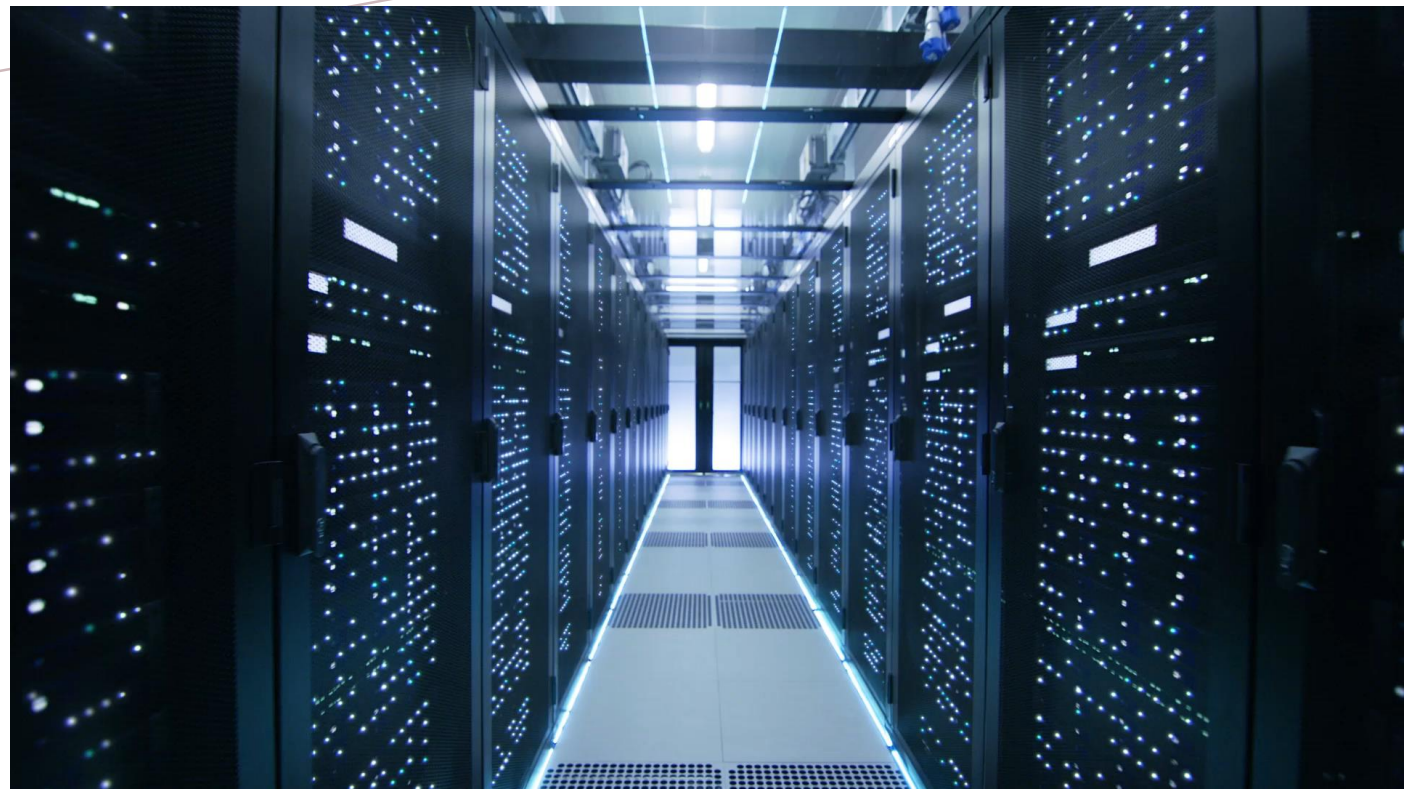


DEBUGOLÁS PL/SQL-BEN

BEMUTATÓT KÉSZÍTETTE: TASNÁDI ISTVÁN-NORBERT

JÁNOSI RANCZ KATALIN TÜNDE VEZETÉSÉVEL

MELY PROGRAMOT FOGJUK VIZSGÁLNI?

- Számos programban van lehetőségünk debugolni PL/SQL programjainkat, mint például Visual Studio, IntelliJ Idea, DataGrip stb.

- Ebben a bemutatóban viszont az Oracle SQL Developer fogjuk megvizsgálni

ELSŐ LÉPÉSEK

- Még mielőtt debugolni tudnánk, meg kell győződjünk először, hogy eleget teszünk a következő feltételeknek:

1) van-e debug jogunk

2) rendesen be van-e konfigurálva az ACL, Network tulajdonságok, port stb.

3) SQL Developer beállításainál Debugger résznél kivan-e pipálva a “Use DBMS_DEBUG” opció

DEBUG JOG MEGADÁS

- Ahhoz, hogy debugolhassuk a PL/SQL programjainkat szükségünk van debug jogokra, nevezetesen a következőkre: ***DEBUG CONNECT SESSION*** és ***DEBUG ANY PROCEDURE***

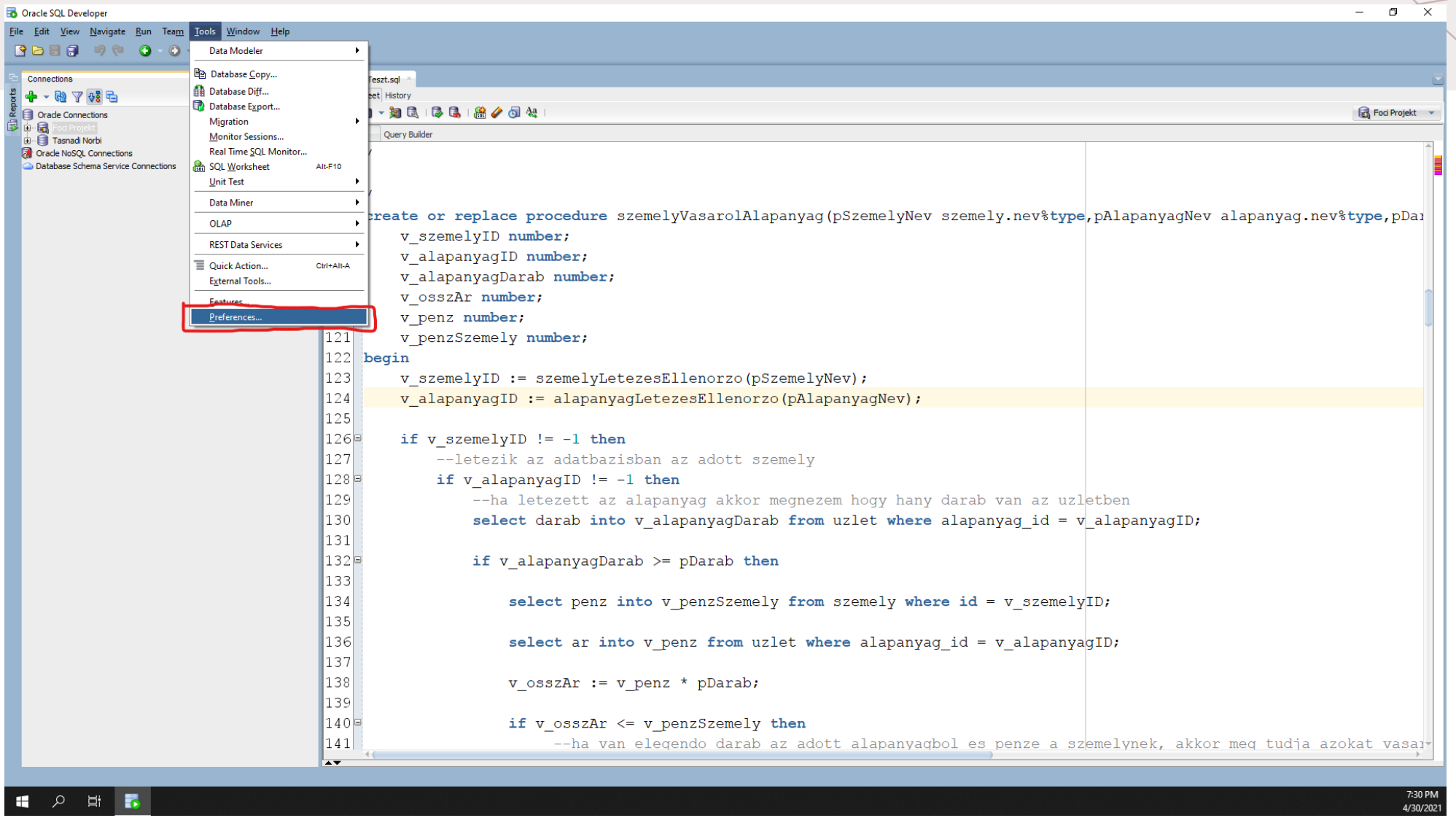
- Ezt a két jogot az adminisztrátor kell megadja a felhasználóknak

- Az adminisztrátor a következőképpen adhatja meg a szükséges jogokat a usereknek:
  - grant DEBUG CONNECT SESSION to <user_name> ;
  - grant DEBUG ANY PROCEDURE to <user_name> ;

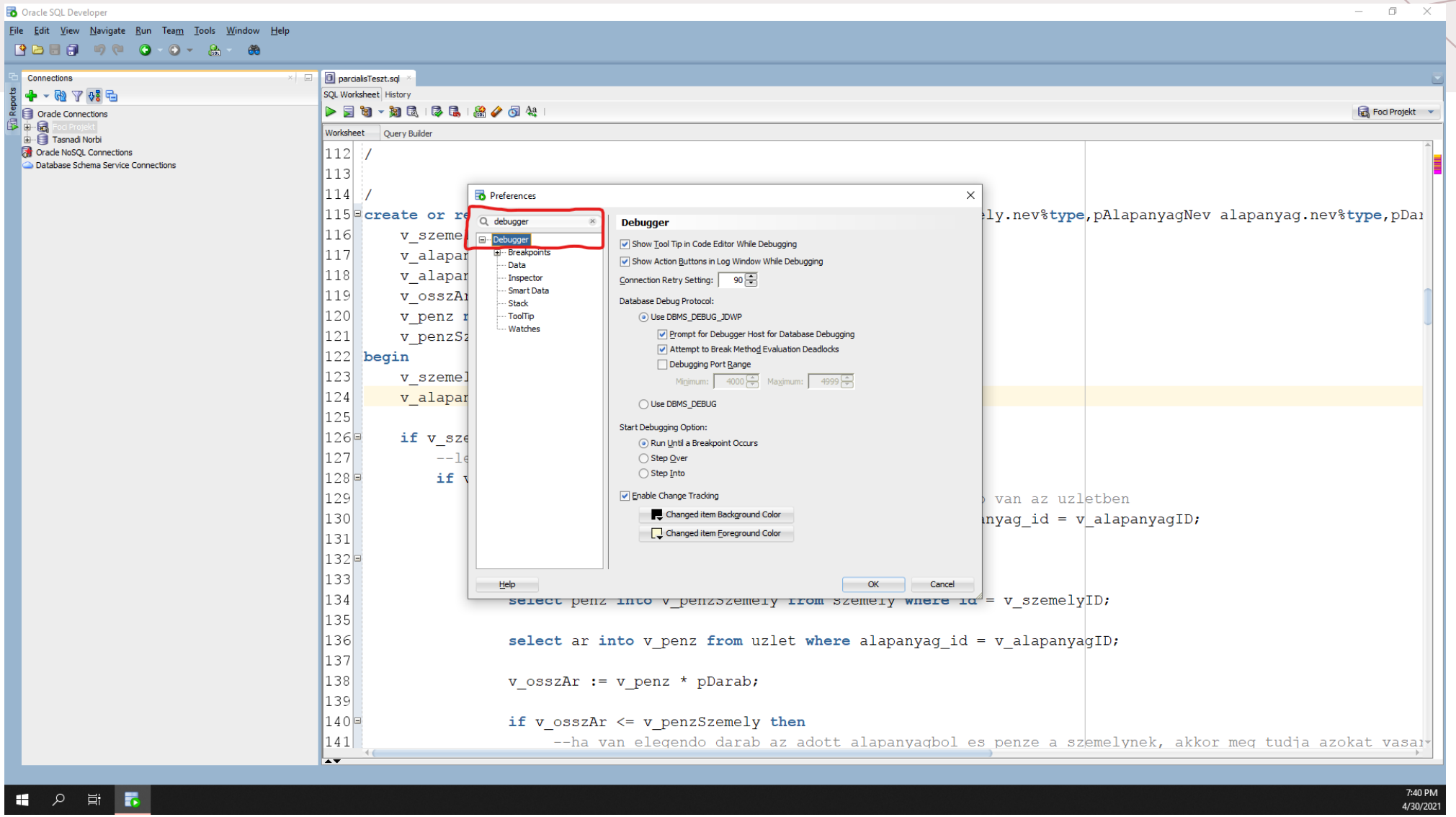
DEBUGOLÁS MÁSODIK KRITÉRIUM OPCIÓK

- A másik feltétele annak, hogy debugolhassunk az az, hogy rendesen be legyen állítva az ACL, Network tulajdonságok
- Meglehetősen macerás ezeket mind beállítani, sok munkát vesz igénybe
- Ezért be lesz mutatva egy másik módszer, nevezetesen , hogy hogyan lehet beállítani az Oracle SQL Developerben a “***Use DBMS_DEBUG*** ” opció, ami szignifikánsan megkönnyíti majd dolgunkat

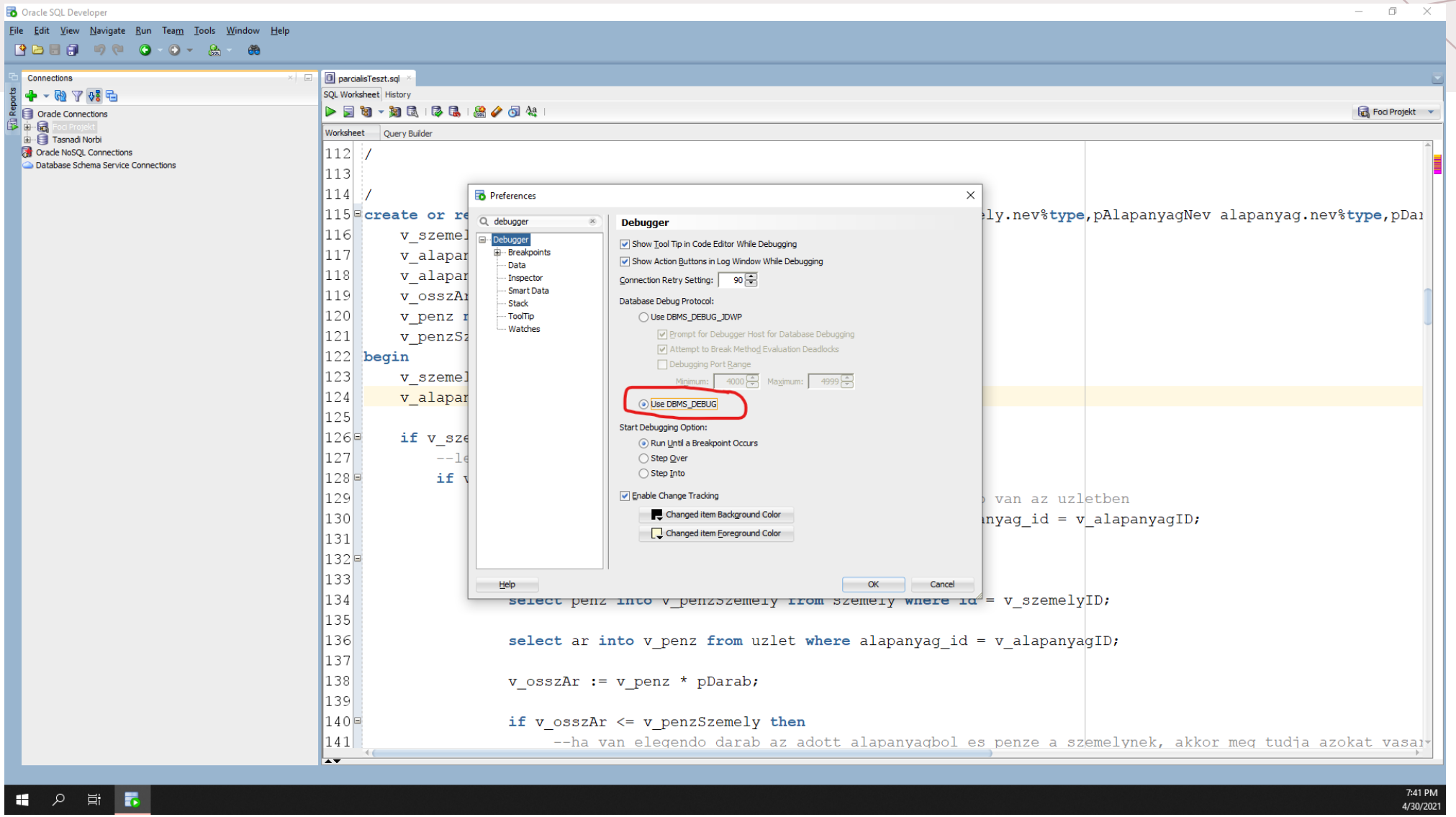
USE DBMS_DEBUG BEÁLLÍTÁSA



USE DBMS_DEBUG BEÁLLÍTÁSA



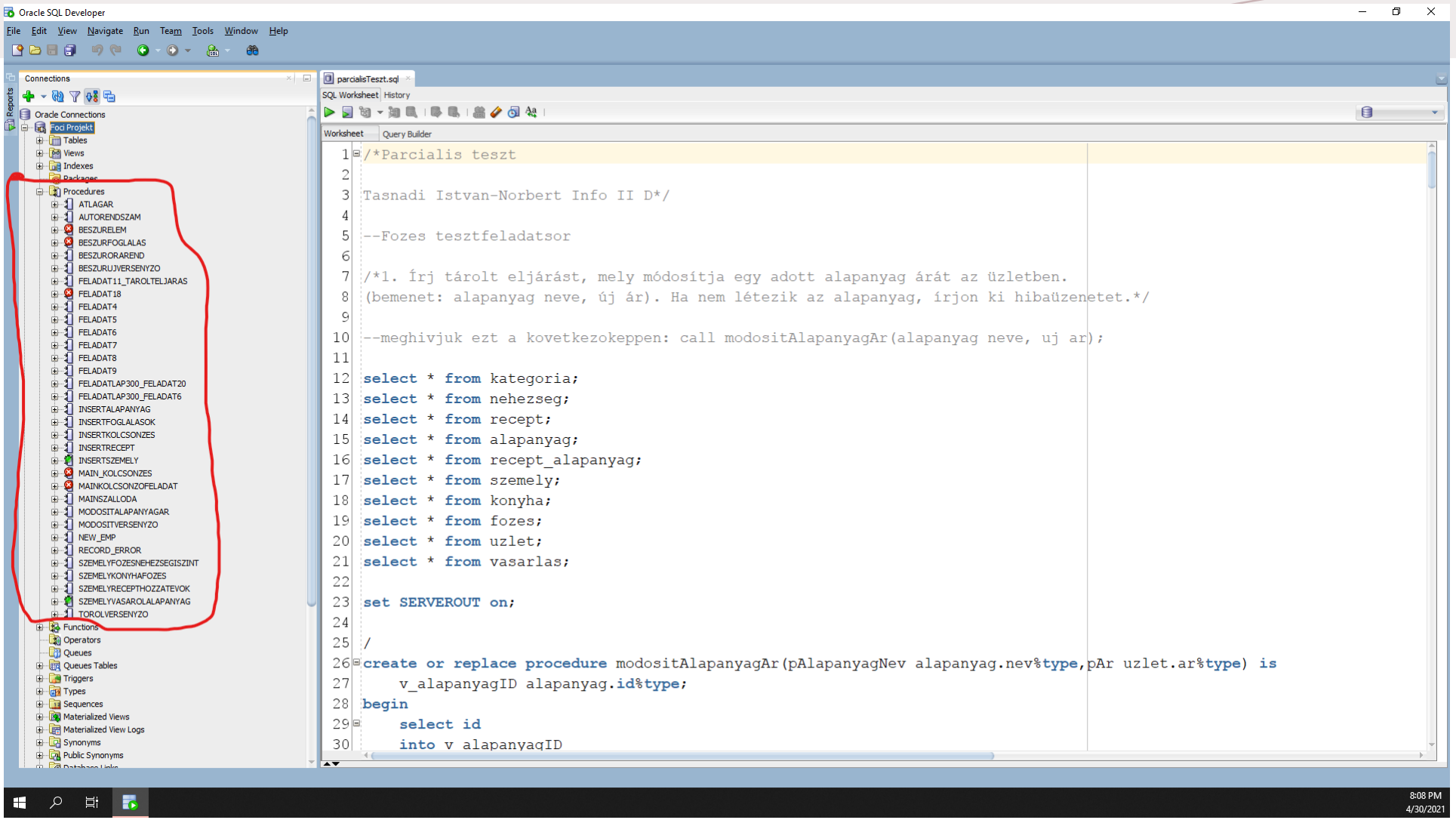
USE DBMS_DEBUG BEÁLLÍTÁSA



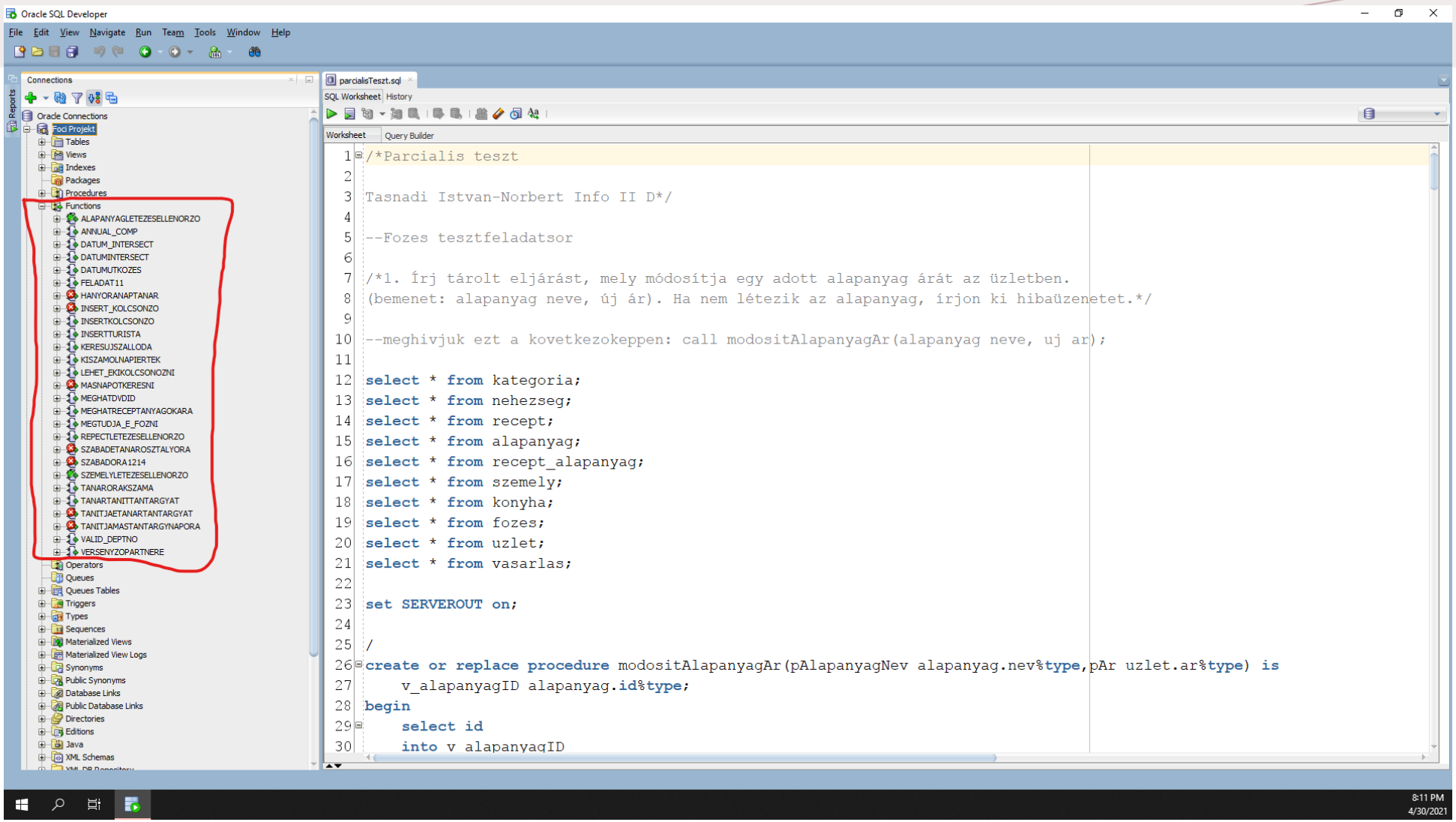
MIKET DEBUGOLHATUNK AZ ORACLE SQL DEVELOPERBEN?

- Az Oracle SQL Developerben debugolhatunk:
  - Procedurákat (procedure)
  - Függvényeket (function)
  - Triggereket (trigger)

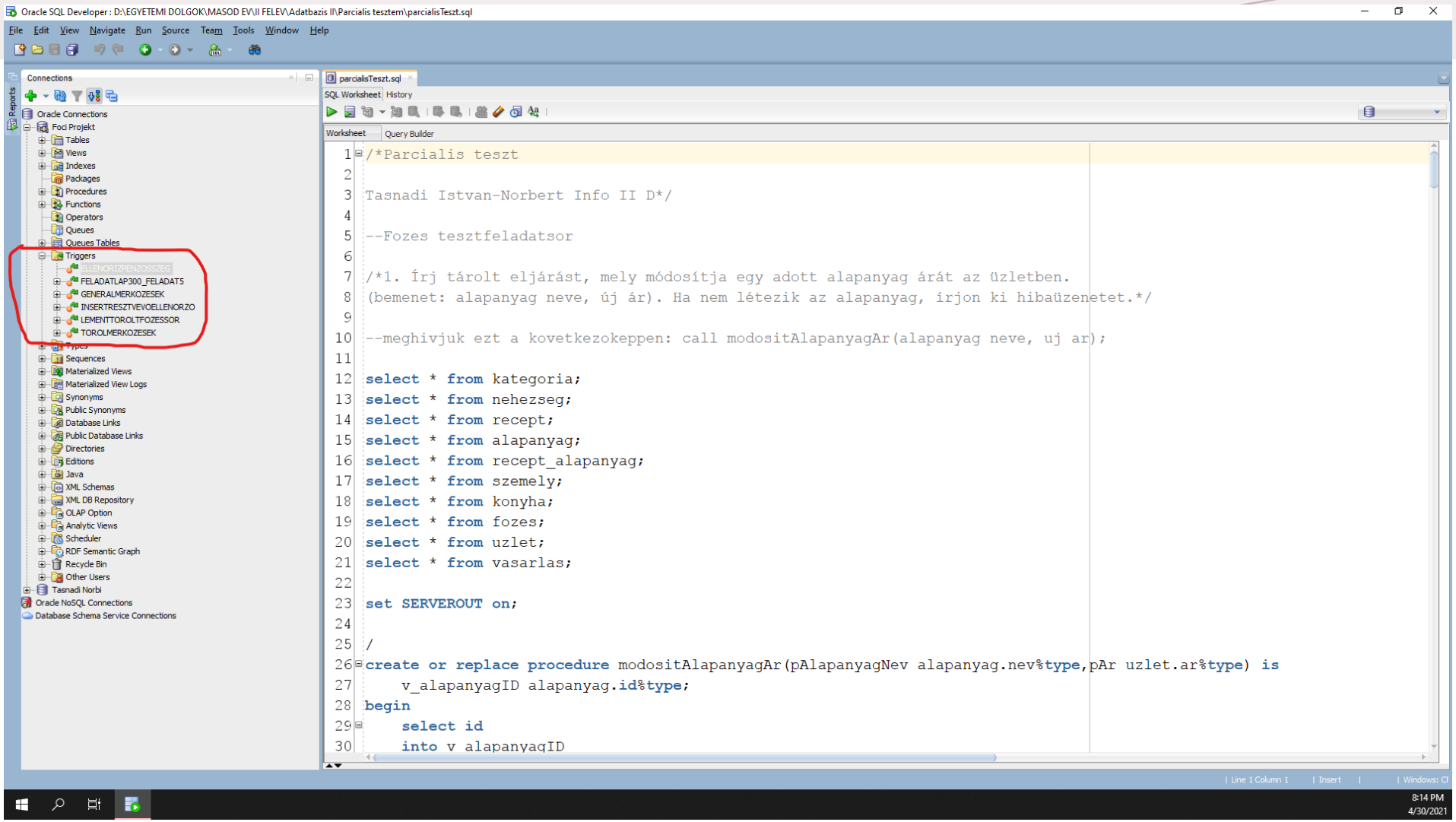
HOL KAPHATOM MEG AZ ÁLTALAM MEGIRT PROCEDÚRÁKAT?



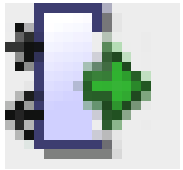
HOL KAPHATOM MEG AZ ÁLTALAM MEGIRT FÜGGVÉNYEKET?



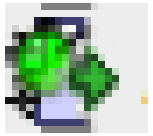
HOL KAPHATOM MEG AZ ÁLTALAM MEGIRT TRIGGEREKET?

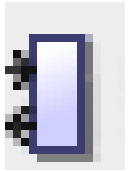


A SÉMÁN LÉVŐ IKONOK JELENTÉSE


Függvény, ami futásra kész állapotú (Ezen nem lehet debugolni)


Függvény, ami nem futtatható (hibák vannak benne)


Függvény, ami debug módban is futtatható


Procedúra, ami futásra kész állapotú (Ezen nem lehet debugolni)


Procedúra, ami nem futtatható (hibák vannak benne)


Procedúra, ami debug módban is futtatható

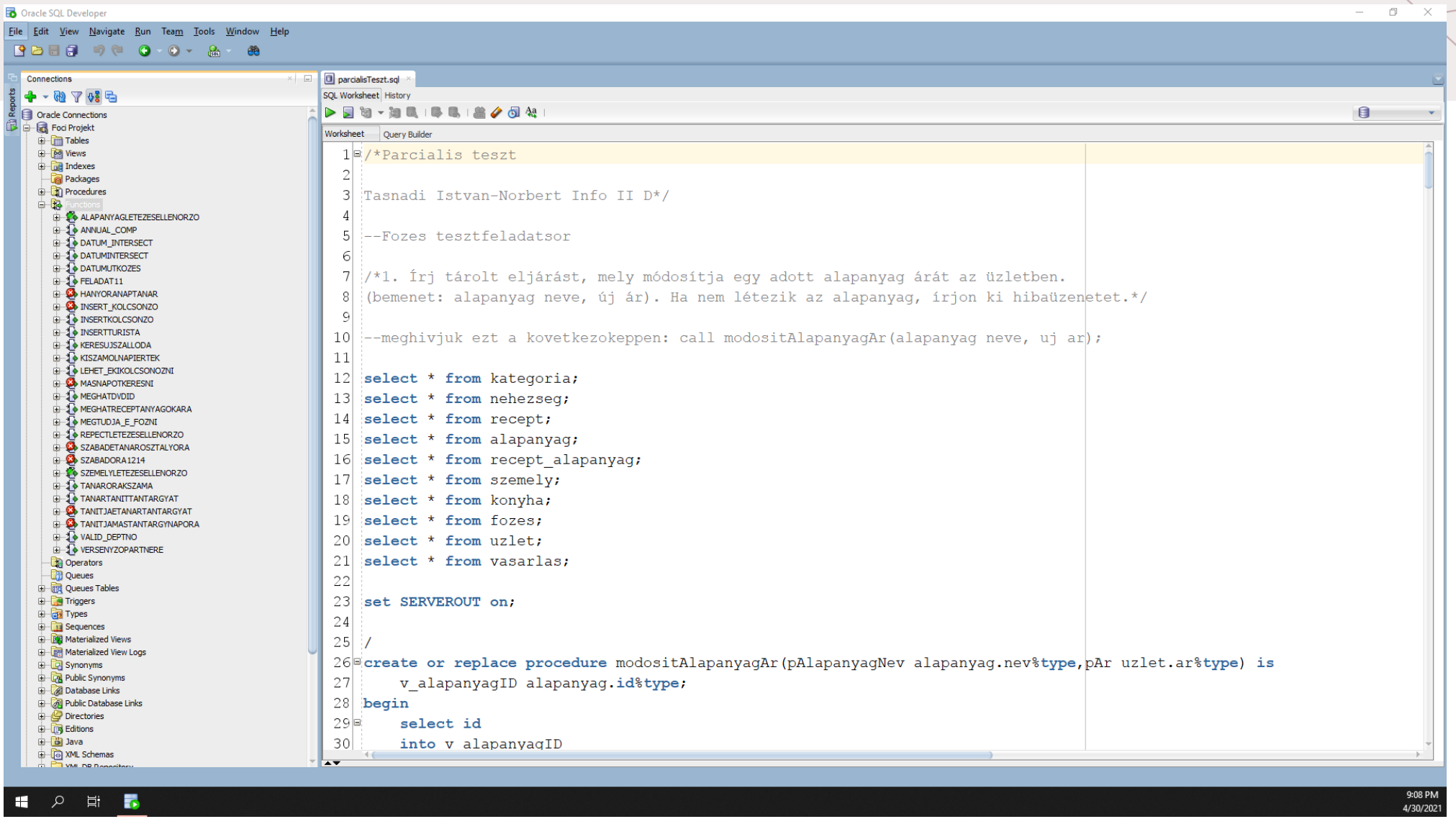

Trigger, ami futásra kész állapotú (Ezen nem lehet debugolni)


Trigger, ami debug módban is futtatható

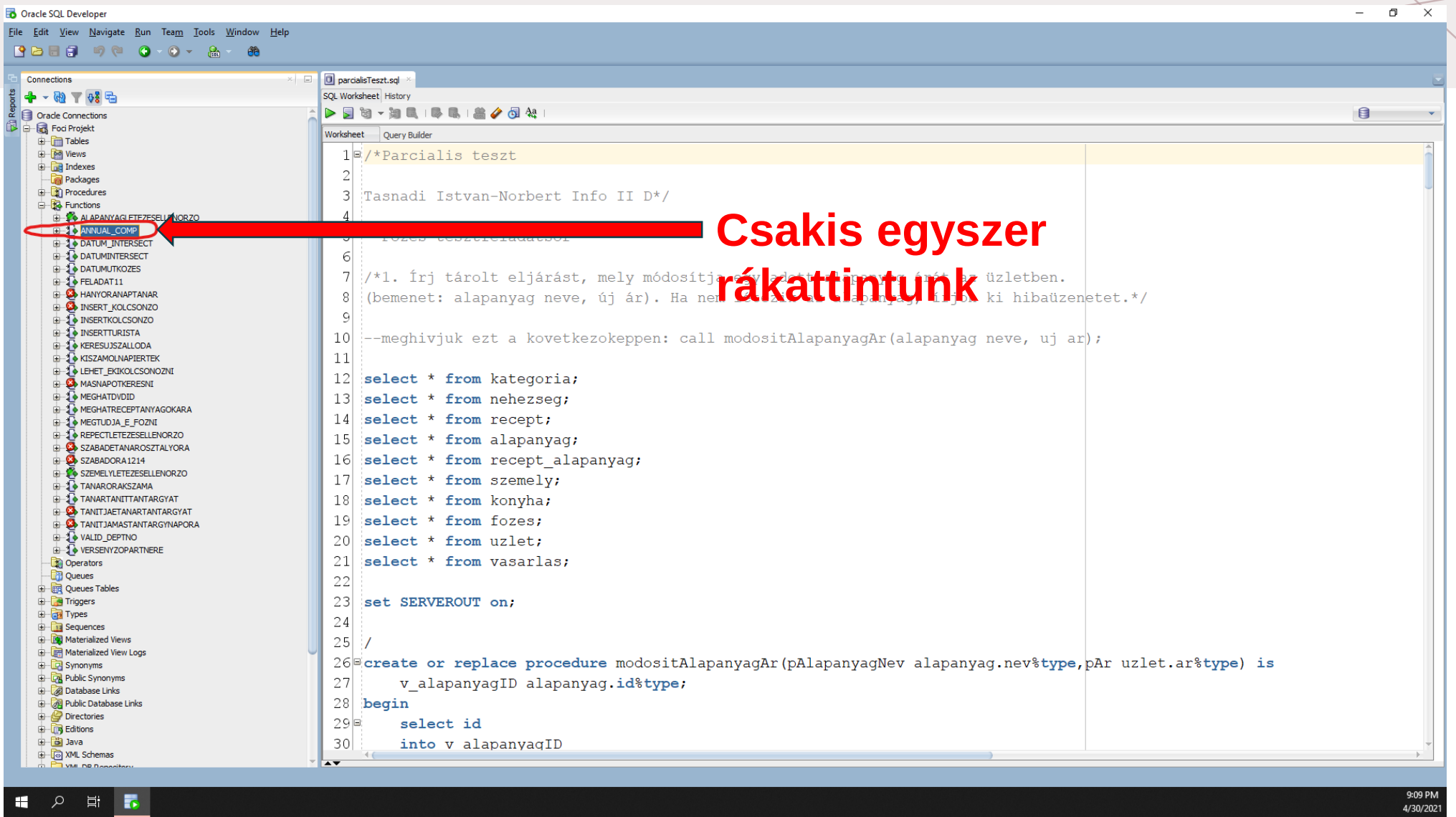
HOGYAN DEBUGOLUNK EGY EGYSZERŰ PROCEDÚRÁT, FÜGGVÉNYT, TRIGGERT?

- 1. Kikeressük a sémából az adott procedúrát, függvényt, triggert
- 2. Megnyitjuk az adott procedúrát, függvényt, triggert (**egyszer rákattintunk**)
- 3. ***Begin-End*** közötti sorokban teszünk egy ***Breakpoint*** -ot
- 4. Az adott procedúrát, függvényt, triggert **kompiláljuk debugra** (***Compile for Debug***)
- 5. A katicabogár ikonra  rákattintva megadjuk az adott procedúra, függvény, trigger **paramétereit** (ha szükséges azokat megadni), majd az ***OK*** gombra rányomva elindítjuk a debugolást.

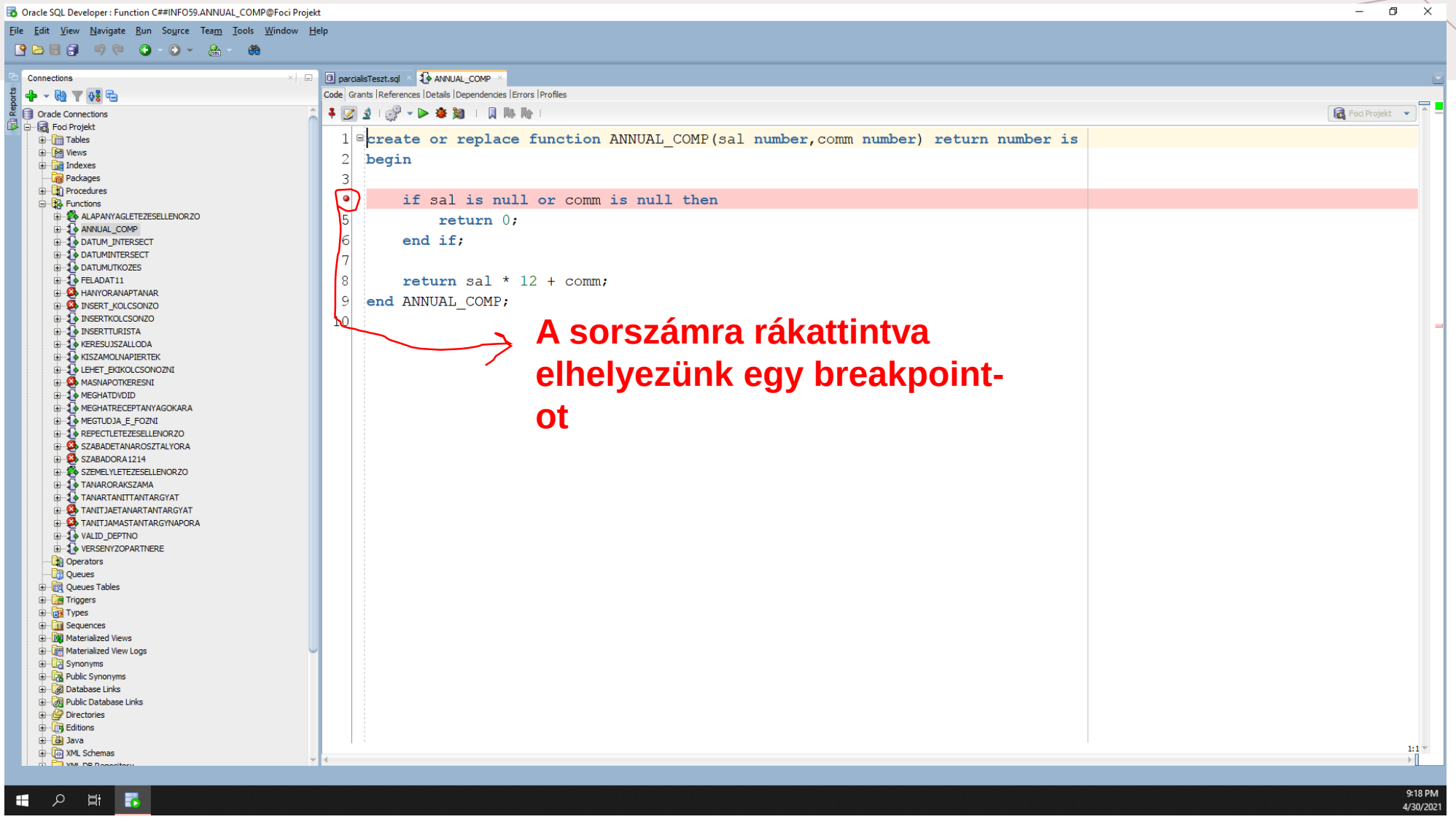
1. KIKERESSÜK A SÉMÁBÓL AZ ADOTT PROCEDÚRÁT, FÜGGVÉNYT, TRIGGERT



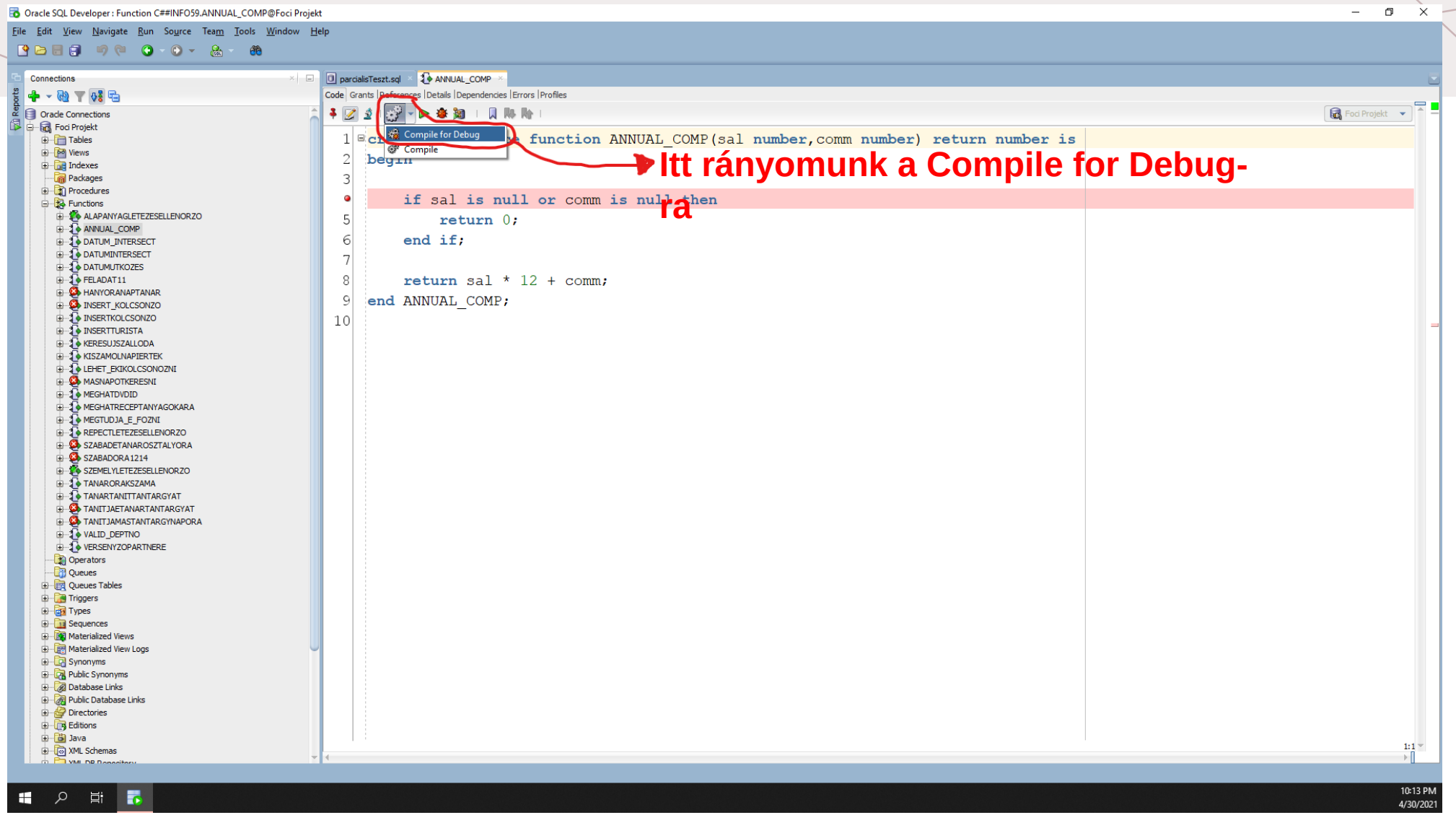
2. MEGNYITJUK AZ ADOTT PROCEDÚRÁT, FÜGGVÉNYT, TRIGGERT (***EGYSZER RÁKATTINTUNK***)

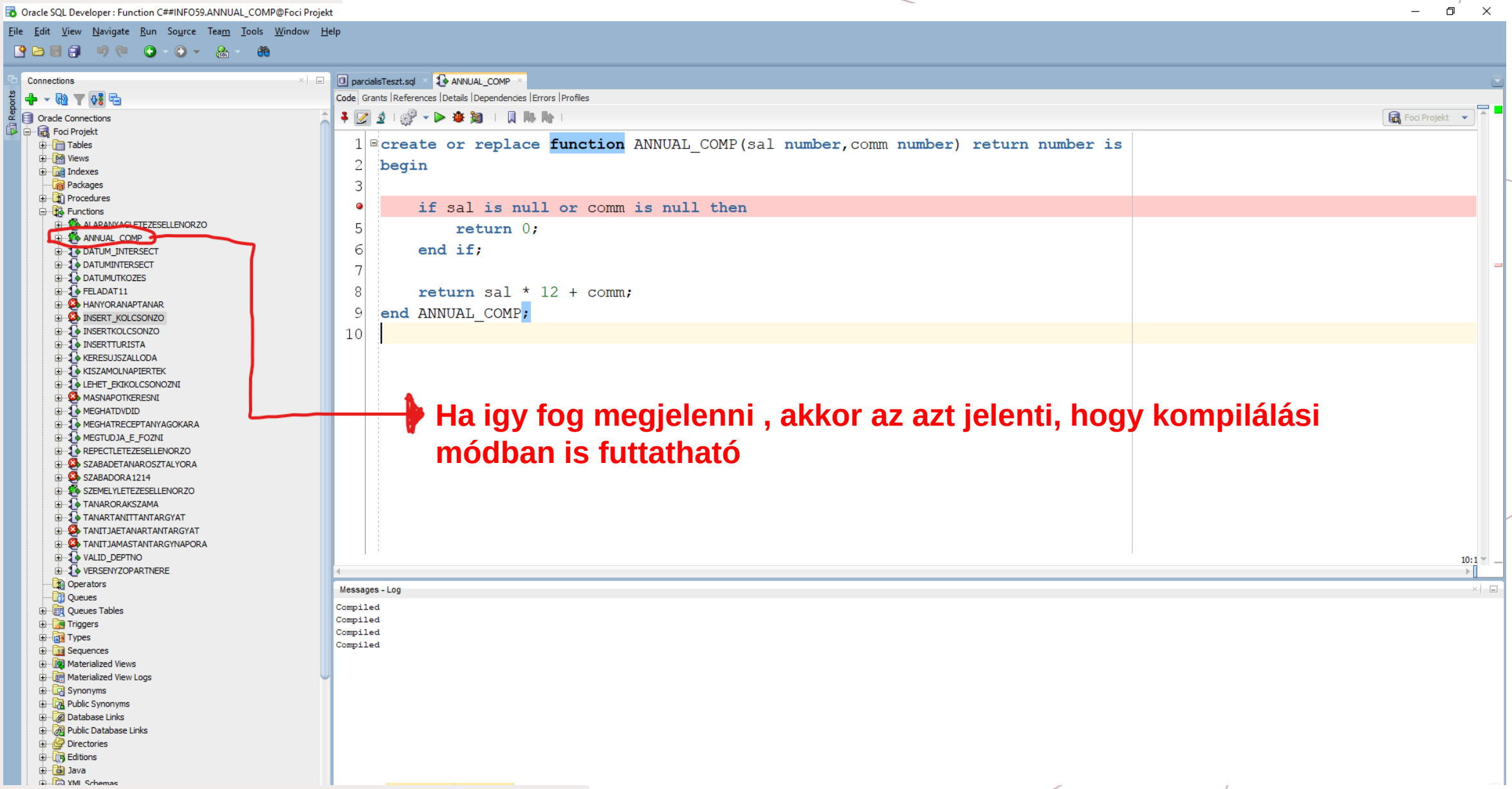


3. **BEGIN-END** KÖZÖTTI SOROKBAN TESZÜNK EGY **BREAKPOINT** -OT



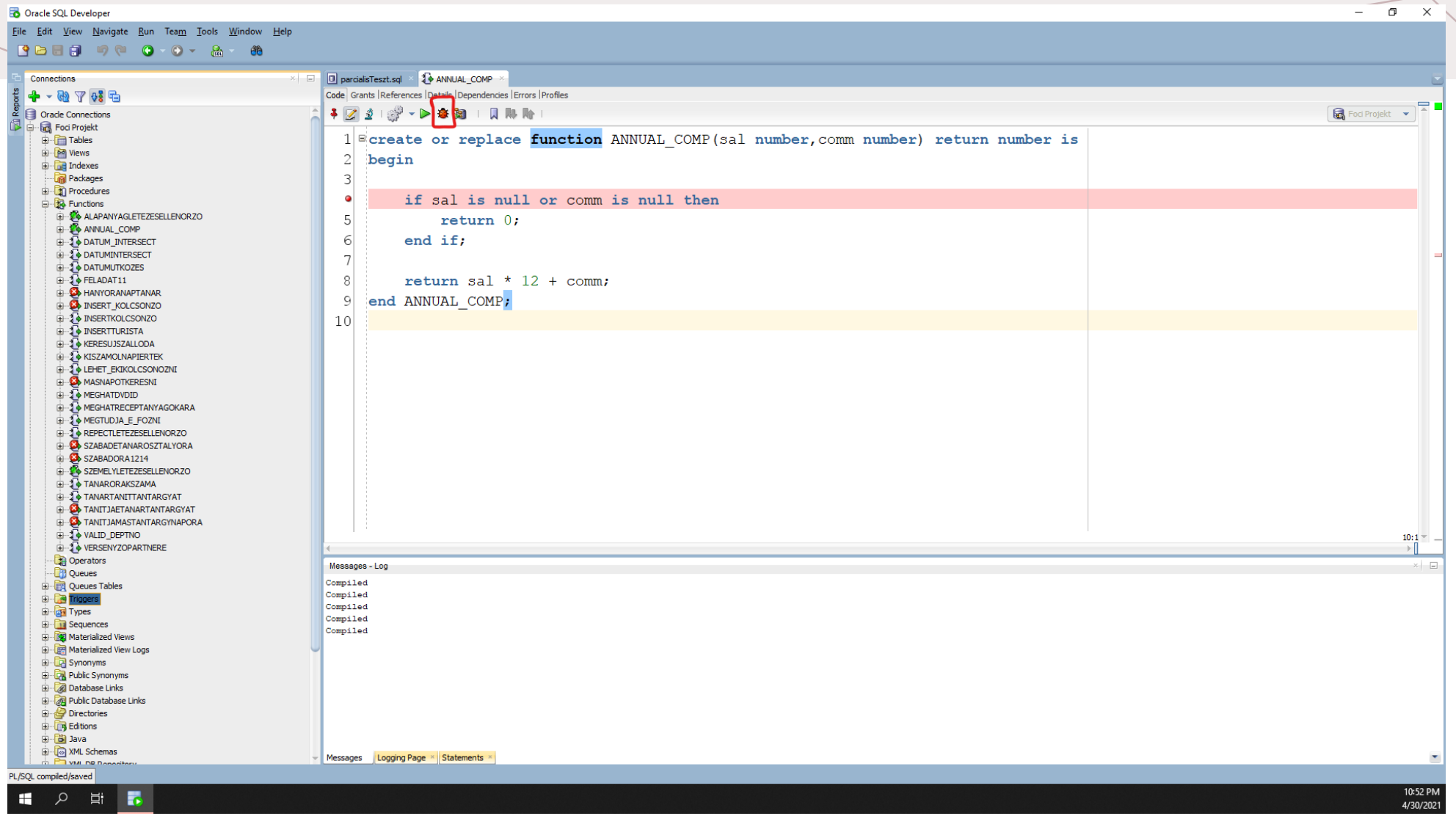
4. AZ ADOTT PROCEDÚRÁT, FÜGGVÉNYT, TRIGGERT ***KOMPILÁLJUK DEBUGRA (COMPILE FOR DEBUG)***



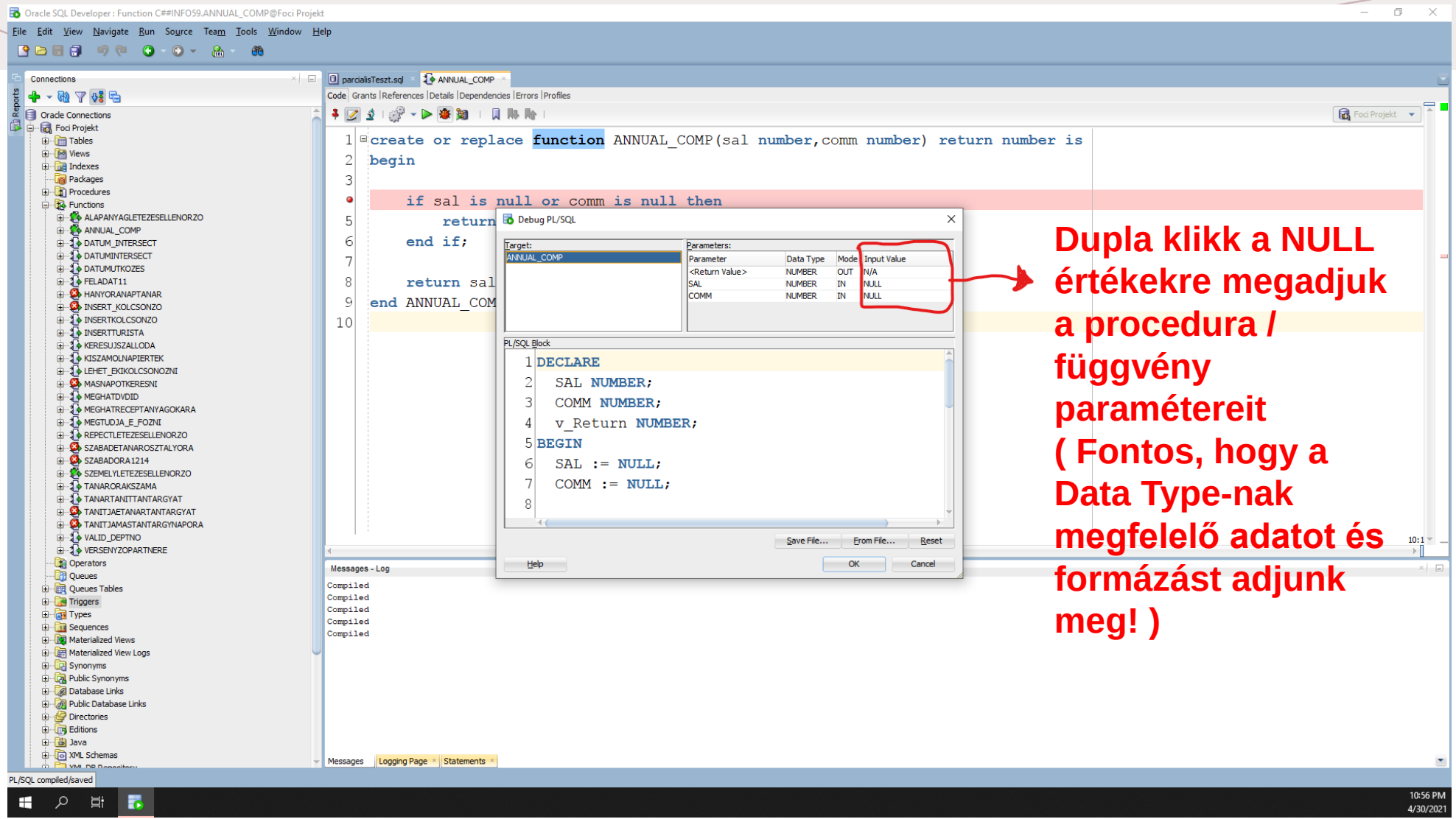


Ha igy fog megjelenni , akkor az azt jelenti, hogy kompilálási módban is futtatható

5. RÁKATTINTUNK A  IKONRA, HOGY EZZEL TUDJUNK MAJD DEBUGOLNI

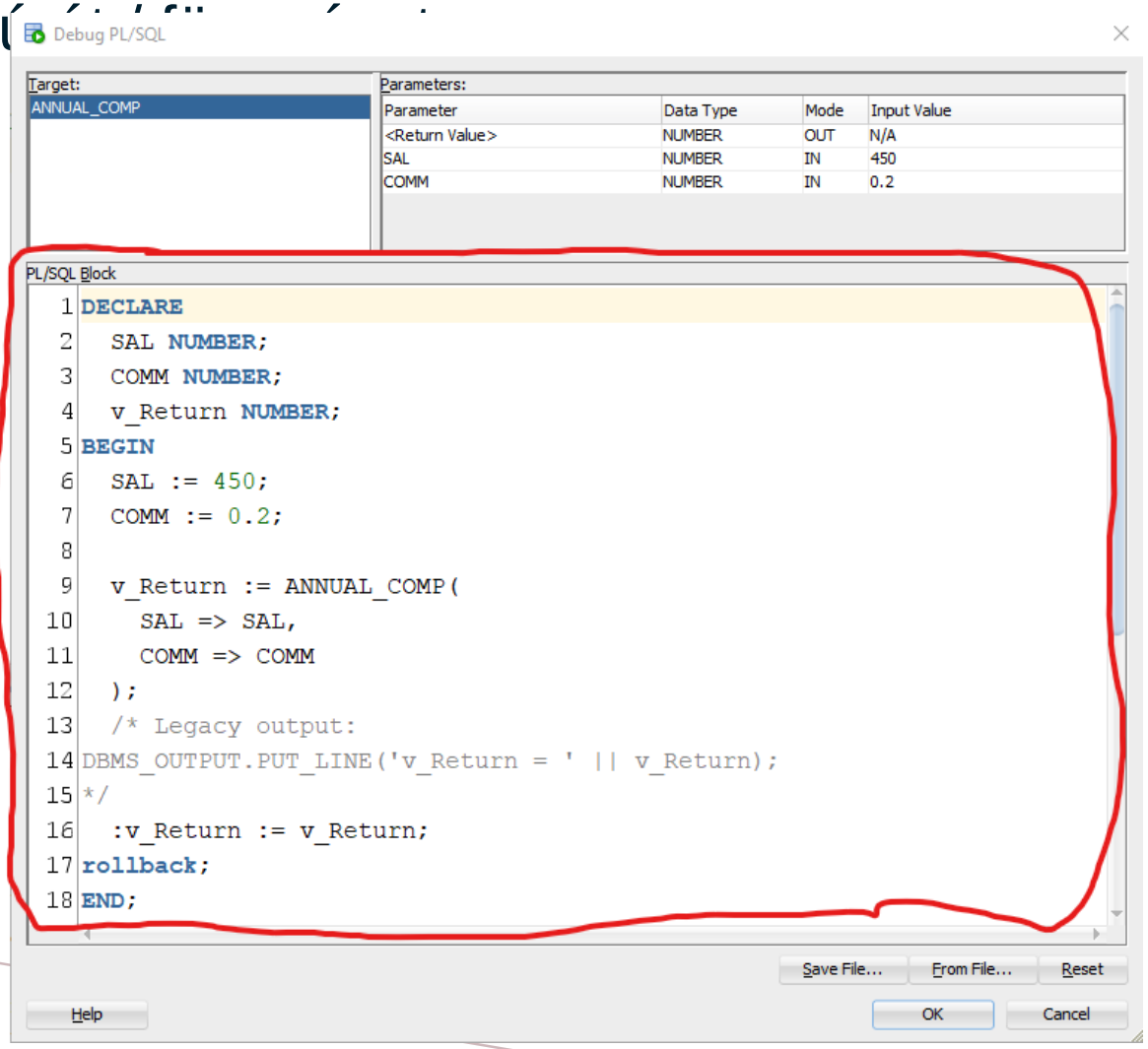


AZ **INPUT VALUE** OSZLOPNÁL MEGADJUK A PROCEDÚRA / FÜGGVÉNY PARAMÉTEREIT

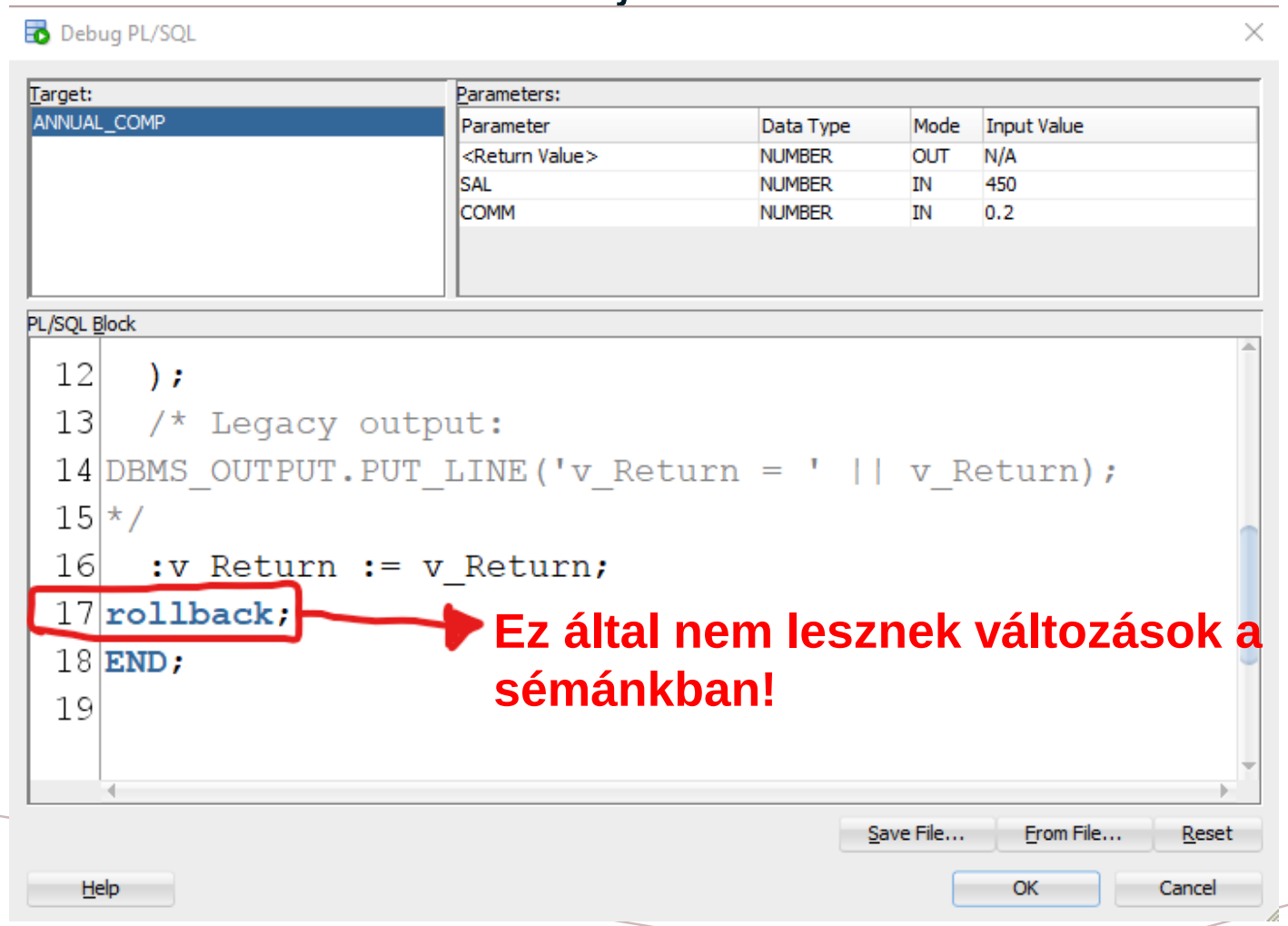


Dupla klikk a NULL értékekre megadjuk a procedura / függvény paramétereit (Fontos, hogy a Data Type-nak megfelelő adatot és formázást adjunk meg!)

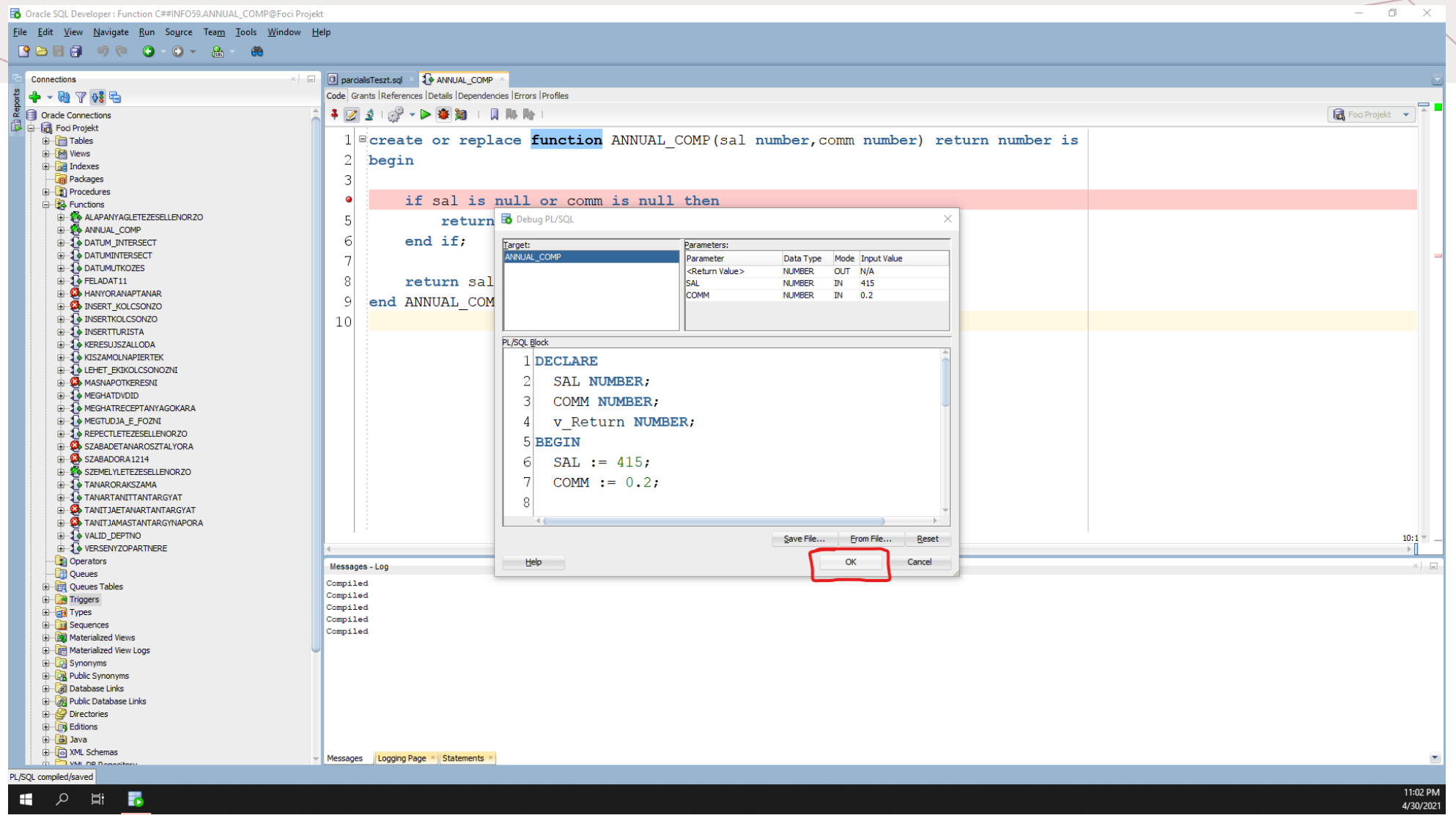
- A **Debug PL/SQL** ablakban látható egy névtelen **PL/SQL blokk**, amelyben az **SQL Developer** a háttérben kb. igy fogja végrehajtani az adott procedú



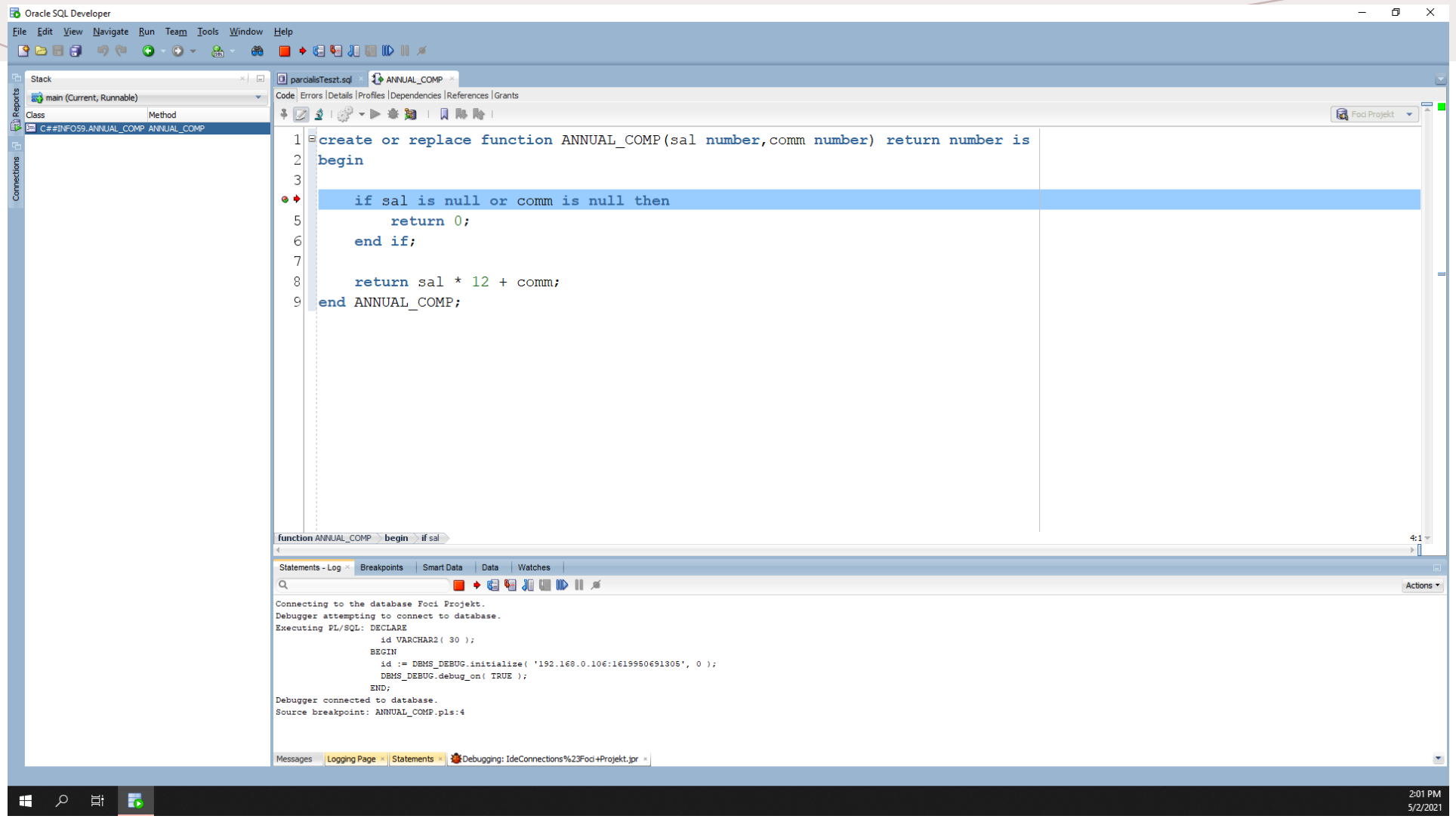
- **Fontos megjegyzés**: A debug során a **sémánk nem fog módosulni**! A megjelenített **Debug PL/SQL** ablakban ha lejjebb görgetünk akkor látni fogjuk, hogy a PL/SQL blokk végén van egy **rollback** utasítás, ami gondoskodni fog arról, hogy a sémánkban valóban ne történjék változás.



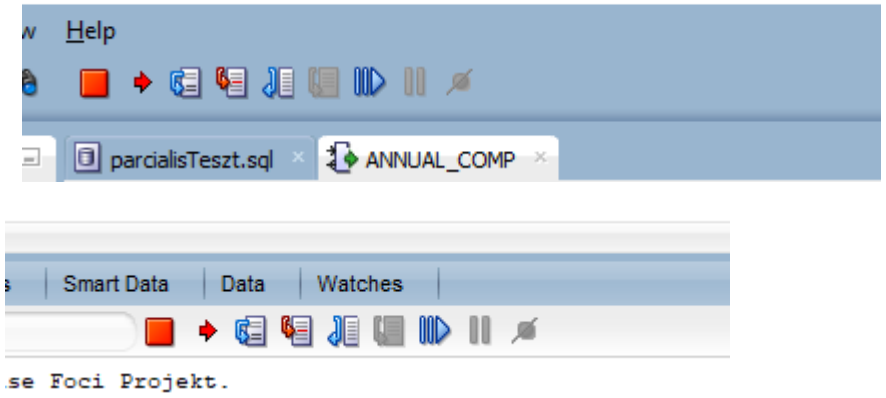
A VÉGÉN PEDIG AZ "OK" GOMBRA RÁNYOMVA ELINDITJUK DEBUGOLÁSRA



EZ A FELÜLET FOGJA MAJD FOGADNI A FELHASZNÁLÓT



NAVIGÁLÁSI MENÜ + IKON MAGYARÁZAT



Ezen a két navigálási menün keresztül tudunk majd lépegetni a debug során, kb. hasonlóan, mint más megszokott programokban (pl: Visual Studio)

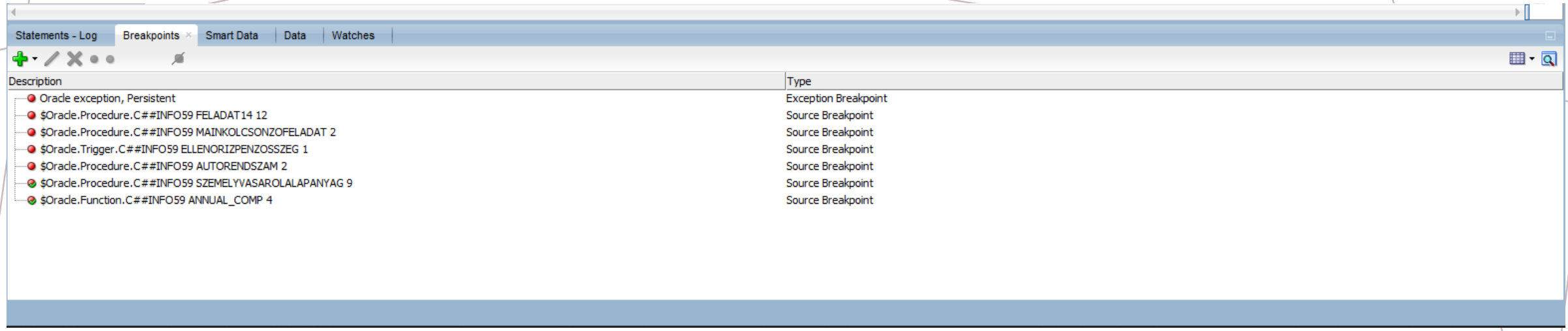
Stop ikon, amivel leállíthatjuk a debugolási folyamatot

Step Over ikon, amivel ugrunk a következő utasítás sorhoz (Gyorsbillentyű : **F8**)

Step Into ikon, ami által **beléphetünk más procedúrába, függvénybe** ; ennek feltétele, hogy az adott **procedúra / függvény**, amibe be szeretnénk lépni **azok legyenek kompilálva debugolásra** (lásd itt) (Gyorsbillentyű : **F7**)

Step Out ikon, ami által **kiléphetünk a jelenlegi procedúrából / függvényből** (Gyorsbillentyű : **Shift + F7**)

Resume ikon, ami által az adott folyamat tovább fog futni és a legközelebbi **Breakpoint**-nál fog **blokkolodni** (Gyorsbillentyű : **F9**)



- **Breakpoints**-nál tudjuk megnézni, hogy adott **felhasználóknak melyik nevű procedúrájában / függvényében / triggerjében hányadik során vannak elhelyezve breakpointok** , illetve itt látható, hogy **mely breakpointok lettek aktiválva és melyek nem**.
- Például: $Oracle.Procedure.C##INFO59.SZEMELYVASAROLALAPANYAG.9

Adott breakpoint aktiválva van

Metodus típusa

User neve, amely birtokolja az adott metodust (jelen esetben procedúrát)

Metodus neve, amelyben található breakpoint (jelen esetben procedúra neve)

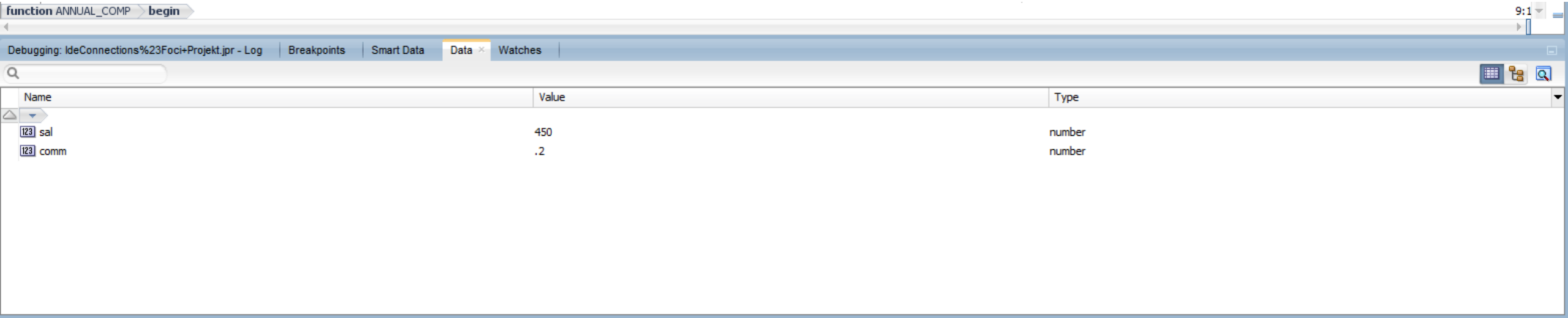
Hányadik sorban található az adott breakpoint

VÁLTOZÓK ÉRTÉKEINEK VIZSGÁLATA

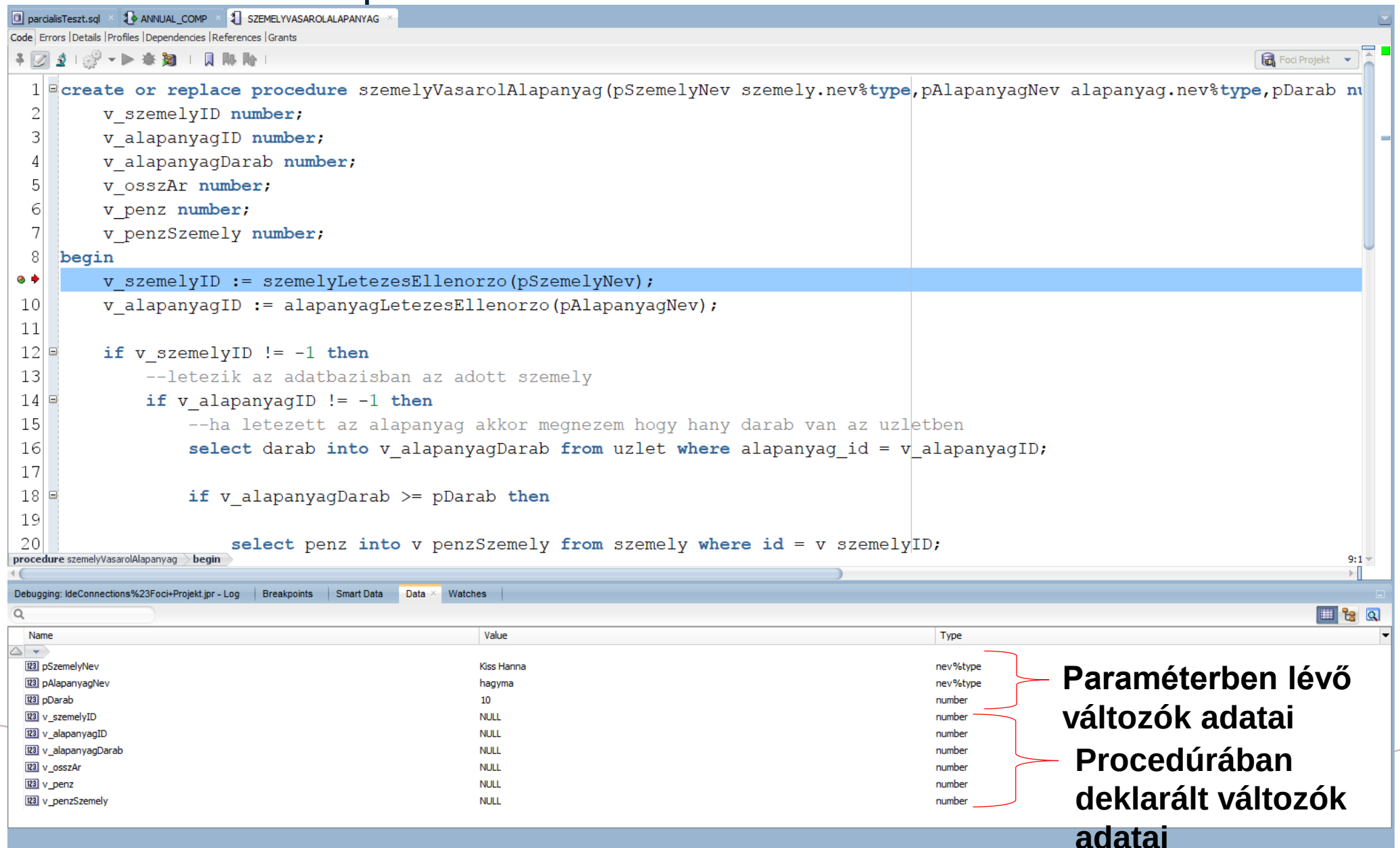
- A debug során a változók jelenlegi értékeit kétféleképpen tudjuk megnézni:

1. ***Data*** ablaknál tudjuk megnézni az adott procedúrában, függvényben definiált összes változó jelenlegi értékét, illetve a paraméterben megadott változók értékeit is

például:



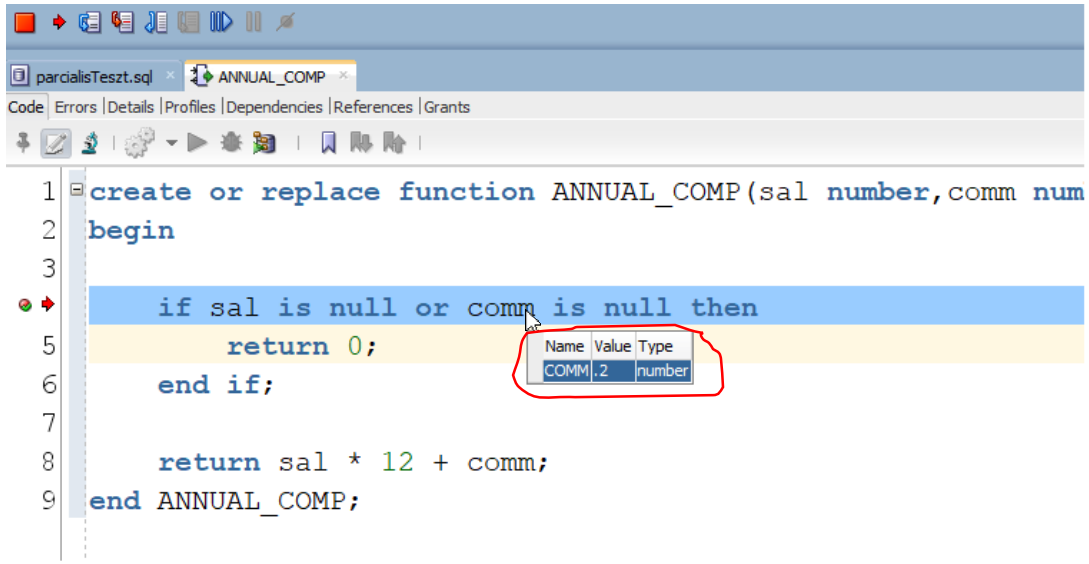
- ***Data*** ablak másik példa:

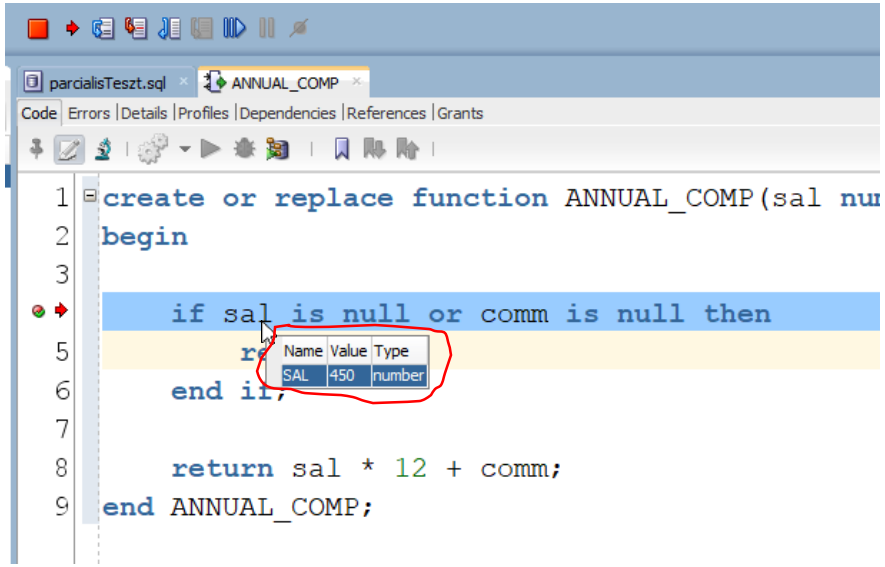


VÁLTOZÓK ÉRTÉKEINEK VIZSGÁLATA

2. Ha az adott **változó nevére rávisszük az egeret**, akkor a debuger meg fogja jeleníteni annak a **változónak az adatait egy kisebb ablakban** (nevezetesen annak **nevét, értékét és típusát**)

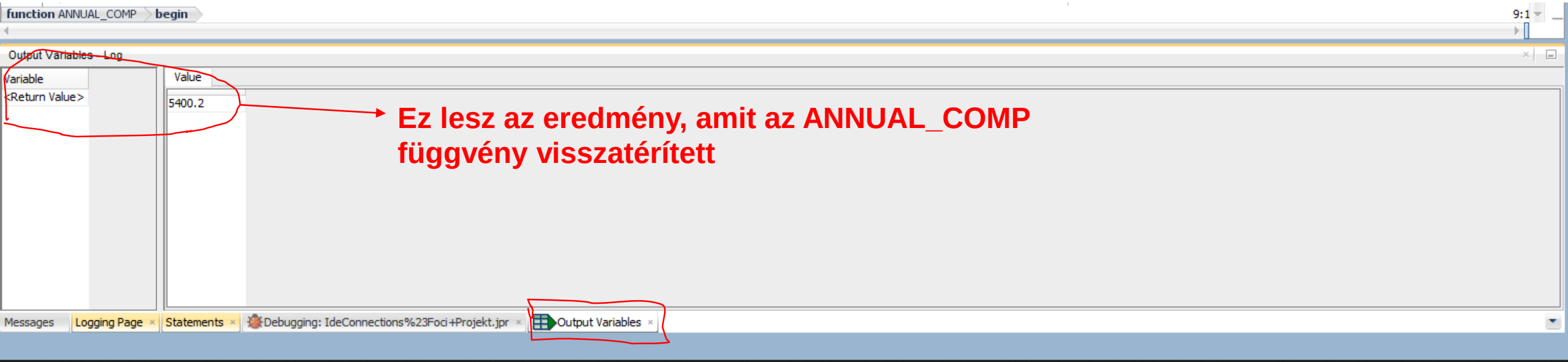
például:





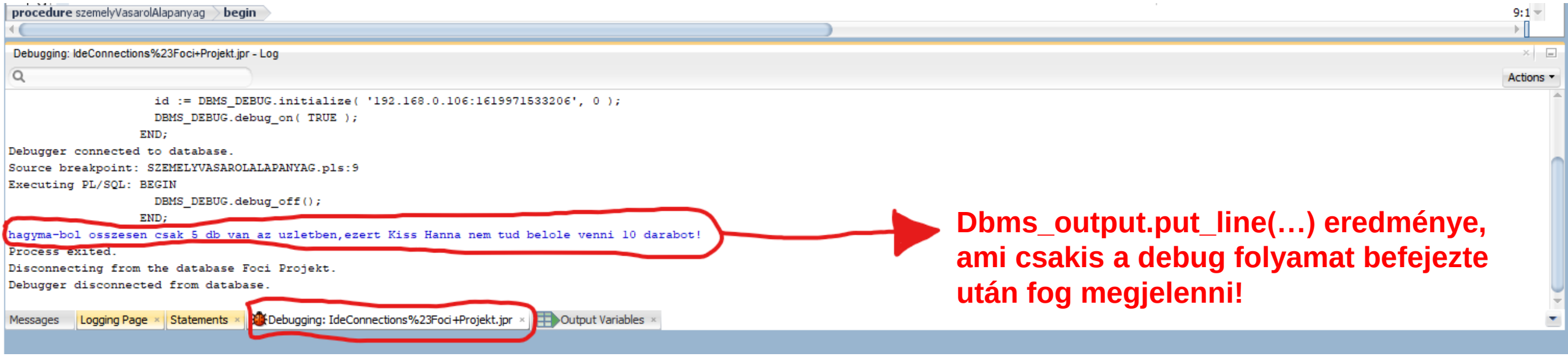
HOL NÉZHETEM MEG, HOGY MIT TÉRIT VISSZA AZ ADOTT PROCEDÚRA / FÜGGVÉNY?

- A debug végén az eredményt, amit vissza fog téríteni az adott procedúra / függvény, azt az ***Output Variables*** ablaknál nézhetem meg.
- Például ha a korábbi slideokban bemutatott **ANNUAL_COMP** debug módban lefuttatom akkor ezt az eredményt fogom látni a korábban említett **Output Variables**-nél:



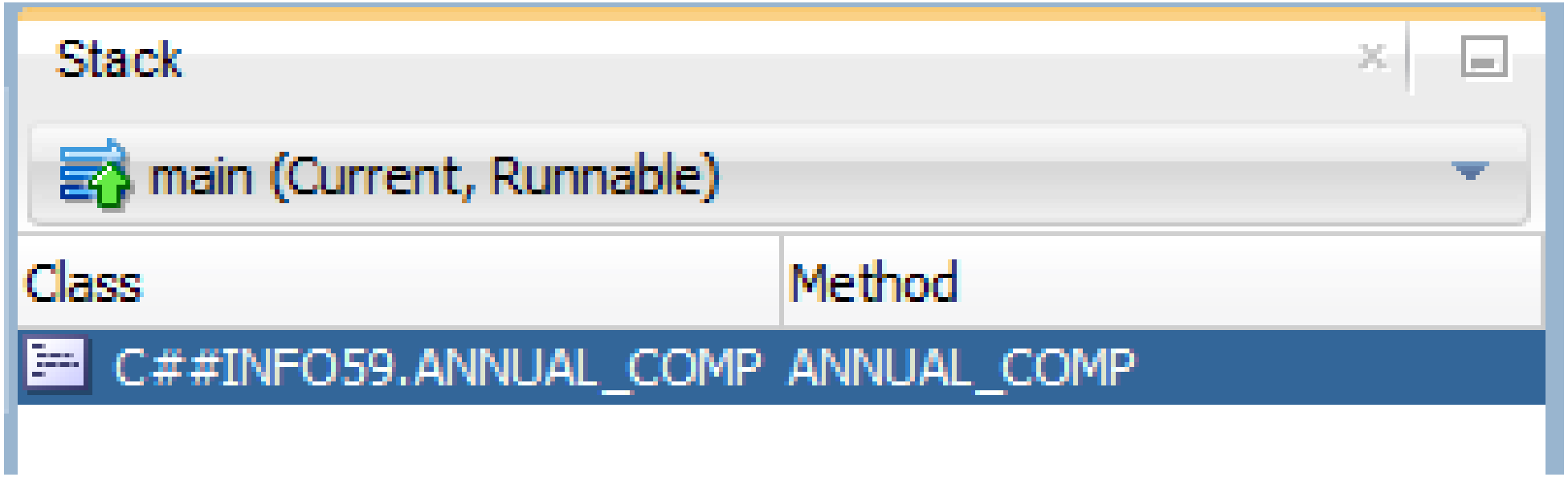
DBMS_OUTPUT UTASITÁS DEBUG MÓDBAN

- A debug során a **dbms_output.put_line(…)** eredménye a **Debugging:IdeConnections…** ablakban fog megjelenni(kék színű szöveggel), **de csakis miután a debug folyamat véget ért.**



STACK ABLAK

- A ***Stack*** ablakban fogjuk látni, hogy a debuger jelenleg mely procedúrában / függvényben van , ahol látni fogjuk a **User** nevét és mellette a **procedúra / függvény nevét**



BONYOLULTABB PL/SQL PROGRAM DEBUGOLÁSA

- Az eddigi slideokban megvizsgáltuk, hogy hogyan tudunk debugolni egyszerű procedúrákat / függvényeket, amelyekben nem voltak egyéb procedúra / függvény hívások
- De viszont felvetődhet a kérdés, hogy mégis hogyan debugolhatunk egy olyan PL/SQL programot, amelyben már több különböző procedúra, függvény is meghívódik?
- Az elkövetkező slideokban erre a kérdésre próbálunk választ adni

BONYOLULTABB PL/SQL PROGRAMOK DEBUGOLÁS LÉPÉSEK

1. Kikeressük a sémából azt a PL/SQL programot (legyen az procedúra vagy függvény attól függően, hogy melyikben van több procedúra / függvény hívás)
2. Kompiláljuk azt debugra (**Compile for Debug**)
3. Elhelyezzük a **Begin-End** közötti sorokban **breakpoint**ot
4. Azokat a **procedúrákat / függvényeket** , amelyekbe be szeretnénk lépni (**Step Into** által) a debug során azokat is **ki kell keresni egyenként a sémából és külön azokat is kompilálni debugolásra (azokra is végrehajtani egy Compile for Debug-ot) , mert anélkül nem fogunk tudni belépni a debug során azokba a procedúrákba / függvényekbe, függetlenül attól, hogy megnyomtuk a Step Into ikont!**
5. Miután **külön lekompiláltuk debugra azokat a procedúrákat / függvényeket is, amelyekbe be szeretnék majd lépni a debug során**, akkor nem marad más hátra , mint megnyomjuk a  ikont, megadjuk az adott PL/SQL program **paramétereit** (ha szükséges azokat megadni), majd végül az **OK** gomb által elinditjuk azt debugolásra.

- Példaképp legyen a következő Főzés nevezetű adatbázisunk megadva:

Kategória (id, név)

Nehézség (id, név)

Recept (id, név, kategória_id, nehézség_id, idő, kalória)

Alapanyag (id, név, mértékegység)

Recept_Alapanyag (recept_id, alapanyag_id, darab)

Személy (id, név, pénz, nehézség_id)

Konyha (személy_id, alapanyag_id, darab)

Főzés (személy_id, recept_id, dátum)

Üzlet (alapanyag_id, ár, darab)

Vásárlás (személy_id, alapanyag_id, dátum, darab)

- A következő feladat van megadva:

2. Írj tárolt eljárást, mely segítségével megvalósítható egy alapanyag megvásárlása egy adott személy által.

Bemenet: személy neve, alapanyag neve, darab.

Ellenőrizd, hogy létezik-e az adott személy és alapanyag. Ha az alapanyag nem létezett, akkor szúrja be új adatként az alapanyag táblába (ár: 1, mértékegység: db). Ha létezett, nézze meg hány darab van belőle az üzletben és csak akkor tudja megvenni ha van elég. Ha meg tudja venni, szúrja be a vásárlás táblába és csökkentse a személy pénzét annyival, amennyibe került az alapanyag.

A következő slideokban meg lesznek adva a feladathoz tartozó PL/SQL program, illetve azon procedúrái, függvényei, amiket meghívunk majd.

```
/
create or replace function szemelyLetezesEllenorzo(pSzemelyNev szemely.nev%type) return number is
    v_szemelyID number;
begin
    select id
    into v_szemelyID
    from szemely
    where nev = pSzemelyNev;

    return v_szemelyID;

    exception when no_data_found then return -1;
    when others then dbms_output.put_line('Hiba lepett fel a szemelyLetezesEllenorzo fuggvenyben');
end szemelyLetezesEllenorzo;
/
```

```
/
create or replace function alapanyagLetezesEllenorzo(pAlapanyagNev alapanyag.nev%type) return number is
    v_alapanyagID number;
begin
    select id
    into v_alapanyagID
    from alapanyag
    where nev = pAlapanyagNev;

    return v_alapanyagID;

    exception when no_data_found then return -1;
    when others then dbms_output.put_line('Hiba lepett fel az alapanyagLetezesEllenorzo fuggvenyben!');
end alapanyagLetezesEllenorzo;
/
```

```
/
create or replace procedure insertAlapanyag(pAlapanyagNev alapanyag.nev%type, pMertekegyseg alapanyag.mertekegyseg%type default 'db') is
    v_id number;
begin
    insert into alapanyag values((select max(id)+1 from alapanyag),pAlapanyagNev,pMertekegyseg)
    returning id into v_id;
    dbms_output.put_line('Sikerult beszurni a(z) ' || pAlapanyagNev || ' az alapanyag tablaba!');

    insert into uzlet values (v_id,1,10);
    dbms_output.put_line('Sikerult beszurni a(z) ' || pAlapanyagNev || ' az uzlet tablaba!');

    exception when others then dbms_output.put_line('Hiba lepett fel az insertAlapanyag proceduraban!');
end insertAlapanyag;
/
```

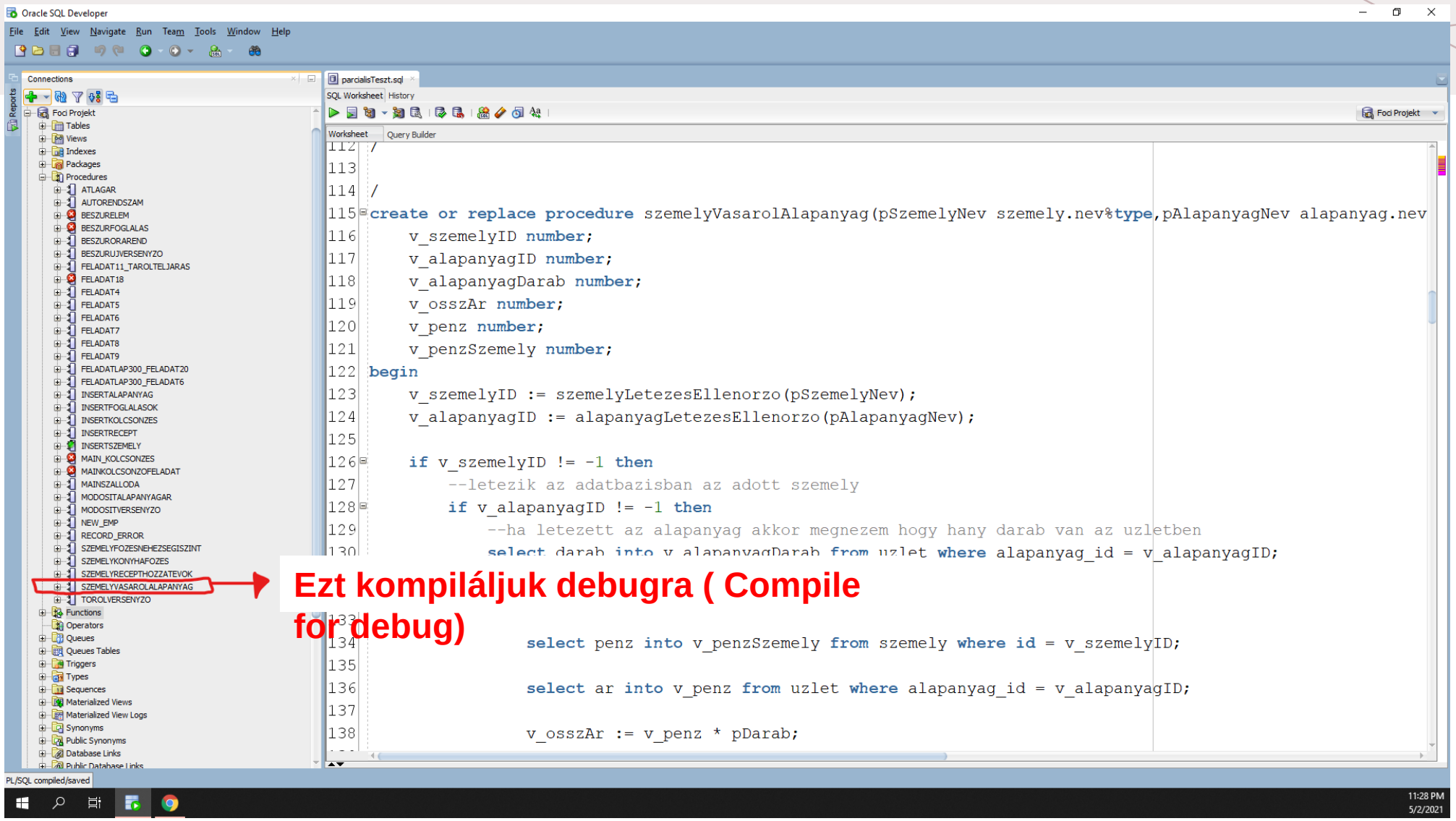
```
/
create or replace procedure insertSzemely(pSzemelyNev szemely.nev%type,pPenz szemely.penz%type default 0 , pNehezseg_id szemely.nehezseg_id%type default 1) is
begin
    insert into szemely values((select max(id) + 1 from szemely),pSzemelyNev,pPenz,pNehezseg_id);
    dbms_output.put_line('Sikeresen be lett szurva a szemely tablaba a kovetkezo szemely: ' || pSzemelyNev);

    exception when others then dbms_output.put_line('Hiba lepett az insertSzemely proceduraban!');
end insertSzemely;
/
```

```
/
create or replace procedure szemelyVasarolAlapanyag(pSzemelyNev szemely.nev%type,pAlapanyagNev alapanyag.nev%type,pDarab number) is
    v_szemelyID number;
    v_alapanyagID number;
    v_alapanyagDarab number;
    v_osszAr number;
    v_penz number;
    v_penzSzemely number;
begin
    v_szemelyID := szemelyLetezesEllenorzo(pSzemelyNev);
    v_alapanyagID := alapanyagLetezesEllenorzo(pAlapanyagNev);

    if v_szemelyID != -1 then
        --letezik az adatbazisban az adott szemely
        if v_alapanyagID != -1 then
            --ha letezett az alapanyag akkor megnezem hogy hany darab van az uzletben
            select darab into v_alapanyagDarab from uzlet where alapanyag_id = v_alapanyagID;

            if v_alapanyagDarab >= pDarab then

                select penz into v_penzSzemely from szemely where id = v_szemelyID;

                select ar into v_penz from uzlet where alapanyag_id = v_alapanyagID;

                v_osszAr := v_penz * pDarab;

                if v_osszAr <= v_penzSzemely then
                     --ha van elegendo darab az adott alapanyagbol es penze a szemelynek, akkor meg tudja azokat vasarolni
                    insert into vasarlas values(v_szemelyID,v_alapanyagID,pDarab,sysdate);

                    update szemely set penz = penz - v_osszAr where id = v_szemelyID;
                    dbms_output.put_line('Sikeres vasarlas tortent,es az adott szemelynek a penze is sikeresen le lett csokkentve!');
```

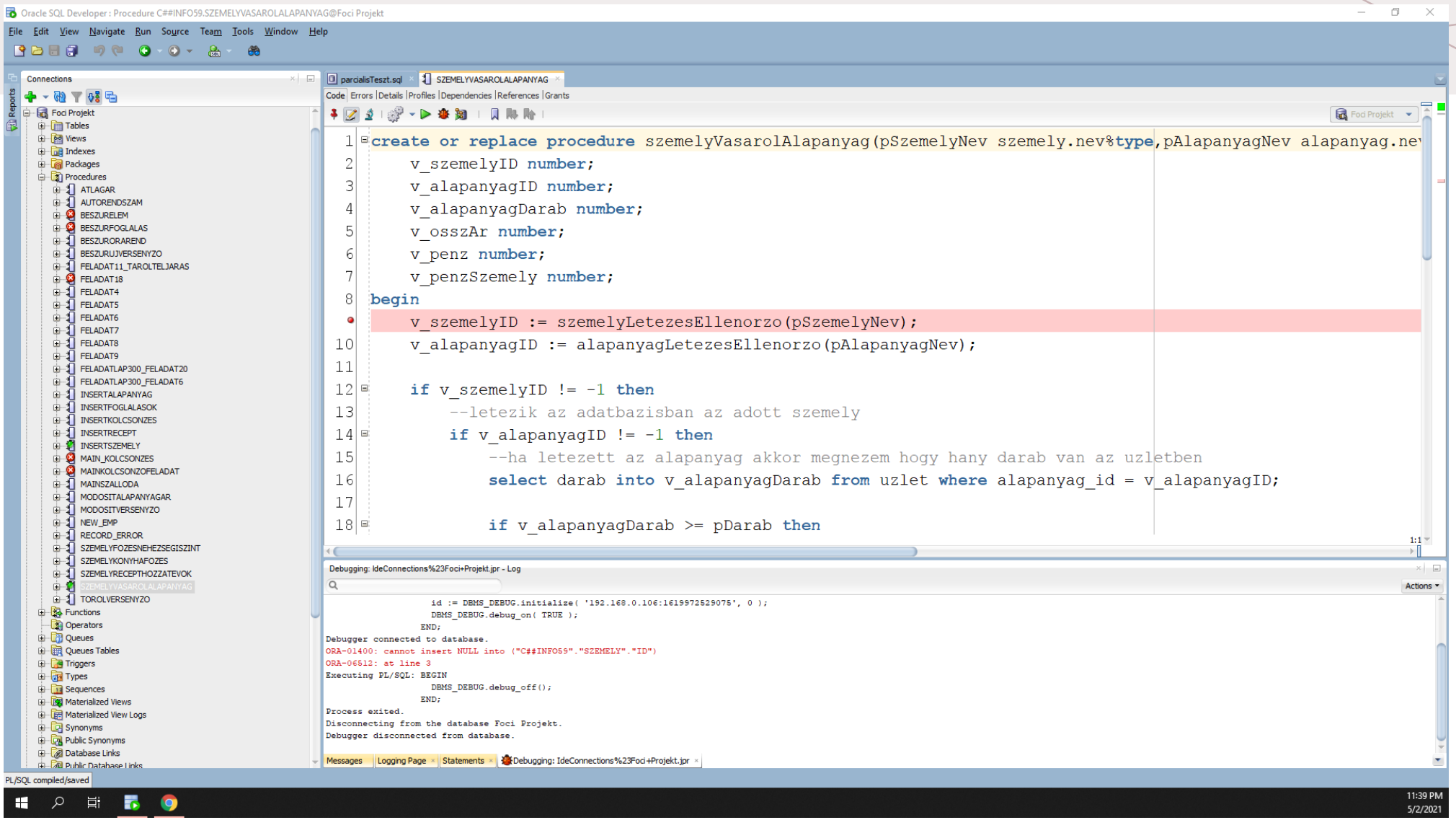
```
                --mivelhogy az adott szemely megvasarolta a recepthez szukseges anyagokat,ezert az uzletbol is csokkeni fog az alapanyagok szamossaga(darabszama)
                update uzlet set darab = darab - pDarab where alapanyag_id = v_alapanyagID;
            else
                dbms_output.put_line(pSzemelyNev || ' nincs elegendo penze,hogy vasaroljon ' || pDarab || ' darab ' ||  pAlapanyagNev || '-t!');
            end if;
        else
            dbms_output.put_line(pAlapanyagNev || '-bol osszesen csak ' || v_alapanyagDarab || ' db van az uzletben,ezert ' || pSzemelyNev || ' nem tud belole venni ' || pDarab || ' darabot!');
        end if;
    else
        --nem letezik az adott alapanyag az adatbazisban ezert be kell majd ezt szurjam
        dbms_output.put_line(pAlapanyagNev || ' nem letezik az adatbazisban,ezert ez be lesz szurva!');
        insertAlapanyag(pAlapanyagNev);
    end if;
else
    dbms_output.put_line(pSzemelyNev || ' nem letezik az adatbazisban,ezert ez be lesz szurva!');
    insertSzemely(pSzemelyNev);
end if;

exception when others then dbms_output.put_line('Hiba lepett fel a szemelyVasarolAlapanyag proceduraban!');
end szemelyVasarolAlapanyag;
/
```

Most pedig ezt a **szemelyVasarolAlapanyag** procedúrát fogjuk debugolni

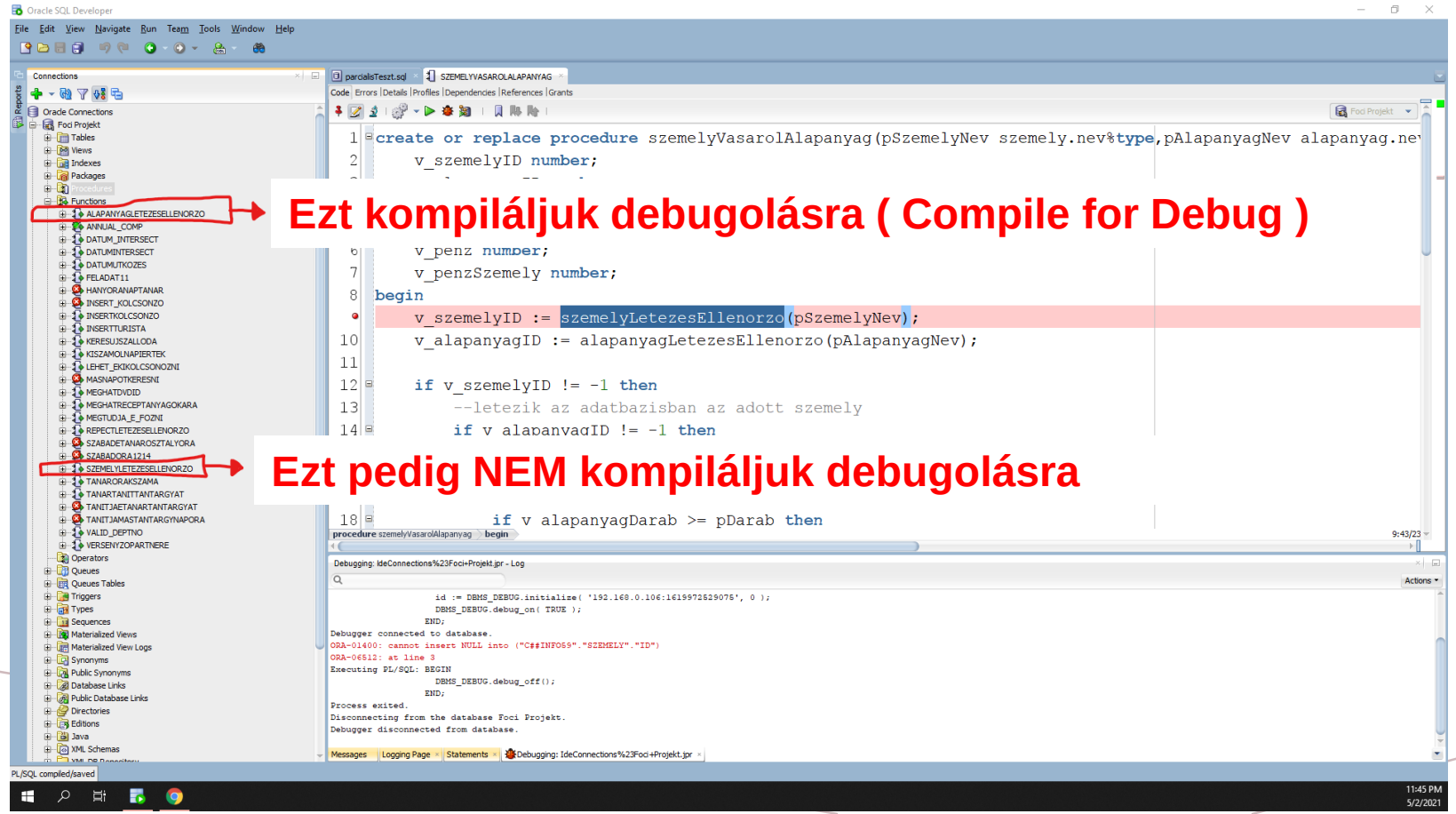
1. MEGKERESSÜK EZT A PROCEDÚRÁT A SÉMÁNKBÓL



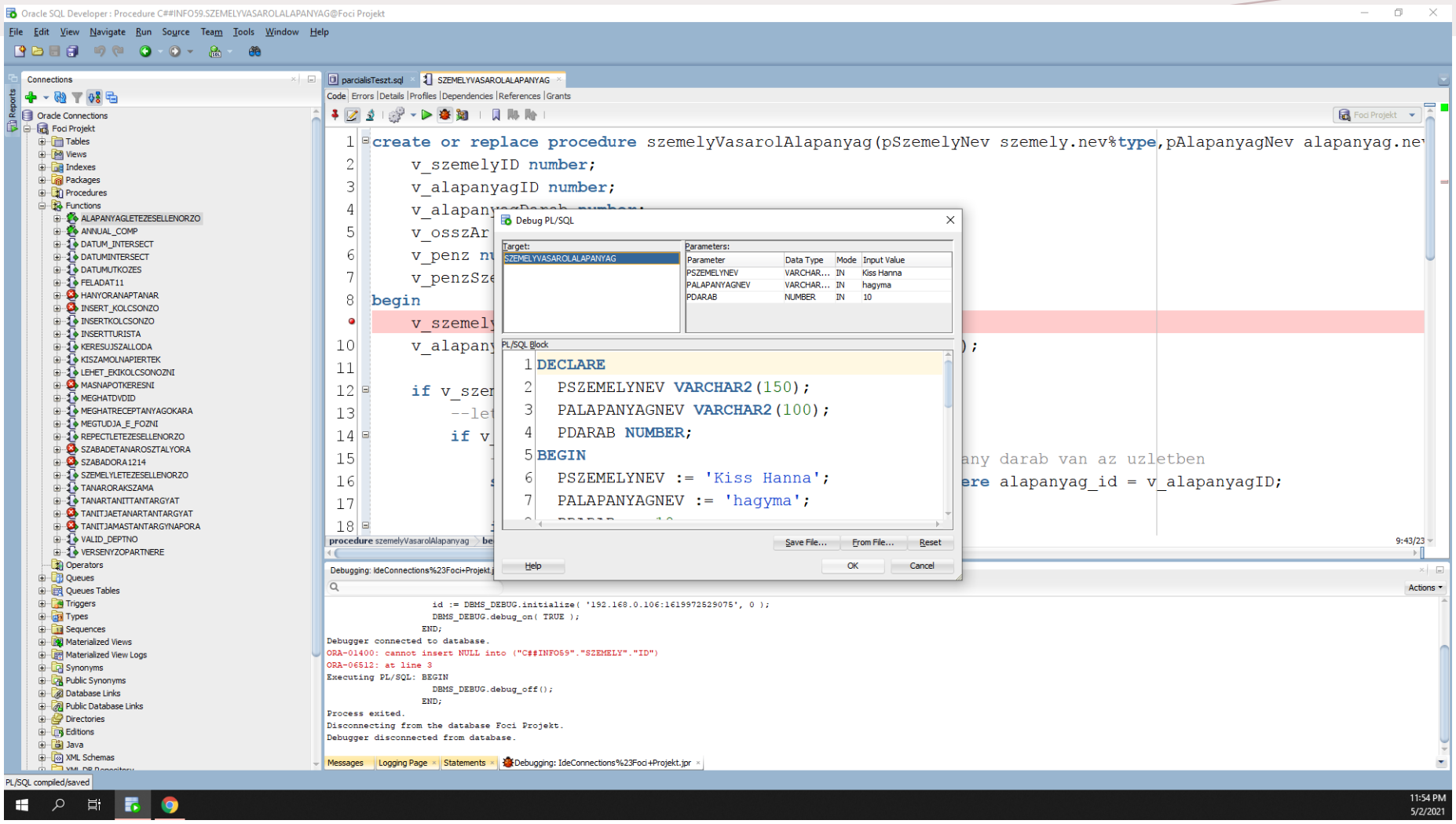
2. TEGYÜNK BREAKPOINTOT A BEGIN ÉS END SOROK KÖZÖTT



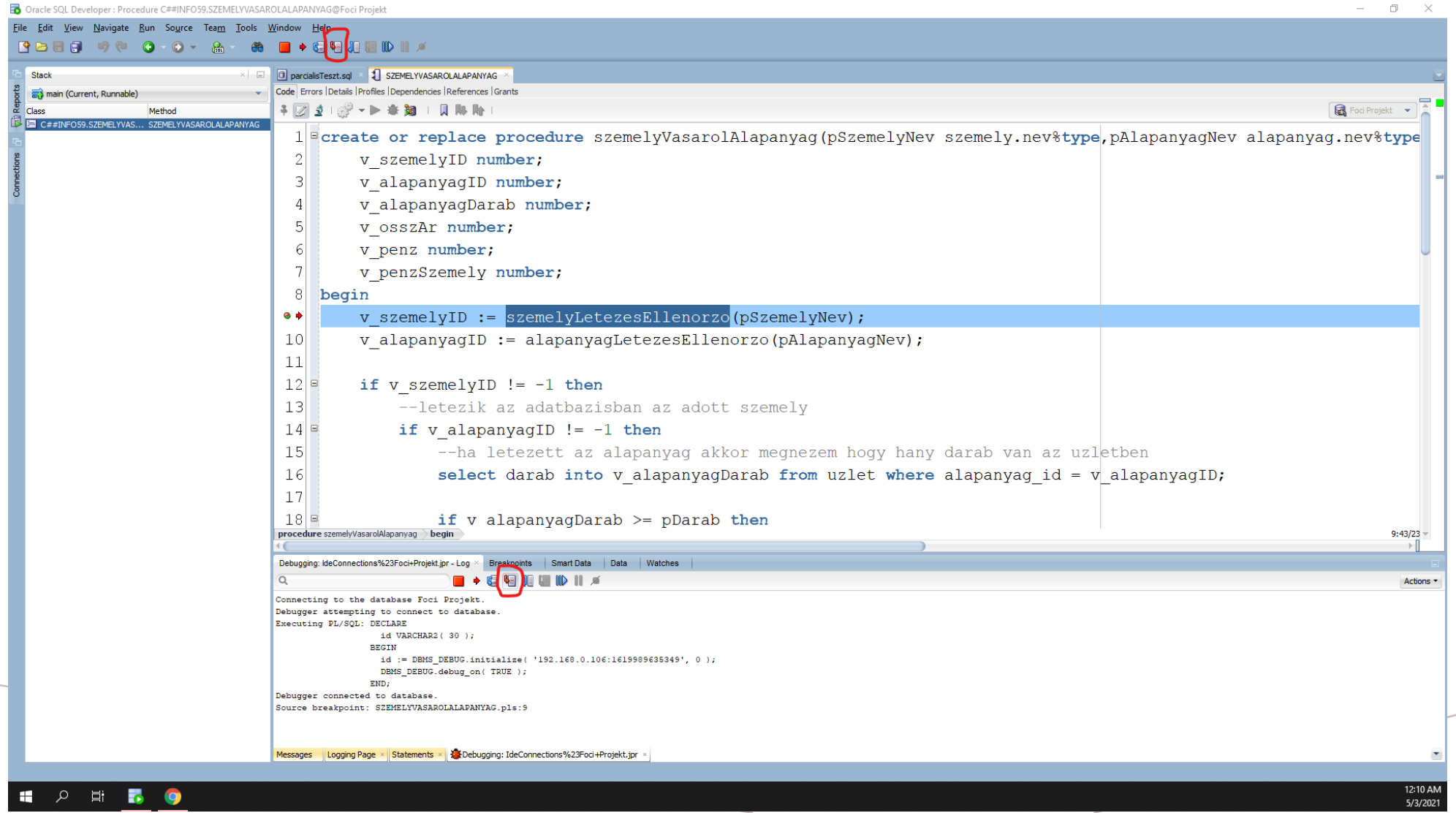
- Továbbá csináljuk a következőket:
  - kompiláljuk az ***alapanyagLetezesEllenorzo*** függvényt debugolásra (Compile for Debug)
  - És a ***szemelyLetezesEllenorzo*** függvényt pedig ne kompiláljuk debugolásra



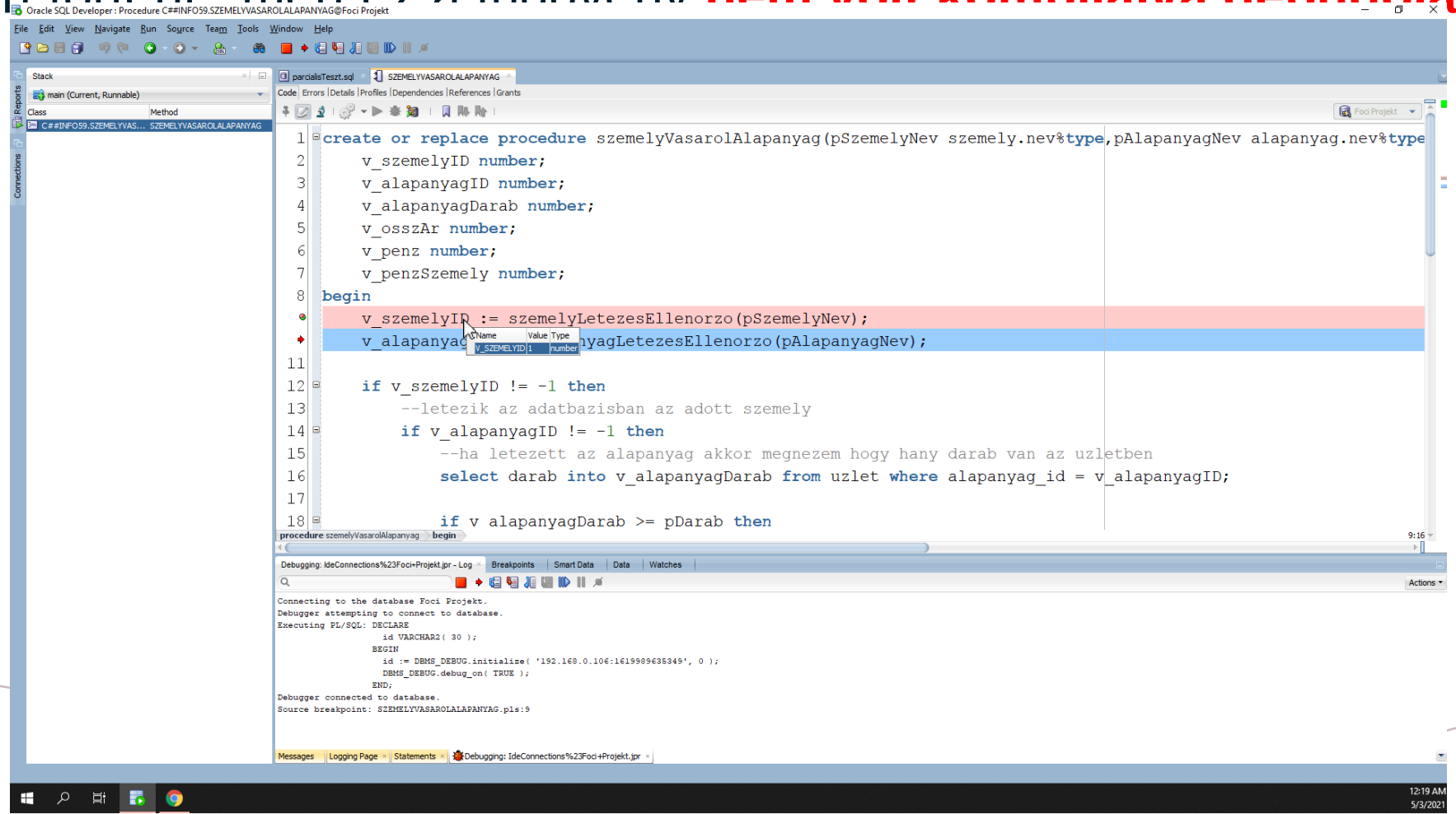
A  IKONRA RÁNYOMVA MEGADJUK A PROCEDÚRA PARAMÉTEREIT, MAJD AZ “**OK**” GOMBRA RÁKATTINTVA ELINDÍTJUK EZT DEBUGOLÁSRA



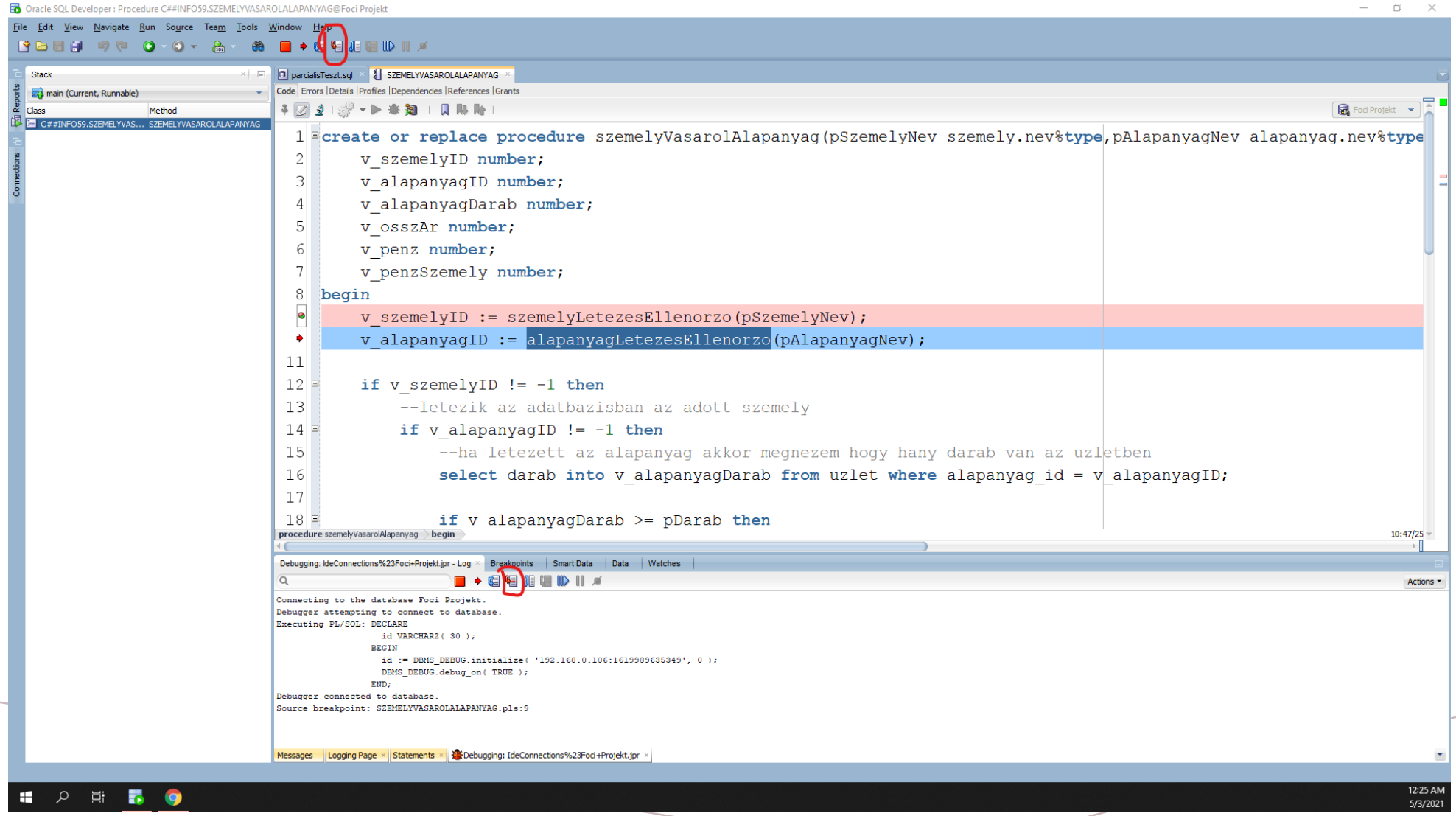
- Amint látható a program megállt a 9-dik sorban.
- Most probáljunk belépni ebbe a **szemelyLetezesEllenorzo** függvénybe a **Step Into** gomb által (ikonra rákattintva vagy F7-et nyomva)



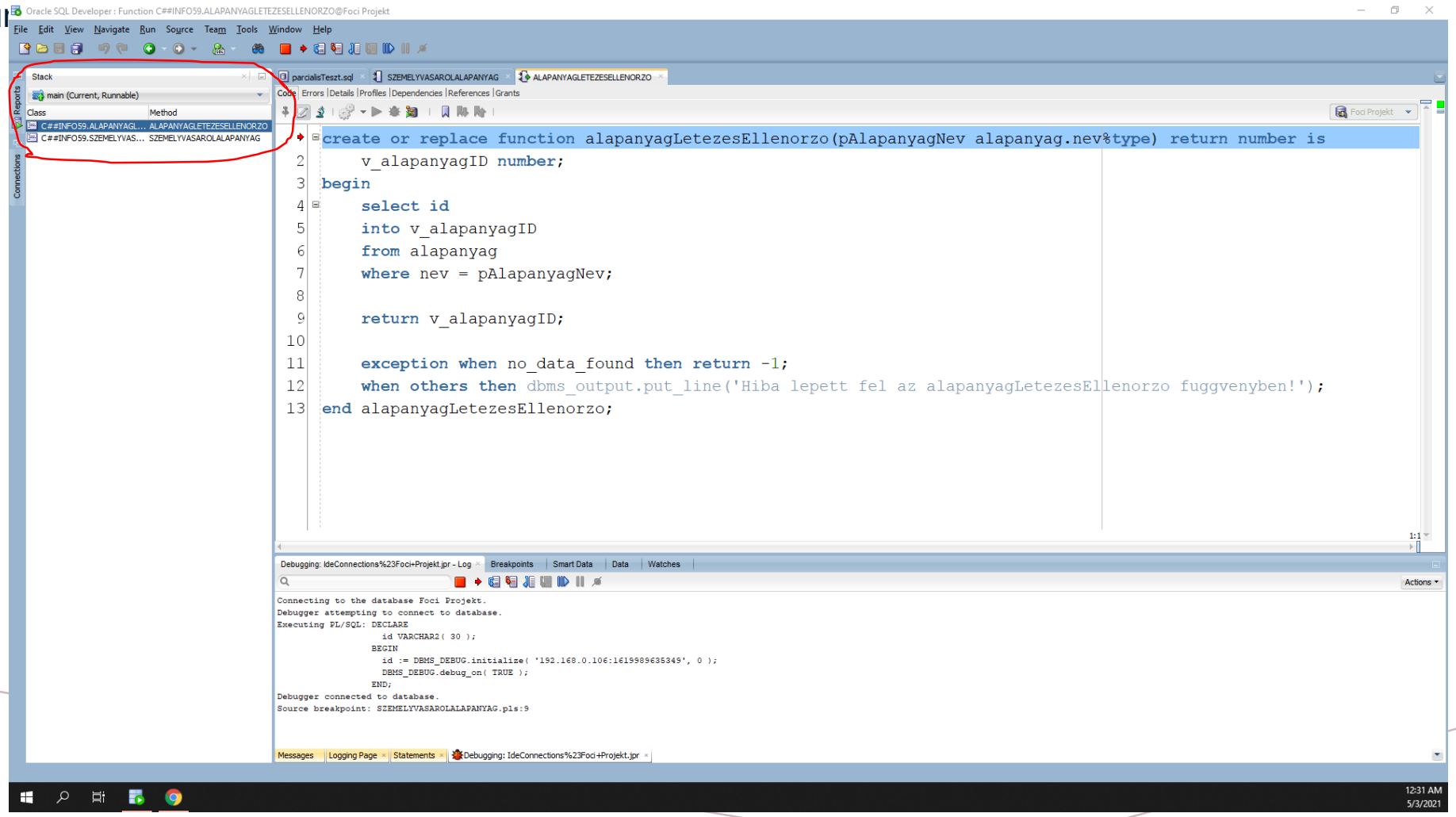
- Amint látható, hiába kattintottunk a **Step Into** gombra, a debuger **nem lépett be** ebbe a **szemelyLetezesEllenorzo** függvénybe, hanem azt csakis végrehajtotta és az eredményt visszatérítette a **v_szemelyID** változóba
- ez azért történt, mert ez a függvény **nem volt kompilálva debugolásra**



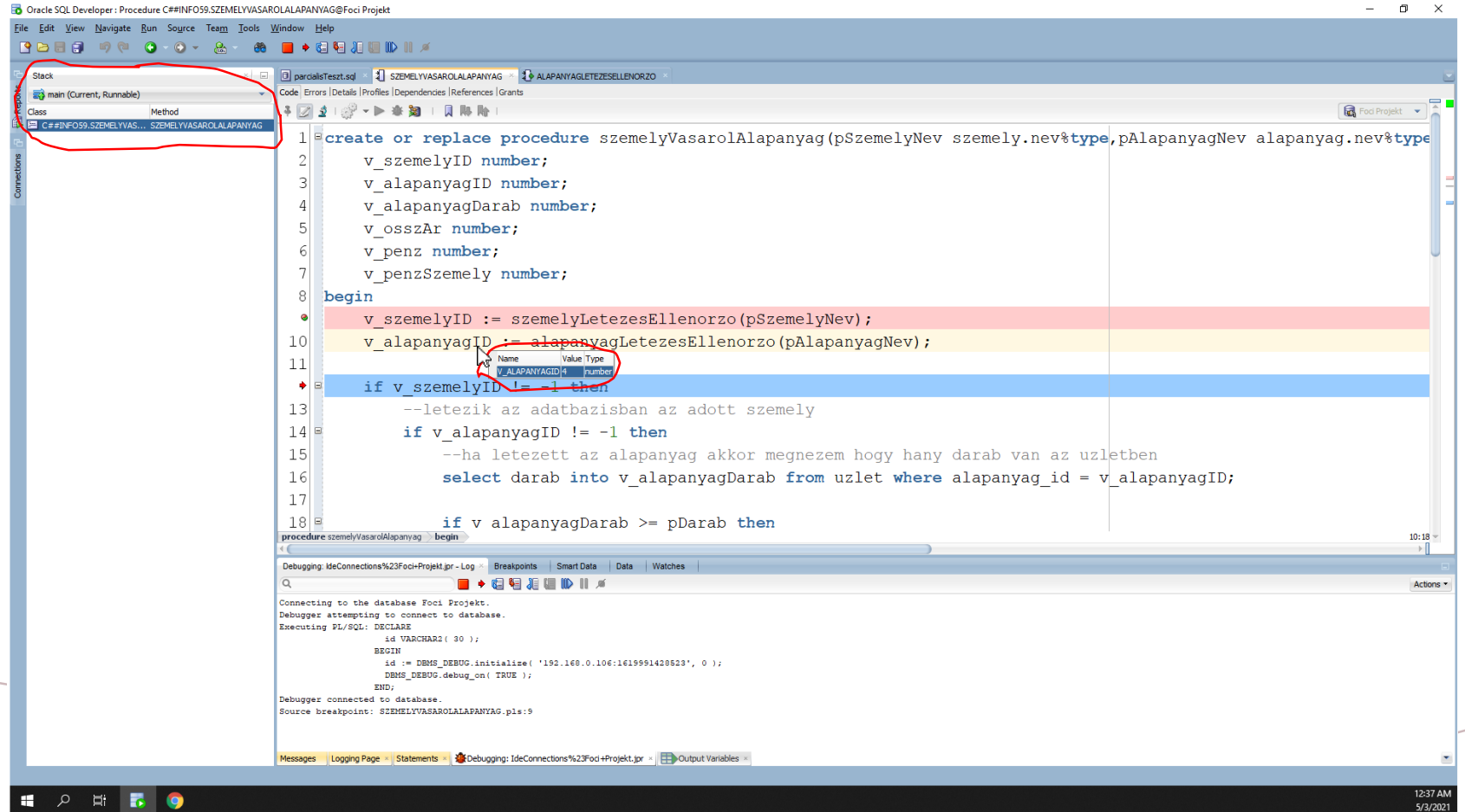
- Most probáljunk belépni az **alapanyagLetezesEllenorzo** függvénybe (**Step Into** segitségével)



- Amint látható most bent vagyunk az **alapanyagLetezesEllenorzo** függvényben, ahol tovább tudunk debugolni használva a **Step Over** gombot (F8 gombot nyomva)
- Ez azért történt meg a másik függvénnyel szemben, mert ezt az **alapanyagLetezesEllenorzo** függvényt **kompiláltuk debugolásra** (míg a **szemelyLetezesEllenorzo** függvényt nem)
- **Ez a dolog érvényes procedúrákra is , nemcsak a függvényekre!**
- Ugyanakkor a **Stack** ablakban is megjelenik, hogy per pillanat az **alapanyagLetezesEllenorzo** függvényben vagyu



- Ha a **Step Over**-t használtuk, hogy kilépjünk az **alapanyagLetezesEllenorzo** függvényből, akkor látni fogjuk, hogy visszakerülünk a **szemelyVasarolAlapanyag** procedúrába, ahol a program a következő sorhoz fog ugorni
- Amit visszatéritett az **alapanyagLetezesEllenorzo** függvény ,az el lesz tárolva a **v_alapanyagID** változóba
- Ugyanakkor azt is látjuk, hogy a Stack-nél az **alapanyagLetezesEllenorzo** függvény eltünt és újra csak a **szemelyVasarolAlapanyag** procedúra van, ez által jelezvén nekünk, hogy újra a **szemelyVasarolAlapanyag** procedúrában vagyunk



MI A TANULSÁG?

- Abban az esetben, ha én akarok debugolni egy bonyolultabb PL/SQL programot úgy, hogy belépjek bizonyos procedúrákba, függvényekbe a **Step Into** által, akkor **meg kell győződjek** először, hogy azok a **procedúrák, függvények le vannak-e kompilálva debugolásra vagy sem.**

Köszönöm szépen a figyelmet! ☺