Database Homework - Assignment 2

Natalia Nazarczuk, 305791

May 23, 2021

1 Introduction

The goal of the task was to prepare a design of a management system for a university, that would manage courses, students and academic teachers. Following that, I proposed a set of appropriate indexes improving the speed of retrieving data from a database, performed examples of rows modification for a selected tables, prepared SELECT queries and a stored procedure.

2 Solution

2.1 *Creating database*

My database consists of many tables representing university structures and relations between them. The main table is Department which has a one to many relation to Room, Course, Student and Teacher tables. The Class table portrays a specific lesson in timetable and following that each teacher and student have a history of the classes they taught or took. Courses have prerequisites that are other courses. The Qualified table links teachers to the courses they are able to teach and the Group table holds a group of students with the teacher assigned as a leader. The source code for database creation can be found in the files titled: uni_create and uni_insert.

2.2 *Modification of rows for a selected table*

To manipulate data in objects such as tables, we can use DML – Data Manipulation Language, which includes commands such as INSERT, UPDATE, DELETE. Firstly, I inserted appropriate sample data into each table with the code in a file uni_insert. Then to modify current data, I used transaction. If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database. If a transaction encounters errors and must be canceled or rolled back, then all of the data modifications are erased. I updated row 'room_type' in Room, row 'day' in Class, and row 'grade' in Taken_History with the below code, checking the results before and after the update.

```
BEGIN TRANSACTION

SELECT * FROM Room WHERE room_type = 'lab';
SELECT * FROM Class WHERE "day"='Tuesday';
SELECT * FROM Taken_History WHERE grade = 5;

UPDATE Room SET room_type = 'lab' WHERE room_id>50;
UPDATE Class SET "day" = 'Tuesday' WHERE semester=2;
UPDATE Taken_History SET grade = 5 WHERE student_id = 3;

SELECT * FROM Room WHERE room_type = 'lab';
SELECT * FROM Class WHERE "day"='Tuesday';
SELECT * FROM Taken_History WHERE grade = 5;

COMMIT;
```

2.3 *Design of indexes*

SQL indexes are used to retrieve data from a database very fast. Indexing a table or view is one of the best ways to improve the performance of queries and applications. A SQL index is a quick lookup table for finding records users need to search frequently. All of my indexes are in uni_queries file. The syntax for index creation is presented below.

```
CREATE INDEX index_name ON table_name(column_name);
```

To improve the performance of my database I have decided on the following indexes:

1. 'course_ID' in a Class table - Foreign key columns are usually non-unique because they're on the "many" side of the one-to-many relationship so it will be useful.
2. 'class_ID' in tables Taken_History and Teach_History - Similarly as with 'course_ID', indexes on foreign key columns speed up operations.
3. covering index on 'name' and 'surname' from tables Student and Teacher to easily retrieve full name of a student or teacher

2.4 *SQL SELECT reports*

2.4.1 *A total number of students at each department for each year in the database.*

```
select d.department_ID, d."name", in2020.student_count, in2021.student_count
from Department d
join(
        select s.department_ID,  count(distinct th.student_ID) as student_count
        from Student s
        inner join Taken_History th on th.student_ID=s.student_ID
```

```
        inner join Class cl on cl.class_ID = th.class_ID
        where cl."year" =2020
        group by s.department_ID
        ) as in2020 on in2020.department_ID=d.department_ID
join(
        select s.department_ID, count(distinct th.student_ID) as student_count
        from Student s
        inner join Taken_History th on th.student_ID=s.student_ID
        inner join Class cl on cl.class_ID = th.class_ID
        where cl."year" =2021
        group by s.department_ID
        ) as in2021 on in2021.department_ID=d.department_ID
```

2.4.2 *A list of 10 courses taken by most students in a single academic year.*

```
select top 10 * from(
(
        select top 10 c.course_ID, c.course_name, count(distinct th.student_ID)
        join Class cl on cl.course_ID=c.course_ID
        join Taken_History th on th.class_ID=cl.class_ID
        where cl."year"=2020
        group by c.course_ID, c.course_name
        order by students_count desc
)
union all
(
        select top 10 c.course_ID, c.course_name, count(distinct th.student_ID)
        join Class cl on cl.course_ID=c.course_ID
        join Taken_History th on th.class_ID=cl.class_ID
        where cl."year"=2021
        group by c.course_ID, c.course_name
        order by students_count desc
)
) as x
order by students_count desc;
```

2.4.3 *A list of teachers whose classes were taken by more than 150% of average number of students per teacher.*

```
select t.teacher_ID, t."name", t.surname from Teacher t
join Teach_History teh on teh.teacher_ID=t.teacher_ID
join Class c on c.class_ID=teh.class_ID
```

```
join Taken_History th on th.class_ID=c.class_ID
group by t.teacher_ID, t."name", t.surname
having count(th.student_ID) > 1.5*
(
        select avg(x.StudentSum)
        from(
                select t.teacher_ID, count(th.student_ID) as StudentSum
                from Teacher t
                inner join  Teach_History teh on teh.teacher_ID=t.teacher_ID
                inner join Class c on c.class_ID=teh.class_ID
                inner join Taken_History th on th.class_ID=c.class_ID
                group by t.teacher_ID
                ) as x
)
```

2.4.4 *Maximum number of hours taught in a single room in a week, and the year and the semester when it happened.*

```
select r.room_ID, count(c.class_ID) as classes_count, c."year", c.semester
from Class c join Room r on r.room_id=c.room_ID
group by r.room_ID, c."year", c.semester
order by classes_count desc
```

2.4.5 *Total number of students who have classes from 8:00 to 20:00 on Monday, without a break of at least 1h.*

```
select s.student_ID, s."name", s.surname from Student s
join Taken_History th on th.student_ID=s.student_ID
join Class c on c.class_ID=th.class_ID
where c."day"='Monday'
group by s.student_ID, s."name", s.surname
having sum(c.duration)>11
```

3 Stored procedure

The below stored procedure called 'TeacherAssignment' is used for automatic assignment of another teacher to a class because of a sick-leave of the original teacher. The procedure takes the class number as an input argument and looks for a teacher who is qualified to teach the same subject and is available at the time of the class. If such a teacher is not found, the class is marked as canceled that day. Else the report about modified rows is presented. The complete code is in a file uni_procedure.

```
create procedure TeacherAssignment(
@class_id int
) as

begin

declare @class int
declare @classday varchar(40)
declare @classhour int
declare @course int
declare @originalteacher int
declare @teacher int

select @course = c.course_id from Class c where class_ID = @class_id
select @class = c.class_id from Class c where class_ID = @class_id
select @classday = c."day" from Class c where class_ID = @class_id
select @classday = c."hour" from Class c where class_ID = @class_id
select @originalteacher = th.teacher_ID from Teach_History th where th.class_ID=

select top 1 @teacher = q.teacher_ID from Qualified q
where q.course_ID=@course
and q.teacher_ID != @originalteacher
and q.teacher_ID not in
(
        select t.teacher_ID from Teacher t join
        Teach_History th on th.teacher_ID=t.teacher_ID
        join Class c on c.class_ID= th.class_ID
        where c."day"=@classday and c."hour"=@classhour
)

if @teacher IS NULL
        begin
        print 'There is no other available teacher. Class is canceled.'
        select c.class_ID, '1' as Canceled from Class c join Teach_History th
        on th.class_ID=c.class_ID where c.class_ID=@class_id;
        end
else
        begin
        print 'Found teacher:'
        print @teacher
        update Teach_History set teacher_ID=teacher_ID where class_ID=@class
        select * from Class c join Teach_History th on th.class_ID=c.class_ID
        where c.class_ID=@class_id
        end
end
```

4 Declaration

I certify that this assignment is entirely my own work, performed independently and without any help from the sources which are not allowed.

5 Test confirming the correctness of solution

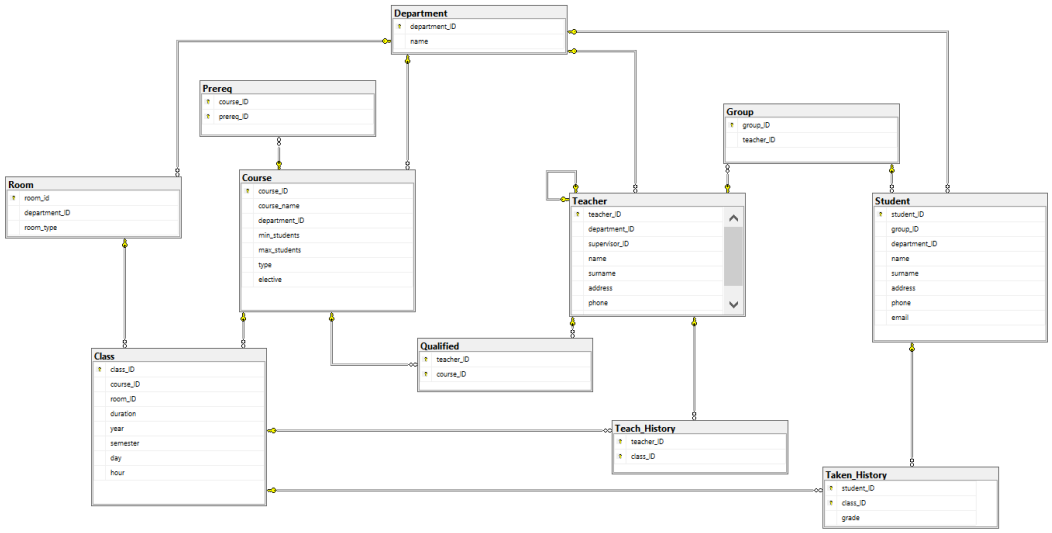

Figure 1: Entity diagram

	room_id	department_ID	room_type
1	35	2	lab
2	41	5	lab
3	54	4	lab
4	67	1	lab
5	72	3	lab

	class_ID	course_ID	room_ID	duration	year	semester	day	hour
1	2	15	83	1	2020	1	Tuesday	13
2	10	13	54	2	2020	1	Tuesday	10
3	12	7	83	2	2020	3	Tuesday	10
4	16	5	83	2	2020	3	Tuesday	13

	student_ID	class_ID	grade
1	3	5	5
2	8	10	5
3	11	2	5
4	15	6	5
5	18	5	5
6	20	6	5

	room_id	department_ID	room_type
1	35	2	lab
2	41	5	lab
3	54	4	lab
4	67	1	lab
5	72	3	lab
6	83	5	lab
7	92	2	lab

	class_ID	course_ID	room_ID	duration	year	semester	day	hour
2	2	15	83	1	2020	1	Tuesday	13
3	8	3	22	1	2021	2	Tuesday	11
4	10	13	54	2	2020	1	Tuesday	10
5	12	7	83	2	2020	3	Tuesday	10
6	13	4	67	1	2021	2	Tuesday	8
7	16	5	83	2	2020	3	Tuesday	13
8	18	14	49	2	2021	2	Tuesday	11
9	19	13	72	2	2021	2	Tuesday	10

	student_ID	class_ID	grade
1	3	5	5
2	3	8	5
3	8	10	5
4	11	2	5
5	15	6	5
6	18	5	5
7	20	6	5

Figure 2: Before and after modification of rows

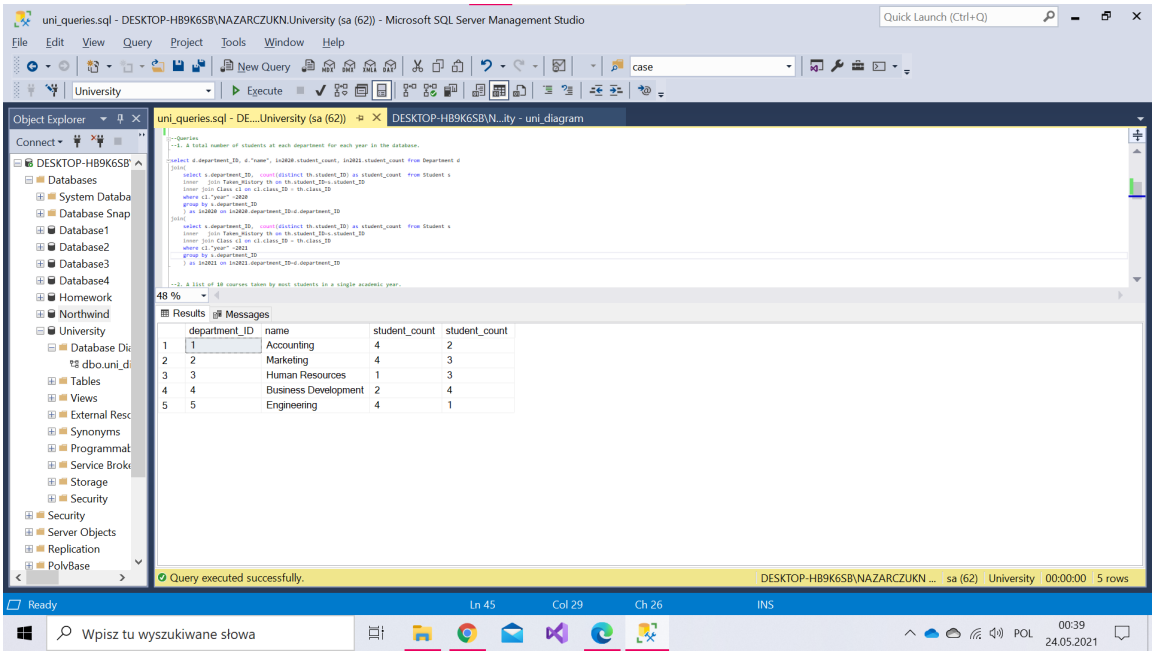

Figure 3: Result of query 1

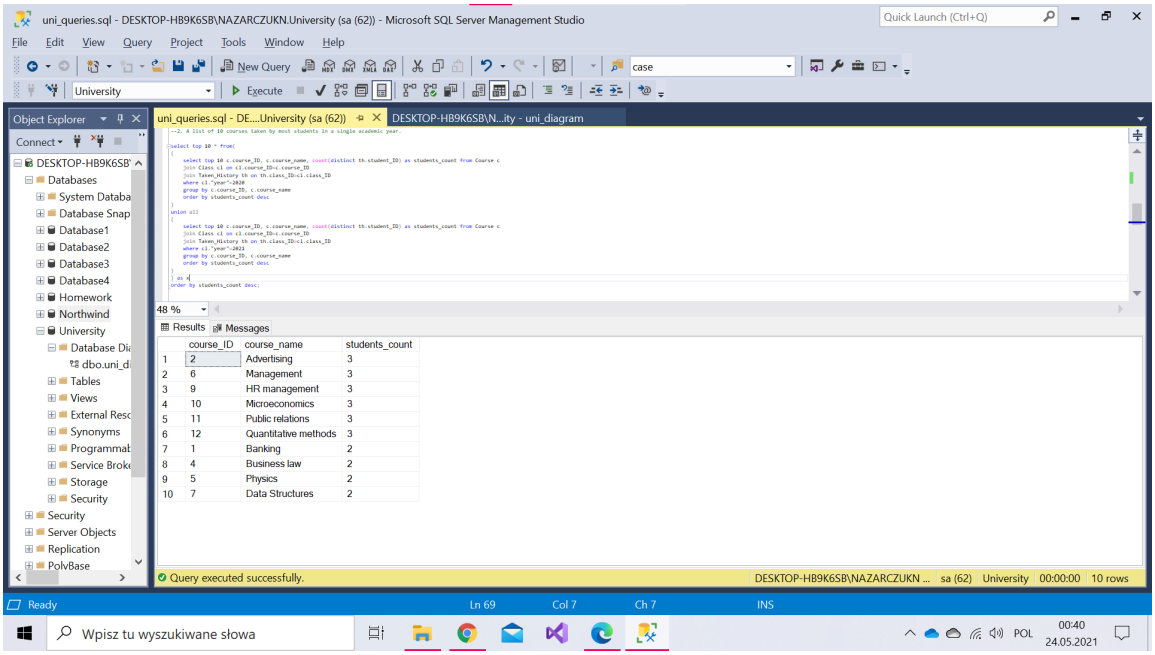

Figure 4: Result of query 2

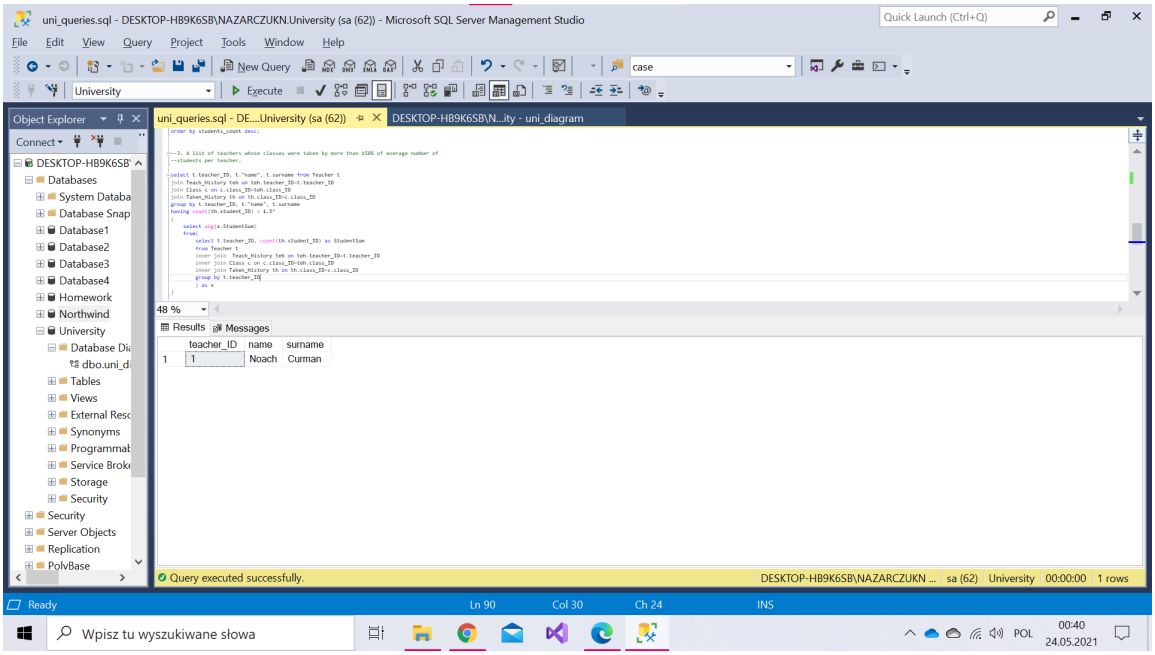

Figure 5: Result of query 3

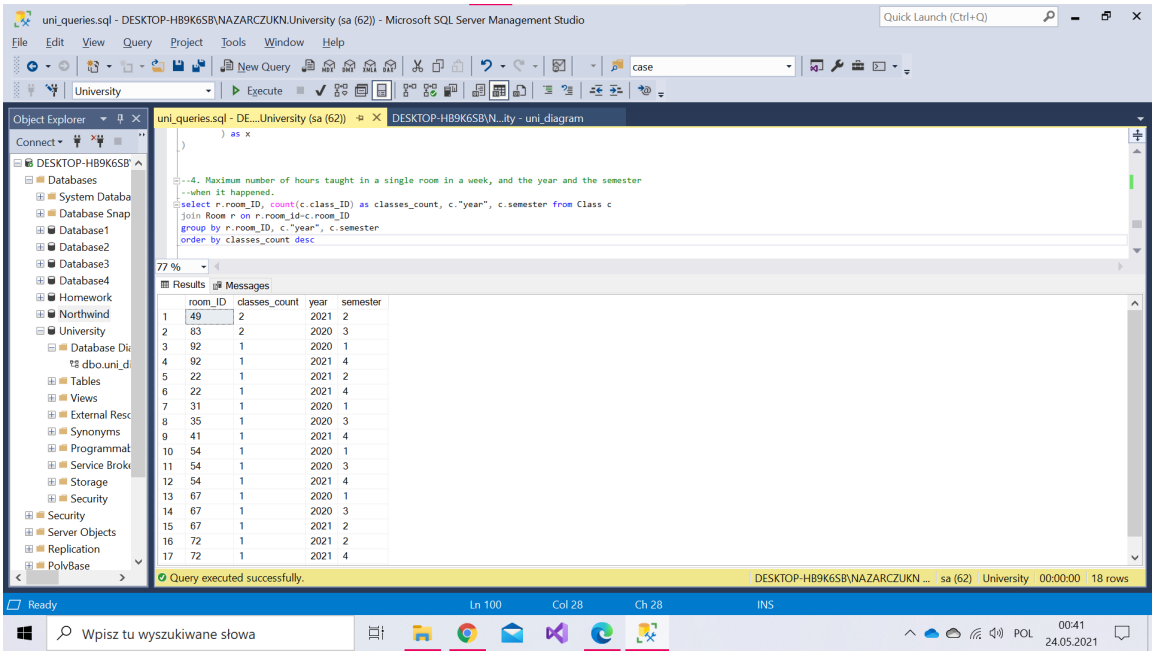

Figure 6: Result of query 4

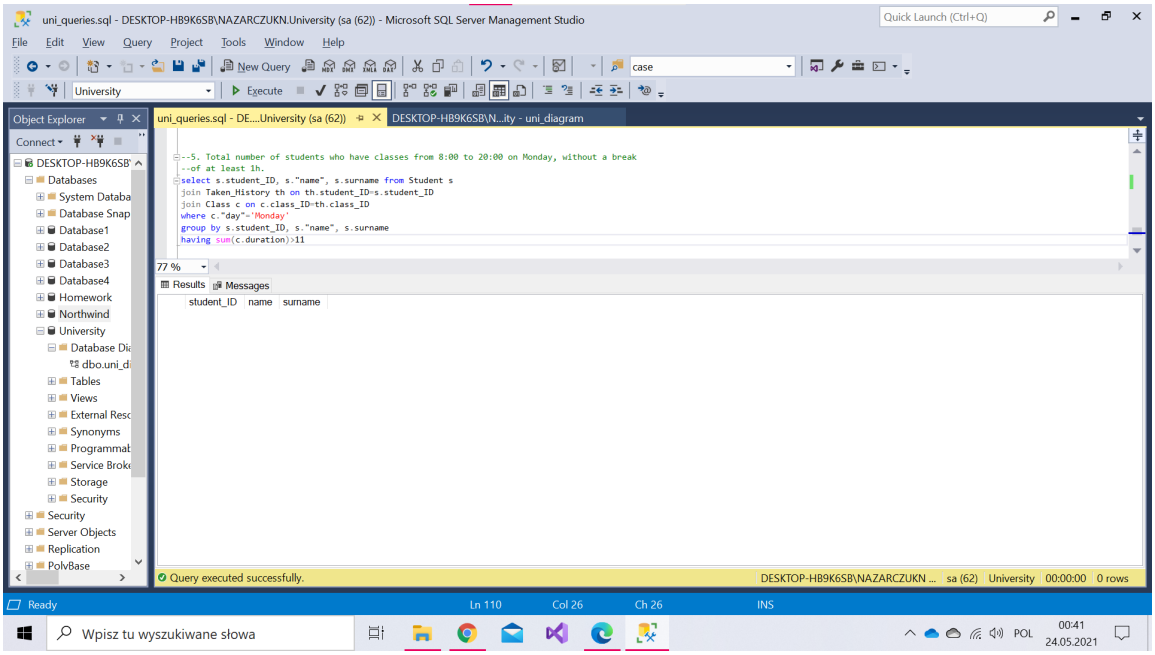

Figure 7: Result of query 5

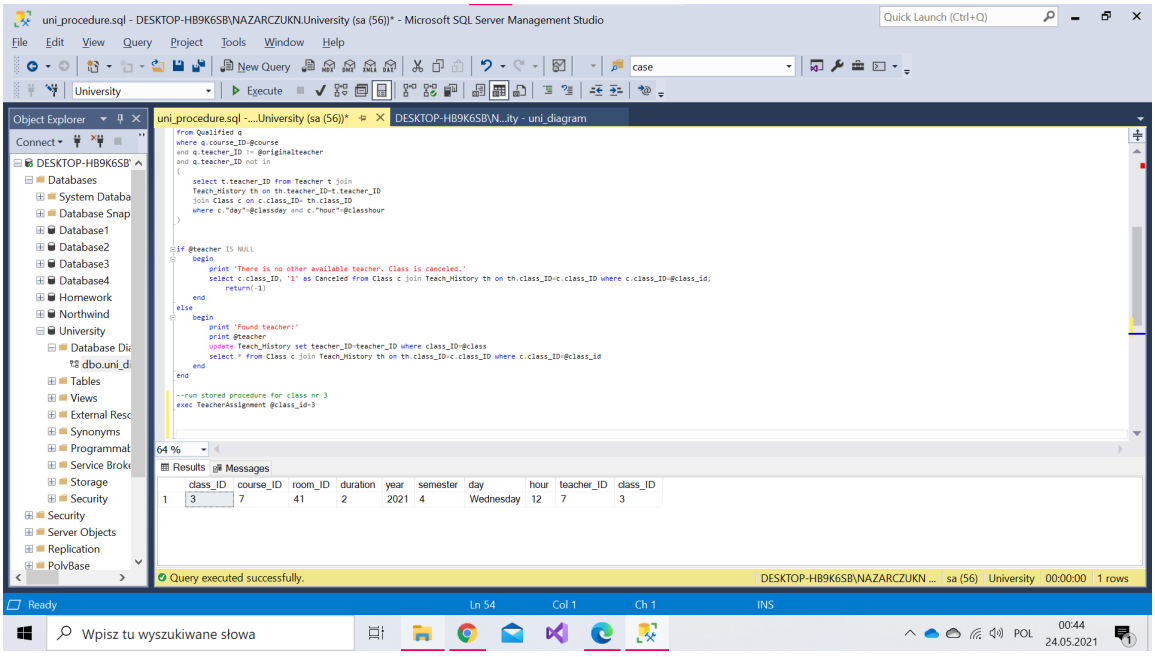

Figure 8: Stored procedure