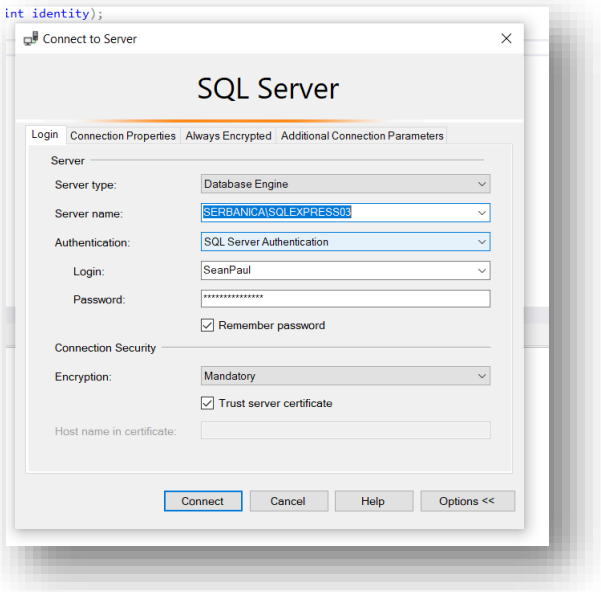
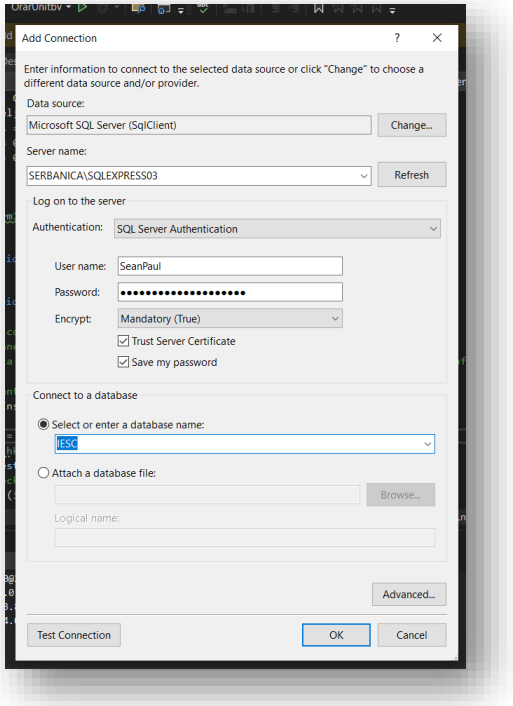
Rezumat 09.08.2024

- Am creat o metodă de autentificare pentru serverul meu local folosind SQL Server Authentification, până acum singura metodă pe care o foloseam pentru conectare era cea standard, selectată implicit de sistem, și anume Windows Authentification.



- În următoare imagine este echivalentul conexiunii din managerul SSMS, dar este în Visual Studio:

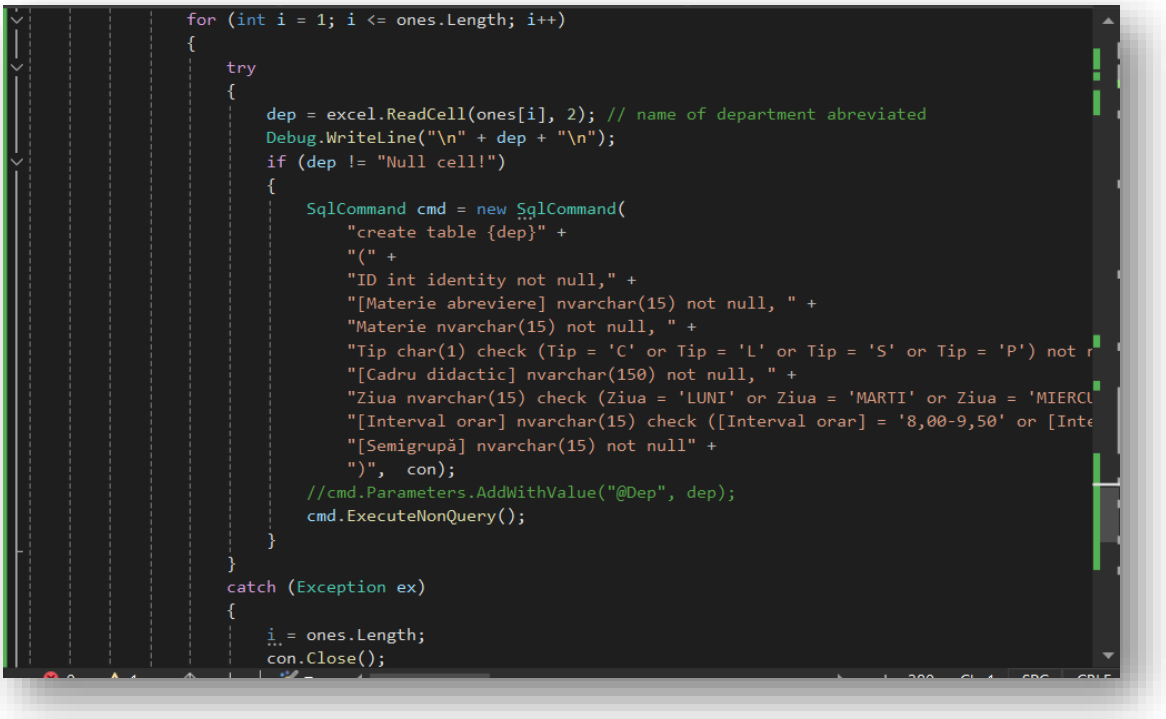


Rezumat 09.08.2024

- Regulile de extragere a datelor din tabel sunt bazate pe instanțe unice, de exemplu, 2020, semestrul 2. Astfel putem stoca în baza de date și locația unde se desfășoară activitatea didactică. Tabelul generat în aplicație va arată în felul următor:

ID	Materie_abreviere	Materie	Tip	Cadru_didactic	Ziua	Interval_orar	Locatie	Semigrupa
1	AMP	Automate si microprograme	C	Itu_L	LUNI	08:00-09:50	VIV7	4LF412A
2	...	...	...	...	...	...	...	...

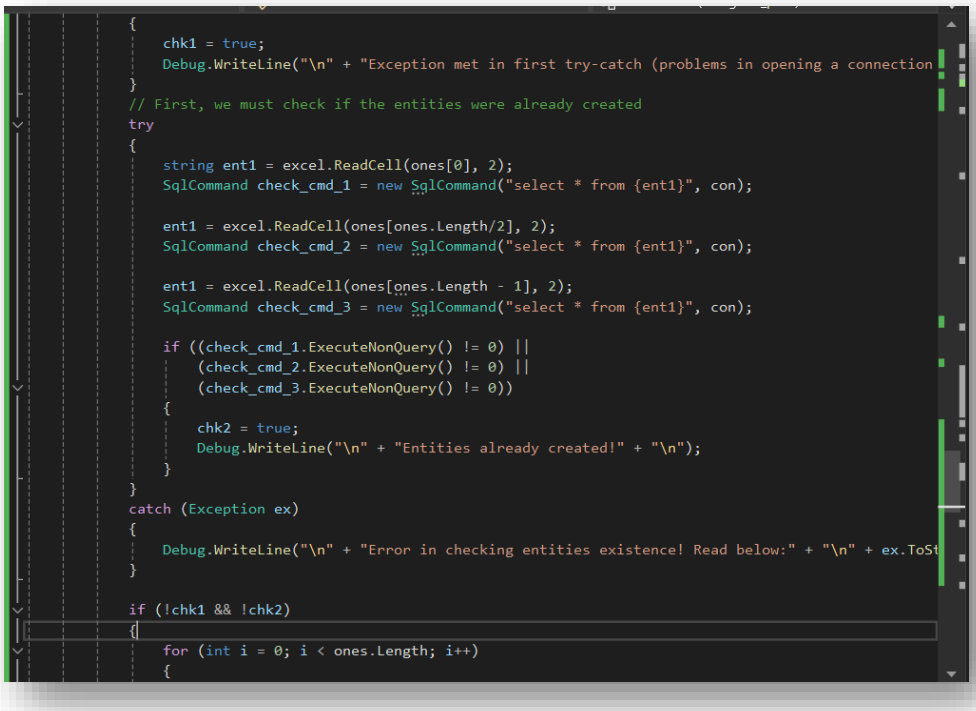
- Codul unei semigrupe este unic, de exemplu, 4LF412A este codul pentru semigrupa mea, acesta îl vom obține prin concatenarea a două celule.



```
for (int i = 1; i <= ones.Length; i++)
{
    try
    {
        dep = excel.ReadCell(ones[i], 2); // name of department abreviated
        Debug.WriteLine("\n" + dep + "\n");
        if (dep != "Null cell!")
        {
            SqlCommand cmd = new SqlCommand(
                "create table {dep}" +
                "(" +
                "ID int identity not null," +
                "[Materie abreviere] nvarchar(15) not null, " +
                "Materie nvarchar(15) not null, " +
                "Tip char(1) check (Tip = 'C' or Tip = 'L' or Tip = 'S' or Tip = 'P') not r
                "[Cadru didactic] nvarchar(150) not null, " +
                "Ziua nvarchar(15) check (Ziua = 'LUNI' or Ziua = 'MARTI' or Ziua = 'MIERCU
                "[Interval orar] nvarchar(15) check ([Interval orar] = '8,00-9,50' or [Inte
                "[Semigrupă] nvarchar(15) not null" +
                ")",  con);
            //cmd.Parameters.AddWithValue("@Dep", dep);
            cmd.ExecuteNonQuery();
        }
    }
    catch (Exception ex)
    {
        i = ones.Length;
        con.Close();
```

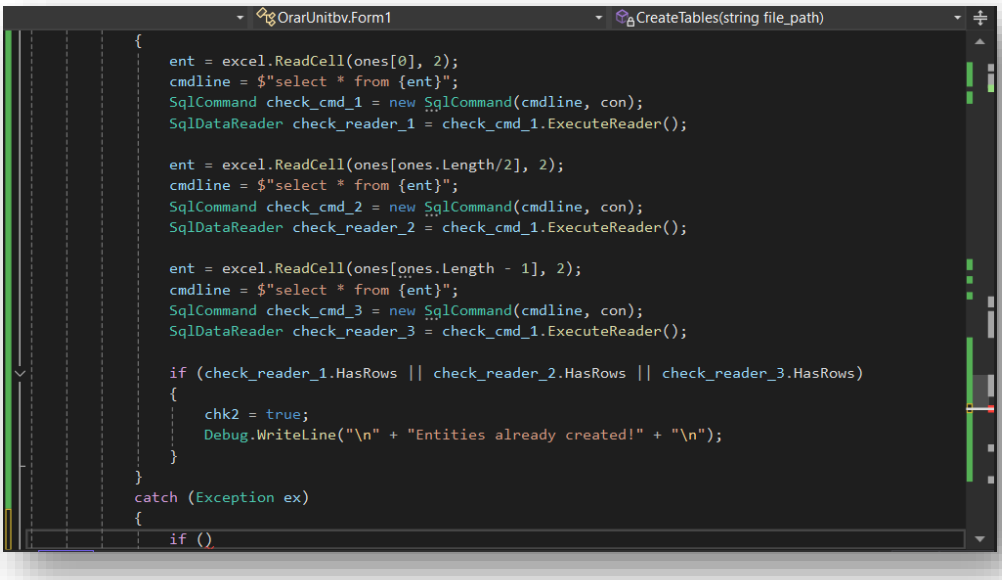
- După crearea entității trebuie considerat și o metodă de prevenire a suprascrierii, tabelele departamentelor trebuie create o singura data per semestru, sau acestea pot fi alterate dintr-o altă interfață. Oricum am considera, trebuie să creăm un mecanism de verificare a existenței entităților, iar aici ne vine în ajutor vectorul cu "1-uri". Array-ul ones[] ne va permite să facem verificări de existență a entităților, astfel vom considera trei comenzi corespunzătoare verificării existenței a trei departamente, și anume vom prelua primul, ultimul și departamentul aflat în mijlocul vectorului. Dacă cel puțin una dintre interogării ne va returna

valoare (ExecuteNonQuery()), atunci înseamnă că minim o entitate cu acel nume deja există și se va trece peste stabilirea entităților. Acest aspect este reflectat în următoarea imagine.



```
{
    chk1 = true;
    Debug.WriteLine("\n" + "Exception met in first try-catch (problems in opening a connection
}
// First, we must check if the entities were already created
try
{
    string ent1 = excel.ReadCell(ones[0], 2);
    SqlCommand check_cmd_1 = new SqlCommand("select * from {ent1}", con);

    ent1 = excel.ReadCell(ones[ones.Length/2], 2);
    SqlCommand check_cmd_2 = new SqlCommand("select * from {ent1}", con);

    ent1 = excel.ReadCell(ones[ones.Length - 1], 2);
    SqlCommand check_cmd_3 = new SqlCommand("select * from {ent1}", con);

    if ((check_cmd_1.ExecuteNonQuery() != 0) ||
        (check_cmd_2.ExecuteNonQuery() != 0) ||
        (check_cmd_3.ExecuteNonQuery() != 0))
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Entities already created!" + "\n");
    }
}
catch (Exception ex)
{
    Debug.WriteLine("\n" + "Error in checking entities existence! Read below:" + "\n" + ex.ToSt
}

if (!chk1 && !chk2)
{
    for (int i = 0; i < ones.Length; i++)
    {
```

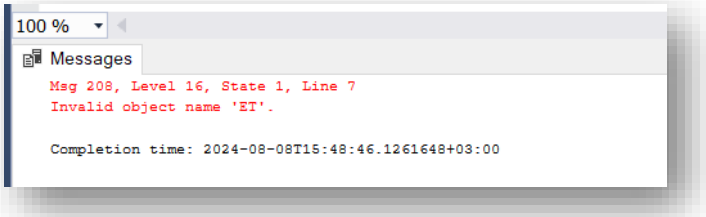
- ExecuteNonQuery nu este potrivit pentru interogări de tipul select, vom opta pentru instanțierea unui obiect de tip SQL reader.

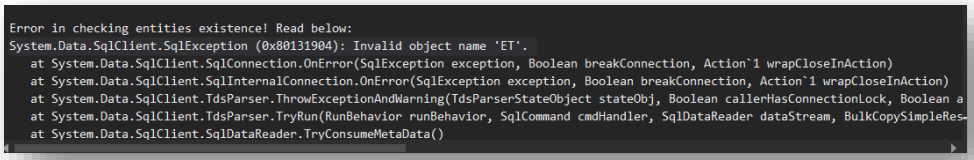


```
OrarUnitbv.Form1                    CreateTables(string file_path)
{
    ent = excel.ReadCell(ones[0], 2);
    cmdline = $"select * from {ent}";
    SqlCommand check_cmd_1 = new SqlCommand(cmdline, con);
    SqlDataReader check_reader_1 = check_cmd_1.ExecuteReader();

    ent = excel.ReadCell(ones[ones.Length/2], 2);
    cmdline = $"select * from {ent}";
    SqlCommand check_cmd_2 = new SqlCommand(cmdline, con);
    SqlDataReader check_reader_2 = check_cmd_1.ExecuteReader();

    ent = excel.ReadCell(ones[ones.Length - 1], 2);
    cmdline = $"select * from {ent}";
    SqlCommand check_cmd_3 = new SqlCommand(cmdline, con);
    SqlDataReader check_reader_3 = check_cmd_1.ExecuteReader();

    if (check_reader_1.HasRows || check_reader_2.HasRows || check_reader_3.HasRows)
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Entities already created!" + "\n");
    }
}
catch (Exception ex)
{
    if ()
```

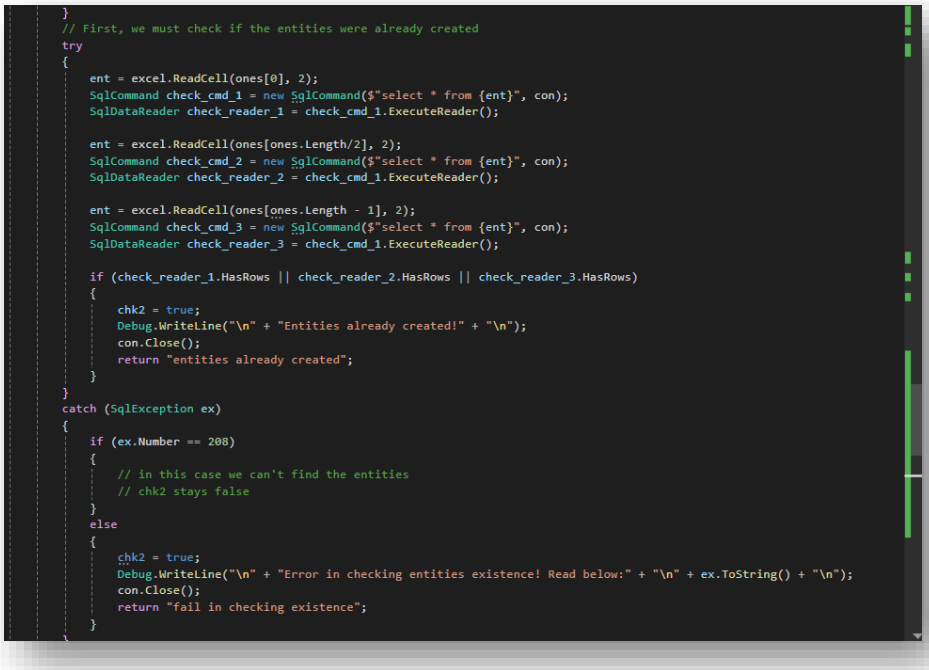
Rezumat 09.08.2024

- Erorile(excepțiile) întâmpinate arată în felul următor:





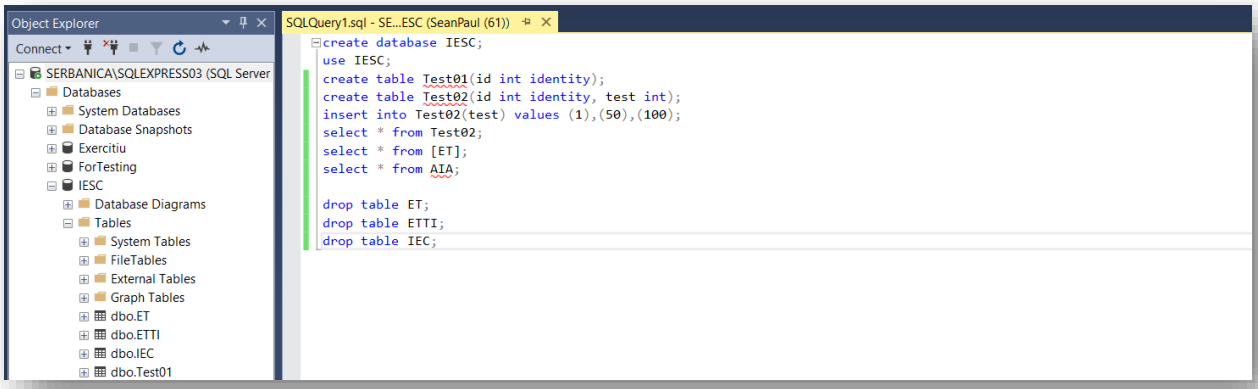
- Cu acestea vom putea trece peste faza de generare de entități în cazul în care ele au fost deja generate. Catch-ul aici poate depista însă și alte excepții care să ascundă alte cazuri indezirabile.
- Dacă excepția are numărul/codul 208 din cadrul bibliotecii de lucru cu SQL Server, atunci înseamnă că aceste entități nu pot fi identificate în baza de date, deci le putem genera pe baza sheet-ului din Excel. Tratarea problemei se va face în felul următor:



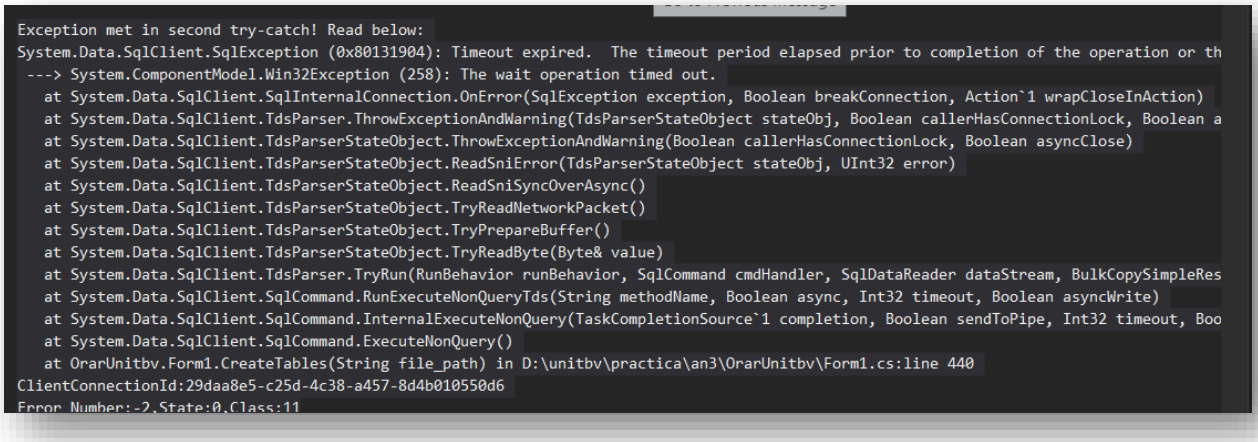
```
}
// First, we must check if the entities were already created
try
{
    ent = excel.ReadCell(ones[0], 2);
    SqlCommand check_cmd_1 = new SqlCommand($"select * from {ent}", con);
    SqlDataReader check_reader_1 = check_cmd_1.ExecuteReader();

    ent = excel.ReadCell(ones[ones.Length/2], 2);
    SqlCommand check_cmd_2 = new SqlCommand($"select * from {ent}", con);
    SqlDataReader check_reader_2 = check_cmd_1.ExecuteReader();

    ent = excel.ReadCell(ones[ones.Length - 1], 2);
    SqlCommand check_cmd_3 = new SqlCommand($"select * from {ent}", con);
    SqlDataReader check_reader_3 = check_cmd_1.ExecuteReader();

    if (check_reader_1.HasRows || check_reader_2.HasRows || check_reader_3.HasRows)
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Entities already created!" + "\n");
        con.Close();
        return "entities already created";
    }
}
catch (SqlException ex)
{
    if (ex.Number == 208)
    {
        // in this case we can't find the entities
        // chk2 stays false
    }
    else
    {
        chk2 = true;
        Debug.WriteLine("\n" + "Error in checking entities existence! Read below:" + "\n" + ex.ToString() + "\n");
        con.Close();
        return "fail in checking existence";
    }
```

Rezumat 09.08.2024

- O altă posibilitate de îmbunătățire ar fi să memorăm în baza de date lungimea și lățimea pe care le considerăm când parcurgem sheet-ul Excel-ului. Reamintim că un sheet are zeci de mii de linii și de coloane, practic, programul s-ar bloca într-o buclă foarte lungă, care de cele mai multe ori s-ar finaliza printr-un crash spontan.
- Am reușit crearea a trei tabele după modelul prezentat mai sus.

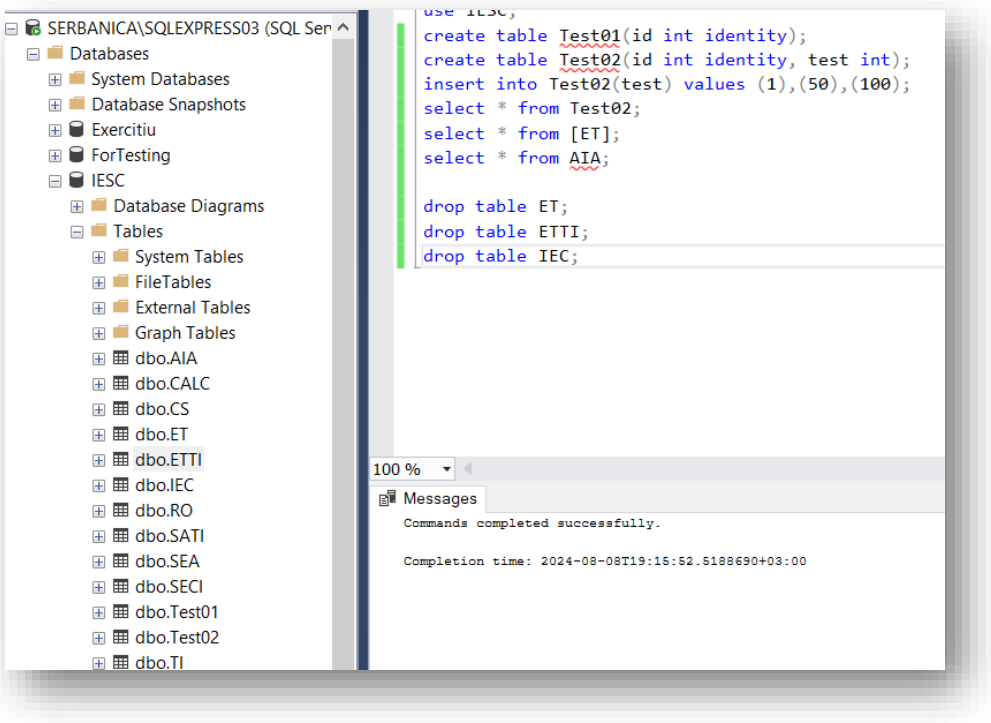


- Dar restul de tabele nu au fost generate, iar în consolă s-a înregistrat următoarea excepție.



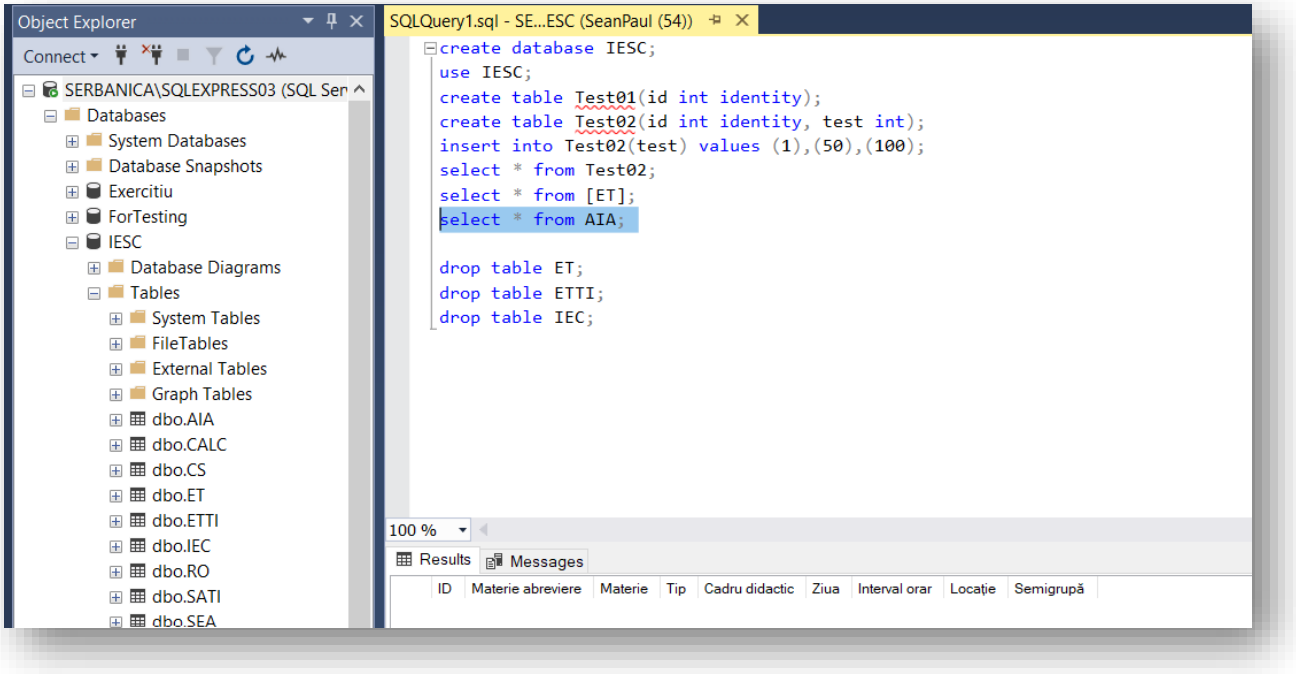
- Excepția de time-out poate fi tratată cu următoarea linie de cod:
  create_cmd.CommandTimeout = 60;
- Într-adevăr, prin adăugarea unui limite de time-out mai mare am rezolvat problema anterioară, rezultatul este prezentat în următoarea figură.

Rezumat 09.08.2024



- Acum trebuie verificat dacă, la următoarea rulare a metodei, va furniza excepție sau va evita crearea tabelelor așa cum am plănuit pentru acest caz și așa cum este vizibil în imaginile de mai sus.
- La a doua rulare obțin excepții la obiectul SqlDataReader care nu este închis după prima rulare.
- O altă problemă care poate apărea este închiderea instanțelor de Excel, daca acestea nu se închid, pot, la un moment dat, să aglomereze memoria și astfel nu numai ca aplicația se oprește forțat, dar și sistemul de operare se oprește de asemenea. Excepția este: **System.InvalidOperationException:** 'There is already an open DataReader associated with this Command which must be closed first.'
- Prin urmare trebuie să facem o verificare progresivă, însemnând că ne folosim de if după verificare instanțiere de reader, apoi îl închidem.
- În debugger întâlnim first chance exceptions, acestea apar în debugger indiferent cum sunt ele tratate ulterior, iar dat fiind că sheet-ul are foarte multe celule nule sau concatenate trebuie să gândim o modalitatea de a filtra aceste excepții care umplu debugger la fiecare celulă nulă sau concatenată cu câte un mesaj de excepție. O soluție(pseudosoluție) pentru cluster-ul care se acumulează în debugger este filtrarea excepțiilor, acestea totuși sunt înregistrate, dar nu mai apar vizul în debugger.

Rezumat 09.08.2024



- În imaginea de mai sus se poate vedea cum arată un tabel asociat unui departament al facultății IESC, așa cum am arătat în schița de tabel de mai sus. Atributele se extrag din celula propriu-zisă prin prelucrarea șirului de caractere, iar celelalte atribute se găsesc fie pe aceeași linie, fie pe aceeași coloană. Semigrupa fiind unică fiecărui grup de studenți, aceasta poate fi folosită pentru a afla când au aceștia ore, unde, etc.
- Trebuie folosită metoda obictului SqlReader care verifică dacă o entitate este populată, adică are cel puțin un rând cu date. Proprietatea este HasRows. Prin acestă metodă am rezolvat cazul în care entitățile au fost create și trebuie sărit peste pasul acesta.
- cmd.CommandTimeout este metoda care ne permite să alocăm mai mult de 30 de secunde, timpul implicit, pentru a rula o comandă în SQL.
- În worksheet există departamente care nu încep cu anul 1, iar acestea vor fi trecute cu vederea în matricea de 1-uri cu care, până acum, am generat tabele, așadar trebuie să mai considerăm încă o metodă prin care putem să generăm tabele și pentru aceste deparamente.
- Ca să nu anulez metoda concepută pentru a găsi departamentele, o să mai fac încă una care o să o verifice pe prima, iar dacă, într-adevăr, există departamente care încep cu anul 2 sau 3, atunci și acestea vor fi considerate când se generează entitățile.
- Zilele următoare o să vă trimit și algoritmul de refiltrare a departamentelor și o să încerc să populez tabelele cu informațiile din celule.