Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Ingineria Software

REPORT

Laboratory work Nr.7

Theme: Diagrams, schemes and synonyms

Realized by: Popa Eugeniu
FAF-202

Chișinău, 2021

The theoretical part

1. Why are the diagrams used in SQL Server databases?
   SQL Server database diagrams provide the capability to create and manage keys on your tables.

2. The advantages of using schemes.
   SQL Server schemas provide the following benefits: Provides more flexibility and control for managing database objects in logical groups. Allows you to move objects among different schemas quickly. Enables you to manage object security on the schema level.

3. Basic syntax of the schema creation instruction.
   CREATE SCHEMA [IF NOT EXISTS] [database.]schema
   ... [AUTHORIZATION username]
   ... [DEFAULT { INCLUDE | EXCLUDE } [SCHEMA] PRIVILEGES]

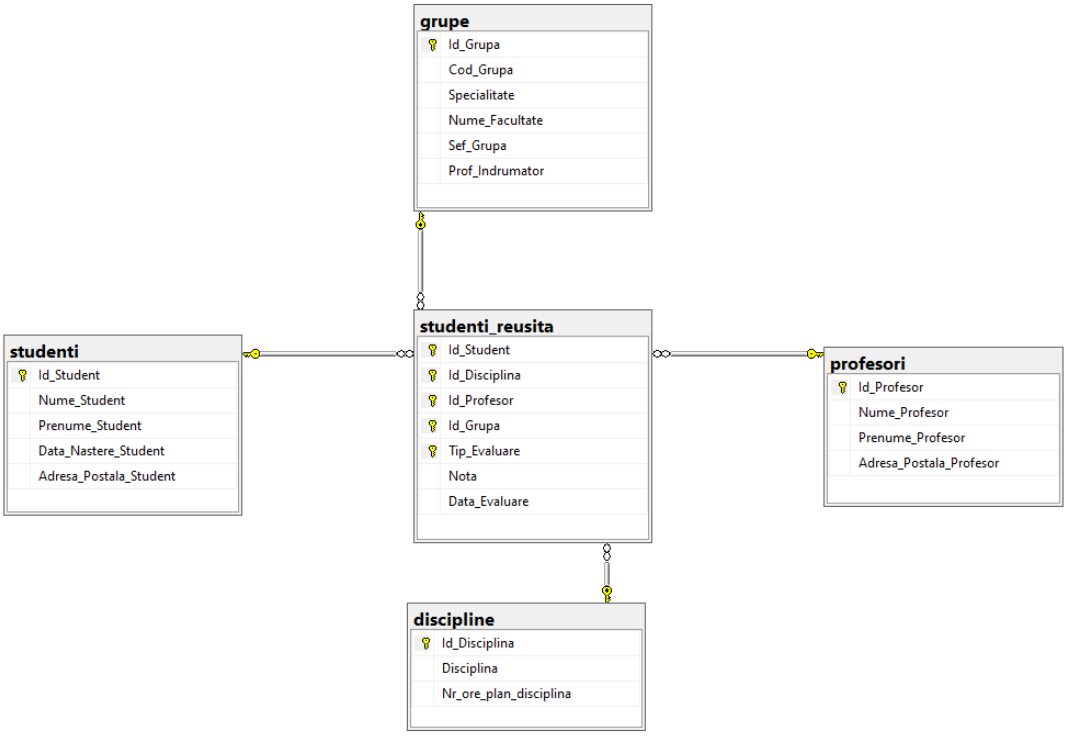
4. Possible constraints on moving objects from one scheme to another and on deleting schemes.
   Moving objects is only possible within a data store. Swapping objects also modifies users access permissions to these objects. All previous access rights are deleted. If the owner of the moved object is an explicitly defined user or role, then they will continue to be the owners of that object. Before deleting a schema, it is important to move or delete the included objects. Attempting to delete a schema containing objects will fail.

5. The advantages of using synonyms.
   Provide a layer of abstraction over the base objects.
   Shorten the lengthy name e.g., a very_long_database_name., with_schema. ...
   Allow backward compatibility for the existing applications when you rename database objects such as tables, views, stored procedures, user-defined functions, and sequences.

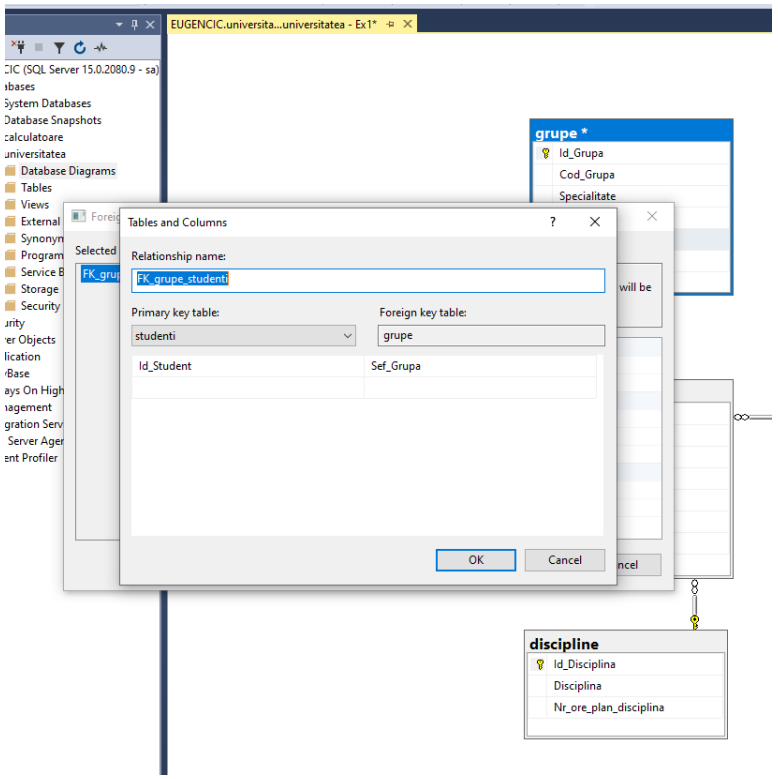
6. The basic syntax of the synonym creation instruction.
   CREATE SYNONYM [schema_name_1.] synonym_name FOR <object>

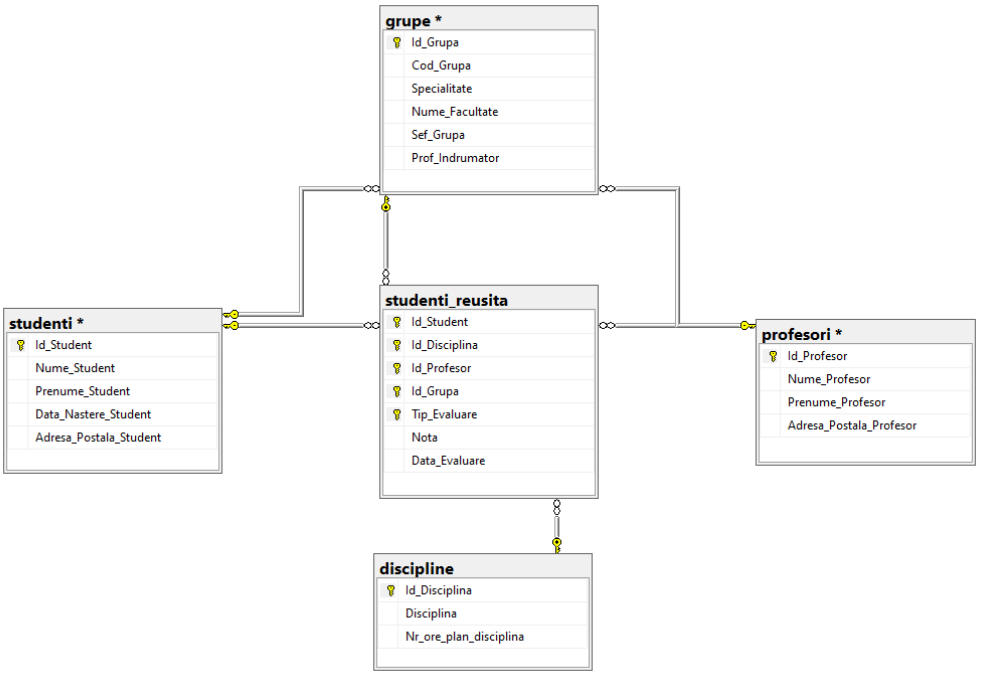
The practical part

1. Create a database diagram using the standard viewer, the structure of which is described at the beginning of the practical tasks in Chapter 4.

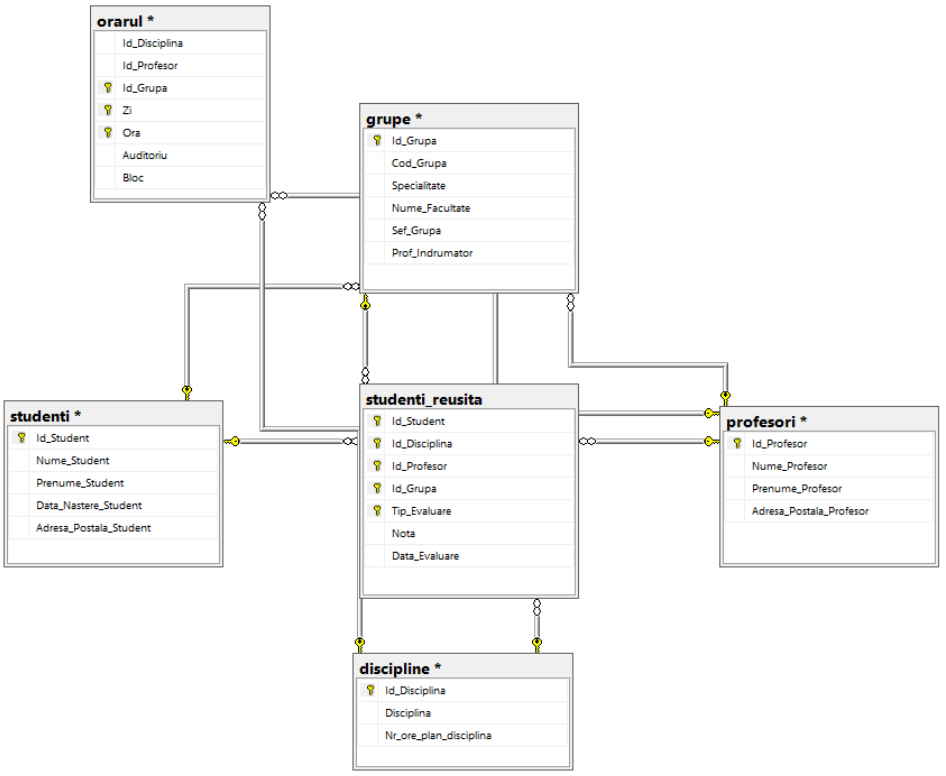


2. Add necessary referential constraints (related to student and teacher tables) for the column Group_Leader and Prof_Guideer columns (task3, chapter 6) in the groups table.

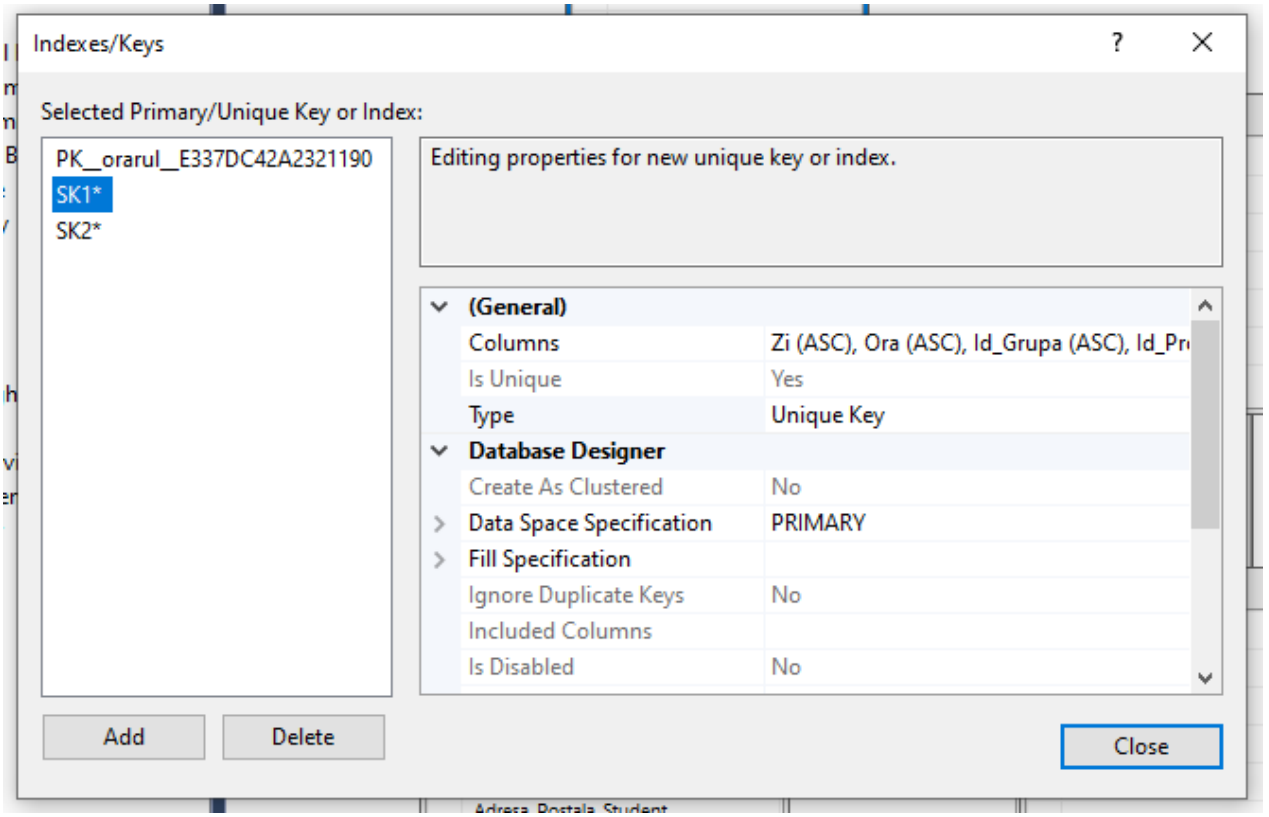




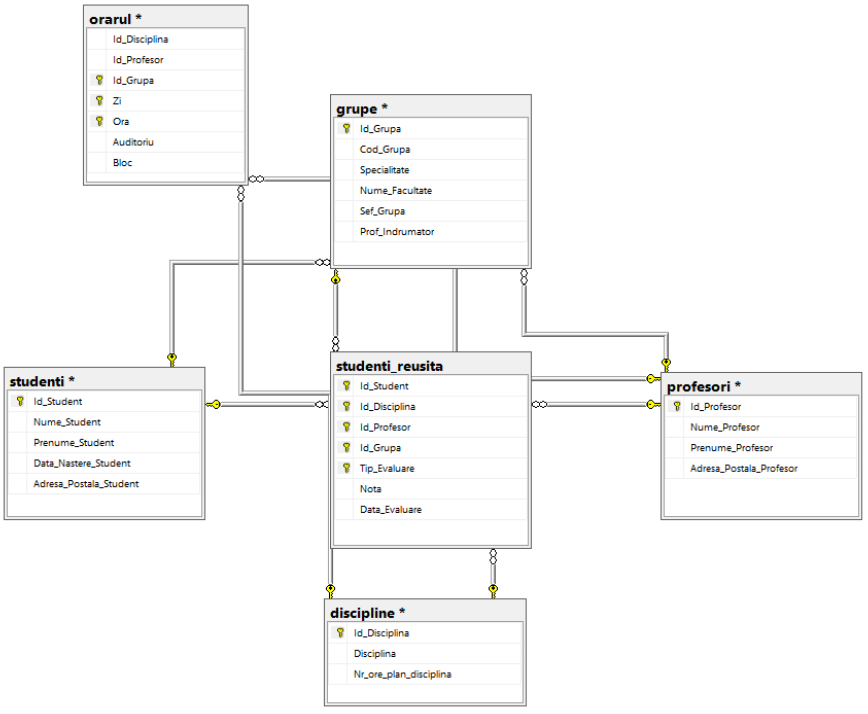
3. To the construction diagram, add the timetable table defined in chapter 6 of this work: the timetable table contains the subject identifier (Id Disciplines), the teacher identifier (Id Professor) and the study block (Block). The table key consists of three fields: group identifier (Group Id), lesson day (Z1), lesson start time (Time), room where the lesson takes place (Auditorium).



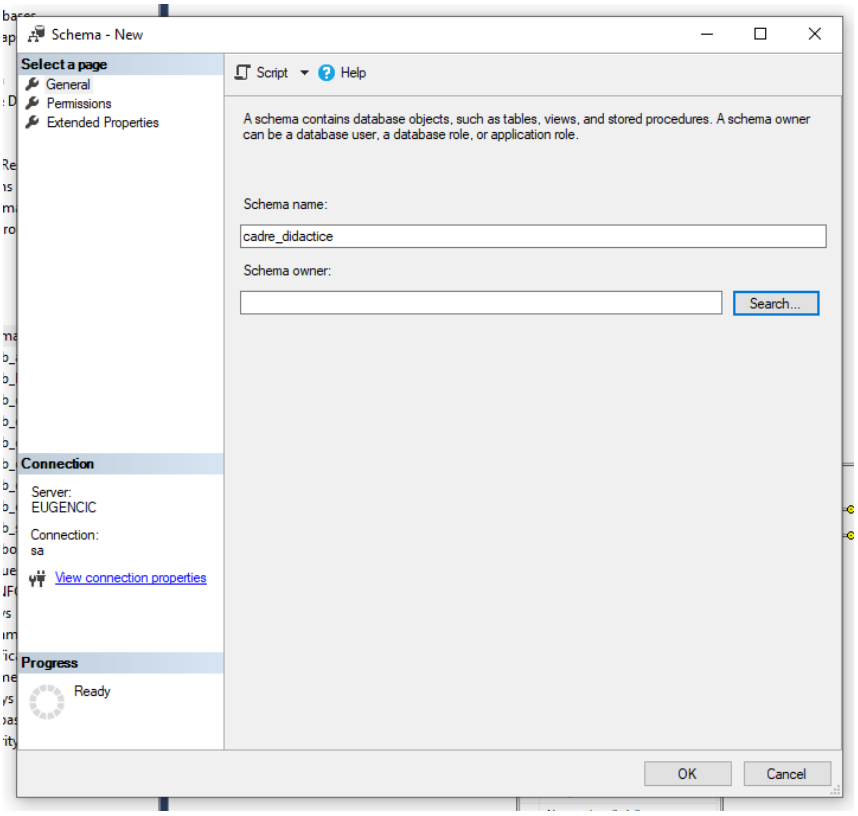
4. The timetable table must contain 2 secondary keys: (Day, Time, Group_Id, Teacher_Id) and (Day, Time, Group Id, Discipline Id). ...



5. The diagram must also define the referential constraints (FK-PK) of attributes Id_Discipline, Id Teacher, Id Group in the table schedule with table attributes respectively.



6. Create, in the university database, three new schemes: teachers, curriculum and students. Transfer the table of teachers from the dbo scheme to the teacher's scheme, taking into account dependencies defined on the mentioned table. In the same way to treat the schedule tables, disciplines that belong to the curriculum plan and tables students, student's success, which belong to student scheme. Write the respective SQL statements.




```
--6.

ALTER SCHEMA cadre_didactice TRANSFER dbo.profesori

ALTER SCHEMA plan_studii TRANSFER dbo.orarul

ALTER SCHEMA plan_studii TRANSFER dbo.discipline

ALTER SCHEMA studenti TRANSFER dbo.studenti

ALTER SCHEMA studenti TRANSFER dbo.studenti_reusita


--SELECT * FROM studenti.studenti_reusita
--SELECT * FROM studenti.studenti
--SELECT * FROM plan_studii.discipline
--SELECT * FROM  cadre_didactice.profesori;
```

7. Modify 2-3 queries on the university database presented in Chapter 4 so that the names of the accessed tables should be explicitly described, taking into account the fact that the tables have been moved to new schemes.

```
--7.

--2

SELECT Disciplina FROM plan_studii.discipline d
ORDER BY d.Nr_ore_plan_disciplina DESC ;

--4
select Disciplina FROM plan_studii.discipline d
WHERE LEN(Disciplina) > 20;

--6
SELECT TOP(5) Nume_Student, Prenume_Student

FROM studenti.studenti_reusita sr INNER JOIN plan_studii.discipline d ON sr.Id_Disciplina = d.Id_Disciplina
            INNER JOIN studenti.studenti s ON s.Id_Student = sr.Id_Student

WHERE sr.Tip_Evaluare = 'Testul 2' AND sr.Nota IS NOT NULL AND d.Disciplina = 'Baze de date '
ORDER BY Nota DESC
```

8. Modify 2-3 queries on the university database presented in Chapter 4 so that the names of the accessed tables should be explicitly described, taking into account the fact that the tables have been moved to new schemes.

```
--8.

CREATE SYNONYM Discipline
FOR universitatea.plan_studii.discipline

CREATE SYNONYM Studenti_Reusita
FOR universitatea.studenti.studenti_reusita

CREATE SYNONYM Studenti
FOR universitatea.studenti.studenti

CREATE SYNONYM Profesori
FOR universitatea.cadre_didactice.profesori

--2.

SELECT Disciplina FROM Discipline d
ORDER BY d.Nr_ore_plan_disciplina DESC ;

--4.
SELECT Disciplina FROM Discipline d
WHERE LEN(Disciplina) > 20;

--6.
SELECT TOP(5) Nume_Student, Prenume_Student

FROM Studenti_reusita sr INNER JOIN Discipline d ON sr.Id_Disciplina = d.Id_Disciplina
            INNER JOIN Studenti s ON s.Id_Student = sr.Id_Student

WHERE sr.Tip_Evaluare = 'Testul 2' AND sr.Nota IS NOT NULL AND d.Disciplina = 'Baze de date '
ORDER BY Nota DESC
```