Michał GRUSZCZYŃSKI, Przemysław IDZIASZEK, Wojciech MUELLER, Jerzy WERES, Janina RUDOWICZ-NAWROCKA, Karolina GÓRNA
Uniwersytet Przyrodniczy w Poznaniu, Wydział Rolnictwa i Bioinżynierii, Instytut Inżynierii Biosystemów
ul. Wojska Polskiego 27, 60-637 Poznań, Poland
e-mail: muellerw@up.poznan.pl

ASSOCIATION OF VECTOR DATA USED FOR TECHNICAL DOCUMENTATION WITH INFORMATION EMBEDDED IN GRAPH DATABASES

Summary

The paper analyzes the possibilities of further and fuller use of geometric models of empirical systems, including agriculture, created at the level of AutoCAD. The analysis focuses primarily on opportunities for better mapping of existing links between graphic elements, unreadable or impossible to present at the level of digital technical documentation in the form of a graph structures offered by Neo4j. Information technologies, recognized and presented as necessary to achieve that objective were subsequently used to design and produce an application as a plug-in for AutoCAD. This application allows generating and management of created connections between geometric objects and their visualization both in the AutoCAD and available client software integrated with Neo4j platform.
Key words: *vector data, technical documentation, graph database, processing, AutoCAD*

KOJARZENIE DANYCH WEKTOROWYCH, TWORZĄCYCH DOKUMENTACJE TECHNICZNĄ Z INFORMACJAMI OSADZONYMI W BAZACH GRAFOWYCH

Streszczenie

Przeanalizowano możliwości dalszego, pełniejszego wykorzystania modeli geometrycznych systemów empirycznych, w tym dotyczących rolnictwa, tworzonych na poziomie AutoCAD-a. Analiza dotyczyła w głównej mierze możliwości pełniejszego odwzorowania istniejących powiązań pomiędzy elementami graficznymi, nieczytelnymi bądź niemożliwymi do przedstawienia na poziomie cyfrowej dokumentacji technicznej w formie struktur grafowych oferowanych przez Neo4j. Rozpoznano i przedstawiono niezbędne do realizacji tego celu technologie informatyczne, które następnie wykorzystano do zaprojektowania i wytworzenia aplikacji w formie plug-in dla AutoCAD. Aplikacja ta pozwala na generowanie, a następnie na zarządzanie utworzonymi powiązaniami pomiędzy obiektami geometrycznymi oraz ich wizualizację zarówno po stronie AutoCAD-a, jak i dostępnego programu klienckiego zintegrowanego z platformą Neo4j.
Słowa kluczowe: *dane wektorowe, dokumentacja techniczna, baza grafowa, przetwarzanie, AutoCAD*

1. Introduction

At present, the technical digital documentation of real systems created by people is mainly vector data. It requires more and more often to use it in different ways and add to it additional character detail. This process can be done in a different manner and it depends on the computer application which is used. The mechanism of connecting graphical elements with relationship database's records provides the widest possibilities and in AutoCAD, product of Autodesk, can be done in the interactive way by navigating through ready-to-use forms [2]. Nevertheless in order to use this approach the database schema and connection have to be created earlier. For this phase of connecting to the external data source, the VBA or .NET platform can be also used. The second option is much more flexible but it needs far more programming work. The mentioned mechanisms pertain to relational databases and in the process of addition of the new data, the relation has to already exist or a new one must be created [3].

The alternative approach where there is no need to create new schemas and relations are graph databases which are non-schema, although there are some solutions that allow to create them [13]. Therefore there is no demand for creating new relations or it is easier than in relationship databases. However, this new way of bonding graphical and character data is not possible to use with the VBA language.

2. The essence of graph databases

In graph databases to present data and relations between them, directed graphs are used since they can be presented in a graphic form. This type for data presentation of executed query is usually desired by the users on condition that the number of returned elements is not high. Vertices and edges are the key concepts of graph databases [20]. Labels and attributes are the next components to build the structure of DB and they are connected with both vertices and edges. The labels are often treated as a type specification, however it can throw serious doubts. The edges describe relations between entities and these are directed which means that there are source node and final node [5].

The process of creation the database can be done by using a query language e.g. Cypher and nowadays the most popular graph database engine Neo4j [7, 23]. Users can also use graphical tool instead of using Cypher. In this case there is no need to know the query language during the database implementation. Mentioned languages are declarative and they allow for both creating and manipulating data. Moreover they allow user not only to build queries for returning data but also for getting information about the structure of graph databases [17].

Important features that decided on use of this type of database consist in simplified phase of modelling when some steps can be omitted. In an extreme case when conceptual modelling is done in a graph form then mentioned in the

literature three phases of modelling can be done in one or two steps. The second option occurs when there is a need to create a scheme which database should follow.

3. The programing interface for the access to the database

The advantages of the given database system must be followed not only in terms of the database itself but also in terms of programing interfaces that are essential in developing the application [15].

The authors for achieving the main goal of this article have chosen free and Open Source graph database – Neo4j. It provides a wide variety of different APIs (Application Programing Interface) for many programing languages such as: C# .NET, PHP, Ruby and many more. On the other hand AutoCAD provides us own API also in .NET technology, the choice of C# language was obvious [19]. Available API for graph databases is provided in Neo4jClient library that with three libraries for AutoDesk AutoCAD is the minimum for implementation of this project [4].

Neo4jClient library is the essential library for opening a connection with the database [8]. The created instance of this class allows user not only to open the connection with Neo4j graph engine and choose database but it allows to construct different queries compatible with CRUD – create, read, update, delete [24]. With available objects in this class it is possible to build queries both in Cypher and Gremlin. Building the select queries with the Cipher language is similar to build queries in LINQ with extension methods. The main difference is that user cannot use the anonymous types and because of it user has to define own classes to intercept the results. The mentioned earlier classes must be defined before the query takes place just like in LINQ [21]. Below snippet (Fig. 1) of code illustrates both the class for results and method with database query.

```
public class DrawingsBlock
{
    4 references
    public DrawingsBlock()
    {
        drawindID = "(main)";
    }
    5 references
    public string id { get; set; }
    19 references
    public string objName { get; set; }
    17 references
    public string handleID { get; set; }
    6 references
    public string drawindID { get; set; }
    2 references
    public string entityType { get; set; }
    1 reference
    public string objectType { get; set; }
}

public static void DLL_Check_Entries_Count_DB()
{
    GraphClient graphClient =
    new GraphClient(new Uri("http://localhost:7474/db/data"));
    graphClient.Connect();

    List<DrawingsBlock> cLista = graphClient.Cypher
                            .Match("(b:DrawingsBlock)")
                            .Return(b => b.As<DrawingsBlock>())
                            .Results.ToList();
    AcAppS.Application.ShowAlertDialog("The entires count in Graph Database: "
                                       + cLista.Count.ToString());
}
```

Source: own work / Źródło: opracowanie własne

Fig. 1. Class of results and exemplary method
Rys. 1. Klasa wynikowa i przykładowa metoda

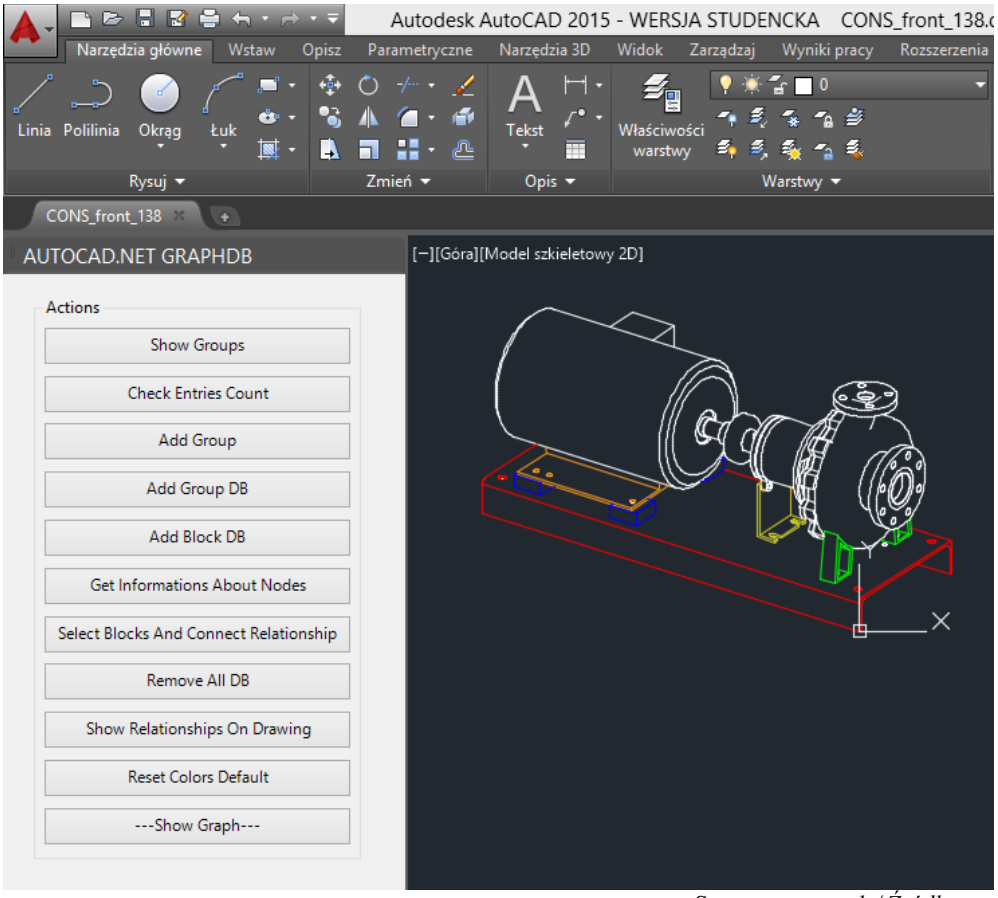
The instance of GraphClient class allows both to connect to the database and through Cypher object, query the Graph Database which are being returned into the collection [12]. Filled collections can be source for another queries where LINQ to Object can be used.

4. Application

Developed and implemented by the authors plug-in application for AutoCAD illustrates connections between the elements in the technical documentation in a digital form in the graph structures. These connections between elements or groups of elements are not directly visible at the level of technical drawing; moreover sometimes they are even impossible to see by the user and they are really important from the point of view of assembly steps or drive transmission [16]. Searching form them at the level of AutoCAD environment cannot be taken into the consideration because of the lack of available tools. These connections are usually linked with complex objects such as groups or blocks. In case when these elements have not been added during the process of drawing, they can be added at the level of discussed in this article application. For this purpose user can use Add Group button of this application or simply use proper AutoCAD command.

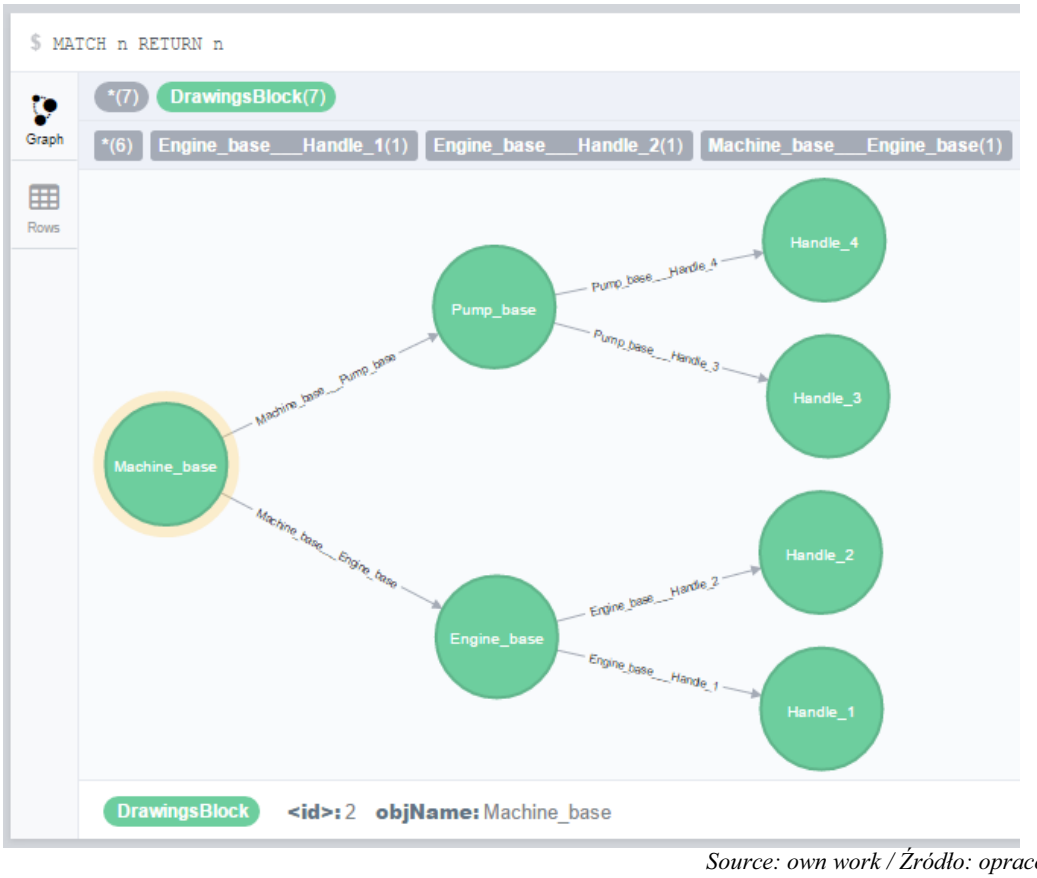
The next step to show explicit connections between elements consists in mapping complex graphical objects in a form of nodes in the graph database [22]. It has been implemented separately for groups and objects by using Add Group DB and Add Block DB respectively. Separation of those is the consequence of different representation for groups and blocks in the AutoCAD's database. For user, the process of selecting drawings' elements is mainly the same. User can view previously added elements by running Show Groups and in a result selected element is highlighted. The finish step is to define the relation between earlier added nodes [9]. This operation is also being made at the level of AutoCAD environment by selecting graphical elements where the relations are being added to. We can see the results of user's actions in the graph database [18]. This functionality is available under the Select Blocks, Groups and Connect Relationship button. This function allows users to add a group or a block and connect it with the relationship in a one-step, even if they have not been added earlier. Thereby, all graph structures in form of nodes and edges will be added automatically and user after the whole process will get the information message about created structures.

Clearly mapped graphical elements and relations between them, from interesting for us perspective, in a form of graph database gives us a wide range of possibilities for searching and returning elements that meets certain parameters, not only for nodes but also for relations. Searching can be made from the client application included in a Neo4j environment with Cypher language that returns the results in a graphical form with nodes and directed edges [6, 14]. Mentioned client application can be launched directly from our application by running Show Graph method. Exemplary results are shown in the below picture (Fig. 2). Show Relationships on Drawing included in our application is the alternative but less responsive tool for searching elements connected with relations. The results are shown directly in the AutoCAD application [1]. User can pick interesting element or relation. In both cases, in a model space, elements that meet search criteria will be shown. It was achieved by changing their colours. Search effects are shown at above illustration (Fig. 3).



Source: own work / Źródło: opracowanie własne

Fig. 2. Management of related parties at AutoCAD plug-in
Rys. 2 .Zarządzanie powiązanymi plug-in AutoCAD



Source: own work / Źródło: opracowanie własne

Fig. 3. Exemplary results of the query
Rys. 3 .Przykładowe wyniki zapytania

5. Conclusions

Digital technical documentation provided by information systems such as AutoCAD support design and manufacturing processes are primarily focused on a creation of geometric models. Further using of a technical documentation related to the life of the product depend on the possibility of enrichment it with new information, which may not necessarily be directly embedded in the digital form of the model [10]. Current available technologies allow to combine graphics with data records from relational databases, but also it is the possibility, which was the subject of this paper, of combining graphic objects with graph structures. They are definitely more flexible from the perspective of mapping the existing connections between geometric model structures, that are difficult to decipher and also create new connections. The mapping of these connections in the graph structures makes them possible to search and thanks to dedicated tools it is possible to get new possibilities for graphical presentation of search results [11]. They may be reproduced in a limited form in a digital technical documentation. Presented by the authors available new technologies which include ObjectARX, LINQ, Neo4j and Neo4jClient not always allow for a convenient way to achieve the intended purpose at the level of AutoCAD system.

6. References

[1] Bethune J.D.: Engineering Graphics with AutoCAD 2015. Peachpit Press, 2014. ISBN: 978-0-13-427101-9.

[2] Celko J.: Joe Celko's Complete Guide to NoSQL. Morgan Kaufman, 2013.

[3] Ellis G.: Getting Started with SQL Server 2014 Administration. Packt Publishing, 2014. ISBN: 978-1-782-17241-3.

[4] Fitzgerald J., Richard P.: Introduction to AutoCAD 2015*:* A Modern Perspective. Peachpit Press, 2014. ISBN: 978-0-13-427102-6.

[5] Fowler A.: NoSQL For Dummies. Wiley, 2015. ISBN: 978-1-118-90574-6.

[6] Goel A.: Neo4j Cookbook. Packt Publishing, 2015. ISBN: 978-1-78328-725-3.

[7] Gupta S.: Neo4j Essentials. Packt Publishing, 2015. ISBN: 978-1-78355-517-8.

[8] Jordan G.: Practical Neo4j. Apress, 2015. ISBN: 978-1-484200-23-0.

[9] Lal M.: Neo4j Graph Data Modeling. Packt Publishing, 2015. ISBN: 978-1-78439-730-2.

[10] Mueller W., Gruszczyński M., Raba B, Lewicki A., Przybył K., Zaborowicz M., Koszela K., Boniecki P.: Visualization of the tire-soil interaction area by means of ObjectARX programming interface. Proc. SPIE. 9159. Sixth International Conference on Digital Image Processing (ICDIP 2014). 91590G (April 16, 2014). DOI:10.1117/12.2064086.

[11] Mueller W., Nowakowski K., Tomczak R. J., Kujawa K., Rudowicz-Nawrocka J., Idziaszek P., Adrian A.: IT system supporting acquisition of image data used in the identification of grasslands. SPIE. 87781. Fifth International Conference on Digital Image Processing (ICDIP2013), 88781T (July 19, 2013). DOI: 10.1117/12.2031602.

[12] Panazarino O.: Learning Cypher. Packt Publishing, 2014. ISBN: 978-1-78328-775-8.

[13] Powell J.: A Librarian's Guide to Graphs, Data and the Semantic Web. Chandos Publishing, 2015. ISBN: 978-1-78063-434-0.

[14] Raj S.: Neo4j High Performance. Packt Publishing, 2015. ISBN: 978-1-78355-516-1.

[15] Redmon E., Wilson J.R.: Seven Databases in Seven Weeks. O'Reilly Media, 2012.

[16] Riley P., Dix M.: Discovering AutoCAD® 2015. Peachpit Press, 2014. ISBN: 978-0-13-427100-2.

[17] Robinson I., Webber J., Eifrem E.: Graph Databases. O'Reilly Media, 2013.

[18] Robinson I., Webber J., Eifrem E.: Graph Databases Second Edition. O'Reilly Media, 2015. ISBN:978-1-491-93200-1.

[19] Schmalz M.: C# Database Basics. O'Reilly Media, 2012. ISBN: 978-1-4493-0998-5.

[20] Sideris Courseware Corp. Data Modeling: Logical Database Design. 2011.

[21] Tiwari S.: Professional NoSQL. Wrox, 2011. ISBN: 978-1-4571-0685-9.

[22] Vaish G.: Getting Started with NoSQL. Packt Publishing, 2013. ISBN: 978-1-84969-498-8.

[23] Van Bruggen R.: Learning Neo4j. Packt Publishing, 2014. ISBN: 978-1-84951-716-4.

[24] Vucotic A., Watt N., Abedrabbo T., Fox D., Partner J.: Neo4j in Action. Manning, 2014. ISBN: 978-1-61729-076-3.