УДК 004.65
DOI https://doi.org/10.32782/2663-5941/2023.2.1/28

Лісовець С.М.
Таврійський національний університет імені В.І. Вернадського

Ківа І.Л.
Таврійський національний університет імені В.І. Вернадського

Гуйда О.Г.
Таврійський національний університет імені В.І. Вернадського

Вишемірська Я.С.
Таврійський національний університет імені В.І. Вернадського

ОРГАНІЗАЦІЯ ДОСТУПУ ДО ДАНИХ В SCADA-СИСТЕМАХ ЗА ДОПОМОГОЮ MICROSOFT SQL SERVER

Сучасні системи керування різними технологічними об'єктами, операціями і процесами, особливо достатньо складними, практично завжди потребують зовнішнього контролю. Для здійснення такого контролю є прекрасно працюючий «механізм» у вигляді SCADA-систем. Але у деяких випадках, коли використовується незначна кількість технологічних параметрів та один-два різні інтерфейси / протоколи, існуючі SCADA-системи відомих виробників можуть бути надлишковими, так як не всі їх можливості можуть бути потрібні. Як один з варіантів створення власних SCADA-систем (під задачі певного невеликого виробництва) є використання мови програмування C#. Перевагою такого підходу є те, що в результаті можна отримати SCADA-систему саме під свої виробничі задачі. Однією з функцій SCADA-систем є робота з даними, які отримуються під час виробництва (вхідні сигнали від датчиків, вихідні сигнали на виконавчі механізми і регулюючі органи, сигнали про порушення режимів роботи тощо). Такі дані постійно змінюються, часто їх необхідно надійно зберігати і мати можливість отримати до них доступ в будь-який момент часу. В проведеному досліджені було показано, що для доступу до даних в процесі виробництва зі SCADA-системи можна використати Microsoft SQL Server в якості сервера SQL-баз даних, а також код невеликого розміру на мові програмування C#, який дозволяє виконувати потрібні команди Transact-SQL. Перевага такого доступу полягає в тому, що C#, як сучасна мова програмування з багатьма можливостями, має вбудовані засоби доступу до SQL-баз даних, і основною задачею розробника в такому випадку є їх ефективне використання. До таких засобів відносяться, зокрема, вбудовані в C# класи SqlConnectionStringBuilder, SqlConnection, SqlCommand і деякі інші. Вони дозволяють, використовуючи методи ExecuteNonQuery(), ExecuteScalar() і/або ExecuteReader(), виконати практично будь-яку команду Transact-SQL.

Ключові слова: *база даних, клас, об'єкт керування, промислова автоматизація, результуючий набір, технологічний параметр.*

Постановка проблеми. SCADA-системи (абр. від англ. Supervisory Control and Data Acquisition, оперативне керування і збір даних) – це пакети програм, які призначені для розробки і/або забезпечення роботи систем збору, обробки, відображення і архівування даних про об'єкт керування (об'єкт автоматизації) [1]. Вони є частиною систем промислової автоматизації та можуть використовуватися практично в усіх галузях промисловості. SCADA-системи можуть бути як самостійними системами, так і входити до складу інших систем (автоматизованих систем керування технологічними процесами/підприємствами, систем автоматизації будівель «розумний дім», систем моніторингу екологічного стану довкілля і так далі). Одним з «недоліків» SCADA-систем, які пропонують різні відомі виробники, є їхня надлишковість – для взаємодії з об'єктом керування не потрібні всі можливості, які такі системи мають (відповідно, вартість таких систем часто є дуже високою). Тому при невеликій складності об'єкта керування іноді доцільно SCADA-системи розробляти самостійно. Одна з необхідних складових таких SCADA-систем – база даних, в якій можуть зберігатися дані про, наприклад, різні технологічні параметри. До систем керування

такими базами даних висуваються кілька вимог. По-перше, часто необхідно в «реальному часі» мати доступ до значень кількох десятків і сотень технологічних параметрів; по друге, необхідно забезпечити високу надійність збереження таких даних. Такими системами можуть бути системи, які використовують мову структурованих запитів Structured Query Language (SQL).

Аналіз останніх досліджень і публікацій. SQL-бази даних є надійним засобом для зберігання даних різного призначення і розміру, які можуть бути представлені в різних форматах [2]. Для доступу до таких SQL-баз використовуються системи керування базами даних різних розробників, найбільш відомими з яких є IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL і деякі інші. Серед них, зокрема, однією з провідних систем керування базами даних є Microsoft SQL Server.

Із Microsoft SQL Server тісно інтегрована програма Microsoft SQL Server Management Studio, яка дозволяє швидко і дуже зручно створювати бази даних, таблиці, курсори, функції і інші елементи баз даних. Але для доступу до баз даних зі сторони клієнтів така програма не дуже підходить через неможливість створення зручного інтерфейсу з клієнтами, кожний з яких повинен вирішувати саме свої специфічні задачі.

Одним із зручних засобів для доступу до SQL-баз даних, зокрема, створених з використанням Microsoft SQL Server, є мова програмування C# [3, 4]. По-перше, вона підтримується тією ж самою Microsoft. По-друге, завдяки технології .NET можна організувати швидкий і надійний доступ до таких баз. Така швидкість і надійність досягається за рахунок того, що мова програмування C# утримує кілька вбудованих класів, призначених для роботи саме з SQL-базами даних. Зокрема, клас SqlConnectionStringBuilder дозволяє створювати рядки підключень до SQL-баз даних і керувати їх умістом, клас SqlConnection – підключатися до SQL-баз даних, клас SqlCommand – виконувати команди Transact-SQL. Задачею розробника засобів доступу до SQL-баз даних в такому випадку є коректне використання таких класів і створення відповідної графічної оболонки (для чого в C# є всі необхідні засоби програмування).

Аналіз останніх досліджень і публікацій показує, що у випадках, коли від комп'ютерної системи не вимагається досягнення максимальної швидкодії (в цьому випадку рекомендується використовувати, зокрема, мови програмування C або C++), мова програмування C# є ефективним засобом отримання програмного коду невеликого розміру.

Постановка завдання. Метою дослідження, результати якого обговорюються в статті, є показати у вигляді невеликого прикладу програмного коду на мові програмування C# як можна організувати, використовуючи наведені вище класи і додатково кілька інших класів, ефективний доступ до SQL-баз даних загального призначення. Аналогічний підхід може бути використаний, зокрема, і при розробці SCADA-систем для зберігання інформації про результати вимірювання, формування керувальних впливів, відхилення від нормальних режимів роботи тощо.

Виклад основного матеріалу дослідження. Сама SQL-база даних створювалася в Microsoft SQL Server Management Studio, а програмний код для доступу до неї – в середовищі Visual Studio 2019. Для початку роботи з SQL-базою даних необхідно отримати до неї доступ (тобто підключитися до неї). Рядок підключення до такої бази даних, в залежності від версії Microsoft SQL Server і інших чинників, може бути різним. Він представляє собою кілька виразів виду «Параметр=Значення», які відокремлені один від одного символом ";". Звичайно використовуються параметри "Data Source" (посилання на Microsoft SQL Server), "Initial Catalog" (посилання на базу даних) і "IntegratedSecurity" (якщо для аутентифікації використовується обліковий запис Windows, то такий параметр повинен мати значення true).

Розглянемо, в якості прикладу, базу даних Electronic_Components, створену в Microsoft SQL Server Management Studio і яка вже утримує в таблиці Resistors вісім записів з інформацією про такі електроні компоненти, як резистори (див. рис. 1).

	Type_Of_Resistors	Nominal_Resistance	Nominal_Power	Number_Of_Resistors
1	МЛТ	100	0,25	20
2	МЛТ	120	0,5	90
3	МЛТ	150	1	100
4	МЛТ	180	2	50
5	МЛТ	220	0,25	120
6	МЛТ	270	0,5	110
7	МЛТ	330	1	70
8	МЛТ	390	2	10

Рис. 1. Таблиця з інформацією про резистори

Наведений нижче фрагмент коду дозволяє створити рядок підключення до такої бази даних (вже в Visual Studio 2019).

```
SqlConnectionStringBuilder _SqlConnectionStringBuilder_ =new SqlConnectionStringBuilder();
_SqlConnectionStringBuilder_.DataSource = @".\SQLEXPRESS";
_SqlConnectionStringBuilder_.InitialCatalog = "Electronic_Components";
_SqlConnectionStringBuilder_.IntegratedSecurity = true;
```

Після його виконання властивість ConnectionString екземпляра класу _SqlConnectionStringBuilder_ буде мати рядок підключення, повністю готовий до використання. Зокрема, для відкриття підключення використовується метод Open() класу SqlConnection, для закриття – метод Close(). Для того, щоб гарантовано закрити підключення до SQL-бази даних, рекомендується відкривати таке підключення в блоці using – таким чином, при виході за межі такого блоку підключення буде гарантовано закрите. Наведений нижче фрагмент коду дозволяє створити, відкрити (і в подальшому закрити) підключення до такої бази даних.

```
using (SqlConnection _SqlConnection_ = new SqlConnection(_SqlConnectionStringBuilder_.ConnectionString))
    {
        _SqlConnection_.Open();
        // Команди Transact-SQL
    }
```

Для виконання команд Transact-SQL передбачається використання класу SqlCommand, який дозволяє виконати за один раз тільки одну таку команду. При цьому властивість CommandText задає команду Transact-SQL, а властивість Connection – підключення до SQL-бази даних. Якщо потрібно виконання команди Transact-SQL без отримання результуючого набору (наприклад, INSERT), використовується метод ExecuteNonQuery() класу SqlCommand, з результуючим набором у вигляді скаляра (наприклад, SELECT разом з Sum) – ExecuteScalar(), з результуючим набором у вигляді таблиці (наприклад, SELECT) – ExecuteReader().

Наприклад, виконання наведеного нижче фрагмента коду (заміщує собою рядок // Команди Transact-SQL) дозволяє додати в таблицю Resistors чотири записи (див. рис. 2).

```
using (SqlCommand _SqlCommand_ = new SqlCommand())
    {
    _SqlCommand_.Connection = _SqlConnection_;
    _SqlCommand_.CommandType = CommandType.Text;
    _SqlCommand_.CommandText = "INSERT Resistors VALUES ( N'МЛТ', 470.0, 0.25, 80) ;";
    _SqlCommand_.ExecuteNonQuery();
    _SqlCommand_.CommandText = "INSERT Resistors VALUES ( N'МЛТ', 560.0, 0.5, 30) ;";
    _SqlCommand_.ExecuteNonQuery();
    _SqlCommand_.CommandText = "INSERT Resistors VALUES ( N'МЛТ', 680.0, 1.0, 60) ;";
    _SqlCommand_.ExecuteNonQuery();
    _SqlCommand_.CommandText = "INSERT Resistors VALUES ( N'МЛТ', 820.0, 2.0, 40) ;";
    _SqlCommand_.ExecuteNonQuery();
    }
```

	Type_Of_Resistors	Nominal_Resistance	Nominal_Power	Number_Of_Resistors
1	МЛТ	100	0,25	20
2	МЛТ	120	0,5	90
3	МЛТ	150	1	100
4	МЛТ	180	2	50
5	МЛТ	220	0,25	120
6	МЛТ	270	0,5	110
7	МЛТ	330	1	70
8	МЛТ	390	2	10
9	МЛТ	470	0,25	80
10	МЛТ	560	0,5	30
11	МЛТ	680	1	60
12	МЛТ	820	2	40

Рис. 2. Таблиця з доданою інформацією про чотири резистори

Виконання іншого наведеного нижче фрагмента коду дозволяє визначити загальну кількість резисторів в таблиці Resistors, яка дорівнює 780 (див. рис. 3).

```
using (SqlCommand _SqlCommand_ = new SqlCommand())
    {
    _SqlCommand_.Connection = _SqlConnection_;
    _SqlCommand_.CommandType = CommandType.Text;
    _SqlCommand_.CommandText = "SELECT SUM(Number_Of_Resistors) FROM Resistors ;";
    Int32 Total_Number_Of_Resistors = (Int32) _SqlCommand_.ExecuteScalar();

    Console.WriteLine(Convert.ToString(Total_Number_Of_Resistors));
    }
```

```
780
```

Рис. 3. Результат визначення загальної кількості резисторів

А виконання ще одного наведеного нижче фрагмента коду дозволяє отримати результуючий набір із записів, для яких номінальний опір резисторів більше 300 Ом (див. рис. 4).

```
using (SqlCommand _SqlCommand_ = new SqlCommand())
    {
    _SqlCommand_.Connection = _SqlConnection_;
    _SqlCommand_.CommandType = CommandType.Text;
    _SqlCommand_.CommandText = "SELECT * FROM Resistors WHERE Nominal_Resistance > 300 ;";
```

```
    using(SqlDataReader _SqlDataReader_ = _
SqlCommand_.ExecuteReader())
    {
    while (_SqlDataReader_.Read())
    {
    Console.WriteLine(String.Format("{0}
{1:f0}  {2:f2}  {3:n0}", _SqlDataReader_[0],
_SqlDataReader_[1], _SqlDataReader_[2], _
SqlDataReader_[3]));
    }}}
```

Наведені вище фрагменти програмного коду дозволяють створювати програмний код у вигляді консольної програми. Але ті ж самі класи можна використати і при створенні, наприклад, програми Windows Forms або WPF.

Висновки. В статті показано, що, використовуючи відповідні класи мови програмування C#, можна достатньо легко організувати надійний доступ до SQL-баз даних, який може бути використаний при створенні власних SCADA-систем.

```
МЛТ        330  1,00  70
МЛТ        390  2,00  10
МЛТ        470  0,25  80
МЛТ        560  0,50  30
МЛТ        680  1,00  60
МЛТ        820  2,00  40
```

Рис. 4. Результат визначення резисторів з номінальним опором більше 300 Ом

Список літератури:

1. Пупена О.М. Розроблення людино-машинних інтерфейсів та систем збирання даних з використанням програмних засобів SCADA/HMI: Навч. посіб. – Київ: Видавництво Ліра-К, 2020. – 594 с.

2. Гайдаржи В.І. Бази даних в інформаційних системах: Підручник / В.І. Гайдаржи, Ізварін І.В. – Київ: Видавництво Університет «Україна», 2018. – 418 с.

3. Standard ECMA-334. C# Language Specification. 5th Edition. – ECMA International. – December 2017.

4. Кравець П.О. Об'єктно-орієнтоване програмування: Навчальний посібник / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.

Lisovets S.M., Kiva I.L., Guida O.G., Vyshemirska Ya.S. ORGANIZING ACCESS TO DATA IN SCADA SYSTEMS USING MICROSOFT SQL SERVER

Modern control systems for various technological objects, operations and processes, especially sufficiently complex ones, almost always require external control. There is a perfectly working «mechanism» in the form of SCADA systems for such control. But in some cases, when a small number of technological parameters and one or two different interfaces / protocols are used, existing SCADA systems from well-known manufacturers may be redundant, as not all their capabilities may be needed. As one of the options for creating your own SCADA systems (under the task of a certain small production) is the use of the C# programming language. The advantage of this approach is that as a result, you can get a SCADA system specifically for your production tasks. One of the functions of SCADA systems is to work with data received during production (input signals from sensors, output signals to executive mechanisms and regulatory bodies, signals about violations of operating modes, etc.). Such data is constantly changing, often it is necessary to store it securely and be able to access it at any time. In the conducted research, it was shown that to access data in the production process from the SCADA system, you can use Microsoft SQL Server as a SQL database server, as well as a small code in the C# programming language that allows you to execute the necessary Transact-SQL commands. The advantage of such access is that C#, as a modern programming language with many features, has built-in means of accessing SQL databases, and the main task of the developer in this case is to use them effectively. These tools include, in particular, the built-in C# classes SqlConnectionStringBuilder, SqlConnection, SqlCommand and some others. They allow you to execute almost any Transact-SQL command using the ExecuteNonQuery(), ExecuteScalar(), and/or ExecuteReader() methods.

Key words: *database, class, control object, industrial automation, result set, technological parameter.*