworzenie aplikacji obiektowej, która jest oparta na modelu obiektowym
5. Do operacji pobrania lub usunięcia obiektów z bazy danych używa się odpowiednim API, które jest dostarczane przez narzędzie ORM.

Standardowo przy operacji pobrania informacji o konkretnym użytkowniku z bazy danych zostałoby użyte zapytanie SQL, które mogłoby wyglądać następująco:

SELECT id, name, email, country, phone\_number

FROM users

WHERE id = 1909

Z kolei z zastosowaniem mapowania obiektowo-relacyjnego zamiast kodu SQL wykorzystałoby metodę do pobierania informacji o użytkowniku, która mogłaby wyglądać następująco:

users.GetById(1909)

**Zalety mapowania obiektowo-relacyjnego:**

* Brak potrzeby znajomości języka SQL oraz szczegółów implementowania bazy danych ze względu na to, że komunikacja z nią jest ukryta za obiektowym interfejsem, co sprawia, że kod jest bardziej przenośny pomiędzy różnymi systemami bazodanowymi, które zawierają pomiędzy sobą różnice, ale także czytelniejszy i łatwiejszy do utrzymania kod
* Zwiększone bezpieczeństwo, ponieważ narzędzia ORM mają wbudowane mechanizmy, które chronią przed atakami typu SQL Injection
* Wyższy poziom abstrakcji – szybsze implementowanie nowych funkcjonalności i skupienie się na logice biznesowej. Dzięki temu można przyspieszyć rozwój projektu i obniżyć jego koszty.
* Większość popularnych frameworków mają wbudowane rozwiązania ORM, co jeszcze bardziej upraszcza tworzenie aplikacji wykorzystującą bazę danych. Dzięki temu można pisać mniej kodu niż przy zastosowania języka SQL
* Ułatwiona skalowalność aplikacji w wypadku zwiększenia zasobów serwera, ale także przy dodawaniu kolejnych obiektów serwera

**Wady mapowania obiektowo-relacyjnego:**

* Zbyt duży poziom złożoności - korzystania z narzędzia ORM może być trudne do nauczenia i konfiguracji, co przekłada się na czasochłonność
* Prawdopodobnie nie będą one działać lepiej w przypadku bardzo złożonych zapytań.
* Narzędzia ORM są z reguły wolniejsze niż SQL – w wypadku nadmiernego używania „Lazy Loading” wpłynie to negatywnie na wydajność

Zasady działania mapowania obiektowo-relacyjnego:

* **Identity Map –** zapewnienie, że każdy obiekt jest jednoznacznie identyfikowany w całym systemie
* **Unit of Work –** zmiany w obiektach są zbierane, które są zapisywane w bazie danych w jednej i spójnej transakcji.
* **Lazy Loading – polega na opóźnieniu wczytania danych do momentu, gdy te dane będą rzeczywiście potrzebne**

**Popularne biblioteki mapowania obiektowo-relacyjnego:**

* **Entity Framework – używana w aplikacjach .NET**
* **Django ORM – powszechnie używana w środowisku Python**
* **Sequelize**  - używana w aplikacjach Node.js
* **Hibernate** – popularna w środowisku Javy

Przykład dla **Entity Framework**

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

public class BloggingContext : DbContext

{

public DbSet<Blog> Blogs { get; set; }

public DbSet<Post> Posts { get; set; }

public string DbPath { get; }

public BloggingContext()

{

var folder = Environment.SpecialFolder.LocalApplicationData;

var path = Environment.GetFolderPath(folder);

DbPath = System.IO.Path.Join(path, "blogging.db");

}

}

public class Blog

{

public int BlogId { get; set; }

public string Url { get; set; }

public List<Post> Posts { get; } = new();

}

public class Post

{

public int PostId { get; set; }

public string Title { get; set; }

public string Content { get; set; }

public int BlogId { get; set; }

public Blog Blog { get; set; }

}

**Porównanie ORM z innymi podejściami do pracy z bazami danych**

* **Bezpośrednie korzystanie z zapytań SQL** – oferuje największą kontrolę nad operacjami bazodanowymi, ale także często prowadzi do większej wydajności, gdyż używane zapytania można optymalizować. Wadą tego podejścia jest wymóg bardzo dobrej znajomości języka SQL, co może być źródłem błędów przy złożonych zapytaniach.
* **Query builders**, takie jak Knex.js dla JavaScript - stanowią pośrednią warstwę abstrakcji, dostarczając wygodniejsze API do generowania zapytań SQL, z jednoczesnym zachowaniem większej kontroli niż w przypadku pełnych ORM.

**Analiza porównawcza wydajności czasowej**

Analiza porównawcza wydajności czasowej zapytań bazodanowych w języku C# opisana w artykule [200] pokazuje, że zwykłe zapytania SQL mają najszybszy czas wykonania ze względu na jego prostą implementację. W Prepared Statement, czyli w zapytaniach SQL, w których znajdował się warunek „WHERE”, wykazywał się dłuższym czasem wykonania niż zwykłe zapytania niezawierające żadnego warunku. Entity Framework Core, który pozwala na mapowanie obiektowo-relacyjne, okazał się najwolniejszym rozwiązaniem, którego jednak atutem jest to, że nie wymaga ona znajomości języka SQL.

**Literatura:**

**100. Mapowanie obiektowo-relacyjne ORM - czy tylko dobra idea,** [**Rosiek Z.**](https://yadda.icm.edu.pl/baztech/contributor/cfb4bb4094a7f8da6f38d295925e00a5)**,**

**200. Analiza porównawcza wydajności czasowej zapytań bazodanowych w języku C#,** [**Nowicki Tomasz**](https://yadda.icm.edu.pl/baztech/contributor/c21952cbc02d0c72bdd72fc5ccf7e9f7)**,**[**Tomczak Sebastian**](https://yadda.icm.edu.pl/baztech/contributor/05093dfc104a37ca714979af7619df18)**,**[**Kozieł Grzegorz**](https://yadda.icm.edu.pl/baztech/contributor/f3e488fcf9dec58e0da840365ad63afd)**, Wydawca** [**Wydawnictwo Politechniki Lubelskiej**](https://yadda.icm.edu.pl/baztech/publisher/bwmeta1.institution.baztech-publisher-wydawnictwo_uczelniane_politechniki_lubelskiej) **,rok wydania: 2022**

**Listingi:**