**Выполнение SQL запросов средствами ORM**

FromSql – строка с интерполяцией, в запрос можно подставить переменные

FromSqlRaw – используются нумерованные параметры

// чтение данных: FromSql / FromSqlRaw

//context.НаборДанных.FromSql($"...")

// чтение значения одного столбца: SqlQuery<тип> - применяется для агрегатных функций

//context.Database.SqlQuery<тип>($"...")

// редактирование: ExecuteSql / ExecuteSqlRaw

//context.Database.ExecuteSql<тип>($"...")

// вызов подпрограмм

Если процедура выводит таблицу, для вызова используется FromSql / FromSqlRaw

Если процедура что-то меняет, для вызова используется ExecuteSql / ExecuteSqlRaw

Работа с выходными параметрами как в ADO.Net

SqlParameter id = new()

{

ParameterName = "@id",

SqlDbType = SqlDbType.Int,

Direction = ParameterDirection.Output

};

var categoryName = "categoryy";

context.Database.ExecuteSql($"AddCategory {categoryName}, {id} OUTPUT");

context.Database.ExecuteSql($"AddCategory @name={categoryName}, @id={id} OUTPUT");

Console.WriteLine(id.Value);

// пример вызова скалярной функции

var categoryId = 1;

var avgPrice = context.Database

.SqlQuery<decimal?>($"SELECT dbo.GetAvgPrice({categoryId}) AS value")

.FirstOrDefault();

Console.WriteLine(avgPrice);

// вызов стандартной функции

// EF.Functions.ИмяФункции

games = context.Games

.Where(g => EF.Functions.Like(g.Name, "[a-l]%"));

foreach (var game in games.ToList())

Console.WriteLine($"{game.Name} {game.Price}");

// вызов табличных функций

categoryName = "шутер";

games = context.Games.FromSql($"SELECT \* FROM dbo.GetGamesByCategory({categoryName})");

foreach (var game in games.ToList())

Console.WriteLine($"{game.Name} {game.CategoryId}");

Console.WriteLine();

Можно выполнить Mapping функции пользователя и процедур в контексте БД