УДК 004.4: 004.4

DOi 10.30837/ bi.2023.1(99).01

Mazurova Oksana[1], Ramazanov Rasul[2]

[1] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine,
oksana.mazurova@nure.ua; ORCID ID: https://orcid.org/0000-0003-3715-3476

[2] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine,
rasul.ramazanov@nure.ua; ORCID ID: https://orcid.org/0009-0000-8656-1869

RESEARCH ON TECHNOLOGIES FOR ACCESSING RELATIONAL DATABASES USING MS SQL SERVER

In the modern world, hundreds of large projects are developed every day, and thousands of startups with a wide variety of topics appear, developers start creating their projects, companies modernize old projects. Each of these activities has something in common - the way they store and manipulate data. Almost all modern projects have a database, because it becomes simply impossible to interact with the client without it. Now, no matter what your application, service, game or anything else is, it is necessary for the user to be able to have his account page, save progress, anything that needs to be saved locally or in the cloud or on the server. And a modern developer can use the database directly or through special database access technologies, including through ORM or Micro ORM. The subject is the study of access technologies for working with relational databases. The goal is to compare the efficiency, flexibility and use of various access technologies when working with the MS SQL SERVER database. Task: to investigate access technologies to MS SQL SERVER. Collect the technical characteristics of queries when using different database access technologies. Compare the results. Methods: analysis of MS SQL SERVER access technologies, experimental research, statistical analysis of results. Results: it is shown how access technologies such as ORM Entity Framework, Micro ORM Dapper and ADO.NET differ in use, it is shown that the performance is most effective in ADO.NET, followed by Dapper and in third place among the access technologies used in terms of efficiency is Entity Framework . But it is shown how the type of request contributes to the efficiency of execution by various technologies. Conclusions: ORMs are used in cases where it is necessary to work with a database using an object-oriented approach. ORM transforms data from the database into objects in the application, which facilitates interaction with the database and reduces the amount of code that must be written to work with the database. Micro ORMs are used when the speed of execution of requests to the database is required, or if the project is small in scope and does not need the full functionality of ORM. Micro ORM is smaller, faster and easier to use than ORM. ADO.NET is recommended when direct interaction with databases is needed. ADO.NET allows you to create multithreaded and distributed applications, interact with databases using transactions, and manage data security. It is more extensible and scalable than ORM and Micro ORM, but requires more code to interact with the database.

ACCESS TECHNOLOGY; DATABASE; MS SQL SERVER; ENTITY FRAMEWORK; DAPPER; ADO.NET.

О.О. Мазурова, Р.Ш. Рамазанов. Дослідження технологій доступу до реляційних баз даних під керуванням MS SQL SERVER. У сучасному світі кожного дня розробляються сотні великих проектів, з'являються тисячі стартапів із найрізноманітнішими тематиками, розробляються нові проекти та модернезують старі. Майже всі подібні проекти мають базу даних, бо зберігати та обробляти дані без неї стає просто неможливо. Сучасний розробник має змогу використовувати базу даних через спеціальні технології доступу до них, в тому числі через ORM або Micro ORM. Предметом дослідження є технологій доступу до реляційних баз даних, які займають значимі позиції в програмних рішеннях. Мета роботи – порівняти продуктивність використання найбільш ефективних на сьогодні технологій доступу до бази даних під керуванням MS SQL SERVER та розробити рекомендації, що дозволять розробникам зробити більш ґрунтовний їх вибір. Методи: аналіз технологій доступу до баз даних MS SQL SERVER, експериментальне дослідження, статистичний аналіз результатів. Результати: зібрано вагомі метрики під час експериментального дослідження продуктивності таких технології доступу як ORM Entity Framework, Micro ORM Dapper та ADO.NET; розроблено рекомендації стосовно використання цих технологій та вибору найбільш ефективного варіанту в залежності від видів запитів до баз даних. Висновки: Micro ORM більш рекомендована до використання, коли критичною є швидкість виконання запитів до бази даних, або якщо проект має невеликий обсяг та не потребує повного функціоналу ORM; ADO.NET рекомендовано в разі, коли потрібно пряма взаємодія з базами даних; ADO.NET дозволяє створювати багатопоточні та розподілені додатки, використовувати механізм транзакцій та керувати безпекою даних. Він є більш розширеним та масштабованим, ніж ORM та Micro ORM, але вимагає більше коду для взаємодії з базою даних.

ТЕХНОЛОГІЯ ДОСТУПУ; БАЗА ДАНИХ; MS SQL SERVER; ENTITY FRAMEWORK; DAPPER; ADO.NET.

Introduction

In today's world, the development of almost any software application is closely related to the use of databases (DB) [1, 2]. Regardless of the type of application, service, or game, there is always a need for users to have their own accounts, store progress, or other information locally, in the cloud, or on a server. Therefore, a modern developer must choose and utilize a specific database access technology, including ORM or Micro ORM, when working with databases in their projects.

For developers, the question arises as to which database access technologies are best to use and which ones are most suitable for their system. Traditional analysis of documentation on various database access technologies

and general recommendations provided by developers often do not provide a detailed description of the technical characteristics regarding the usage of these technologies within a specific stack of software tools.

One of the widely adopted solutions for working with relational databases is developing on the .NET platform using the Microsoft SQL Server database management system (DBMS) [3-4]. Currently, the most popular database access technologies in such an environment are Object-Relational Mapping (ORM) tools like Entity Framework, Micro ORM solutions such as Dapper, and the well-established ADO.NET technology.

When designing a system that interacts with SQL databases, it is crucial for the developer to have a clear understanding of the database's logic and the tools offered by the chosen database access technology [1, 5]. Due to limited practical information available about specific software connections, many commercial projects hesitate to adopt new database access technologies, as implementing such a transition requires significant time for performance modeling and data migration [6].

Therefore, a relevant direction for research is to establish clearer, practically validated recommendations for using database access technologies based on experimental measurements and comparisons of significant database performance metrics [7].

1. Analysis of the problem and existing methods

Relational databases have been widely used until now for developing applications that require strong support for ACID transaction properties, such as in the fields of banking, healthcare, and so on. With the emergence of the first technologies for accessing relational databases, many new trends and improvements to existing approaches have appeared [8, 9].

On the .NET platform, there are several ORM frameworks (Object-Relational Mapping) [10-11] that allow for easy and convenient interaction with relational databases. A thorough analysis has been conducted on several of them, namely:

– Entity Framework (EF) is an ORM framework developed by Microsoft for the .NET platform. EF allows developers to work with databases using an object-oriented approach rather than writing SQL queries. It supports various relational databases, including Microsoft SQL Server [12-13], Oracle, and MySQL;

– NHibernate is a popular ORM framework for the .NET platform. It allows developers to work with databases by mapping objects to database tables, making the interaction much simpler;

– Dapper is a lightweight ORM framework developed by StackExchange. It falls under the category of Micro ORM and enables developers to have more precise control over the database interaction process by using simple SQL queries;

– DevExpress XPO is an ORM framework that provides a wide range of functionality for interacting with relational databases, including code generation based on the database schema.

These frameworks enable developers to interact easily and conveniently with relational databases on the .NET platform, while also providing support for various functional capabilities such as data caching, lazy loading, data migrations, and more.

The rapid and widespread adoption of ORM and Micro ORM is driven by their simplicity. They allow programmers to work with databases using familiar objects and programming languages, instead of complex SQL code. ORM enables developers to interact with databases through an object-oriented interface, making the process more intuitive. Additionally, ORM automatically generates SQL code for interacting with the database, making the development process faster and less error-prone.

Micro ORM is a simplified version of an ORM that provides only basic database interaction functionalities. This makes it even simpler and easier to use, resulting in faster execution of database queries and reduced memory consumption.

As a result, the simplicity of working with ORM and Micro ORM [14] allows developers to focus on developing program functionality rather than spending time writing and testing complex SQL queries to the database.

Additionally, the well-established object-oriented technology ADO.NET [15-16], which is part of the .NET Framework, is quite commonly used on the .NET platform. ADO.NET enables developers to establish connections with a database, execute SQL queries, and retrieve query results in the form of a DataSet or DataReader.

During the selection of an access technology, one can rely on research dedicated to the characteristics of access technologies and their architectural features. For instance, in a three-tier architecture, ORM such as Entity Framework resides in the Data Access Layer and serves as a wrapper that communicates with the database and maps data from the database to the data layer model used by the developer. This accelerates development and data manipulation processes.

In addition to the architectural considerations, developers can also rely on analyzing current trends in database development [1, 17]. When considering the .NET platform, the two most popular ORM technologies are Entity Framework and the micro ORM Dapper. Until recently, Dapper and ADO.NET were comparable in terms of performance, but with each new version of .NET developed by Microsoft, the efficiency of development using Entity Framework has been increasing and, in some cases, may surpass Dapper and ADO.NET. The availability of open-source products has fostered a large community of developers dedicated to the advancement of not only Entity Framework but also other ORM and micro

ORM solutions. However, it is unfortunate that some access technologies are not open-source. ORM has become so prevalent that some developers may not be familiar with SQL and rely solely on ORM for writing queries. This can be a significant problem because such developers become heavily dependent on a single ORM and may not recognize alternative options or understand the criteria for selecting more efficient technologies for specific software solutions.

2. Objective of the Work

The purpose of this article is to conduct an experimental investigation of database access technologies on the .NET platform, specifically focusing on a relational database managed by the MS SQL Server DBMS. The goal is to evaluate their productivity and develop practical recommendations for their effective usage in various software projects.

For the experimental research, the most popular representatives of their respective classes have been selected: the ORM Entity Framework, the Micro ORM Dapper, and the object technology ADO.NET.

This research requires the following steps to be conducted: Analysis of the domain-specific application area and designing a database based on it for further experimental research.

– Development of software solutions based on Entity Framework, Dapper, and ADO.NET database access technologies.

– Conducting experimental research on the performance of implemented database access technologies and providing recommendations on the suitability of using these technologies.

The evaluation of the effectiveness of using database access technologies should be conducted considering the following metrics: query execution speed (in milliseconds), query execution speed (in ticks), and consumed resources of the working memory (in bytes).

3. Materials and Methods

The chosen subject area for designing the database for the research is the field of e-commerce. E-commerce, or electronic commerce, refers to the process of buying and selling goods and services over the Internet. It can include online stores, internet auctions, digital goods (such as music and videos), online booking and payment services (such as hotels, airline tickets, etc.), electronic marketplaces, and more. For conducting the experiments, a simplified database [18] for an online clothing store was designed.

A database containing the following basic concepts and their interrelationships has been developed for conducting experiments:

Season: Can be described by attributes such as "name" and "start date" (modeled by the "Seasons" table);

– Catalog: Within a season, there can be multiple catalogs of different categories (the "Catalogs" table);

– Category: Has an attribute "name" (the "Categories" table);

– Product: Can be described by attributes such as name, price, color, description, and category (the "Products" table);

– Good: Represents the relationship between a product and a catalog (the "Goods" table);

– Order: References the user who placed the order and the products included (the "Orders" table);

– User: Can be described by attributes such as name, email, and phone number (the "Users" table).

Classes were designed [16] for the use of Entity Framework and Dapper technologies based on the developed database model. The Code First approach was employed during the creation of software solutions using these technologies. The diagram of the developed classes can be seen in Fig. 1.

The solutions for the research were developed as web applications using ASP.NET Core Web API. In general, the architecture of ASP.NET Core Web API allows for the development of fast, scalable, and reliable web services that can be integrated into various applications and platforms.

Experimental research planning was conducted, and queries were developed as the basis for measuring performance metrics and investigating the productivity of access technologies.

The following queries were developed and used for conducting series of experiments:

– GetUsers query: retrieves all fields from the Users table.

– GetUserWithOrders query: retrieves data that requires joining the Users, Orders, Products, Goods, Catalogs, and Seasons tables (see Fig. 2).

– GetSeasonsQuery: retrieves the count of products present in a season; this query utilizes join operations between tables and the aggregate function Count() (see Fig. 3).

– ultiple aggregate functions.

– CreateCategory query: creates a new category with a specified name.

– DeleteCategory query: deletes a category with a specified name.

4. Research results and their discussion

Let's consider the results of executing queries for the investigated technologies: Entity Framework, Dapper, and ADO.NET. MS SQL Server was used as the database management system, and the tables in the database contained 1000 and 10,000 records, respectively.

For the purpose of conducting a pure research study, caching was disabled for Entity Framework. Caching allows Entity Framework to execute the same query in a very short time, which is one of the advantages of an ORM. By disabling caching, we can observe the actual performance of Entity Framework without the influence of cached results.

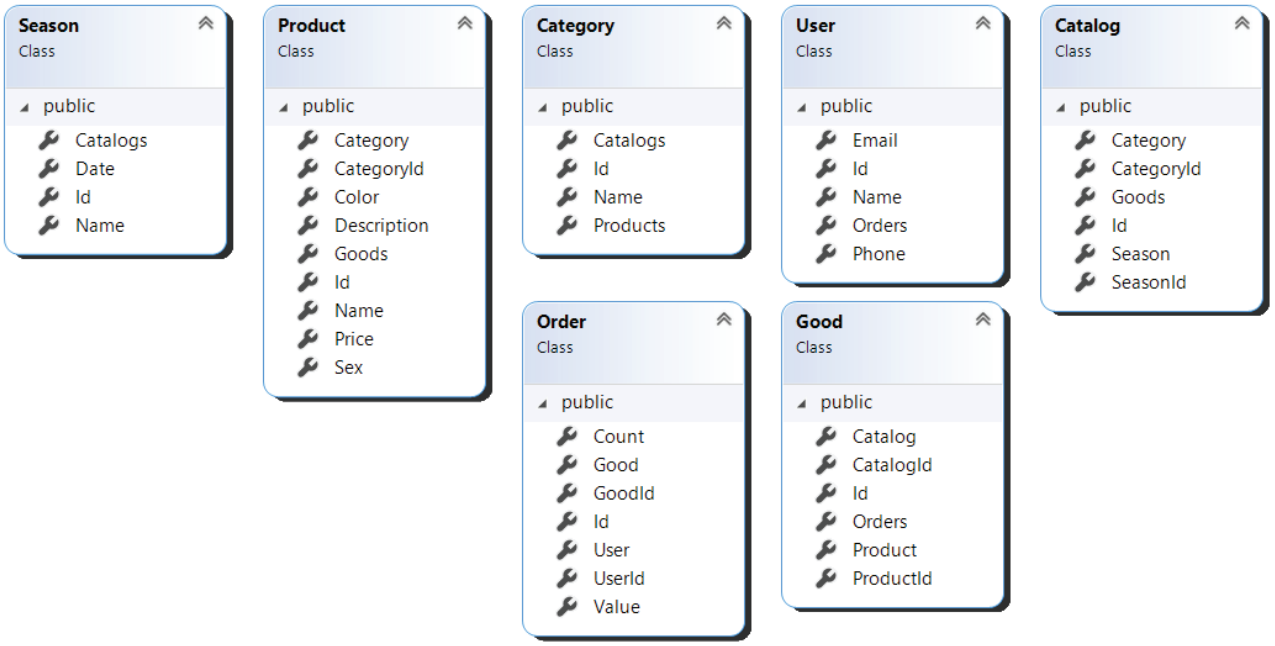

Fig. 1. Class diagra

```
SELECT users.Name as UserName, products.Id as ProductId,
               products.Name as ProductName, orders.Count, seasons.Date

               FROM Users as users,
               Orders as orders,
               Products as products,
               Goods as goods,
               Catalogs as catalogs,
               Seasons as seasons

               WHERE orders.UserId = users.Id
               AND orders.GoodId = goods.Id
               AND goods.ProductId = products.Id
               AND goods.CatalogId = catalogs.Id
               AND catalogs.SeasonId = seasons.Id
               AND seasons.Date < GETDATE()

               GROUP BY users.Name, products.Id, products.Name, orders.Count, seasons.Date
```

Fig. 2. Query GetUserWithOrders

```
SELECT seasons.Name AS SeasonName, seasons.Date, catalogs.Id AS CatalogId, COUNT(goods.CatalogId) AS GoodsCount
          FROM Seasons AS seasons
          JOIN Catalogs AS catalogs ON catalogs.SeasonId = seasons.Id
          JOIN Goods AS goods ON goods.CatalogId = catalogs.Id
          JOIN Products AS products ON products.Id = goods.ProductId
          GROUP BY seasons.Name, seasons.Date, catalogs.Id
```

Fig. 3. Query GetSeasonsQuerry

In the research, it was also decided to measure the performance metrics of queries for Dapper and ADO. NET, taking into account the resources spent on database connection and excluding the connection time. This is reasonable because, in the case of Dapper and ADO. NET, the database connection is established explicitly by the developer when executing queries. Therefore, the measurement of query performance includes the time required for establishing the database connection.

Let's also note that the first query to the database takes

longer to execute compared to subsequent queries because there may be additional tasks to perform before executing the query, such as query plan preparation and query optimization. Query plan preparation is the process of creating an execution plan that outlines the necessary steps to execute the query. It involves selecting the access method for database tables, determining the join method for table joins, and deciding the order of operations in the query. Query optimization is the process of improving query performance by modifying the query or utilizing additional indexes to speed up data retrieval and selection. The execution speed of the first query depends on the MS SQL Server and the time it takes to establish a connection to the database. The average query execution time for the server and database residing in the same physical space depends on the hardware and was computed based on 1000 runs in a series of experiments. Considering the consumed resources is important to understand how much memory is required for executing a specific query and to properly configure the database server.

The performance evaluation of the queries was conducted taking into account the following metrics:

– speed of execution of the first launch (ticks);
– speed of execution of the first launch (ms);
– average request execution time (ticks);
– average request execution time (ms);
– spent RAM resources (bytes) when executing the request.

Fig. 4 shows the results of a series of experiments on the GetUsers request. We see that the average execution time is equal to one for all access technologies, taking into account the execution of the request with and without connection to the database server. This means that the request is running too fast to be measured in ms, which shows us how beneficial it is to use more precise units of measurement like ticks.

Request	GetUsers				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	2005217	3851348	3392629	3216040	206956
First run (ms)	182	400	352	321	20
Average execution time (ticks)	5623	5806	6181	4702	3209
Average execution time (ms)	1	1	1	1	1
Spent memory (bytes)	8512	4571	4384	4544	704

Fig. 4. Results of experiments with the request GetUsers

The first query run takes significantly longer than all other queries. If we take the ratio of the speed of the first request in ticks, then in ADO.NET w/o connection the execution speed is the best. This means that the server took less time to process the first request in ADO.NET w/o connection than for other access technologies. Also, the experiment made it possible to observe significantly lower costs in ADO.NET w/o connection for the first start in ms.

If we look at the average execution time in ticks, we will see that it will be less for ADO.NET and ADO.NET w/o connection than for Entity Framework. Dapper by this metric remains the longest running.

In terms of memory consumption, ADO.NET w/o connection consumed significantly less resources, namely 704 bytes. Then comes Dapper w/o connection, followed by ADO.NET and Dapper, and Entity Framework required the most resources, almost twice as much as ADO.NET and Dapper.

Fig. 5 shows the results of experiments with GetUsersWithOrders queries. Execution of this request is also faster than can be measured in ms. Let's look at the measurements in ticks. The first run shows the same ratios as the GetUsers query, namely ADO.NET w/o connection is the fastest, followed by Entity Framework and ADO.NET, followed by Dapper w/o connection and Dapper. The same behavior for the first run can be seen in the execution speed in ms. If you look at the average query execution time in ticks, then ADO.NET w/o connection is in first place, followed by ADO.NET, Dapper w/o connection and Dapper, and Entity Framework took the longest time to execute this query. According to the consumed memory resources, it can be found that ADO.NET with and without connection requires much less memory than Dapper, and Entity Framework consumes the most of this resource.

Request	GetUserWithOrders				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	2555442	4138181	4002818	3283229	244381
First run (ms)	260	391	352	346	27
Average execution time (ticks)	15413	12814	11782	11867	8633
Average execution time (ms)	1	1	1	1	1
Spent memory (bytes)	12496	10552	10329	1731	1324

Fig. 5. Results of experiments with the request GetUsersWithOrders

Fig. 6 shows the results of experiments with the GetSeasonsQuerry query. The speed of execution of the first run is similar to the GetUsers and GetUsersWithOrders queries. But according to the average request execution time, it can be emphasized that ADO.NET w/o connection and Dapper w/o connection are executed the fastest, followed by ADO.NET and Dapper, and Entity Framework, which executes in almost 9 times longer than ADO.NET w/o connection. According to the used resources, we can also say that ADO.NET and Dapper use 3 times less memory resources than Entity Framework.

Request	GetSeasonsQuerry				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	2510343	4273887	3499499	3447487	246980
First run (ms)	248	414	355	351	40
Average execution time (ticks)	44839	8590	6888	7092	5319
Average execution time (ms)	1	1	1	1	1
Spent memory (bytes)	5048	1720	1688	1688	896

Fig. 6. Results of experiments with the request GetSeasonsQuerry

Fig. 7 shows the results of experiments with the GetSeasonsTotalPrice query. In terms of the speed of the first run, the situation is similar to the previous requests, judging by the metrics in ticks and ms. In terms of average execution time, ADO.NET w/o connection and ADO.NET lead, followed by Dapper w/o connection and Dapper, and the Entity Framework query took the longest, which lagged behind ADO.NET w/o connection by almost 2 times.

Request	GetSeasonTotalPrice				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	2983407	4065444	3811651	4184960	199574
First run (ms)	281	441	359	349	27
Average execution time (ticks)	13621	9117	9328	8586	7400
Average execution time (ms)	1	1	1	1	1
Spent memory (bytes)	9992	3022	2978	2764	1586

Fig. 7. Results of experiments with the request GetSeasonsTotalPrice

Fig. 8 shows the results of experiments with the CreateCategory request. Technologies can be divided according to the execution time of the first launch as follows: ADO.NET w/o connection, Entity Framework, ADO.NET, Dapper w/o connection and Dapper. According to the average execution time in ms, you can see that Entity Framework lags behind significantly. A more revealing ratio can be obtained by comparing the average execution time in ticks, where it can be seen that ADO.NET w/o connection is 2.6 times faster than Entity Framework. According to the used memory, similar to previous requests, Entity Framework uses 3 times more resources than other access technologies.

Request	CreateCategory				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	1592580	3900297	3601395	3294704	214774
First run (ms)	161	421	329	313	16
Average execution time (ticks)	26597	11690	11098	10710	9817
Average execution time (ms)	2.5	1.5	1	1	1
Spent memory (bytes)	7664	2554	2474	2468	1238

Fig. 8. Results of experiments with the request CreateCategory

Fig. 9 shows the results of experiments with DeleteCategory queries. By the time of execution of the first run, the same sequence of technologies is preserved, as in the previous requests. But in terms of average execution time, Entity Framework is now the fastest: 5 times faster than Dapper and 4 times faster than ADO.NET. This can be seen by the average execution time in ticks and ms. In terms of memory usage, everything is similar to the previous queries, where Entity Framework uses many times more memory than the other investigated technologies.

Request	DeleteCategory				
	Entity Framework	Dapper	Dapper W/O connection	ADO.NET	ADO.NET W/O connection
First run (ticks)	2949541	3714834	3741236	3655405	196057
First run (ms)	273	375	318	329	17
Average execution time (ticks)	15092	77176	73560	71484	63049
Average execution time (ms)	1.3	7	6.9	6.4	6.7
Spent memory (bytes)	9064	2288	1934	936	772

Fig. 9. Results of experiments with the request DeleteCategory

Based on the analysis of the results of the experiments and taking into account the provided functionality, separate recommendations can be formulated for each of the technologies.

ADO.NET technology has shown itself to be very resource-efficient compared to the ORM and Micro ORM studied. The execution of the request is almost the fastest for almost all types of requests. Certain problems can arise only due to the incorrect writing of queries in the SQL language. That is, the use of this technology requires the developer to have some experience with SQL. Other complications may arise when it is necessary to take the data returned from the database, because there is no internal automapping in ADO.NET.

Dapper technology showed itself very well in the speed of execution of requests, where there was almost no lag behind ADO.NET. This Micro ORM outperformed Entity Framework by several times. You can see from the memory consumption why Dapper is considered a lightweight Micro ORM. It consumes resources in almost the same way as ADO.NET. As with ADO.NET, Dapper requires SQL knowledge to use, but it has no problem with automapping extracted data. This is a very useful tool that even allows you to populate fields in custom classes from Dapper extracted data belonging to other classes.

Entity Framework is a very popular ORM and it is fully confirmed by experiments why this is so. Thanks to the internal functionality, you can use different development approaches, such as Code First, DataBase and Model First, perform database migrations, easily write queries that the ORM will automatically send to the DB server. But there are disadvantages to this. As you can see, the more complex the query, the longer it took to use Entity Framework compared to other access technologies. But when the queries are simple, Entity Framework took less time than Dapper.

5. Conclusions

In the work were investigated such database access technologies as ORM Entity Framework, Micro ORM Dapper and ADO.NET from the point of view of performance when working with the popular RDBMS MS SQL Server. A series of experiments was conducted to measure the performance of database queries.

To conduct the research, a software solution was developed using the .NET platform, C# 7, ASP.NET Core Web API, Swagger. To conduct experiments, a relational database in the field of e-commerce and a set of requests for performing CRUD operations were designed, the performance of which was investigated.

During the experiments, metrics were used regarding the speed of the first request and the average speed in milliseconds and ticks, the amount of memory spent on the request (byte).

The research showed that none of the technologies used can be called unequivocally the best. Based on the results of the experiments and taking into account the features of the functionality, we can conclude that if the development of a simple application is planned or it is necessary to speed up the execution of requests to the database as much as possible, it is better to use Micro ORM Dapper. If a large and complex program is being created, and at the same time it is planned to use an object-oriented approach, then ORM Entity Framework may be the best choice. ADO.NET is more efficient for executing complex queries, especially if they require optimization or use special database functions, ADO.NET works at a lower level of abstraction compared to Dapper and Entity Framework and provides direct control over transaction management, i.e. the ability to manually start, commit or cancel transactions, which gives you more flexibility and control over this process.

REFERENCES

[1] Filatov, V., & Semenets, V. (2018). Methods for Synthesis of Relational Data Model in Information Systems Reengineering Problems. In 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). IEEE.

[2] Maran, V., Machado, A., Machado, G. M., Augustin, I., de Oliveira, J. P. M. (2018), "Domain content querying using ontology-based context-awareness in information systems", Data and Knowledge Engineering, No. 115, P. 152–173. DOI: 10.1016/j.datak.2018.03.003.

[3] Michael Lee, Gentry Bieker: SQL Server 2008. DOI: https://doi.org/10.1002/9781118257388.ch17

[4] Christian Nagel Professional C# 7 and .NET Core 2.0 DOI: https://doi.org/10.1002/9781119549147.ch31

[5] Pérez-Castillo, R., De Guzmán, I. G. R., Caivano, D., Piattini, M. (2012), "Database schema elicitation to modernize relational databases", ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems, P. 126–132.

[6] Maran M. M., Paniavin N. A., Poliushkin I. A. Alternative Approaches to Data Storing and Processing. V International Conference on Information Technologies in Engineering Education (Inforino). 2020. P. 1–4, DOI: https://doi.org/10.1109/inforino48376.2020.9111708

[7] Renée M. P. Teate SQL for Data Scientists: A Beginner's Guide for Building Datasets for Analysis. DOI: https://doi.org/10.1002/9781119669388.ch1

[8] Filatov, V., Radchenko, V. (2015), "Reengineering relational database on analysis functional dependent attribute", Proceedings of the X Intern. Scient. and Techn. Conf. "Computer Science & Information Technologies" (CSIT'2015), 14-17 sept. 2015, Lviv, Ukraine, P. 85–88.

[9] Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., Ismaili, F., (2018), "Comparison between relational and NOSQL databases", 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), P. 216-221. DOI: https://doi.org/10.23919/mipro.2018.8400041

[10] Ying Bai SQL Server Database Programming with Visual Basic.NET: Concepts, Designs and Implementations. DOI: https://doi.org/10.1002/9781119608493.ch3

[11] Ying Bai Oracle Database Programming with Visual Basic. NET: Concepts, Designs, and Implementations. DOI: https://doi.org/10.1002/9781119734529.ch3

[12] Itzik Ben-Gan. Microsoft SQL Server 2012 T-SQL Fundamentals - Microsoft Press, 1st edition July 15, 2012.- 442 c.

[13] Christian Nagel Entity Framework Core. DOI: https://doi.org/10.1002/9781119549147.ch26

[14] Riadh Ghlala Analytic SQL in SQL Server 2014/2016. DOI: https://doi.org/10.1002/9781119649540.ch1

[15] Ying Bai Practical Database Programming with Visual C# .NET DOI: https://doi.org/10.1002/9780470567845.ch5

[16] Jonathan Eckstein, Bonnie R. Schultz Introductory Relational Database Design for Business, with Microsoft Access. DOI: https://doi.org/10.1002/9781119430087.ch4

[17] Paulraj Ponniah Ph.D. Database Design and Development: An Essential Guide for IT Professionals. DOI: https://doi.org/10.1002/0471728993.ch1

[18] Bagui, S., Earp, R. (2011), Database Design Using Entity-Relationship Diagrams (Foundations of Database Design), Auerbach Publications, 371 P., ISBN 978-143-986-177-6. DOI: https://doi.org/10.1201/9781439861776

The article was delivered to editorial stuff on the 15.02.2023