Paper received: 11.05.2019. Journal of Agronomy, Technology and Engineering Management
Paper accepted: 06.08.2019. ISSN: 2620-1755 Šimović and Varga, 2019. Vol. 2 SI(1): 15-34

Review paper

APPLYING SQL DATABASE QUERY TO ACCESS SQL SERVER 2019 – VISUAL STUDIO 2019

Vladimir Šimović[1,*], Matija Varga[2]

[1] The University of Applied Sciences Baltazar Zaprešić, Vladimira Novaka 23, 10290 Zapresic, Croatia
[2] University NORTH, Trg Žarka Dolinara 1, 48000 Koprivnica, Croatia

*Corresponding author:
E-mail address: vladimir.simovic@bak.hr

ABSTRACT: The paper presents: relational data model, which is the basis for the development of relational database management system (RDBMS) and the development of query languages for working with databases, relational schema, complex SQL queries and subqueries and the results obtained based on queries in relations. MS SQL Server 2019 accesses data in the database and can extend MS SQL Server 2019 application modules (eg. MS Visual Studio) that are created as a graphical interface for end users, with the addition of new capabilities that will enhance the execution of processes for participants within the company and other application modules of the organization. Each organization's database contains information of great importance, so queries and backups will be made using relational algebra operators and warned of potential threats and dangers of database corruption.

Key words: *RDBMS, relational data model, relational schema, SQL, data manipulation language, dana management, relational algebra operators, database protection*

INTRODUCTION

Working with the database is about retrieving data, adding data, modifying, deleting and preserving the integrity of the data, backing up the database, allowing simultaneous access to the database for multiple users, verifying the identity of users, etc. The database in Croatian has been created entirely by author named: "Applying SQL Complex Queries to Access Data in a Database Using MS SQL Server 2019".

The goals of the paper are: (1) to show certain capabilities of MS SQL Server 2019, (2) to show ways of querying using aggregate functions, (3) to show how to set nested queries i.e. subqueries, (4) to warn about possible database corruption, (5) present the ability to collect data from multiple relationships, (6) show and explain, by way of example, basic concepts in databases such as relation, relational schema, RMP model, etc., (7) present more complex queries to quickly find specific data in a database, to (8) demonstrate how to delete data on MS SQL Server 2019, (9) explain ways to protect data from unauthorized and simultaneous access to data.

RELATIONAL DATA MODEL

The relational data model forms the basis for the development of the RDBMS[1] and the development of query languages for working with databases. RDM (Relational Data Model) is a solid theoretical apparatus that has a dominant influence on the further development of theory and practice in the field of databases. The RMP was constructed based on the EVA[2] model [4]. Also, model creation represents the second stage in data modelling. RDM is created automatically so that each object type from the ELA model is converted to a RDM relation. The object properties become attributes of the relational schema, and the primary object key becomes the primary relational key. An attribute in a relational schema is called a foreign or foreign key if it has (has) a primary key function in another relational schema. The relational model shows everything that the ELA model does.

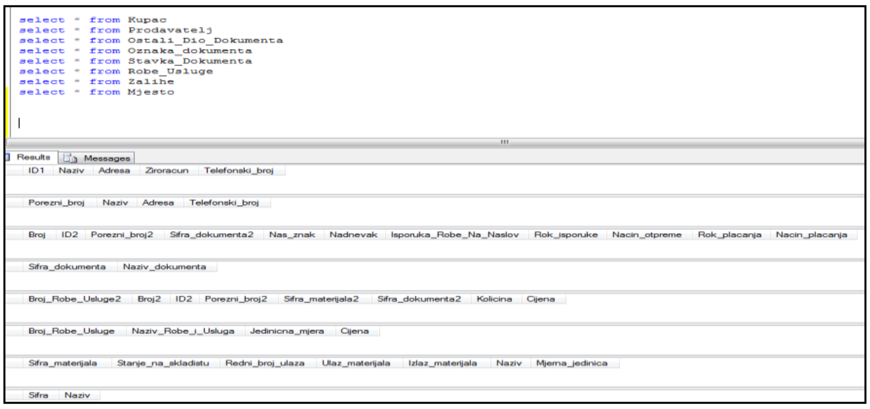
SCOPE OF PROJECTS

The scope of the project depends on three key elements: quality to be delivered, time to be spent and resources to be engaged. These three key elements are the basic pillars of project management. Project quality can be defined as a series of activities that must be carried out for a project to meet agreed specifications. Specifically, the keyword here is an activity for the simple reason that quality control enters the activity. When planning, activities that are part of the quality assurance system should be envisaged. If these activities are neglected, the quality may be unsatisfactory.
The time it takes to complete project activities is defined in the planning process. However, it is in this segment that there can be gross omissions. In such a case, the recovery of the project can be problematic.
Resources are not just about money; this segment is far broader and includes other hard-to-define moments in addition to financial moments. Knowledge and experience are typical project resources that are difficult to define. They are abstract in themselves, but the members of the project team in charge of carrying out the project are the bearers of knowledge and experience.
All told so far applies to software projects, especially in the area of human resource management. The software is mainly a product of the experience and knowledge of the project participants. On the other hand, when the software becomes too complex then the division of work can take place.

RELATIONAL SCHEME

A relational schema is a data model based on set theory that represents the form by which data is stored. Each relational scheme has meaning and content [2].

[1] RDBMS – Relational Database Management System.
[2] ELA – Entities, Links, and Attributes.



```sql
select * from Kupac
select * from Prodavatelj
select * from Ostali_Dio_Dokumenta
select * from Oznaka_dokumenta
select * from Stavka_Dokumenta
select * from Robe_Usluge
select * from Zalihe
select * from Mjesto
```

```
ID1  Naziv  Adresa  Ziroracun  Telefonski_broj

Porezni_broj  Naziv  Adresa  Telefonski_broj

Broj  ID2  Porezni_broj2  Sifra_dokumenta2  Nas_znak  Nadnevak  Isporuka_Robe_Na_Naslov  Rok_isporuke  Nacin_otpreme  Rok_placanja  Nacin_placanja

Sifra_dokumenta  Naziv_dokumenta

Broj_Robe_Usluge2  Broj2  ID2  Porezni_broj2  Sifra_materijala2  Sifra_dokumenta2  Kolicina  Cijena

Broj_Robe_Usluge  Naziv_Robe_i_Usluga  Jedinicna_mjera  Cijena

Sifra_materijala  Stanje_na_skladistu  Redni_broj_ulaza  Ulaz_materijala  Izlaz_materijala  Naziv  Mjerna_jedinica

Sifra  Naziv
```

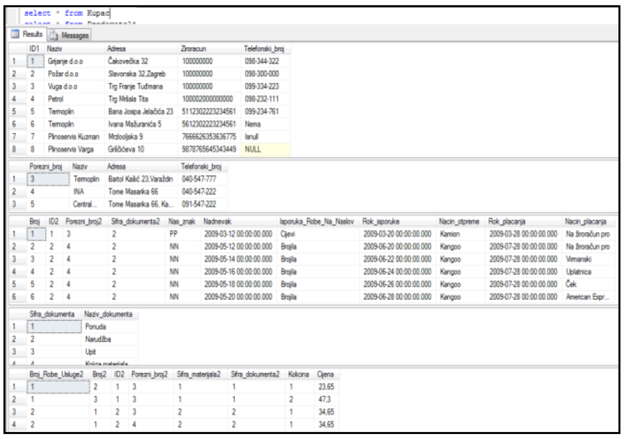
Figure 1. View of relational schemas in MS SQL Server 2019 (Source: Authors)

Figure 1 shows the relational schemas. In the lower part of the image, the *'Results'* option contains the attributes of relations, that is, the set of attributes that make up the domain elements. Figure 1 also shows the relational schema and the queries invoking the relational schema and showing its attributes. Such a data model is based on set theory. For example, we can denote a relation by *R*, while a set of attributes can be denoted by *A. R(A)*, *A* is a set of attributes of a relation *R*. Example: $R(A_1, A_2, A_3, \ldots, A_n)$. Let $R(A_1, A_2, A_3, \ldots, A_n)$ be a relational scheme and let each attribute A_i be an element of the relation *R* that can represent a domain (depending on which level we are observing), given by the corresponding domain Di. Let D be the union of all domains D_i, ($i = 1, 2, \ldots, n$). The relation *R* on the relational scheme is a finite set of *n-vertices* obtained by mapping from *R* to *D* with the constraint that for every *n-torque* value to which the attribute A_i is mapped, there must be an element of domain D_i, ($i = 1, 2, \ldots, n$). The set of attributes that participate in the formation of key candidates are called *key attributes*, because they are key in defining relationships, and other attributes can be said to be *non-key attributes* [4].

RELATIONAL DATABASE MANAGEMENT SYSTEM – RDBMS

A 'Relational Database Management System' (RDBMS) is a computer system with software to create, update, search and maintain a database whose structure is described by a relational model. The properties of RDBMS are: functionalities of RDBMS, which provides easier administration of data, provides a better understanding of data integrity requirements, enables a more efficient data redundancy[3] control process, supports more complex database lookup procedures, enables data relationships to be displayed. In fact, RDBMS[4] has a rigorous mathematical basis established on the concept of relation and set theory, etc. [4]. The relational system in use here is a database management system called 'MS SQL Server 2019'.

[3] Isti podatci koji su pohranjeni više puta (zalihost). Nastoji se izbjeći.
[4] Relational Database Management System' = 'RDBMS'.



```
select * from Kupac
```

Results | Messages

	ID1	Naziv	Adresa	Ziroracun	Telefonski_broj
1	1	Grijanje d.o.o	Čakovečka 32	100000000	098-344-322
2	2	Požar d.o.o	Slavonska 32,Zagreb	100000000	098-300-000
3	3	Vuga d.o.o	Trg Franje Tuđmana	100000000	099-334-223
4	4	Petrol	Trg Mršala Tita	100002000000000	098-232-111
5	5	Termoplin	Bana Josipa Jelačića 23	5112302223234561	099-234-761
6	6	Termoplin	Ivana Mažuranića 5	5612302223234561	Nema
7	7	Plinoservis Kuzman	Mrzloojlska 9	7666626353636775	lanull
8	8	Plinoservis Varga	Gršičeva 10	9878765645343449	NULL

	Porezni_broj	Naziv	Adresa	Telefonski_broj
1	3	Termoplin	Bartol Kašić 23,Varaždin	040-547-777
2	4	INA	Tome Masarka 66	040-547-222
3	5	Central...	Tome Masarka 66, Ka...	091-547-222

	Broj	ID2	Porezni_broj2	Sifra_dokumenta2	Nas_znak	Nadnevak	Isporuka_Robe_Na_Naslov	Rok_isporuke	Nacin_otpreme	Rok_placanja	Nacin_placanja
1	1	1	3	2	PP	2009-03-12 00:00:00.000	Cijevi	2009-03-20 00:00:00.000	Kamion	2009-03-28 00:00:00.000	Na žiroračun pro
2	2	2	4	2	NN	2009-05-12 00:00:00.000	Brojila	2009-06-20 00:00:00.000	Kangoo	2009-07-28 00:00:00.000	Na žiroračun pro
3	3	2	4	2	NN	2009-05-14 00:00:00.000	Brojila	2009-06-22 00:00:00.000	Kangoo	2009-07-28 00:00:00.000	Virmanski
4	4	2	4	2	NN	2009-05-16 00:00:00.000	Brojila	2009-06-24 00:00:00.000	Kangoo	2009-07-28 00:00:00.000	Uplatnica
5	5	2	4	2	NN	2009-05-18 00:00:00.000	Brojila	2009-06-26 00:00:00.000	Kangoo	2009-07-28 00:00:00.000	Ček
6	6	2	4	2	NN	2009-05-20 00:00:00.000	Brojila	2009-06-28 00:00:00.000	Kangoo	2009-07-28 00:00:00.000	American Expr...

	Sifra_dokumenta	Naziv_dokumenta
1	1	Ponuda
2	2	Narudžba
3	3	Upit
4	4	Knjiga materijala

	Broj_Robe_Usluge2	Broj2	ID2	Porezni_broj2	Sifra_materijala2	Sifra_dokumenta2	Kolicina	Cijena
1	1	2	1	3	1	1	1	23,65
2	1	3	1	3	1	1	2	47,3
3	2	1	2	3	2	2	1	34,65
4	2	1	2	4	2	2	1	34,65

Figure 2. View of the RDM (Relational Data Model) in 'MS SQL Server 2019'
(Source: Authors)

Figure 2 shows the 'Relational Data Model' (RDM) and database entry commands. This is a data model because the data for each attribute within the created relation is displayed. Figure 2 also shows the relational database. Relational databases consist of at least two tables or relations. Five of the seven relationships can be seen from Figure 2. From all relationships, we can print or call data using queries contained in the database. The relational schemas shown show that attributes represent the domain (depending on which level we are looking at).

STANDARD STRUCTURED QUERY LANGUAGE (SQL)

The abbreviation SQL means 'structured query language'. SQL (Structured Query Language) is a commercial query language for working with relational data base. It is widely known that SQL was developed by IBM [8]. SQL is the most popular, well-known and commonly used query language. SQL is used to manipulate and manage the data contained in the database. The SQL language was created in the 1970s in the development of the 'System R' development research project within IBM [8]. The project manager and chief designer of SQL was Donald Chamberlin. The language was gradually refined, and its refined version appeared in today's IBM [8] RDBMS called DB2. In the spread of SQL during the 1980s, Oracle Corporation, a software company that incorporated SQL into its DBMS, played an important role, making it accessible and popular on all major computer platforms. Other DBMS vendors, such as Digital Equipment Corporation (DEC) and Informix Inc., have closed acceptable SQL on the market and abandoned their own languages [3]. The relational database provided the greatest flexibility in decomposition data, which provides a major advantage in scheduling the distribution of data across individual system nodes. They can divide relationships vertically and horizontally so that they can also be reunited by joining operations.

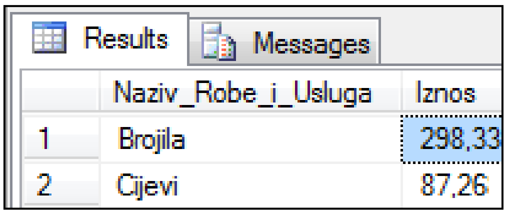
For example, '*Customer*' relationship (in Figure 1) has a set of attributes: *ID1, Name, Address, Giro_Account, Telephone_number*. The relational schema and attribute set can be displayed as an example: *Customer(ID, Name, Address, Giro_Account, Phone_Number)*[5]. No matter what the attributes in the 2019 MS SQL Server are sorted in the order they are in the ELA model, in the relational schema the order of the attributes does not matter. In MS SQL Server 2019 they are arranged in this way for the sake of transparency. Example of a query to call the displayed relationships by name *Buyer, Seller, Document_Label, Goods_Services, Inventory, Other_Document_Part, Document_Item*[6] in Figure 1. report(reads): "*select * from table*".

COMPLEX QUERIES AND SUBQUERIES

A simple query is set with the "*select*" command, with the help of which we can get information from a specific table (relation). A simple subquery is set with the 'SELECT' command which is nested within the commands *SELECT, SELECT ... INTO, INSERT ... INTO, DELETE, UPDATE*, or within other subquery. In the subquery, we use the *SELECT* command to derive a set of specific values to evaluate in a *WHERE* or *HAVING* condition statement [6]. Encoding the first complex query (below, in Croatian) the relation with the following data was obtained (Figure 3): the name of the goods and the amount of the total price with VAT for all the goods purchased (pipes and counters):

*select Naziv_Robe_i_Usluga, (sum(st.Kolicina * ru.Cijena)*23)/100+sum(st.Kolicina * ru.Cijena) as 'Iznos' from Stavka_Dokumenta st,Robe_Usluge ru*

where st.Broj_Robe_Usluge2=ru.Broj_Robe_Usluge group by Naziv_Robe_i_Usluga



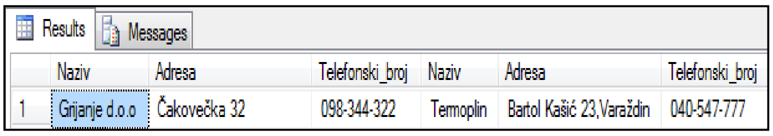
	Naziv_Robe_i_Usluga	Iznos
1	Brojila	298,33
2	Cijevi	87,26

Figure 3. View of the relation obtained after the first complex query is set (Source: Authors)

The name of the goods and services and the amount represent the attributes of the displayed relation that we can view as a domain with the elements: counters 298.33 and pipes 87.26. When asked for a second complex query (below, in Croatian), information about the buyer's name, address, buyer's phone number and seller's name, seller's address and seller's phone number from the buyer and seller relations was requested.

select k.Naziv,k.Adresa,k.Telefonski_broj,p.Naziv,p.Adresa,p.Telefonski_broj from Kupac k, Prodavatelj p

[5] In Croatian: *Kupac(ID, Naziv, Adresa, Žiroračun, Telefonski_broj)*.
[6] In Croatian: *Kupac, Prodavatelj, Oznaka_dokumenta, Robe_Usluge, Zalihe, Ostali_Dio_Dokumenta, Stavka_Dokumenta*.

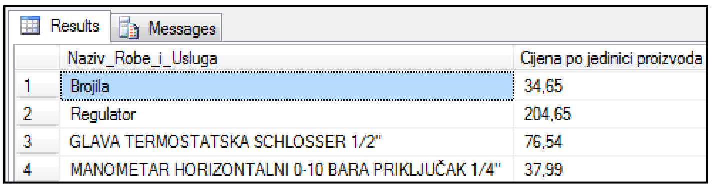


	Naziv	Adresa	Telefonski_broj	Naziv	Adresa	Telefonski_broj
1	Grijanje d.o.o	Čakovečka 32	098-344-322	Termoplin	Bartol Kašić 23,Varaždin	040-547-777

Figure 4. View of the relation obtained after the second complex query is set (Source: Authors)

With the third complex query (below, in Croatian), the price of material per unit of product above HRK 30 was requested and obtained using comparison operators: greater than (>) and equal (=) (Figure 5).

select Naziv_Robe_i_Usluga, Cijena as 'Cijena po jedinici proizvoda' from Robe_Usluge where Cijena >= 30

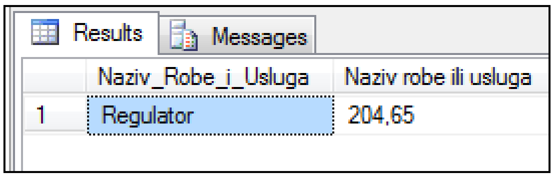


	Naziv_Robe_i_Usluga	Cijena po jedinici proizvoda
1	Brojila	34,65
2	Regulator	204,65
3	GLAVA TERMOSTATSKA SCHLOSSER 1/2"	76,54
4	MANOMETAR HORIZONTALNI 0-10 BARA PRIKLJUČAK 1/4"	37,99

Figure 5. View of the relation obtained after the third complex query is set (Source: Authors)

With the set fourth complex query (below, in Croatian), the price information of all products beginning with the letter R was requested and obtained (Figure 6).

select Naziv_Robe_i_Usluga, Cijena as 'Naziv robe ili usluga'from Robe_Usluge where Naziv_Robe_i_Usluga like 'R%'

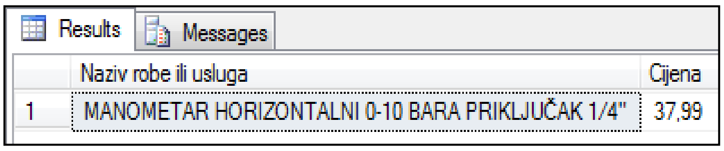


	Naziv_Robe_i_Usluga	Naziv robe ili usluga
1	Regulator	204,65

Figure 6. View of the relational representation of the price information of all products beginning with the letter R obtained after the fourth complex query is set (Source: Authors)

With the set of the fifth complex query (below, in Croatian), the price information of the horizontal pressure gauge 0-10 bar, connection "¼" was requested and obtained (Figure 7).

select Naziv_Robe_i_Usluga as 'Naziv robe ili usluga' , Cijena from Robe_Usluge where Naziv_Robe_i_Usluga like 'MANOMETAR HORIZONTALNI 0-10 BARA PRIKLJUČAK „¼“'



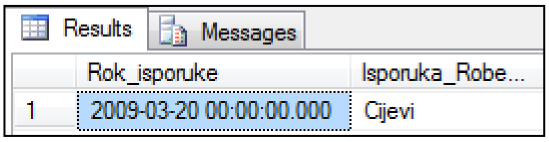
```
Results | Messages
   Naziv robe ili usluga                           Cijena
1  MANOMETAR HORIZONTALNI 0-10 BARA PRIKLJUČAK 1/4"  37,99
```

Figure 7. View of the relational representation of the price information of the horizontal pressure gauge 0-10 bar, connection “¼” obtained after the fifth complex query is set

(Source: Authors)

With the sixth complex query (below, in Croatian), the date of delivery and the name of the product delivered (the pipe) were requested and obtained (Figure 8).

select Rok_isporuke,Isporuka_Robe_Na_Naslov from Ostali_Dio_Dokumenta where Rok_isporuke = '2009-03-20'



```
Results | Messages
   Rok_isporuke             Isporuka_Robe...
1  2009-03-20 00:00:00.000  Cijevi
```

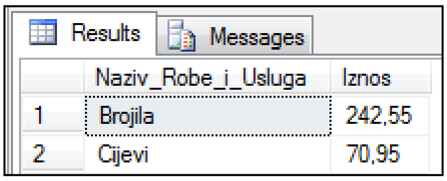
Figure 8. View of the relational representation of the date of delivery and the name of the product delivered (the pipe) obtained after the sixth complex query is set (Source: Authors)

With the seventh complex query (below, in Croatian), information on the name of the goods and the amount of the price of the goods excluding VAT were requested and obtained (Figure 9). The query uses the HAVING operator, which is used for filtering, that is, it limits the groups that appear in the query. The HAVING condition has similarities to the WHERE condition. WHERE comes usually before GROUP BY, while HAVING comes after GROUP BY and executes after grouping records.

select Naziv_Robe_i_Usluga, sum(st.Kolicina * ru.Cijena)as 'Iznos' from Stavka_Dokumenta st,Robe_Usluge ru

where st.Broj_Robe_Usluge2=ru.Broj_Robe_Usluge group by Naziv_Robe_i_Usluga

having sum(st.Kolicina*ru.Cijena)>45.23


Results | Messages

	Naziv_Robe_i_Usluga	Iznos
1	Brojila	242,55
2	Cijevi	70,95

Figure 9. View of the relational representation of the name of the goods and the amount of the price for counters and pipes obtained after the seventh complex query is set (Source: Authors)

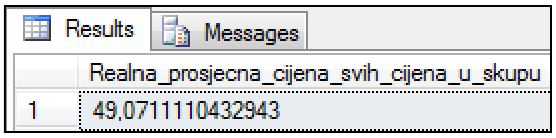
In SQL, commands are often used to delete data, such as (below, in Croatian):

delete from Stavka_Dokumenta where Cijena=23.64

The command deletes all the quoted prices from the relation named *Item_Document* (in Croatian: *Stavka_Dokumenta*), i.e. it destroys the information that the material price is 23.64 HRK.

With the eighth complex query (below, in Croatian), average data calculated on the basis of the prices of all products recorded in the database were requested and obtained (Figure 10). *AVG*'s function is a commonly known function for calculating averages. Numerous authors in addition to the *AVG* function and the min, max, count and sum sums are aggregate functions. So instead of *AVG* it is possible to combine in queries the basic functions: *min, max, count* and *sum*.

select avg(cast(Cijena as real)) as Realna_prosjecna_cijena_svih_cijena_u_skupu from Robe_Usluge

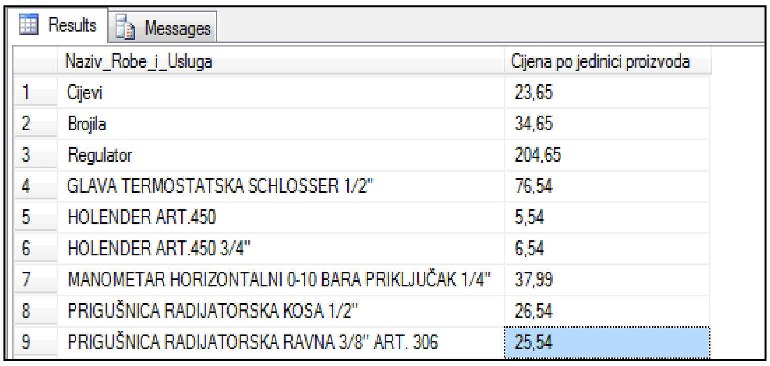

Results | Messages

	Realna_prosjecna_cijena_svih_cijena_u_skupu
1	49,0711110432943

Figure 10. View of the relational representation of the query results for the average price calculated based on all the prices entered for each product obtained after the eighth complex query is set (Source: Authors)

With the ninth complex query (below, in Croatian), information on the name of the goods and the price per unit of product were requested and obtained, but only at the price other than HRK 23.64 (Figure 11). The operator *! = other than*. Was used for this query. Other comparison operators that can be combined in this type of query are: *equal (=), greater than (>), less than (<), equal to or greater than (=>), equal to or less than (= <), different from (<>), not greater than (!>), not less than (! <)*.

select Naziv_Robe_i_Usluga,Cijena as 'Cijena po jedinici proizvoda' from Robe_Usluge where Cijena != 23.64.



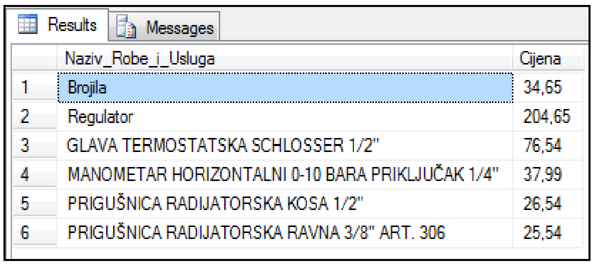
Results | Messages

	Naziv_Robe_i_Usluga	Cijena po jedinici proizvoda
1	Cijevi	23,65
2	Brojila	34,65
3	Regulator	204,65
4	GLAVA TERMOSTATSKA SCHLOSSER 1/2"	76,54
5	HOLENDER ART.450	5,54
6	HOLENDER ART.450 3/4"	6,54
7	MANOMETAR HORIZONTALNI 0-10 BARA PRIKLJUČAK 1/4"	37,99
8	PRIGUŠNICA RADIJATORSKA KOSA 1/2"	26,54
9	PRIGUŠNICA RADIJATORSKA RAVNA 3/8" ART. 306	25,54

Figure 11. View of the relational representation of the results for query requesting information on the name of the goods and a price different from 23.64 obtained after the ninth complex query is set (Source: Authors)

With the 10th complex query (below, in Croatian), information on the name and price of the goods was requested and obtained, costing between 25 HRK and 208 HRK (Figure 12). The *between* operator specifies the ranges we are testing, and it is combined with the logical operator *and*.

select ru.Naziv_Robe_i_Usluga,ru.Cijena from Robe_Usluge ru where ru.Cijena between 25 and 208



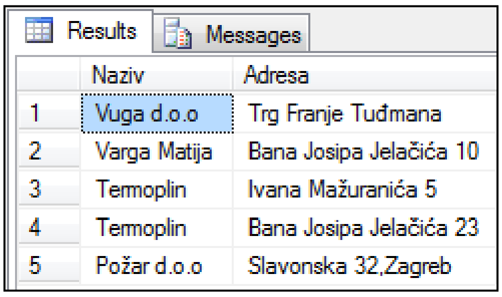
Results | Messages

	Naziv_Robe_i_Usluga	Cijena
1	Brojila	34,65
2	Regulator	204,65
3	GLAVA TERMOSTATSKA SCHLOSSER 1/2"	76,54
4	MANOMETAR HORIZONTALNI 0-10 BARA PRIKLJUČAK 1/4"	37,99
5	PRIGUŠNICA RADIJATORSKA KOSA 1/2"	26,54
6	PRIGUŠNICA RADIJATORSKA RAVNA 3/8" ART. 306	25,54

Figure 12. View of the relational representation of the information on the name and price of the goods costing between 25 HRK and 208 HRK (operator *between* was combined with the logical operator *and*) obtained after the 10th complex query is set (Source: Authors)

With the eleventh complex query (below, in Croatian), the requested and obtained information about the customer's name and customer's address are sorted by descending by name. The query defines that the user of RDBMS wants only the first five customers, which is defined by the *top* command and the number (*value*) *5* (Figure 13).

select top 5 Naziv, Adresa FROM Kupac ORDER BY Naziv desc



Results | Messages

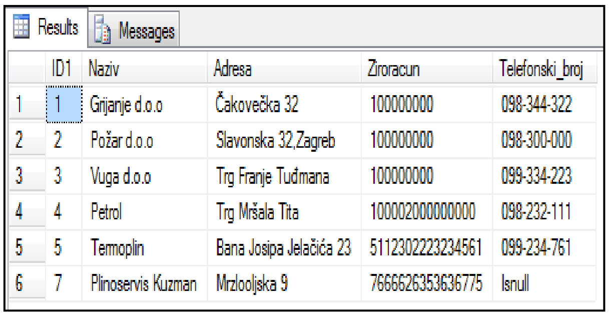
	Naziv	Adresa
1	Vuga d.o.o	Trg Franje Tuđmana
2	Varga Matija	Bana Josipa Jelačića 10
3	Termoplin	Ivana Mažuranića 5
4	Termoplin	Bana Josipa Jelačića 23
5	Požar d.o.o	Slavonska 32,Zagreb

Figure 13. View of the relational representation of the information about the customer's name and customer's address (which were sorted by descending by name) and only for the first five customers obtained after the 11th complex query is set (Source: Authors)

When asked for the twelfth complex query (below, in Croatian), the information about the customers which left their phone number was requested and obtained, and their number was entered into the database (Figure 14). The query used the *where* condition. In *where* condition, aggregate values or groups cannot be referenced. *Where* condition applies to single rows.

1st example: *SELECT * FROM Kupac WHERE Telefonski_broj <> '' AND Telefonski_broj != 'nema';*

2nd example: *SELECT * FROM Kupac WHERE NOT (Telefonski_broj='' OR Telefonski_broj = 'nema');*



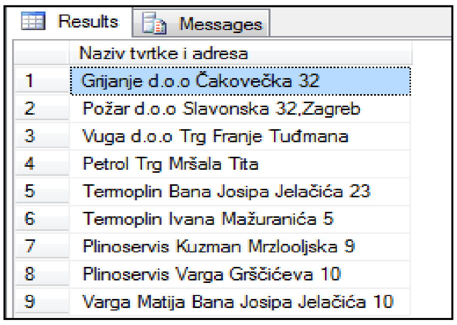
Results | Messages

	ID1	Naziv	Adresa	Ziroracun	Telefonski_broj
1	1	Grijanje d.o.o	Čakovečka 32	100000000	098-344-322
2	2	Požar d.o.o	Slavonska 32,Zagreb	100000000	098-300-000
3	3	Vuga d.o.o	Trg Franje Tuđmana	100000000	099-334-223
4	4	Petrol	Trg Mršala Tita	100002000000000	098-232-111
5	5	Termoplin	Bana Josipa Jelačića 23	5112302223234561	099-234-761
6	7	Plinoservis Kuzman	Mrzlooljska 9	7666626353636775	Isnull

Figure 14. View of the relational representation of the information about the customers which left their phone number obtained after the 12th complex query is set (Source: Authors)

With setting up the thirteenth complex query (below, in Croatian), the values of the two attributes were combined into one column (Figure 15).

select Naziv+' '+Adresa as 'Naziv tvrtke i adresa' from Kupac

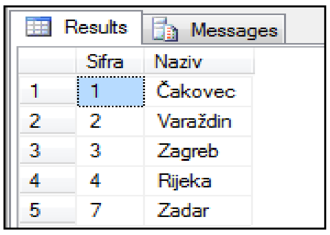


	Naziv tvrtke i adresa
1	Grijanje d.o.o Čakovečka 32
2	Požar d.o.o Slavonska 32,Zagreb
3	Vuga d.o.o Trg Franje Tuđmana
4	Petrol Trg Mršala Tita
5	Termoplin Bana Josipa Jelačića 23
6	Termoplin Ivana Mažuranića 5
7	Plinoservis Kuzman Mrzlooljska 9
8	Plinoservis Varga Grščićeva 10
9	Varga Matija Bana Josipa Jelačića 10

Figure 15. View of the relational representation of the information about the data collected from 2 columns in 1 column obtained after the 13th complex query is set (Source: Authors)

The fourteenth complex query (below, in Croatian) shows cities that do not have the letter S in their name (Figure 16).

*select * from Mjesto where Naziv not like '%s%'*



	Sifra	Naziv
1	1	Čakovec
2	2	Varaždin
3	3	Zagreb
4	4	Rijeka
5	7	Zadar

Figure 16. View of the relational representation of the information about the cities that do not have the letter S in their name obtained after the 14th complex query is set (Source: Authors)

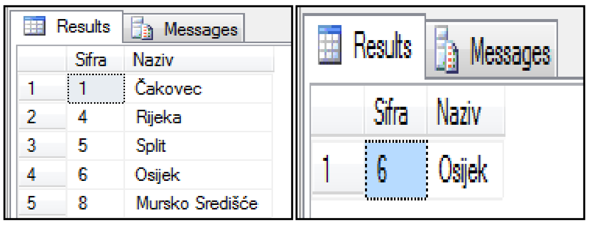
For the fifteenth complex query (below, in Croatian), the cities whose name does not begin with the letters V and Z are printed. The *LIKE* operator is used to print the results, which is used to compare *strings*, i.e. *characters* (Figure 17).

*select * from Mjesto where Naziv*

like '[^VZ]%'

*select * from Mjesto where Naziv*

like '_s%'



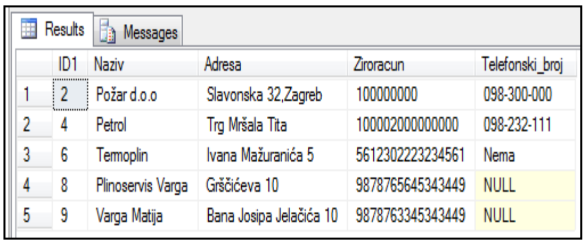
	Sifra	Naziv
1	1	Čakovec
2	4	Rijeka
3	5	Split
4	6	Osijek
5	8	Mursko Središće

	Sifra	Naziv
1	6	Osijek

Figure 17. View of the relational representation of the information about the cities whose name does not begin with the letters V and Z, and the cities which has the second letter S in the name and contains the first of any letters in the name obtained after the 15th complex query is set (Source: Authors)

For the sixteenth complex query (below, in Croatian), all buyers that do not have ID1 equal to 1, 3, 5, 7 are printed (Figure 18).

*select * from Kupac where ID1 NOT IN (1,3,5,7)*

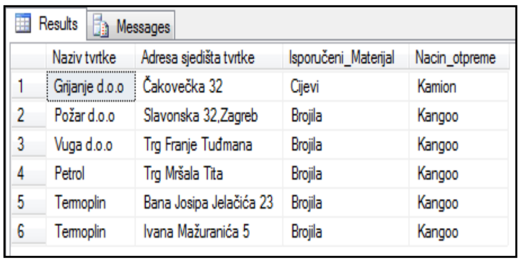


	ID1	Naziv	Adresa	Ziroracun	Telefonski_broj
1	2	Požar d.o.o	Slavonska 32,Zagreb	100000000	098-300-000
2	4	Petrol	Trg Mršala Tita	10000200000000	098-232-111
3	6	Termoplin	Ivana Mažuranića 5	5612302223234561	Nema
4	8	Plinoservis Varga	Grščićeva 10	9878765645343449	NULL
5	9	Varga Matija	Bana Josipa Jelačića 10	9878763345343449	NULL

Figure 18. View of the relational representation of the information about all buyers that do not have ID1 equal to 1, 3, 5, 7 obtained after the 16th complex query is set (Source: Authors)

When asked for the seventeenth complex query (below, in Croatian), all information about the company name, the address of the company headquarters, the delivered material and the method of dispatch of the goods are printed. The query used an *internal join* to retrieve records from multiple tables and to provide one set of records as the final query solution (Figure 19).

select Naziv as 'Naziv tvrtke',Adresa as 'Adresa sjedišta tvrtke', Isporuka_Robe_Na_Naslov as Isporuceni_Materijal, Nacin_otpreme from Kupac inner join Ostali_Dio_Dokumenta ON Kupac.ID1=Ostali_Dio_Dokumenta.Broj



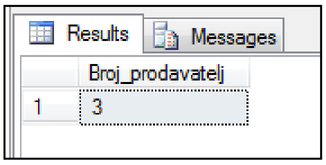
Results | Messages

	Naziv tvrtke	Adresa sjedišta tvrtke	Isporučeni_Materijal	Nacin_otpreme
1	Grijanje d.o.o	Čakovečka 32	Cijevi	Kamion
2	Požar d.o.o	Slavonska 32,Zagreb	Brojila	Kangoo
3	Vuga d.o.o	Trg Franje Tuđmana	Brojila	Kangoo
4	Petrol	Trg Mršala Tita	Brojila	Kangoo
5	Termoplin	Bana Josipa Jelačića 23	Brojila	Kangoo
6	Termoplin	Ivana Mažuranića 5	Brojila	Kangoo

Figure 19. View of the relational representation of all information about the company name, the address of the company headquarters, the delivered material and the method of dispatch of the goods obtained after the 17th complex query is set (Source: Authors)

With the eighteenth complex query (below, in Croatian), the number of the seller having a different name in the relation called the *seller* is printed. *COUNT* returns the number of elements in the group. *COUNT* returns an integer value, while *DISTINCT* considers different values (Figure 20).

select COUNT(DISTINCT Naziv) as Broj_prodavatelj from Prodavatelj



Results | Messages

	Broj_prodavatelj
1	3

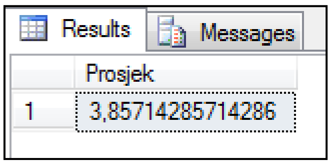
Figure 20. View of the relational representation of all information about the number of the seller having a different name in the relation called the *seller* obtained after the 18th complex query is set (Source: Authors)

When asked for the nineteenth complex query (below, in Croatian), the average grade data is printed as a real number. Grades from the fourth exam date, 2011, were averaged. The query used the aggregate function AVG (Figure 21).

select AVG(CAST(ocjena as float)) as Prosjek from MV_predmet

where DATEPART(D, ispitni_rok)=4

AND DATEPART(YY,ispitni_rok)=2011



Results | Messages

	Prosjek
1	3,85714285714286

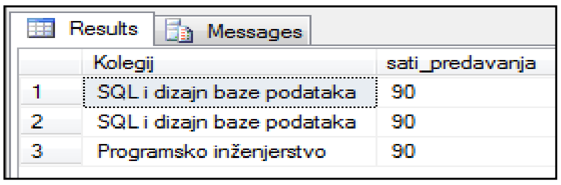
Figure 21. View of the relational representation of all information about the average grade (as a real number) obtained after the 19th complex query is set (Source: Authors)

When asked for the 20th complex query (below, in Croatian), all information about the all courses which are completed and take more hours than the average time for teaching all the subjects individually (Figure 22).

select naziv as 'Kolegij', sati_predavanja from MV_predmet

where sati_predavanja > (select AVG(sati_predavanja) from

MV_predmet)

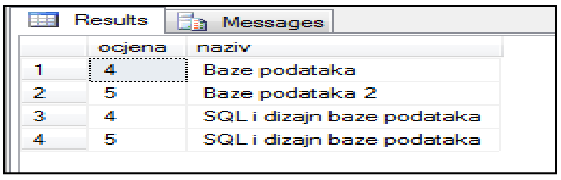


	Kolegij	sati_predavanja
1	SQL i dizajn baze podataka	90
2	SQL i dizajn baze podataka	90
3	Programsko inženjerstvo	90

Figure 22. View of the relational representation of all information about the all courses which are completed and take more hours than the average time for teaching all the subjects individually obtained after the 20th complex query is set (Source: Authors)

With the 21st complex query (below, in Croatian), all grades above the average grade and subject name are printed (Figure 23).

select ocjena, naziv from MV_predmet where ocjena>(select AVG(ocjena) from MV_predmet)



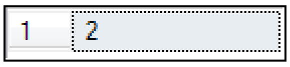
	ocjena	naziv
1	4	Baze podataka
2	5	Baze podataka 2
3	4	SQL i dizajn baze podataka
4	5	SQL i dizajn baze podataka

Figure 23. View of the relational representation of all grades above the average grade and subject name obtained after the 21st complex (sub)query is set (Source: Authors)

The 22nd complex query (below, in Croatian) shows the number of faculties with more than 15,000 students enrolled. The aggregate *count* function and the comparison operator more than (>) were used (Figure 24).

select count(*)from College

where enrollment > 15000; [7]



1	2

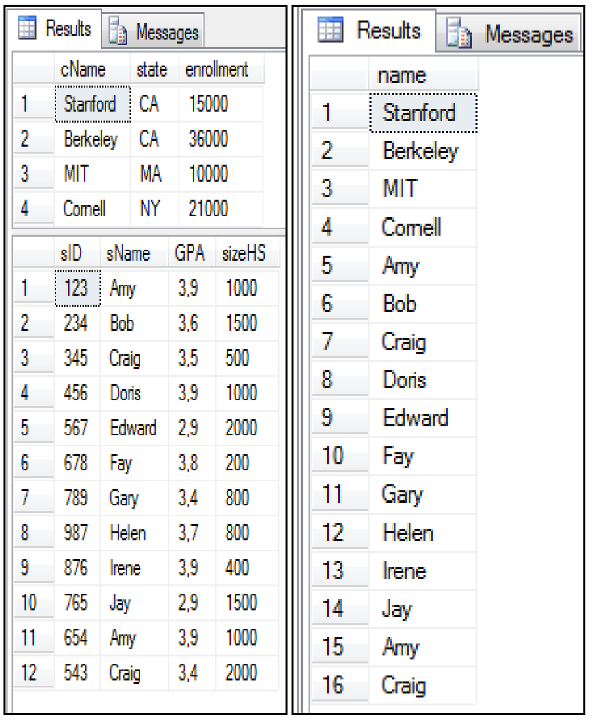
Figure 24. View of the relational representation of the number of faculties with more than 15,000 students enrolled obtained after the 22nd complex (sub)query is set (Source: Authors)

With the 23rd complex query (below, in Croatian), the union operation was applied with the *union* command (Figure 25).

select cName as name from College

union all

select sName as name from Student;[7]



Results | Messages

	cName	state	enrollment
1	Stanford	CA	15000
2	Berkeley	CA	36000
3	MIT	MA	10000
4	Cornell	NY	21000

	sID	sName	GPA	sizeHS
1	123	Amy	3,9	1000
2	234	Bob	3,6	1500
3	345	Craig	3,5	500
4	456	Doris	3,9	1000
5	567	Edward	2,9	2000
6	678	Fay	3,8	200
7	789	Gary	3,4	800
8	987	Helen	3,7	800
9	876	Irene	3,9	400
10	765	Jay	2,9	1500
11	654	Amy	3,9	1000
12	543	Craig	3,4	2000

Results | Messages

	name
1	Stanford
2	Berkeley
3	MIT
4	Cornell
5	Amy
6	Bob
7	Craig
8	Doris
9	Edward
10	Fay
11	Gary
12	Helen
13	Irene
14	Jay
15	Amy
16	Craig

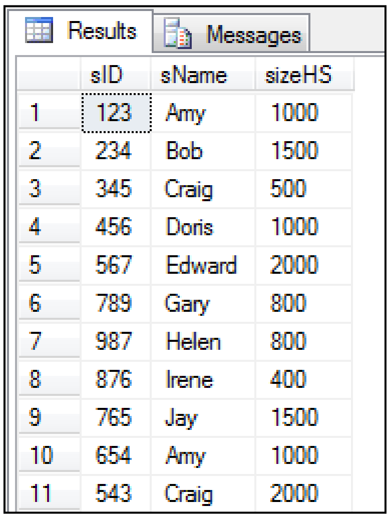
Figure 25. View of the relational representation of the view operation (relational algebra) *union* with attribute data *sName* and *sName* from *College* and *Student* relations obtained after the 23rd complex (sub)query is set (Source: Authors)

The existing *exists* operator was applied to the set 24th complex query (below, in Croatian). Operator *exists - true* if the subquery contains at least one record (Figure 26).

select sID, sName, sizeHS

from Student S1

*where exists (select * from Student S2 where S2.sizeHS < S1.sizeHS);*[7]



Results | Messages

	sID	sName	sizeHS
1	123	Amy	1000
2	234	Bob	1500
3	345	Craig	500
4	456	Doris	1000
5	567	Edward	2000
6	789	Gary	800
7	987	Helen	800
8	876	Irene	400
9	765	Jay	1500
10	654	Amy	1000
11	543	Craig	2000

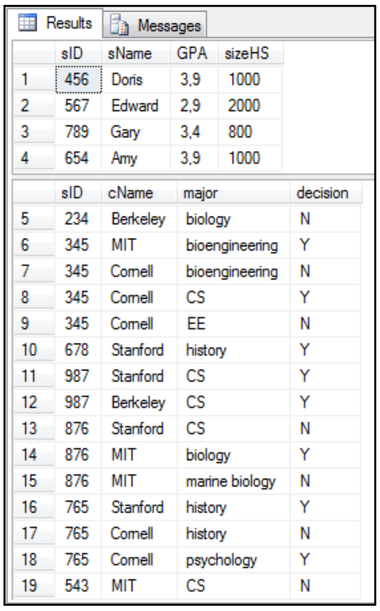
Figure 26. View of the relational representation of the view the subquery where the *exists* operator is used obtained after the 24th complex (sub)query is set (Source: Authors)

With the 25th complex subquery set (below, in Croatian), the data is written to a new relation that does not have data for the *sID* attribute the same as the records in the *Apply* relation (Figure 27).

*select * from Student*

where sID not in (select sID from Apply);

*select * from Apply* [7]



Results | Messages

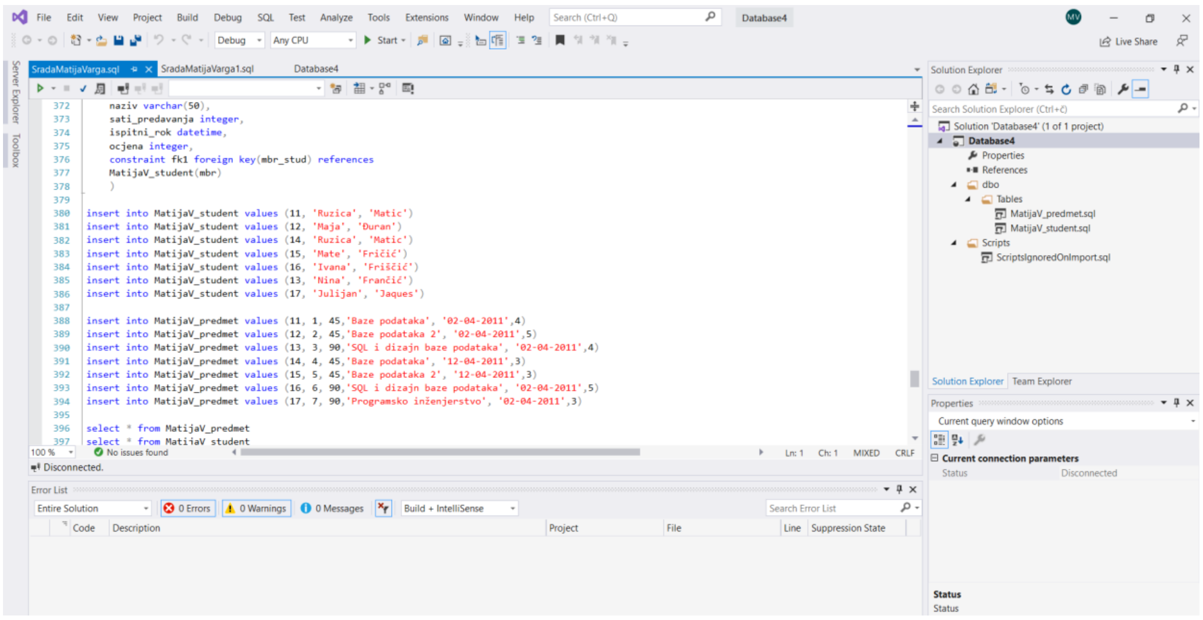
	sID	sName	GPA	sizeHS
1	456	Doris	3,9	1000
2	567	Edward	2,9	2000
3	789	Gary	3,4	800
4	654	Amy	3,9	1000

	sID	cName	major	decision
5	234	Berkeley	biology	N
6	345	MIT	bioengineering	Y
7	345	Cornell	bioengineering	N
8	345	Cornell	CS	Y
9	345	Cornell	EE	N
10	678	Stanford	history	Y
11	987	Stanford	CS	Y
12	987	Berkeley	CS	Y
13	876	Stanford	CS	N
14	876	MIT	biology	Y
15	876	MIT	marine biology	N
16	765	Stanford	history	Y
17	765	Cornell	history	N
18	765	Cornell	psychology	Y
19	543	MIT	CS	N

Figure 27. View of the relational representation of the view the subquery using the *not* and *in* operators[7] obtained after the 25th complex (sub)query is set (Source: Authors)

[7] *Not* to negate other logical operators. *True* if the operand is in the default list of values.

Figure 28 shows a file called “*SradaMatijaVarga.sql*” that created the database and shows the application of the *DDL, DQL, DML* and *DCL* structural languages[8], and also shows the appearance of the “*Visual Studio 2019*” *user interface*.



```sql
    naziv varchar(50),
    sati_predavanja integer,
    ispitni_rok datetime,
    ocjena integer,
    constraint fk1 foreign key(mbr_stud) references
    MatijaV_student(mbr)
    )

insert into MatijaV_student values (11, 'Ruzica', 'Matic')
insert into MatijaV_student values (12, 'Maja', 'Đuran')
insert into MatijaV_student values (14, 'Ruzica', 'Matic')
insert into MatijaV_student values (15, 'Mate', 'Fričić')
insert into MatijaV_student values (16, 'Ivana', 'Friščić')
insert into MatijaV_student values (13, 'Nina', 'Frančić')
insert into MatijaV_student values (17, 'Julijan', 'Jaques')

insert into MatijaV_predmet values (11, 1, 45,'Baze podataka', '02-04-2011',4)
insert into MatijaV_predmet values (12, 2, 45,'Baze podataka 2', '02-04-2011',5)
insert into MatijaV_predmet values (13, 3, 90,'SQL i dizajn baze podataka', '02-04-2011',4)
insert into MatijaV_predmet values (14, 4, 45,'Baze podataka', '12-04-2011',3)
insert into MatijaV_predmet values (15, 5, 45,'Baze podataka 2', '12-04-2011',3)
insert into MatijaV_predmet values (16, 6, 90,'SQL i dizajn baze podataka', '02-04-2011',5)
insert into MatijaV_predmet values (17, 7, 90,'Programsko inženjerstvo', '02-04-2011',3)

select * from MatijaV_predmet
select * from MatijaV_student
```

Figure 28. View of a file called “*SradaMatijaVarga.sql*” and view of a “*Visual Studio 2019*” *user interface* layout obtained (Source: Authors)

POSSIBLE CASE OF DAMAGE AND RENEWAL OF DATABASE

Database corruption can occur for a variety of reasons, such as a failure of the computer hardware partition (disk and other storage media) or the occurrence of a system software error. The database may be damaged by malicious or accidental behavior. Regardless of the causes and causes of the damage, the database must return to a state of preserved physical integrity. In order for the database to be restored, it is necessary to first back up the data from the database (which must be protected by cryptographic encryption) to some medium outside the database. Therefore, the entire database is backed up to a separate medium periodically, and any changes to the database are recorded in the change log [5]. For business companies and other organizations, it is usually recommended that the backup of data from the database to the media be done on average every five days. But, backing up your data every five days is too rare. The question is what would happen if the big company remained without important information for five days. Any amount less than the loss of data would pay off to back up your data more often. It’s safer to back up your data every day. The recommendation for non-IT companies is to hire their own IT manager to handle the database administration, i.e. the database administrator, who can take care of the backups and be responsible for all data and information (or knowledge) in the database (which must be

[8] According to source [9]: GeeksforGeeks. URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (3/2/2020), “DDL, DQL, DML and DCL structural languages are abbreviation for SQL commands, which are mainly categorized into next four categories: DDL - Data Definition Language, DQL - Data Query Language, DML - Data Manipulation Language, DCL - Data Control Language.”

protected by cryptographic encryption). The "(Object) Relational Database Management Systems" (abbreviation: (O)RDBMS) and RDBMS in business companies (and other organizations) need to be available every minute, so backups are enabled for 24 hours (for every day, week and month during the whole year).

Backups can be made during operation. In order to prevent malicious intrusion of companies into the information system, it is necessary to change passwords minimally every month. Also, for each part of the application there must be a separate password and authority to access individual modules. Only certain employees have the right to access certain program modules. All passwords in business companies (and other organizations) should be set to contain a combination of uppercase and lowercase letters and numbers. To protect all the data in the database (which must be protected by cryptographic encryption), business companies (and other organizations) minimally use one (light) server (net) antivirus protection application programme and different (heavier) antivirus protection application programme for client computers (Personal Computer – PC, or notebook/laptop/tablet) and central firewall system, also. All antivirus software updates must be up-to-date, and central firewall system (also). Each time when computers (server and clients) are used, all system user must update the antivirus programmes. Also, disk scan is required at least once a week, i.e. a complete scan of the disk.

PROTECTION OF THE DATABASE FROM UNAUTHORIZED ACCESS

Databases must be protected against unauthorized access by malicious (physical and/or legal) persons. In addition to physical protection, there is also a software security method, in which certain software is embedded in some (O)RDBMS or RDBMS. It restricts the use of the database to users who have the ability to physically access the computer (or computer terminals), in fact to server and to client computers (PC, or notebook/laptop/tablet). There are several ways to protect the encrypted database: user identification, various security mechanisms, and authorizations. When it comes to protecting a database by identifying users, it is generally known that each user of the database has his (or her) *username* and *password*. The user must identify himself/herself with his/her *username* and *password* in some (O)RDBMS or RDBMS, thus proving his/her identity. If the user does not enter a *username* and *password* corresponding to the list in some (O)RDBMS or RDBMS, that system refuses to give his/her permission to operate the database.

Many database management systems can be managed over the Internet/intranet/extranet. Also, it is common knowledge that a database can be accessed from a remote computer (PC, or notebook/laptop/tablet) that can be located anywhere in the world. How to protect a database that can be accessed online (over the Internet/intranet/extranet)? We can temporarily protect it by turning off the network capabilities of the some (O)RDBMS or RDBMS, or to restrict options so that only local users (developers) can access the database (i.e. local network clients), and to restrict options so that only some external users may be allowed access to the database, but only with client identification, by using encrypted communication such as *SSL/SSH*, duplicate keys, etc. When it comes to *views*, the user can be *granted access* to only a certain part of the data, what means that other parts of the database are unavailable to a specific user.

The protection of the database by *granting authority* determines what the user can do with the database information at his (or her) disposal, and where security permissions can be: *read/select, update, insert,* and *delete*. The (O)RDBMS or RDBMS must "remember", i.e. have a stored list of authorizations for each user and each relation from a specific view. If the user tries to perform an activity that is not authorized, the (O)RDBMS or RDBMS will not execute it, but will print a warning that the user is not authorized to perform the specified activity. In most cases, a developer named *"Database Administrator"* is concerned about database protection. The *database administrator* has a user list, provides view options, and regulates authorizations. Only the *database administrator* has the ability to regulate authority, which ordinary users do not have. In order to protect the database, it is necessary to properly monitor, analyse changes in the data by period and record the actions of the person who can access the database at the moment when he (or her) intentionally (or not) wishes to cause harm.

The (O)RDBMS or RDBMS can be vulnerable, also. The overall functionality of some (O)RDBMS or RDBMS can also be an indicator of its security. By backup (system) we can protect data in the database, also. Today, systems can be used to automatically back up and recover the system, providing the ability to back up (archived) data for up to 5 years and eliminate the costs caused by human errors associated with data storage. This kind of protection provided by systems to automatically back up and recover the system is better than backing up the entire database, because the system restores only the data that has been changed. The (O) RDBMS or RDBMS must provide and guarantee serial use (serial capability or batch execution)[9], whereby serial capability is achieved by special control during the execution of transactions.

CONCLUSIONS

SQL query language (opposite to "Query by Example" (QBE) language) consists of English words that can be divided minimally into three groups, which are (three basic groups of SQL): *DCL*, *DML*, and *DDL* [1] [9], where *DCL* is a *Data Control Language*, *DML* is a *Data Manipulation Language*, *DDL* is a *Data Definition Language* for defining data, and is in some ways almost a fundamental part of these languages because it serves to define the structure of an object in a database without which the database would not exist [1] [9]. The manuals or papers entitled something like this (theme) "*Applying SQL Complex Queries to Access Database Data Using MS SQL Server 2019*" usually lists the most commonly used *DML* keywords, and the most commonly used *SELECT* query, where SELECT is used to retrieve rows from relationships to find the desired data. Also, the most commonly associated with the key word *SELECT* are these (sub)keywords*: from, where, group by, having* and *order by*. Also, usually at the end of these kind of manuals or papers are detail explanations about the controls over databases and mentions about the protection of the database against unauthorized access to the data, etc.

[9] Have several transactions run concurrently in a multiuser database so that certain parts of those transactions are timed mixed, and if the ultimate effect of their execution is the same as if they were executed batch wise or sequentially, we say that it is a batch or serializable execution of transactions.

REFERENCES

[1] **CARIC, T. and BUNTIC, M.** (2015). Introduction to Relational Databases. Zagreb, 2015. Expert reviewers: prof. dr. sc. Hrvoje Gold doc. de. sc. Edouard Ivanjko. 2015. (In Croatian: Carić, T. and Buntić, M. Uvod u relacijske baze podataka. Zagreb, 2015. Stručni recenzenti: prof. dr. sc. Hrvoje Gold doc. de. sc. Edouard Ivanjko.)

[2] **VARGA, M.** (2013). Bernstein algorithm for vertical 3NF normalization by synthesis. Review article. Economic Bulletin. EFOS. God. XXVI, BR. 1/2013. pp. 315-322. URL: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=159539. (3/2/2020). (In Croatian: Varga, M. Bernsteinov algoritam za vertikalnu 3NF normalizaciju sintezom. Pregledni znanstveni članak. Ekonomski vjesnik. EFOS. God. XXVI, BR. 1/2013. str. 315-322. URL: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=159539. (3.2.2020))

[3] **MANGER, R. DATABASE.** (2012). The script. Faculty of Natural Sciences and Mathematics. 1st edition. Zagreb 2012. University of Zagreb. Element. Department of Mathematics. (In Craotian: Manger, R. Baze podataka. Skripta. Prirodoslovno-matematički fakultet. 1. izdanje. Zagreb 2012. godina. Sveučilište u Zagrebu. Element. Matematički odsjek.)

[4] **SKOCIR, Z., MATASIC, I. and VRDOLJAK, B.** (2007). Organization of data processing. Zagreb: Merkur A.B.D., FER, 2007. ((In Croatian: Skočir, Z., Matasić, I.; Vrdoljak, B. Organizacija obrade podataka. Zagreb: Merkur A.B.D., FER, 2007.)

[5] **VARGA, M., CURKO, K., PANIAN, Z., CERIC, V., VUKSIC BOSILJ, V., SRICA, V., POZGAJ, Z ., STRUGAR, I., SPREMIC, M., PEJIC BACH, M., VLAHOVIC, N. and JAKOVIC, B.** (2007). Informatics in business. Zagreb: University of Zagreb, 2007. (In Croatian: Varga, M.; Ćurko, K..; Panian, Ž.; Čerić, V.; Vukšić Bosilj, V.; Srića, V.; Požgaj, Ž.; Strugar, I.; Spremić, M.; Pejić Bach, M.; Vlahović, N.; Jaković, B. Informatika u poslovanju. Zagreb: Sveučilište u Zagrebu, 2007.)

[6] **SQL subqueries** (In Croatian: pod-upiti). (2011). URL: https://support.office.com/hr-hr/article/sql-podupiti-d41fc0b1-1c88-40d4-bbd1
951de6e94e2a?ocmsassetID=HA001231513&CorrelationId=10e79c01-2685-4c9a-9de7-4328db44c503&ui=hr-HR&rs=hr-HR&ad=HR. (3/2/2020).

[7] **Widom, J. SQL.** (2020). Aggregation. URL: http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=IntroToDatabases&doc=/docs/course-materials.html. (3/2/2020).

[8] **Company IBM.** (2020). URL: https://www.ibm.com/hr-en. (3/2/2020).

[9] **GeeksforGeeks.** (2020). URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (3/2/2020).