Kako dibagirati Forms programe ili (alternativan naziv) Kako debagirati (engl. debugging) Forms programe

Zlatko Sirotić, univ.spec.inf.

ISTRA TECH d.o.o.

Pula

ISTRA TECH d.o.o. & me

- ISTRA TECH je novo ime (od 2015.) poduzeća **Istra informatički inženjering**, osnovanog 1990. godine.
- Radim na informatičkim poslovima od 1984. godine.
- Oracle softverske alate (baza, Designer CASE, Forms 4GL, Reports, Java) koristim oko 25 godina.
- Objavljivao sam stručne radove na kongresima / konferencijama HrOUG, JavaCro, CASE, KOM, "Hotelska kuća", te u časopisima "Mreža", "InfoTrend" i "UT".
- Neka moja programska rješenja objavljivana su na web stranicama firmi Oracle i Quest.
- Od 2012. sam vanjski suradnik na Fakultetu informatike Pula.

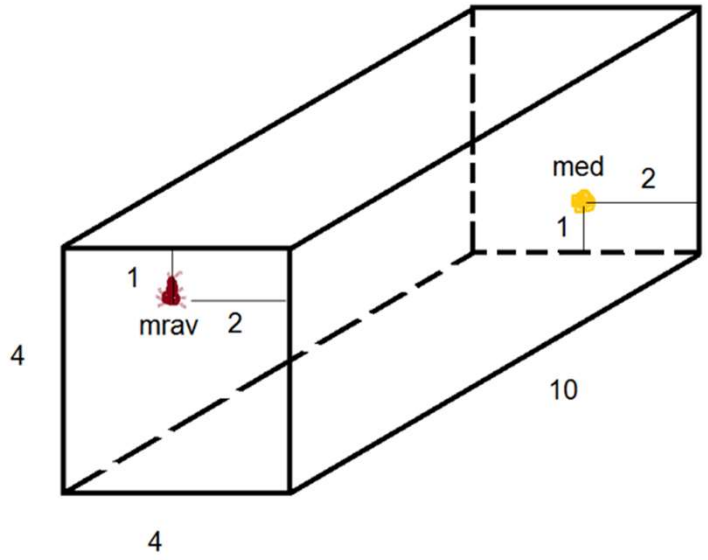
HrOUG i JavaCro & me

- Prvi put predavač na HrOUG 2002.
  Sudjelovao sam 18 puta i održao 24 predavanja.
- Prvi put predavač na JavaCro 2014.
  Sudjelovao sam 4 puta i održao 4 predavanja.
- "Odijelo ne čini čovjeka"
  i "Naslov ne čini prezentaciju", ali možda ipak pomaže:
  2004. Kako spriječiti "začarani krug"
  (rješavanje određenog tipa poslovnih pravila u Oracle BP)
  2007. Objektno-relacijske baze podataka – postoje li?
  2013. Što poslije Pascala? Pa … Scala!
  2014. Trebaju li nam distribuirane baze u vrijeme oblaka?
  2015. Višestruko nasljeđivanje - san ili Java 8?
  2020. Postoji li samo jedna "ispravna" arhitektura
  web poslovnih aplikacija

Mrav i med na prizmi (i valjku)

- Na HrOUG 2015. prvi put sam spomenuo zadatak Mrav i med na prizmi (verzija 0), unutar predavanja Povratak u Prolog.

  Na HrOUG 2019. sam najavio prezentaciju (verzija 1).



- Na stranici http://www.istratech.hr/category/blog/

nalazi se najnovija verzija (2), uz još neka predavanja:

- Mrav i med na prizmi - verzija 2
- Povratak u Prolog - verzija 2
- Strukturna složenost algoritama

Teme

- Oracle Forms
- Usporedba Oracle Forms : Oracle ADF
- Oracle Forms 6i debugging
- Oracle Forms 12c debugging

Je li izrada web poslovnih aplikacija postala previše kompleksna?

- Različiti pristupi pisanju "tankog" ili "debelog" koda u različitim arhitekturnim slojevima; 1. varijanta je najčešća
- 8. varijanta (sve tanko) ne postoji – ipak treba nešto (i) programirati ☺

Oracle Forms alat

- Oracle Forms je Rapid Application Development (RAD) alat, koji je Oracle napravio početkom 80-ih, nedugo nakon nastanka Oracle baze verzije 2 (verzija 1 nije nikada postojala), i radio je kao znakovno orijentirana (character mode) aplikacija na serveru.
- Sredinom 90-ih napravljena je klijent-server GUI varijanta. Klijent-server varijanta pratila je baze 6, 7 i 8, a Forms verzije bile su 4, 4.5, 5, 6 i 6i. U Forms verziji 6 pojavila se paralelno i web varijanta - Web Forms.
- Nakon verzije 6i, klijent-server varijanta više ne postoji, tj. verzije od 9i do 12.2 (verzije 7 i 8 nikad nisu postojale) rade isključivo kao web Forms aplikacija (koja nije baš jeftina).
- Zadnja verzija je 12.2.1.4.

Oracle Forms
- web varijanta

- Web varijanta radi tako da Java applet (ili samostalni Java kod) na klijentu (koji se brine samo za kreiranje korisničkog sučelja) surađuje sa Forms servisom na aplikacijskom serveru.
- Forms servis manje-više čini onaj isti kod (pisan u C-u) kao i u klijent-server varijanti.
- **I u ovoj varijanti:**
  - Forms servis drži stalnu konekciju s bazom;
  - korisnik stalno drži Forms module s kojima radi, sve dok ne završi rad.
- U suštini, ovaj način rada nema nekih značajnih razlika u odnosu na klijent-server rad.

Forms COMMIT i POST procesi

- Među najvažnijim, ali i najsloženijim Forms procesima, jesu COMMIT i POST procesi.
- COMMIT proces može se jednostavno (i točno) prikazati kao:

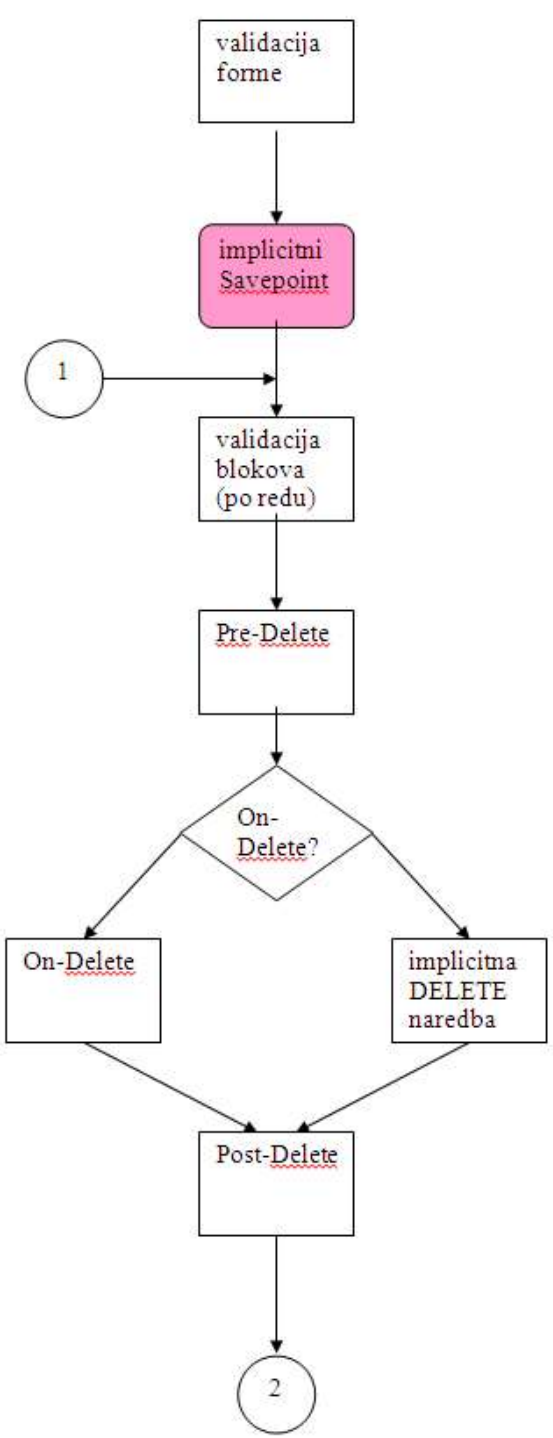
  COMMIT proces =

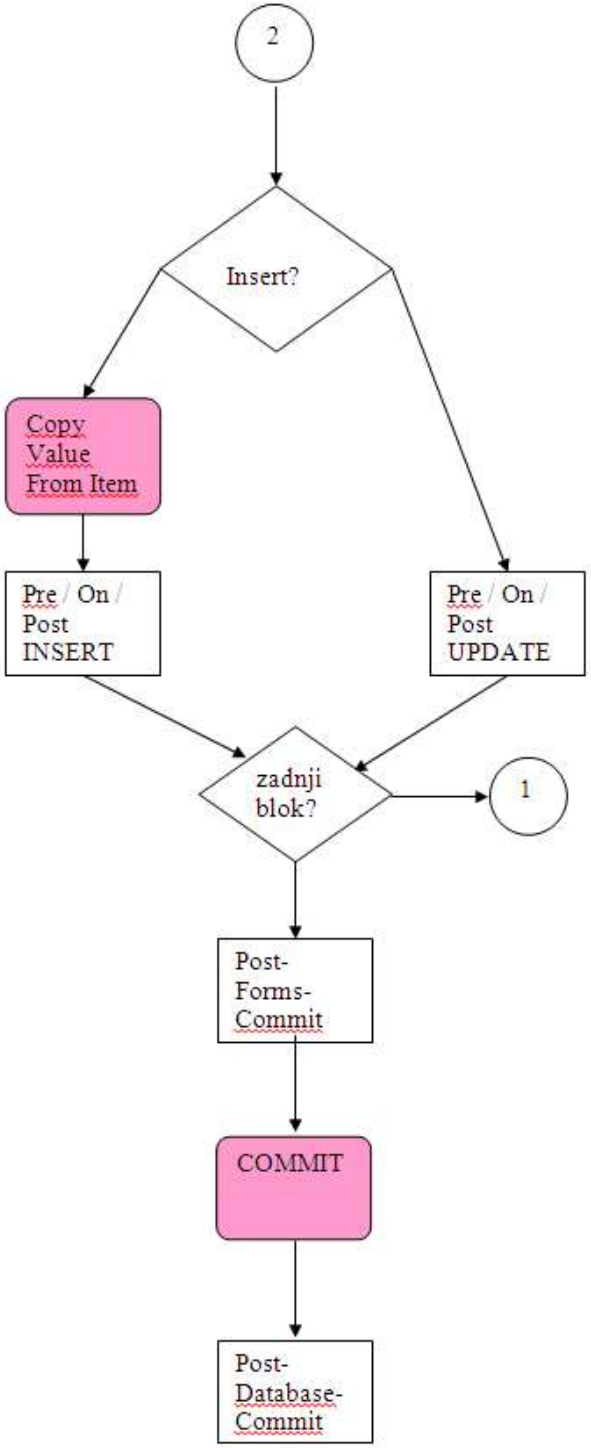
  POST proces +

  COMMIT naredba na bazi +

  Forms okidač Post-Database-Commit

- Suština je u tome da se ažuriranje redaka (unos / izmjena / brisanje) prvo radi samo unutar Forms bloka. Kad korisnik odabere gumb Save ili funkcijsku tipku F10, Forms šalje odgovarajuće DML naredbe na bazu, te daje COMMIT na bazu (ako je riječ o COMMIT procesu).

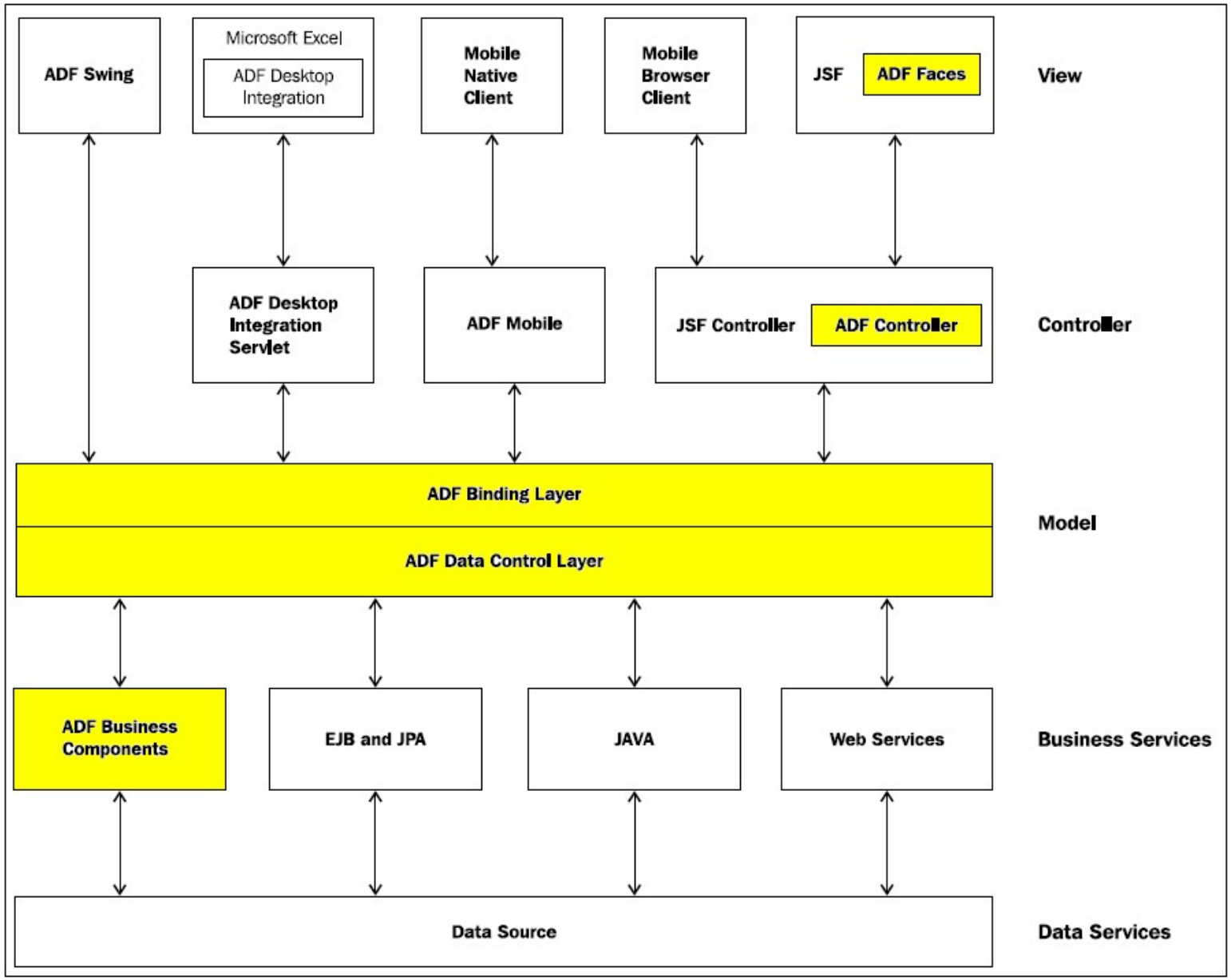

validacija forme
implicitni Savepoint
1
validacija blokova (po redu)
Pre-Delete
On-Delete?
On-Delete
implicitna DELETE naredba
Post-Delete
2


2
Insert?
Copy Value From Item
Pre / On / Post INSERT
Pre / On / Post UPDATE
zadnji blok?
1
Post-Forms-Commit
COMMIT
Post-Database-Commit

Oracle ADF (Application Development Framework)

- 6 mjeseci nakon što je Sun izdao verziju Jave 1.0, u Oracleu su odlučili raditi RAD alat temeljen na jeziku Java. Prvo izdanje frameworka, koji se tada nije zvao ADF već JBO (Java Business Objects), uslijedilo je 1999. godine. Ubrzo mu je ime promijenjeno u BC4J (Business Components for Java).
- BC4J je pokrivao onaj dio koji danas pokriva ADF BC.
  Uz BC4J, Oracle je počeo razvijati i odgovarajući IDE JDeveloper, licencirajući 1998. tadašnji Borlandov alat JBuilder. Oracle je 2001. temeljito preradio JDeveloper, pri čemu ga je u potpunosti "prepisao" u Java kod.
  Ubrzo je termin BC4J zamijenjen sa ADF.
- Od 2005. godine Oracle JDeveloper IDE je besplatan.
- **Verzija ADF Essentials je besplatna od ljeta 2012.**
- Zadnja verzija je 12.2.1.4.

Oracle ADF - struktura



Forms i ADF primjeri veza klijent - aplikacijski modul - baza podataka

Alat i varijanta	Veza između klijenta i aplikac. modula	Veza između aplikac. modula i baze podataka	Broj potrebnih aplikac. modula i konekcija na bazu za 1000 klijenata
Oracle Forms	stalna	stalna	1000 modula 1000 konekcija na bazu
Oracle ADF varijanta 1	Reserved (stalna)	stalna - default	1000 modula 1000 konekcija na bazu
Oracle ADF varijanta 2	Managed - default (relativno stalna)	stalna - default	npr. 100 do 1000 modula npr. 100 do 1000 konekcija na bazu
Oracle ADF varijanta 3	Managed - default (relativno stalna)	nestalna	npr. 100 do 1000 modula npr. 10 konekcija na bazu
Oracle ADF varijanta 4	Unmanaged (nestalna)	nestalna	npr. 100 modula npr. 10 konekcija na bazu

Forms 6i debugging

- U Forms Builderu se na određenom Forms modulu pokrene ikona **Run Form Debug** (ikona prikazuje bubu).
- U Forms Runtime, u Forms Debugger prozoru, odabere se željeni kod, tj. Forms trigger ili Forms procedura / funkcija, u srednjem dijelu Forms Debugger prozora.
- Može se odabrati i pohranjena procedura na bazi (u Database Objects), ali mora biti kompajlirana sa

  ALTER FUNCTION sekvenca COMPILE DEBUG;
- Na gornjem dijelu prozora (gdje se prikazuje izvorni kod sa brojevima redaka) kreira se barem jedan Break Point, npr. tako da se napravi dvostruki klik na neku izvršnu naredbu (ne može biti BEGIN, END, komentar i sl.).
- Zatim se pokrene ikona Close (crveni X), čime se starta Forms modul.

Forms 6i debugging

- Kada program u toku izvođenja dođe do naredbe na kojoj je postavljen Break Point, opet se prikaže Forms Debugger.
- Sada možemo npr. gledati sadržaj parametara ili lokalnih varijabli, gledati izvorni kod i:

- ići naredbu po naredbu kroz izvorni kod, ili ući u potprogram (**Step Into**)
- izvršiti cijeli potprogram kao jednu naredbu (**Step Over**)
- izaći iz potprograma (**Step Out**)
- vratiti se u Forms Runtime (**Go** – slika munje).

- Treba naglasiti da:

- **Break Point vrijedi samo za Forms trigger u kojem je postavljen, a ne za drugi Forms trigger**
- **Forms Debugger ne prolazi kroz trigger baze, niti procedure / funkcije koje se iz njega pozivaju, a nemaju Break Point.**

Forms 6i debugging

- Prikazat ćemo jednostavan Forms modul,
  - baza generira ID kroz BIR trigger i funkciju sekvenca:

```
CREATE TABLE M_ARTIKLI (
  ID      NUMBER (8)     NOT NULL,
  SIFRA   VARCHAR2 (10) NOT NULL,
  NAZIV   VARCHAR2 (40) NOT NULL,
  CONSTRAINT ART_UK UNIQUE       (SIFRA) USING INDEX,
  CONSTRAINT ART_PK PRIMARY KEY ( ID )   USING INDEX
);
CREATE OR REPLACE TRIGGER BIR_M_ARTIKLI
 BEFORE INSERT ON M_ARTIKLI
 FOR EACH ROW
BEGIN
  :NEW.id := sekvenca (:NEW.id);
END;
```

Forms 6i debugging

- Funkcija na bazi izgleda ovako:

```
CREATE FUNCTION sekvenca (id_p NUMBER) RETURN NUMBER IS
  id_l NUMBER;
BEGIN
  /* ako je klijent poslao vrijednost ID-a,
  ne mijenja se
  */
  IF id_p IS NULL THEN
    id_l := obuka_seq.NEXTVAL;
  ELSE
    id_l := id_p;
  END IF;

  RETURN id_l;
END;
```

Forms 6i debugging

- Forms modul ima PRE-INSERT trigger,
  koji služi samo za demonstraciju debugging procesa
  (dio koda je za sada komentiran):

```
MESSAGE ('In pre-insert trigger');
PAUSE;

/*
DECLARE
  i INTEGER := 0;
BEGIN
  i := sekvenca (NULL);
  MESSAGE ('i: ' || i);
  PAUSE;
END;
*/
```

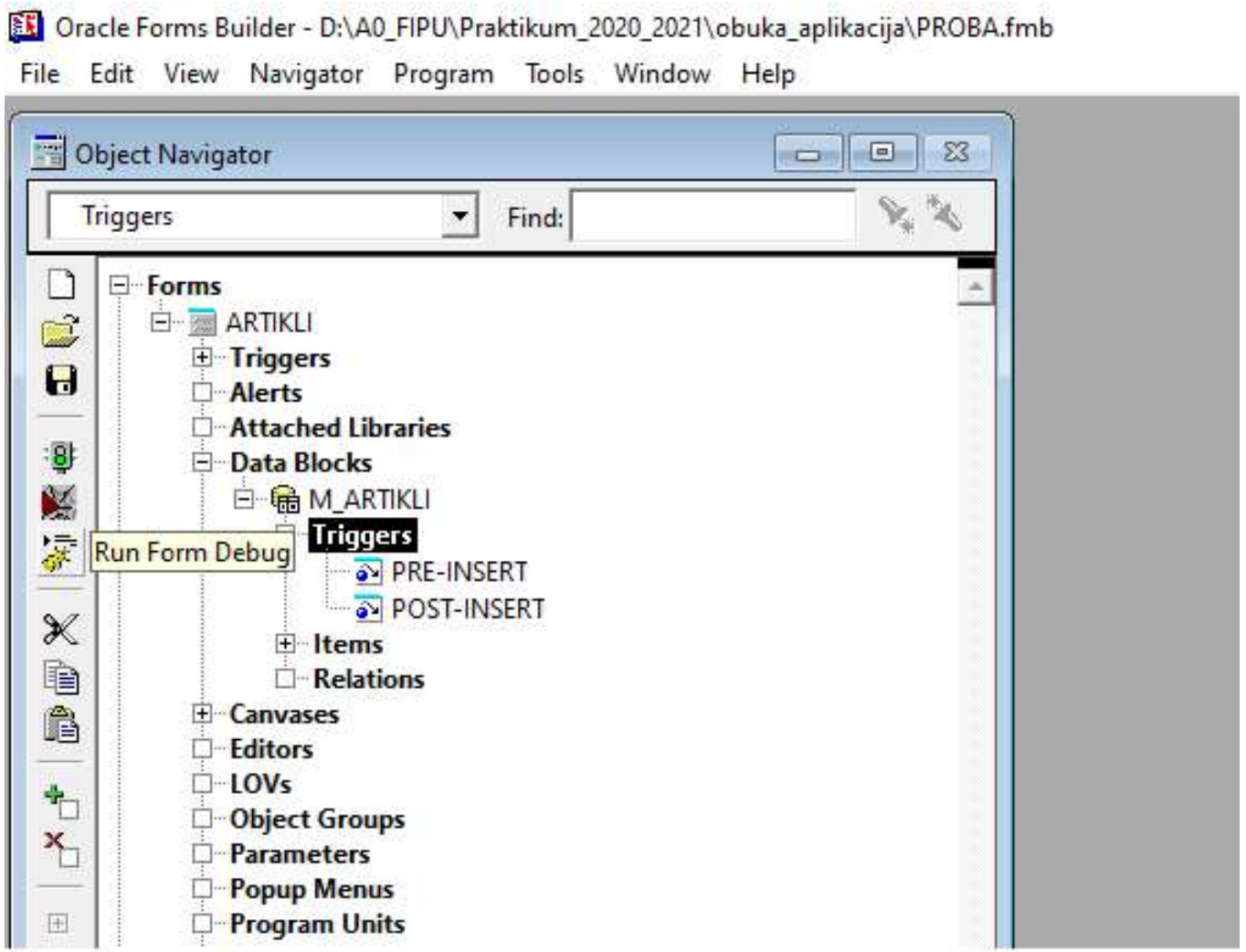
Forms 6i debugging

- Forms modul ima i POST-INSERT trigger,
  koji služi za sinkronizaciju Forms modula s bazom
  (jer baza generira ID kroz BIR trigger i funkciju sekvenca):

```
/*
sinkronizacija - id se generira na bazi
*/
SELECT id INTO :m_artikli.id
  FROM m_artikli
 WHERE :m_artikli.ROWID = ROWID;
```

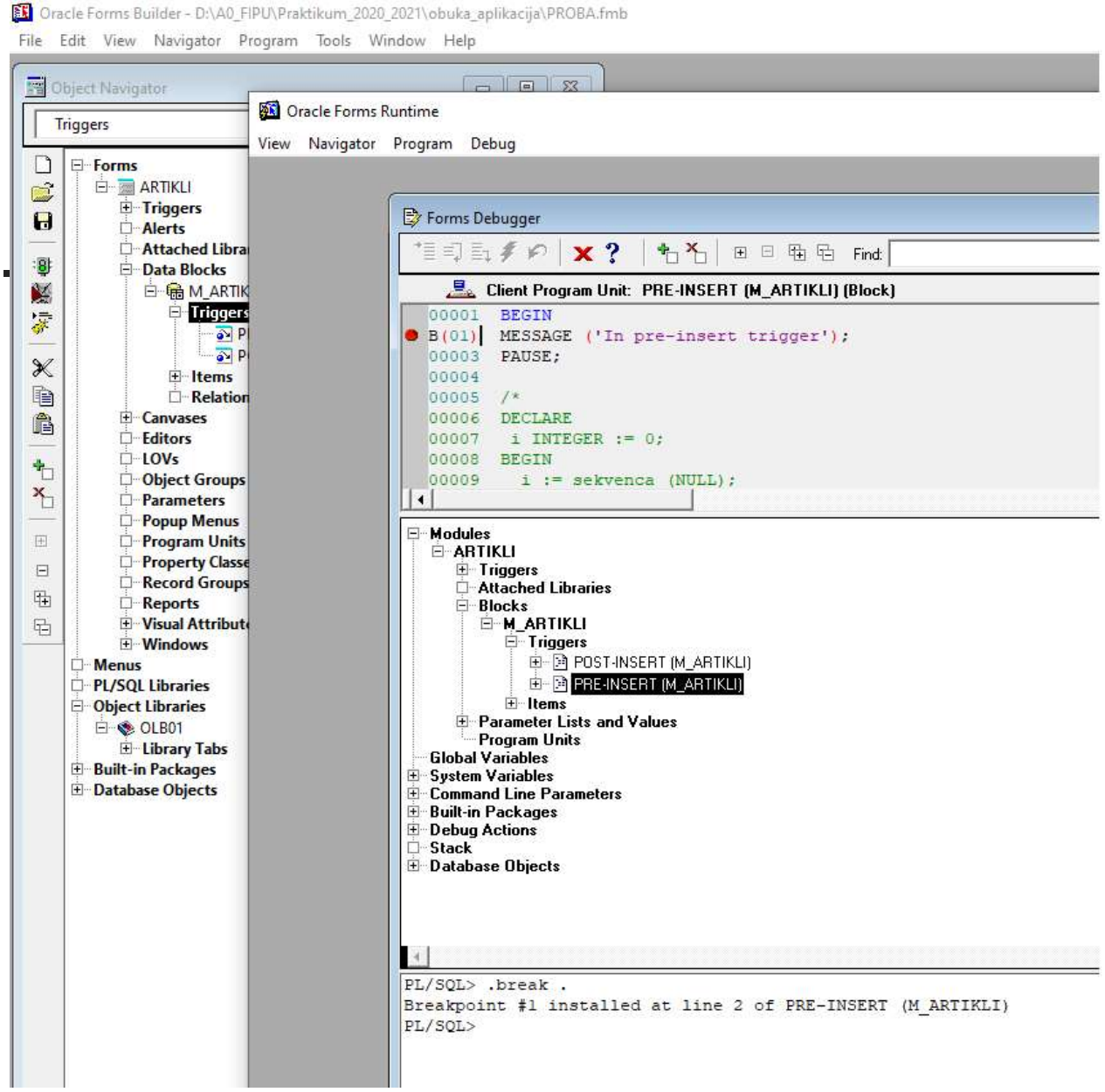
Forms 6i debugging

- U Forms Builderu na određenom Forms modulu pokrenemo ikonu **Run Form Debug** (ikona prikazuje bubu).



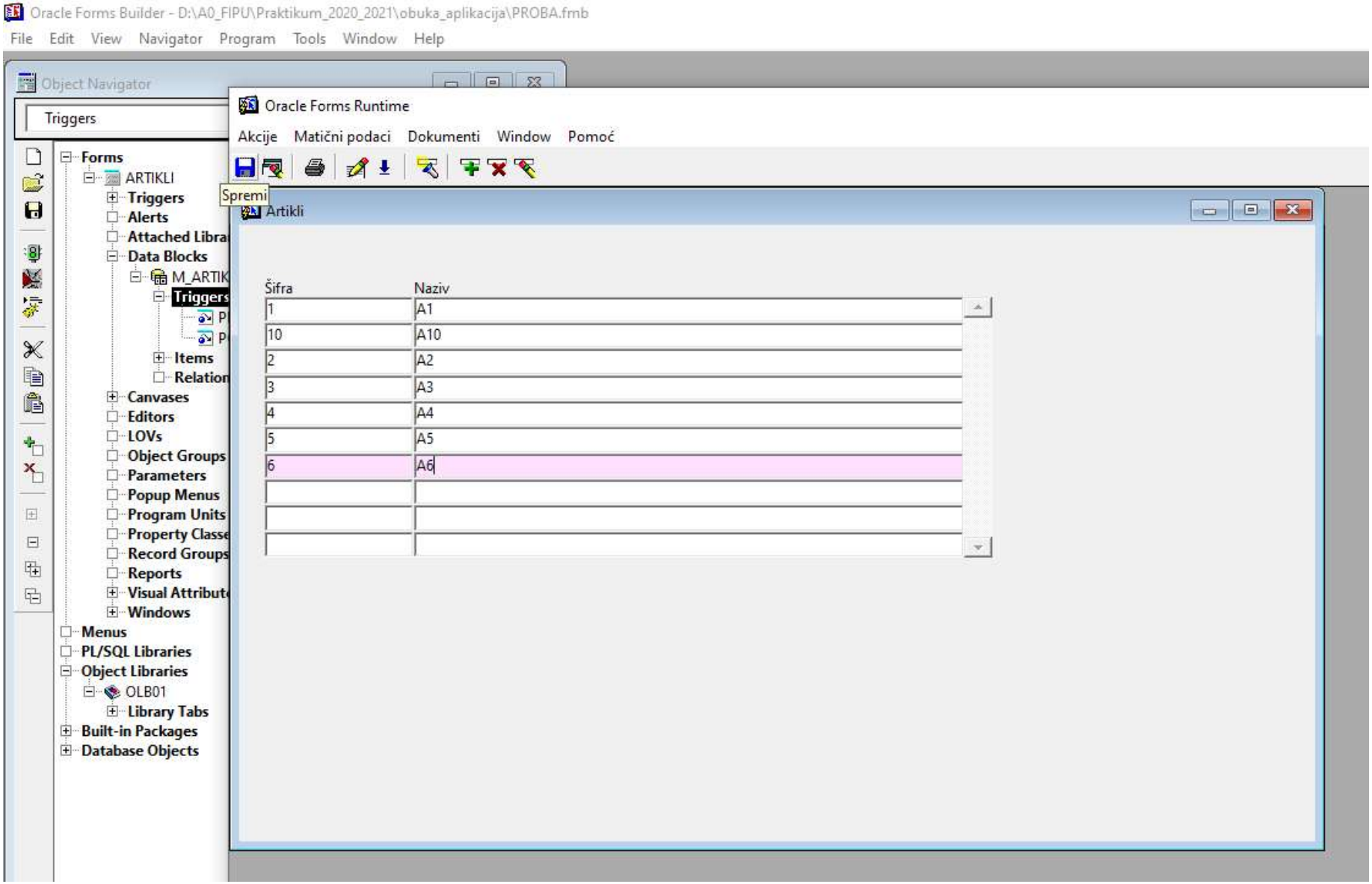
Forms 6i debugging

- U Forms Runtime, u Forms Debugger prozoru, odaberemo naredbu, kreiramo Break Point, pa sa Close (crveni X) startamo Forms modul.



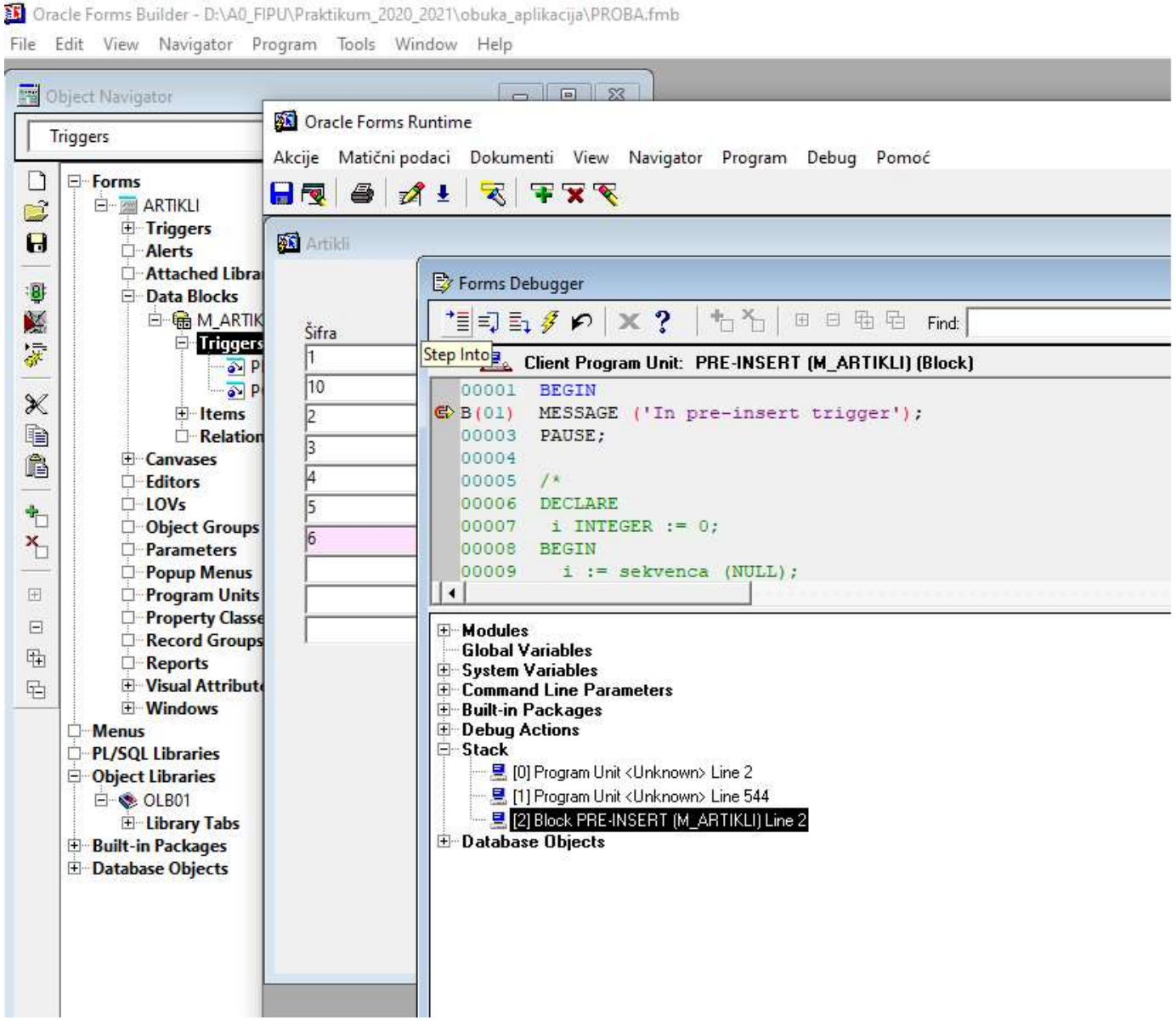
Forms 6i debugging

- U Forms Runtime, unesemo novi (npr. šesti) redak i damo Spremi (disketa), čime pokrećemo Forms COMMIT proces.



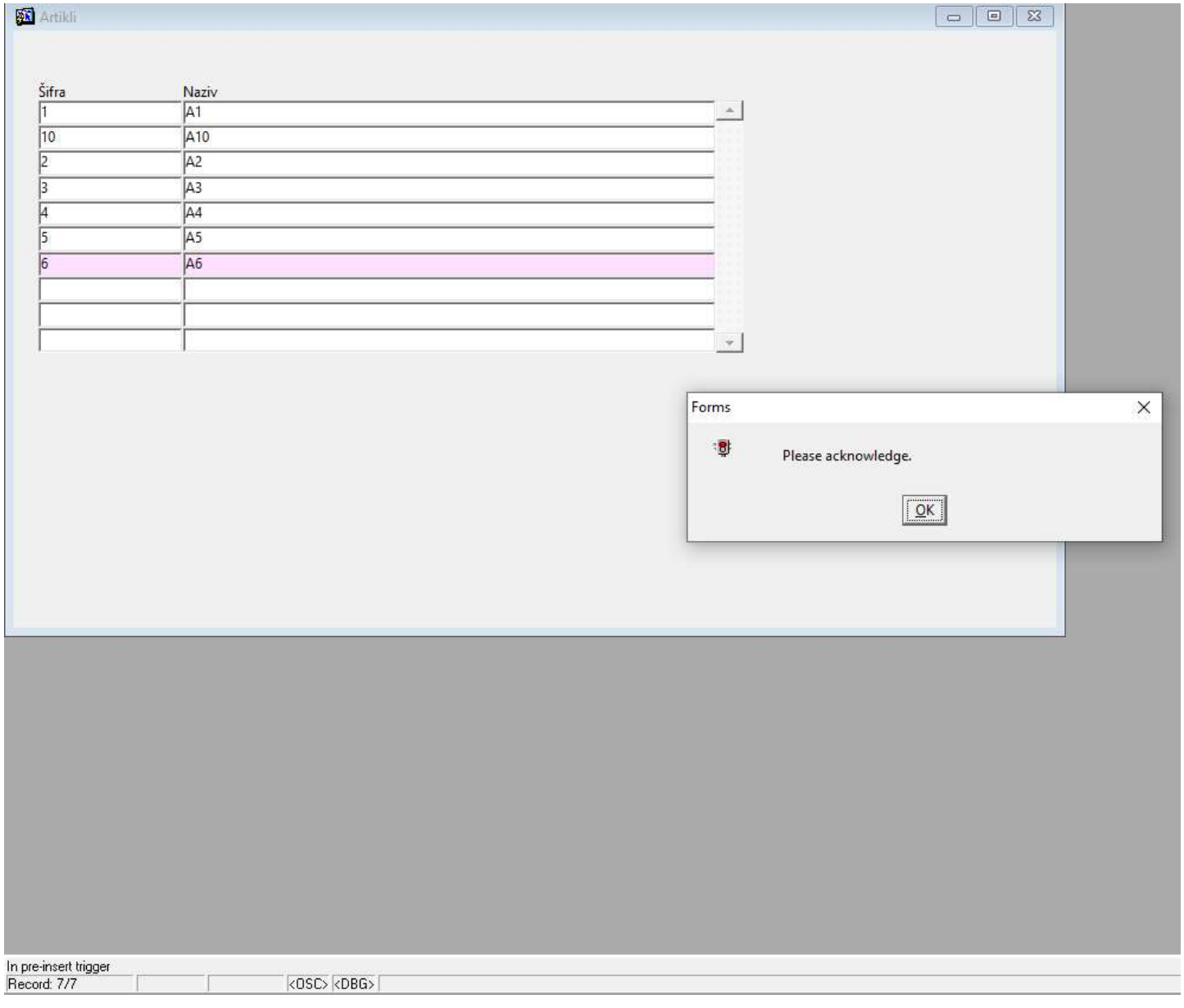
Forms 6i debugging

- Kad modul dođe do Break Pointa (u PRE-INSERT triggeru), prikaže se Forms Debugger - stane na naredbi MESSAGE.



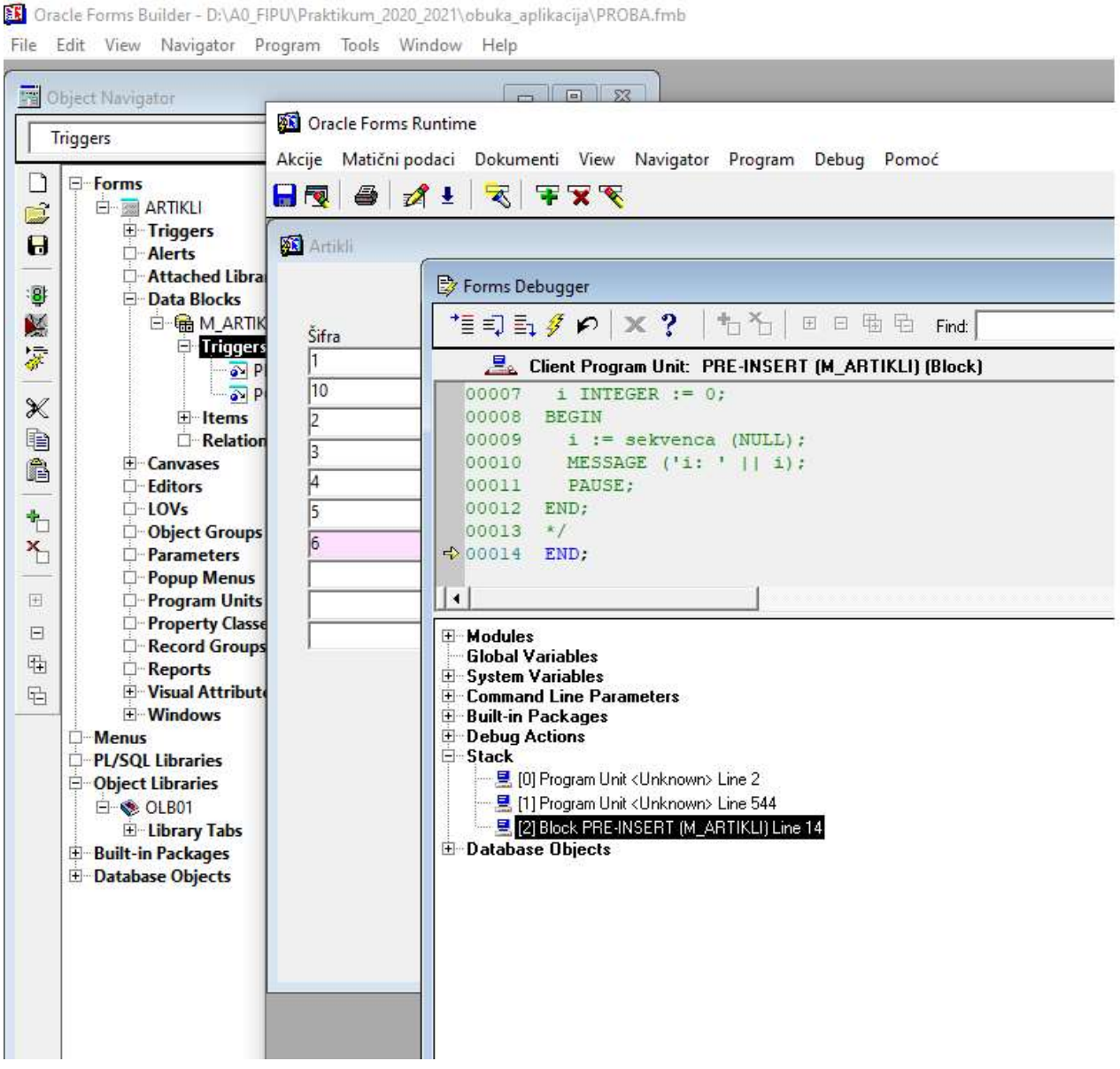
Forms 6i debugging

- Kad damo Step Into na sljedećoj naredbi (PAUSE), ponovno se prikaže Forms runtime – prikazuje se poruka.



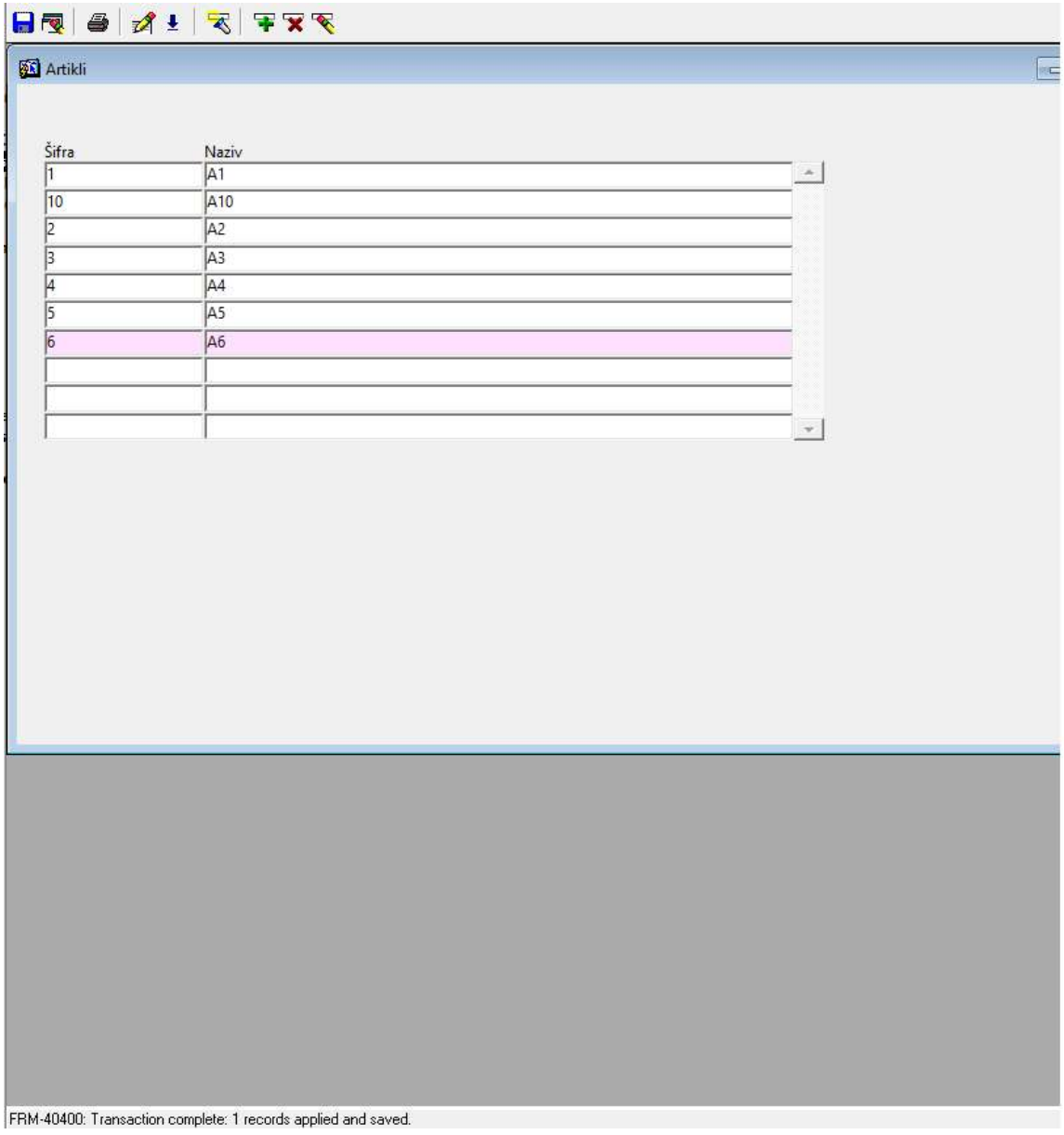
Forms 6i debugging

- Nakon toga se vrati u Forms Debugger, na sljedeću naredbu (END, jer komentar preskače).



Forms 6i debugging

- Forms Debugger nestane i Forms runtime prikaže da je jedan redak upisan u bazu - završen je Forms COMMIT proces.



Forms 6i debugging

- Dakle, vidjeli smo ono što smo već prije rekli:

- Break Point vrijedi samo za Forms trigger u kojem je postavljen (u ovom slučaju PRE-INSERT), a ne za drugi Forms trigger (POST-INSERT).
- Ako bismo htjeli prolaziti kroz naredbe drugog Forms triggera, trebali bismo i u njemu postaviti Break Point.

- Forms Debugger ne prolazi kroz trigger baze, niti procedure / funkcije koje se iz njega pozivaju, a nemaju Break Point, iako su oni kompajlirani sa ALTER … COMPILE **DEBUG.**
- **No, Forms Debugger može prolaziti kroz programski kod procedura / funkcija na bazi koje imaju Break Point, ili one koje nemaju Break Point, ali se direktno pozivaju iz Forms triggera koji ima Break Point.**

Forms 6i debugging

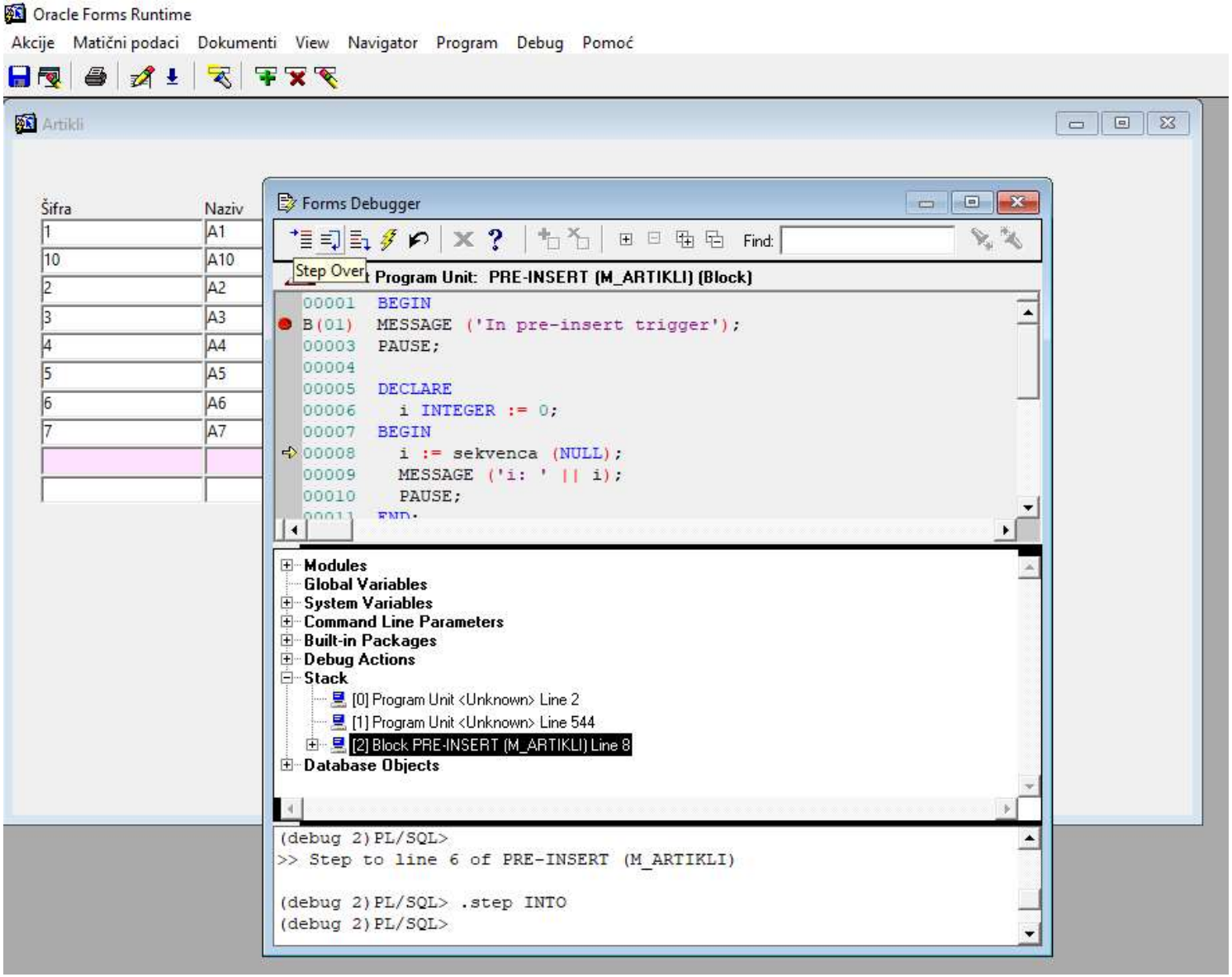
- Skinimo komentar na DECLARE – END bloku u PRE-INSERT triggeru:

```
MESSAGE ('In pre-insert trigger');
PAUSE;

DECLARE
   i INTEGER := 0;
BEGIN
   i := sekvenca (NULL);
   MESSAGE ('i: ' || i);
   PAUSE;
END;
```

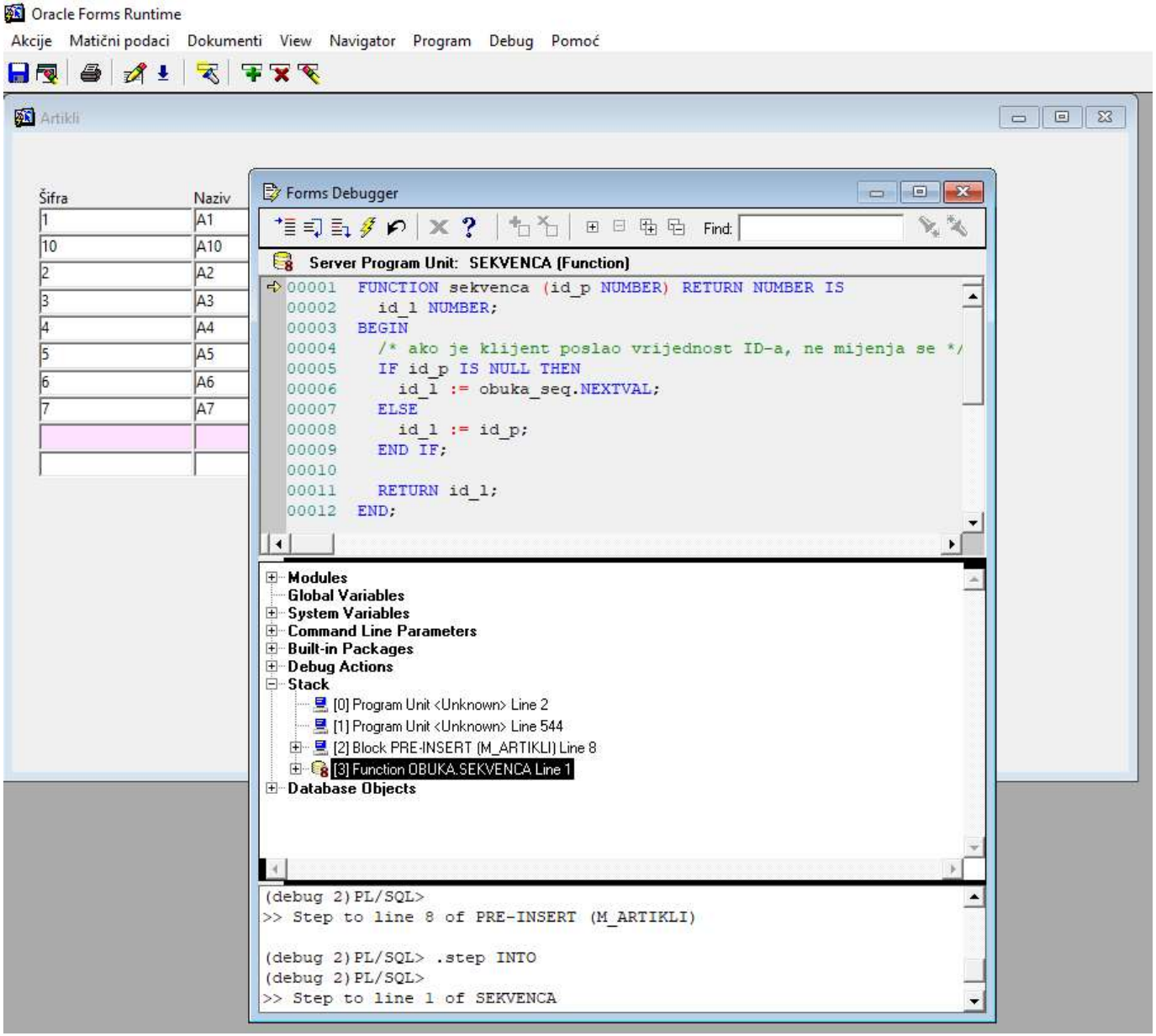
Forms 6i debugging

- Postavimo Break Point kao prije, i u Forms Debuggeru dođimo do naredbe koja poziva funkciju s baze:



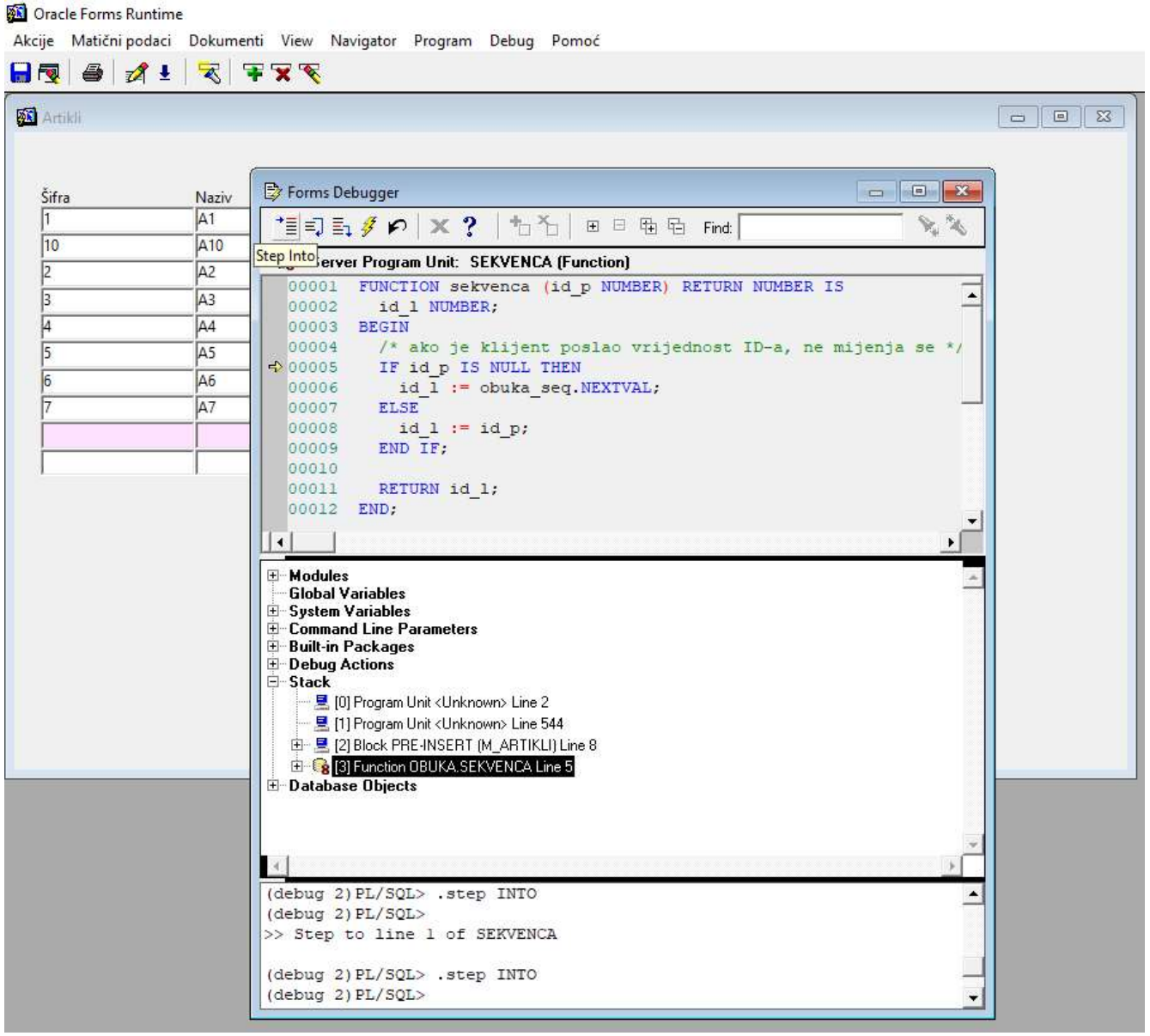
Forms 6i debugging

- Vidimo da se Forms Debugger postavio na početak funkcije:



Forms 6i debugging

- Kroz funkciju na bazi možemo ići isto kao kroz programski kod u Forms modulu (Step Into itd.):



Forms 12c debugging

- MOS dokument ID2480942.1: How To Debug Form When Start Forms 12c With FSAL Mode Or WebStart?

Please follow these steps to **Debug remotely running forms** using Forms debugger:

1. Open Forms Builder

2. Open a Form which you plan to debug. Add a breakpoint, compile and transfer to server from where its being executed.

3. Start the Forms with FSAL mode or Webstart mode.

For example:

java -jar frmsal.jar -url "http://hostname:port/forms/frmservlet?config=standaloneapp&form=mytest.fmx&**debug=yes**&userid=<username>/<password>@<tnsalias>"

Forms 12c debugging

4. Once executed, it will prompt HOST and PORT
(it has choices for the Debugger) then click button "OK"

5. At Form builder, choose **Debug -> "Attach Debug"**

6. It will prompt to enter Host and Port.
Enter the values as Prompted by step 4 then click button "OK”

7. Run the form and follow the break points, Debug Window,
Debug console to watch the values for the forms variables.

Forms 12c debugging

- Napomena – u drugom koraku piše:
  2. Open a Form which you plan to debug.
  Add a breakpoint, **compile and transfer to server**
  from where its being executed.

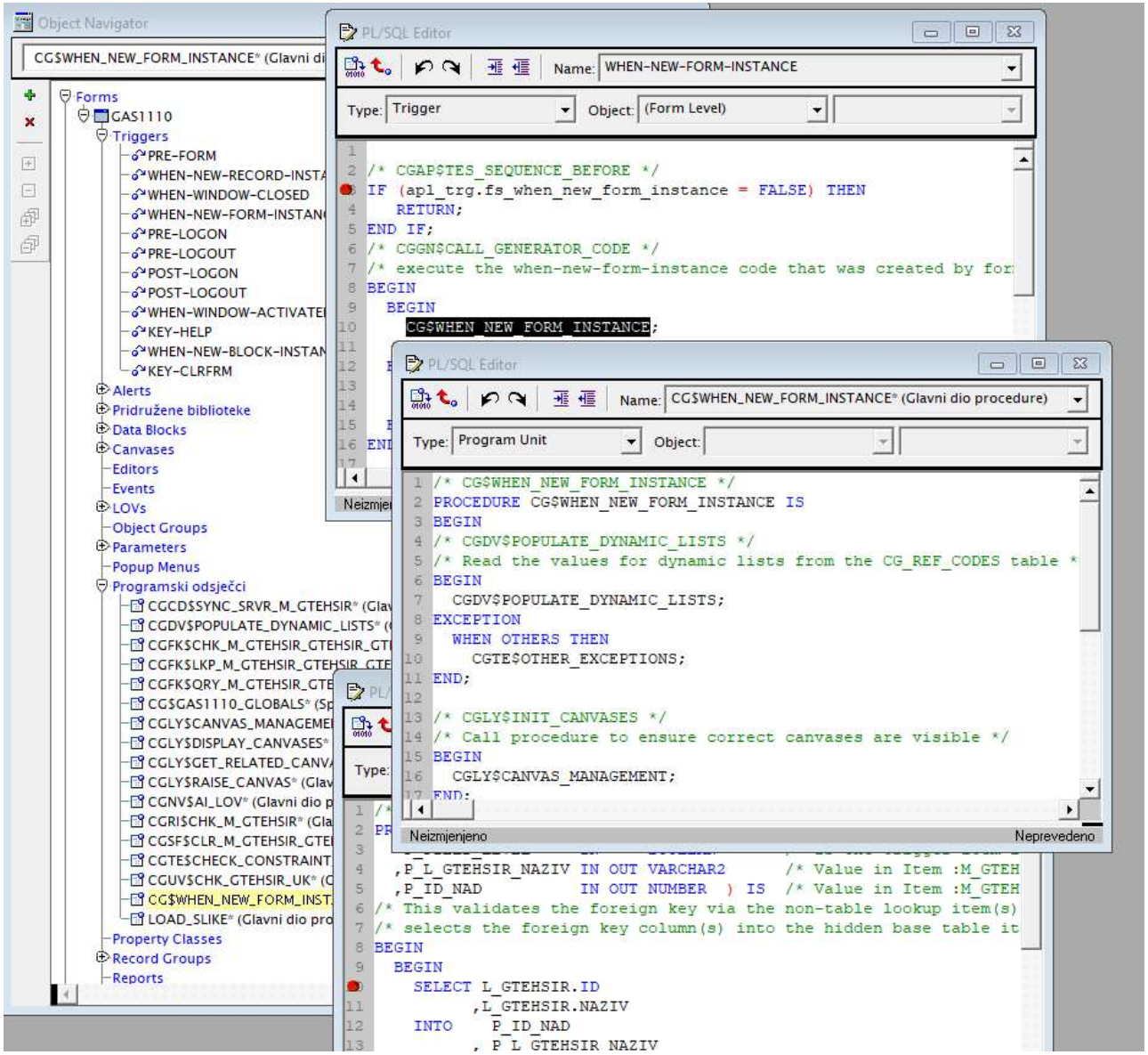
- Međutim, zaključili smo da to nije nužno!

- Ako je modul koji otvorimo u Forms Builderu
  jednak onome koji se izvodi na serveru (naravno,
  na serveru je fmx, a u Forms Builderu otvaramo fmb),
  ne treba transferirati modul na server.

- Dodatna napomena: firewall treba podesiti tako da
  dopusti Forms debugging TCP portove na AS-u.

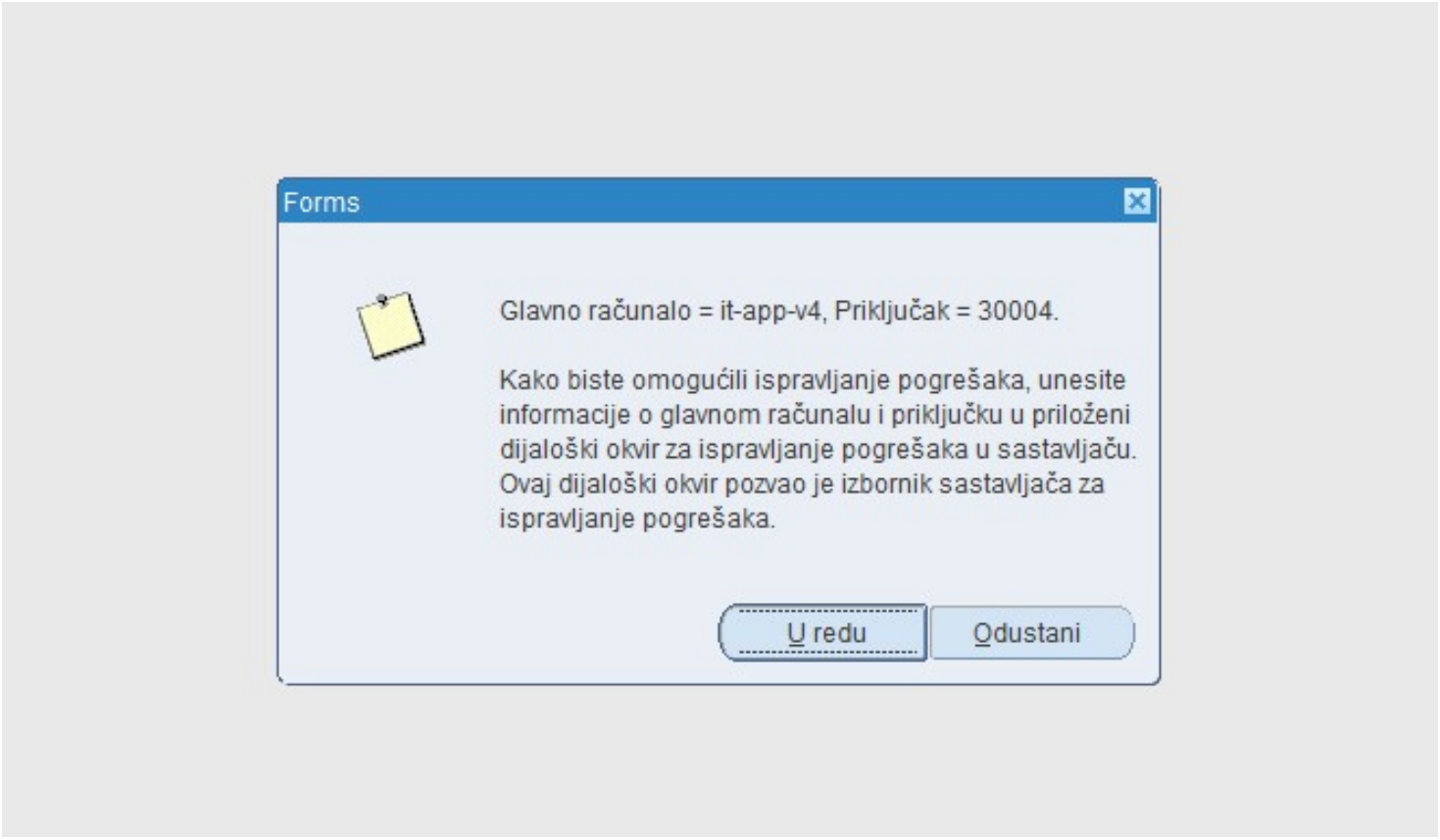
Forms 12c debugging

- 1. Open Forms Builder
- 2. Open a Form which you plan to debug. Add a breakpoint …



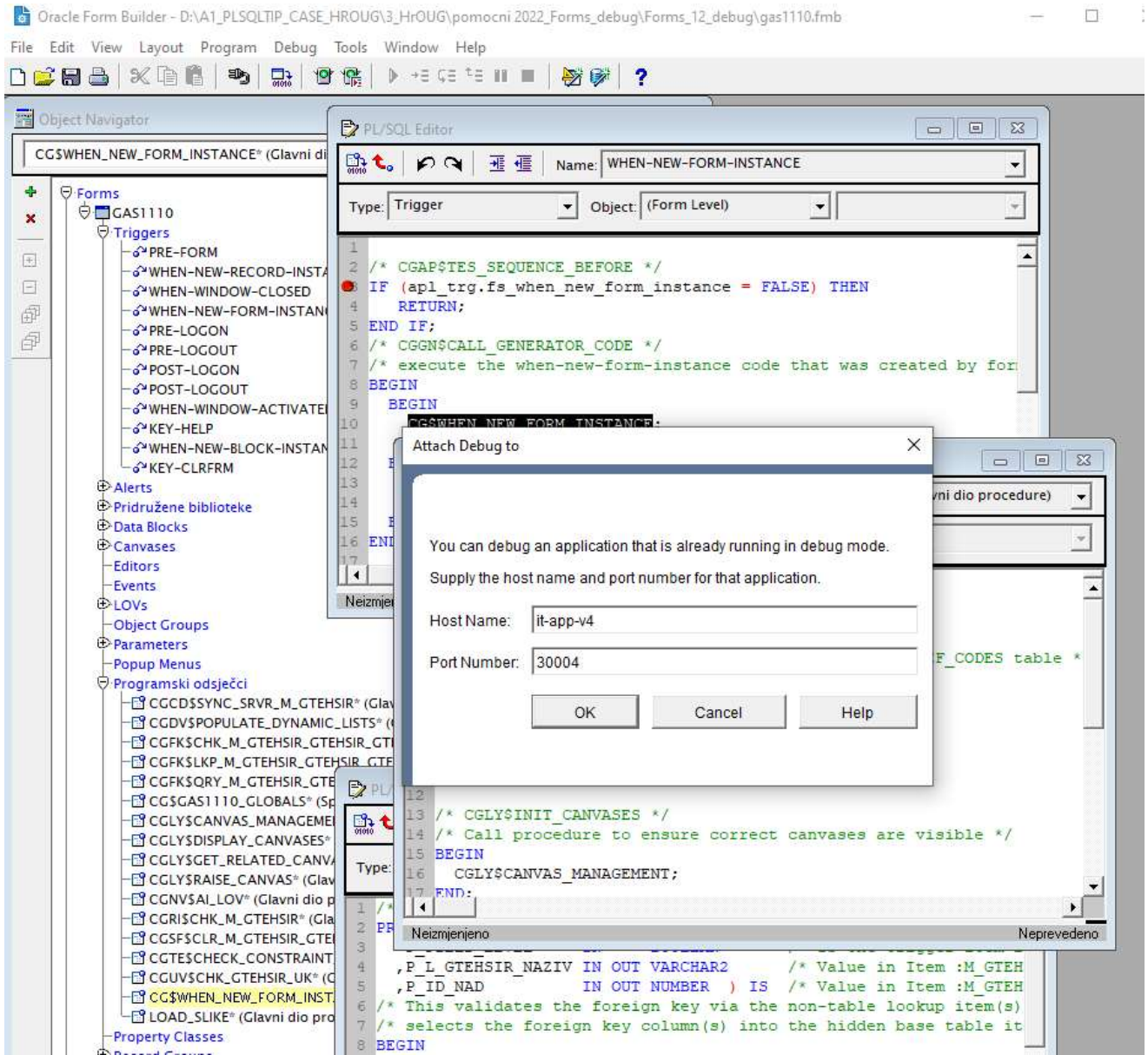
Forms 12c debugging

- 3. Start the Forms with FSAL mode or Webstart mode…
- 4. Once executed, it will prompt HOST and PORT…
  "Odustani" pokreće aplikaciju u normalnom radu.
  "U redu" znači da želimo debuggirati (zamrzne se i čeka …).



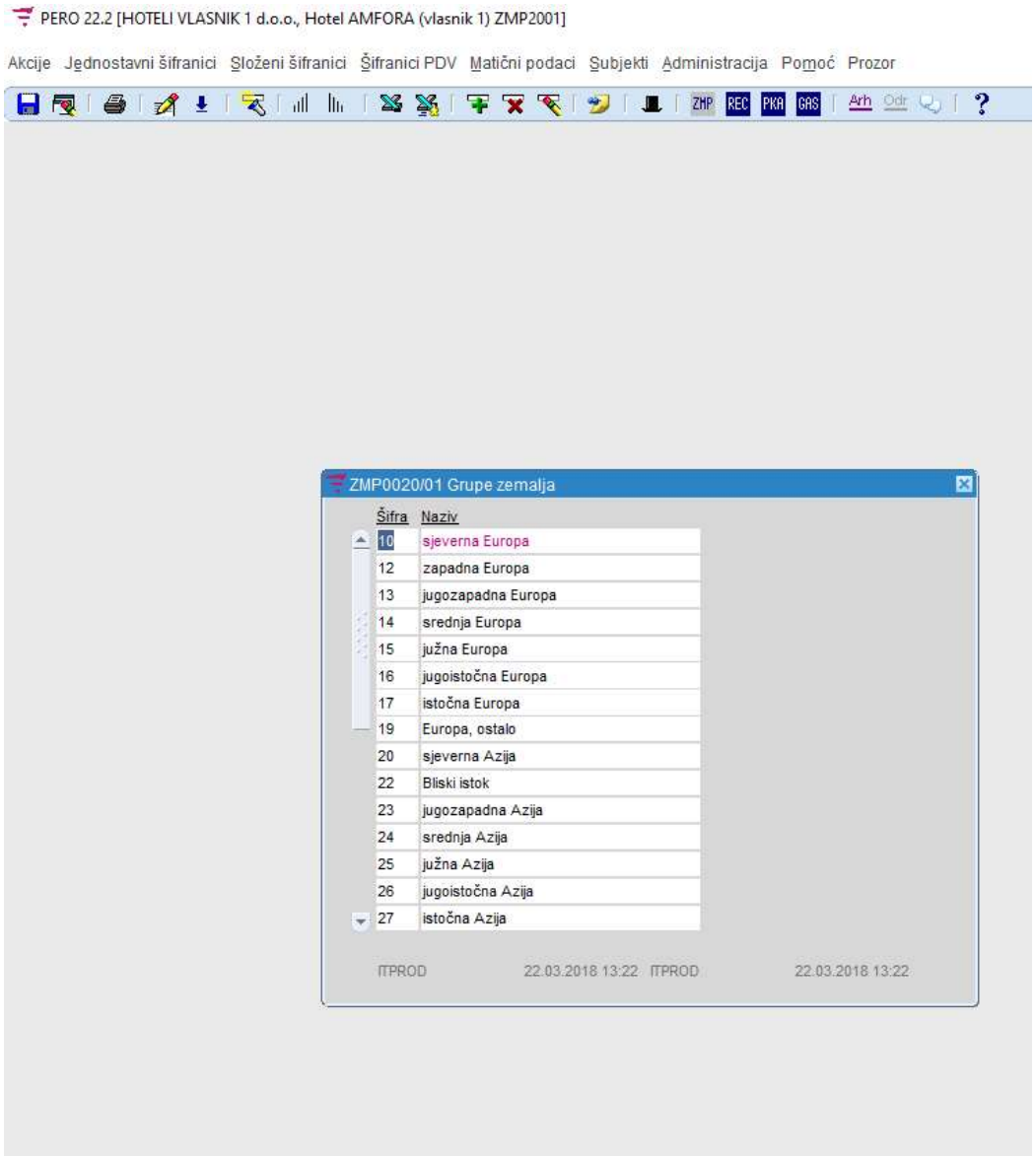
Forms 12c debugging

- 5. At Form builder, choose Debug -> "Attach Debug"
- 6. It will prompt to enter Host and Port … click button "OK"



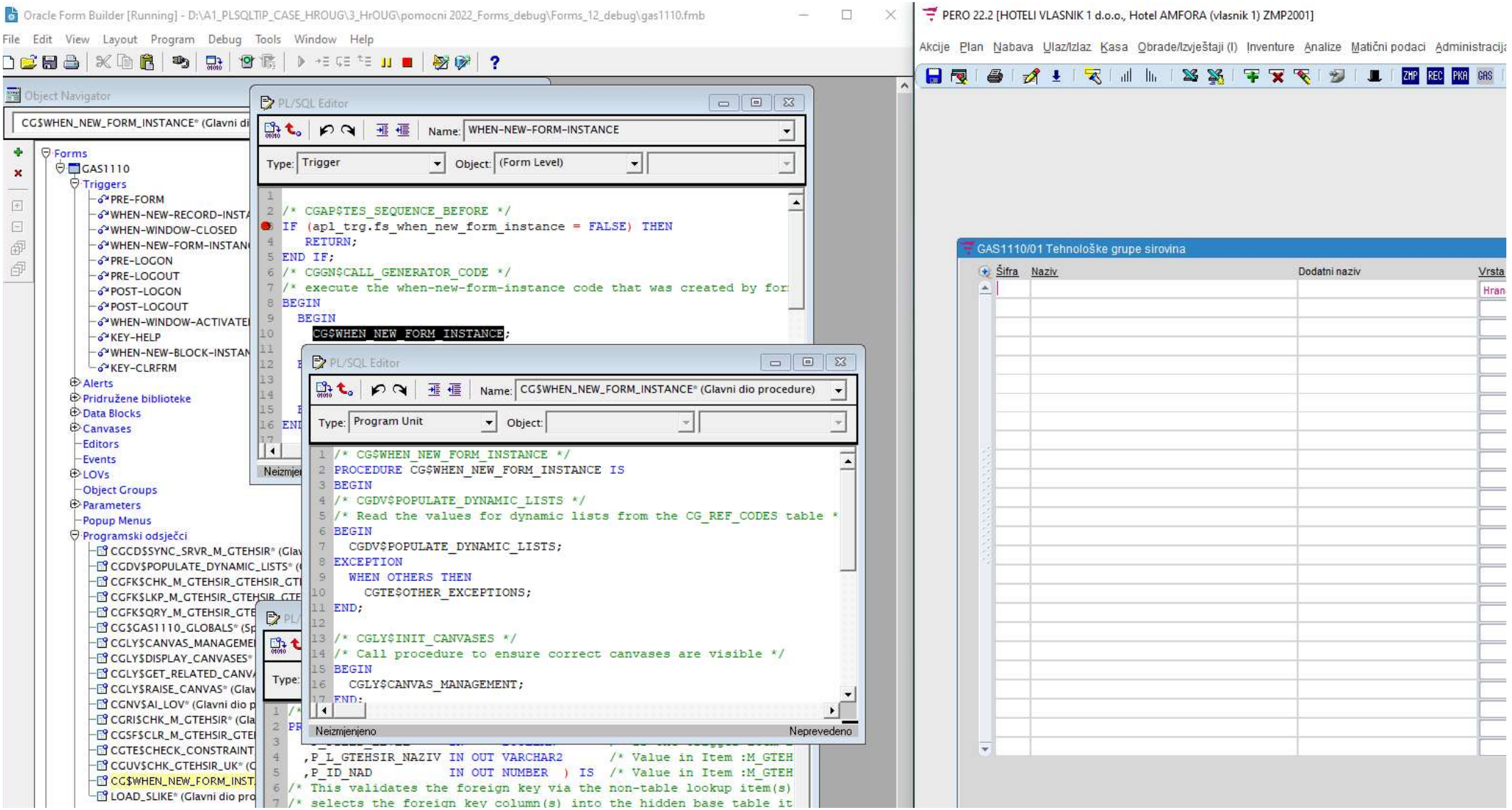
Forms 12c debugging

- 7. Run the form and follow the break points …
- Možemo normalno raditi s ostalim programima aplikacije.



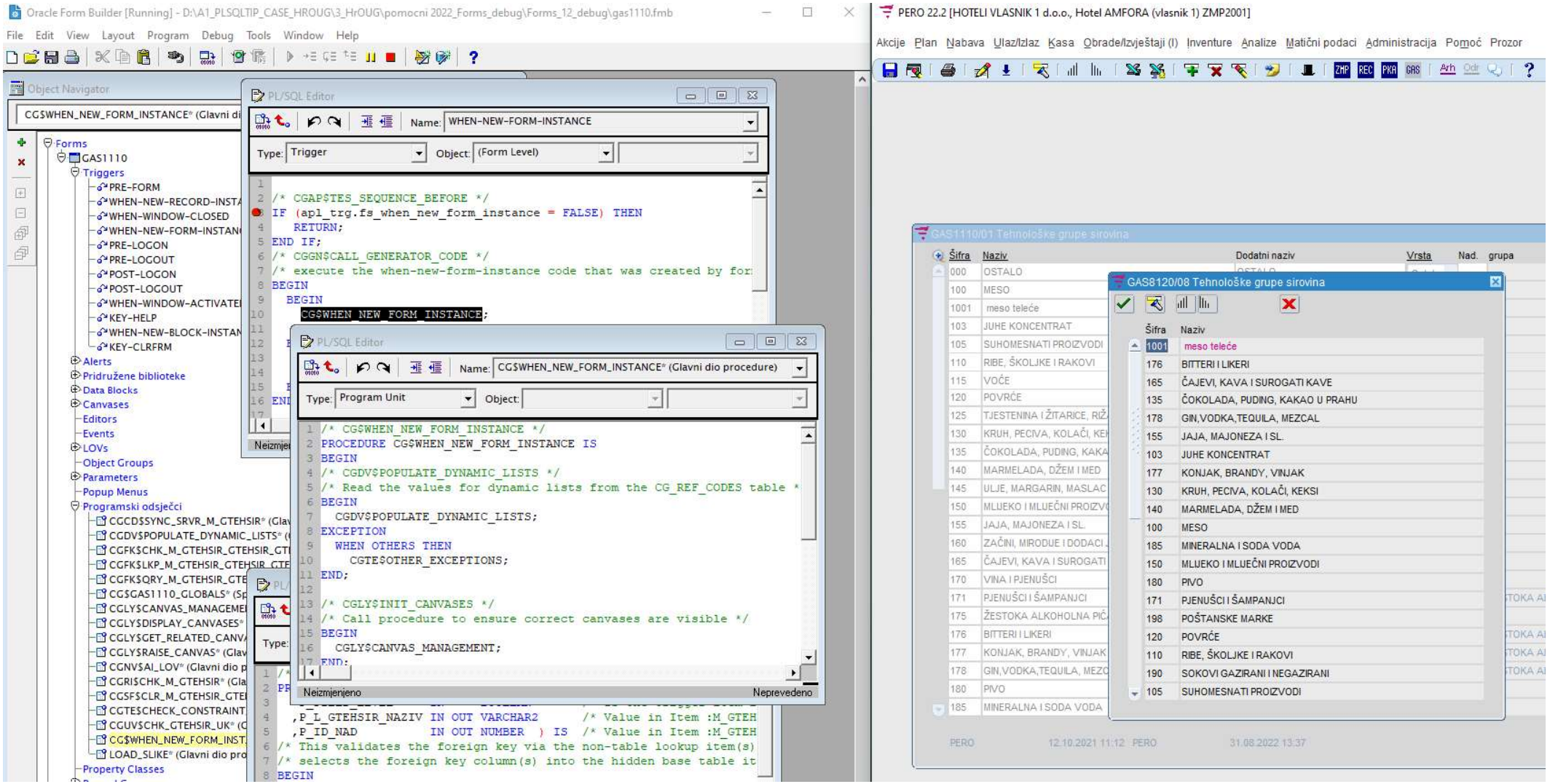
Forms 12c debugging

- Pokrećemo program otvoren u Forms Builderu. **Vidimo da se ne izvodi debugging, iako trigger WNFI ima Break Point.**



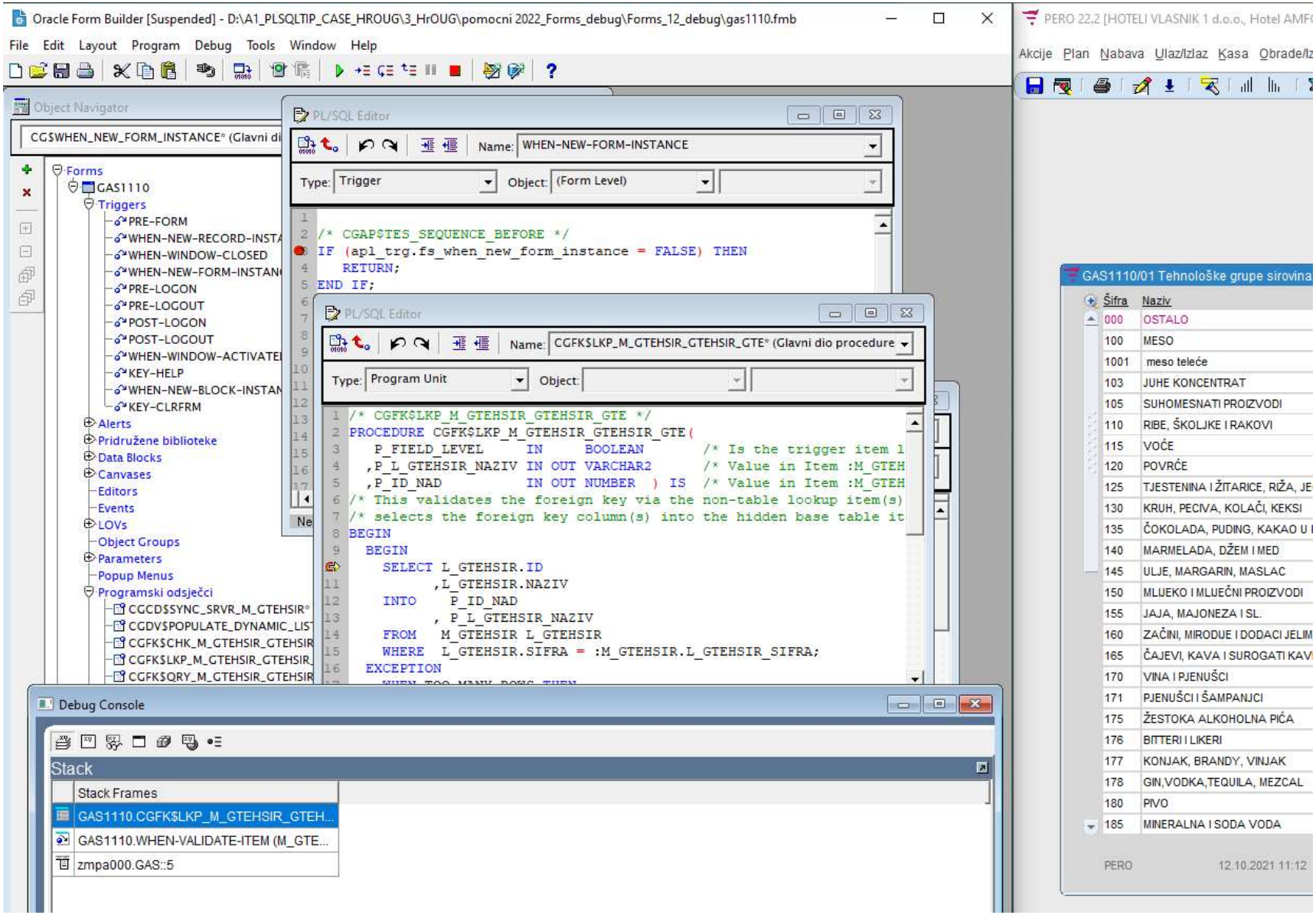
Forms 12c debugging

- Dolazimo u aplikaciji do radnje prije poziva Forms procedure u koju smo dodali Break Point.



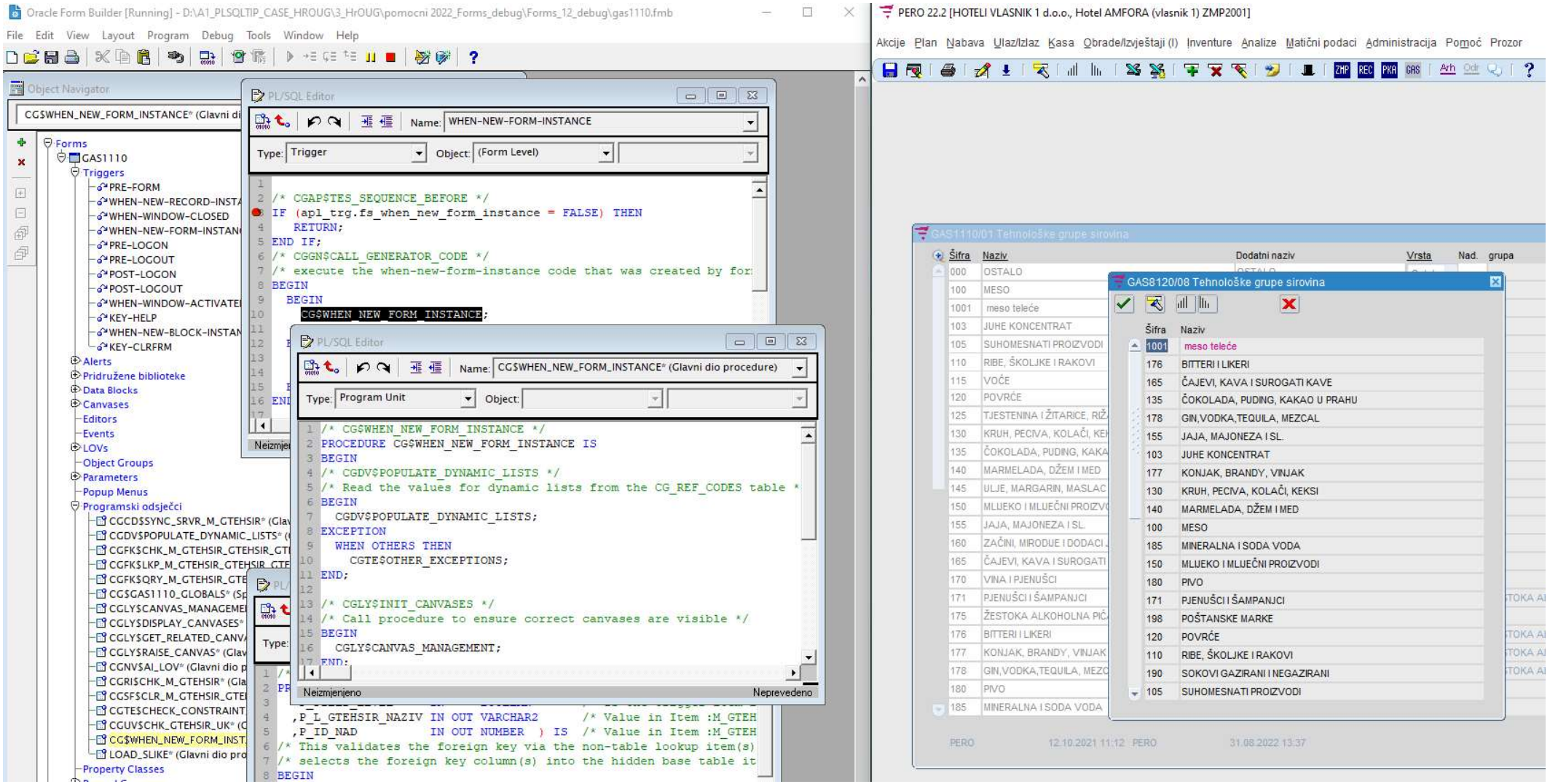
Forms 12c debugging

- Sada počinje debugging proces.



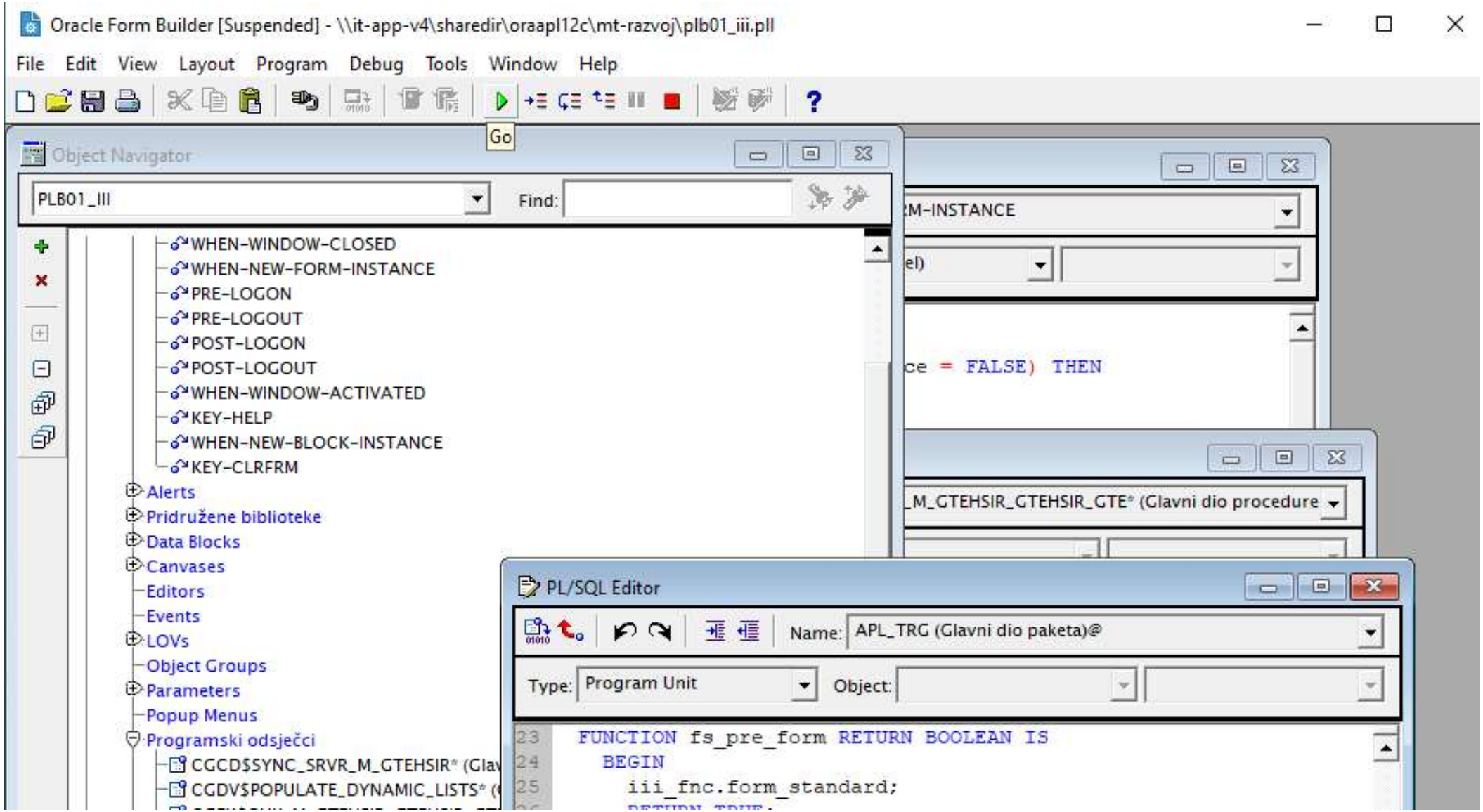
Forms 12c debugging

- Prikaže se čak i kod iz PL/SQL lib-a, koji nismo eksplicitno otvorili u Forms Builderu (ali je na registry FORMS_PATH).



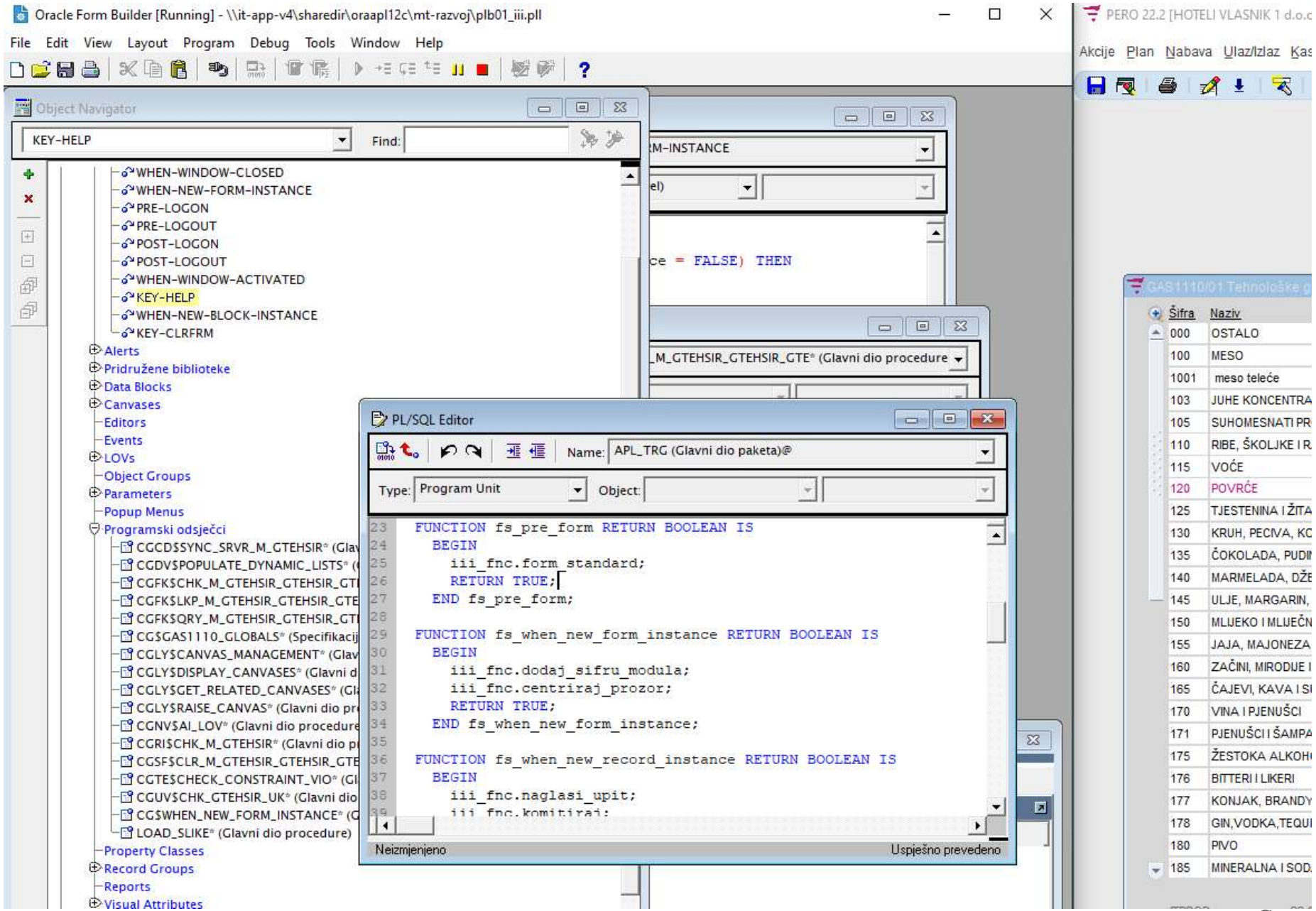
Forms 12c debugging

- Tipkom Go prekidamo debuggiranje.



Forms 12c debugging

- Aplikacija nastavlja s radom, do sljedećeg Break Pointa.



Forms 12c debugging

- Kako smo vidjeli, za razliku od Forms 6i debuggera, **Forms 12c debugger ne prikazuje prolaz kroz Forms triggere**, a stavili smo Break Point u Forms trigger WHEN-NEW-FORM-INSTANCE
  (u Forms 12c, Break Point se postavlja direktno u editoru).

- **Forms Builder prikazuje zbivanja u proceduri** CGFK$LKP_M_GTEHSIR_GTEHSIR_GTE,
  gdje smo stavili Break Point.

- Nakon što se pokrene debugger (tj. dođe na Break Point u Forms proceduri / funkciji), i dalje ne prikazuje zbivanja u Forms triggerima, ali ide korak po korak kroz njih,
  pa možemo u Forms Builderu gledati programski kod triggera **i pretpostaviti koja se naredba trenutačno izvršava.**

Forms 12c debugging

- Kao i Forms Forms 6i debugger, niti Forms 12c Debugger ne prikazuje prolaz kroz triggere na bazi, niti kroz procedure / funkcije koje se pozivaju iz triggera baze, a nemaju Break Point.

- Za razliku od Forms 6i debuggera, **Forms 12c debugger ne prikazuje prolaz niti kroz procedure / funkcije na bazi koje imaju Break Point, niti one koje Forms direktno poziva.**

- Srećom, za razliku od vremena kada smo imali samo Forms 6i, već odavno imamo SQL Developer, koji može raditi debugging koda na bazi.

SQL Developer debugging

- SQL Developer omogućava (i) remote debugging.

- **Ako je Oracle baza 12.2 ili novija, remote debugging se može raditi bez mijenjanja koda na bazi**
  (jedino taj način je detaljnije opisan u nastavku).

- Koristi se naredba
  DBMS_DEBUG_JDWP.CONNECT_TCP s 4 parametra:
  IP adresa, TCP port, SID, SERIAL# (sesije na bazi).

- Napomena: IP adresa i TCP port su
  IP adresa i TCP port lokalnog računala
  (onog na kojem se izvodi SQL Developer debugging),
  a ne aplikacijskog servera (kao kod Forms remote debugging)
  ili baze (što bi se možda očekivalo).

SQL Developer debugging

- Ako je baza starija od 12.2, onda treba koristiti **DBMS_DEBUG_JDWP.CONNECT_TCP s 2 parametra:** IP adresa, TCP port, tj. bez navođenja informacija o sesiji baze (SID i SERIAL#).

- Ali, tu naredbu onda treba staviti u proceduru / funkciju na bazi na kojoj se radi debugging, ili (ako želimo izvršiti samo proceduru na bazi, bez prethodnog Forms koda), možemo napisati neimenovani PL/SQL blok koji prvo ima tu naredbu, pa onda poziv procedure na bazi.

SQL Developer debugging

- **1. Ne možemo paralelno koristiti Forms Builder remote debugging i SQL Developer remote debugging.**
  Naime, kad u Forms Builderu uključimo debugger i pokrenemo aplikaciju s uključenim debuggingom (bez daljnjeg rada, samo uđemo u aplikaciju), u SQL Developeru više ne prolazi exec DBMS_DEBUG_JDWP.CONNECT_TCP … - javlja se **"session is already connected to a debugger"**.

  To znači da prvo trebamo analizirati Forms kod (kod na bazi se samo izvede), a onda po potrebi analizirati samo kod na bazi (Forms kod se samo izvede). Forms 6i može oboje (jedino se ne može raditi debugging triggera na bazi).

- 2. SQL Developer remote debugging ponekad ne radi (na nekim računalima) **dok se (privremeno) ne isključi Windows Defender firewall.**

SQL Developer debugging

 Postupak (s bazom 12.2. ili većom):

1. Firewall na bazi ne želimo iskopčati. U Tools -> Preferences za Debugger, pod Use DBMS_DEBUG_JDWP treba postaviti Debugging Port Range npr. od 4000 do 4999.

2. Kompajliranje procedure / funkcije sa
ALTER PROCEDURE … COMPILE DEBUG;

3. U Sql Developeru pokrenuti remote debugging:
- desni klik na konekciji baze, pa Debug -> Remote Debug
- unijeti IP adresu lokalnog računala, zapamtiti port i dati OK.

4. U Sql Developeru postaviti Break Point u proceduri.

SQL Developer debugging

5. Izvršiti npr. za Forms sesije (prije i poslije pokretanja određenog Forms modula - da lakše nađemo željenu sesiju):

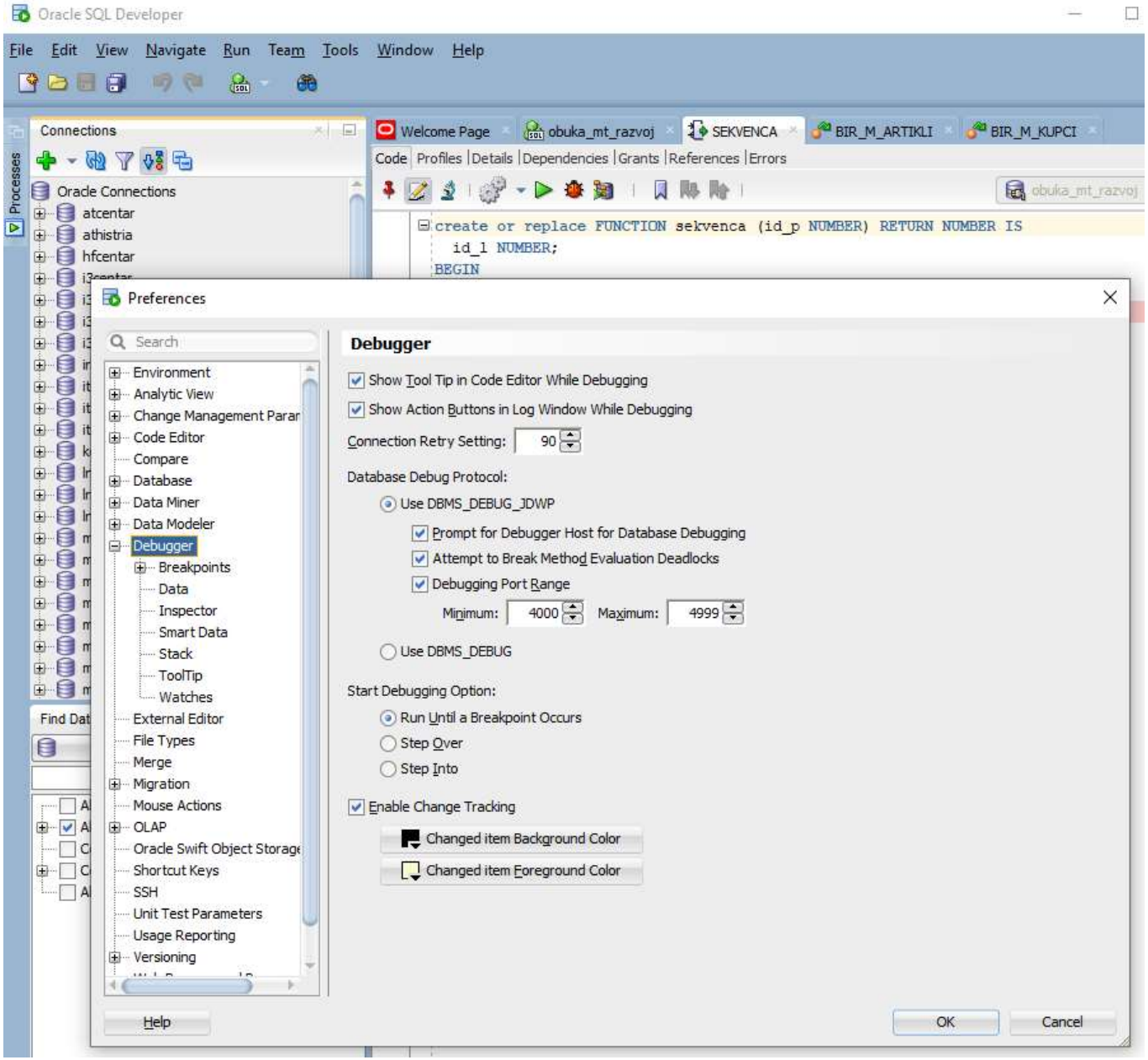
```
select *
  from v$plsql_debuggable_sessions
 where username = 'PERO'
   and program like 'frmweb%'
 order by process, sid;
```

6. Izvršiti DBMS_DEBUG_JDWP.CONNECT_TCP, kojem proslijedimo IP adresu, TCP port, SID, SERIAL#.

7. Kad klijent pokrene proceduru i ona dođe do Break Pointa, kontrolu preuzima remote debugging proces.

SQL Developer debugging

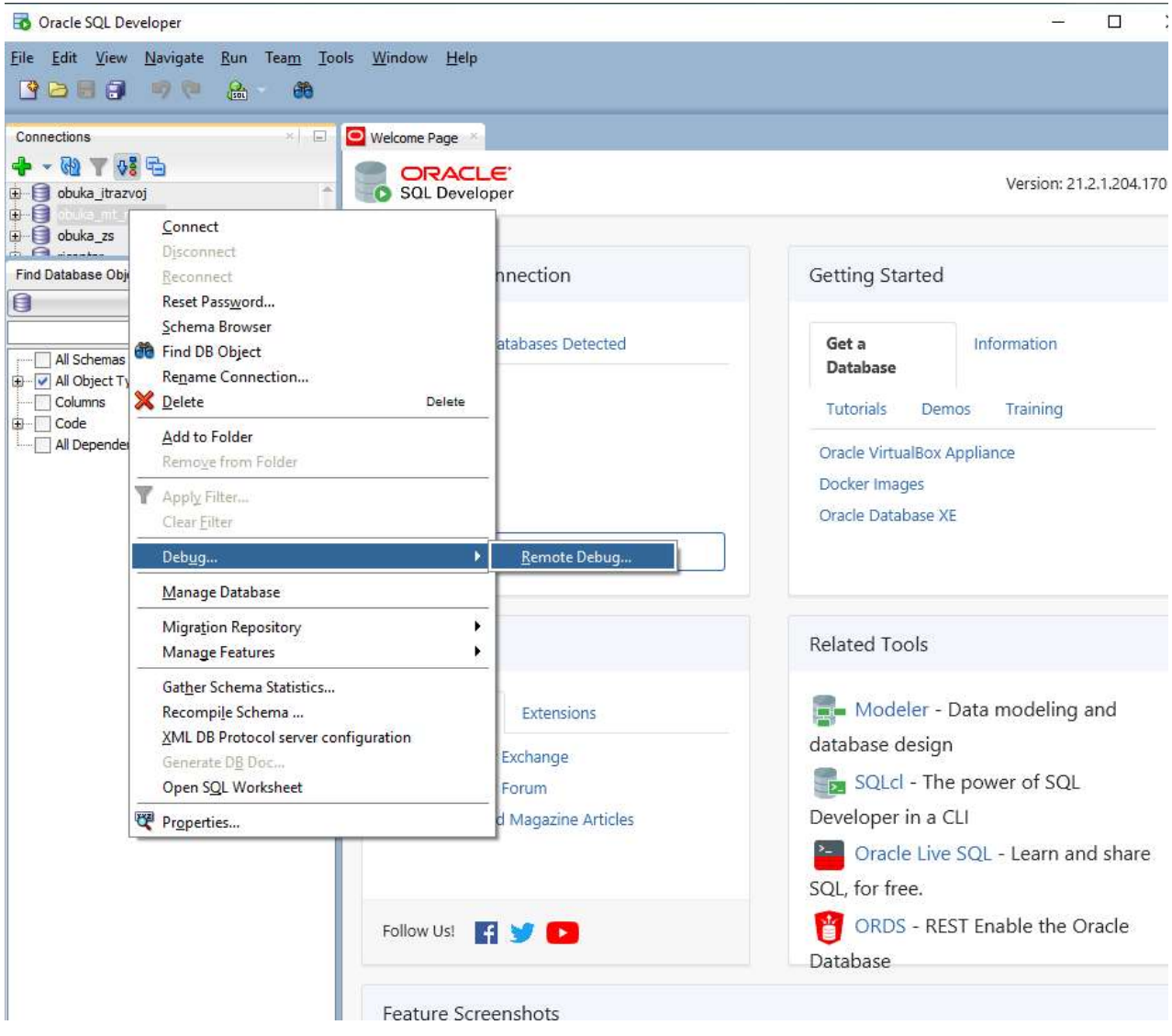
- 1. … postaviti Debugging Port Range



SQL Developer debugging

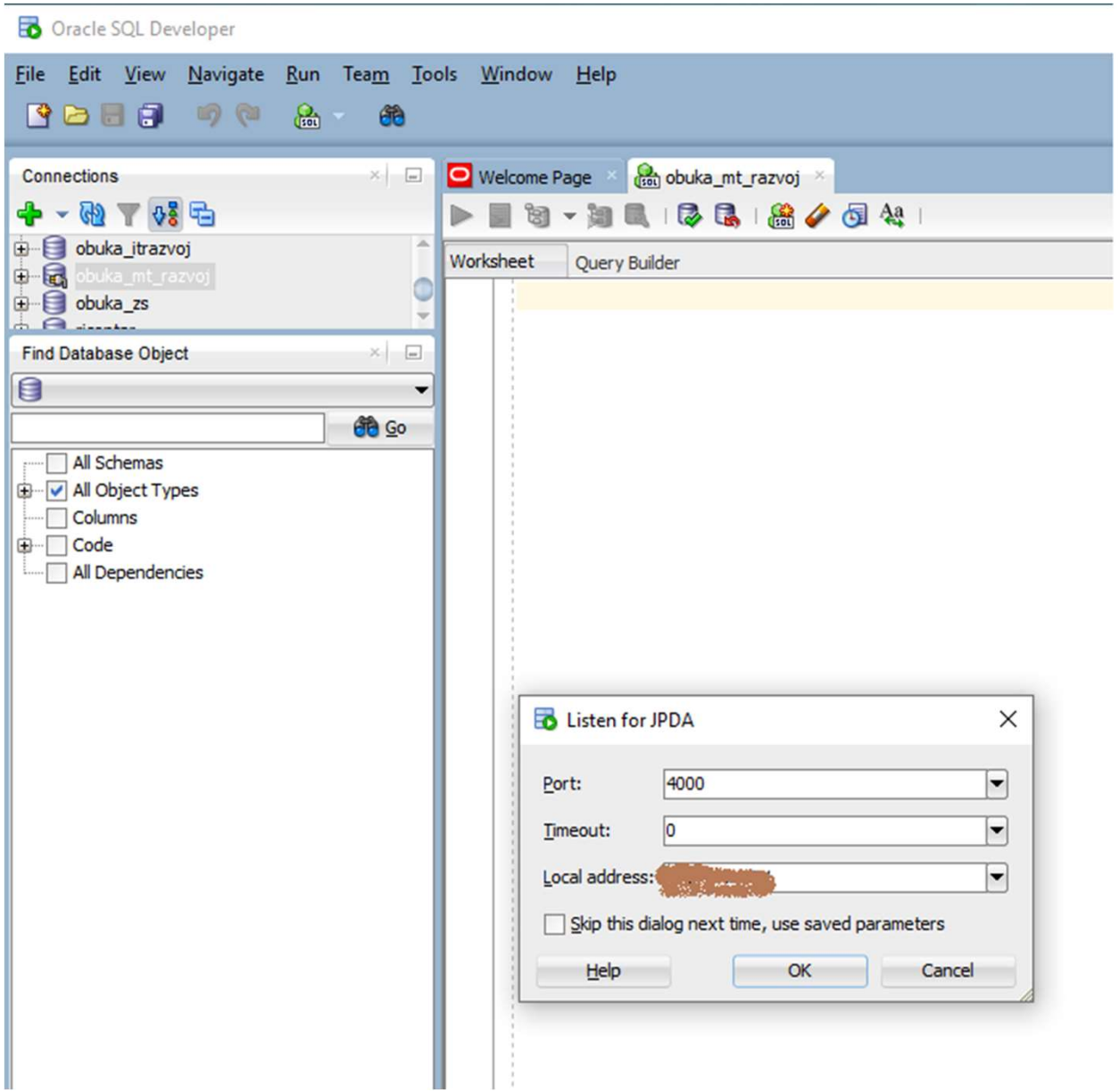
- 2. ALTER … COMPILE DEBUG; (može kroz SQL Developer)
- 3. U Sql Developeru pokrenuti remote debugging:

- desna tipka miša na konekciju … Debug -> Remote Debug



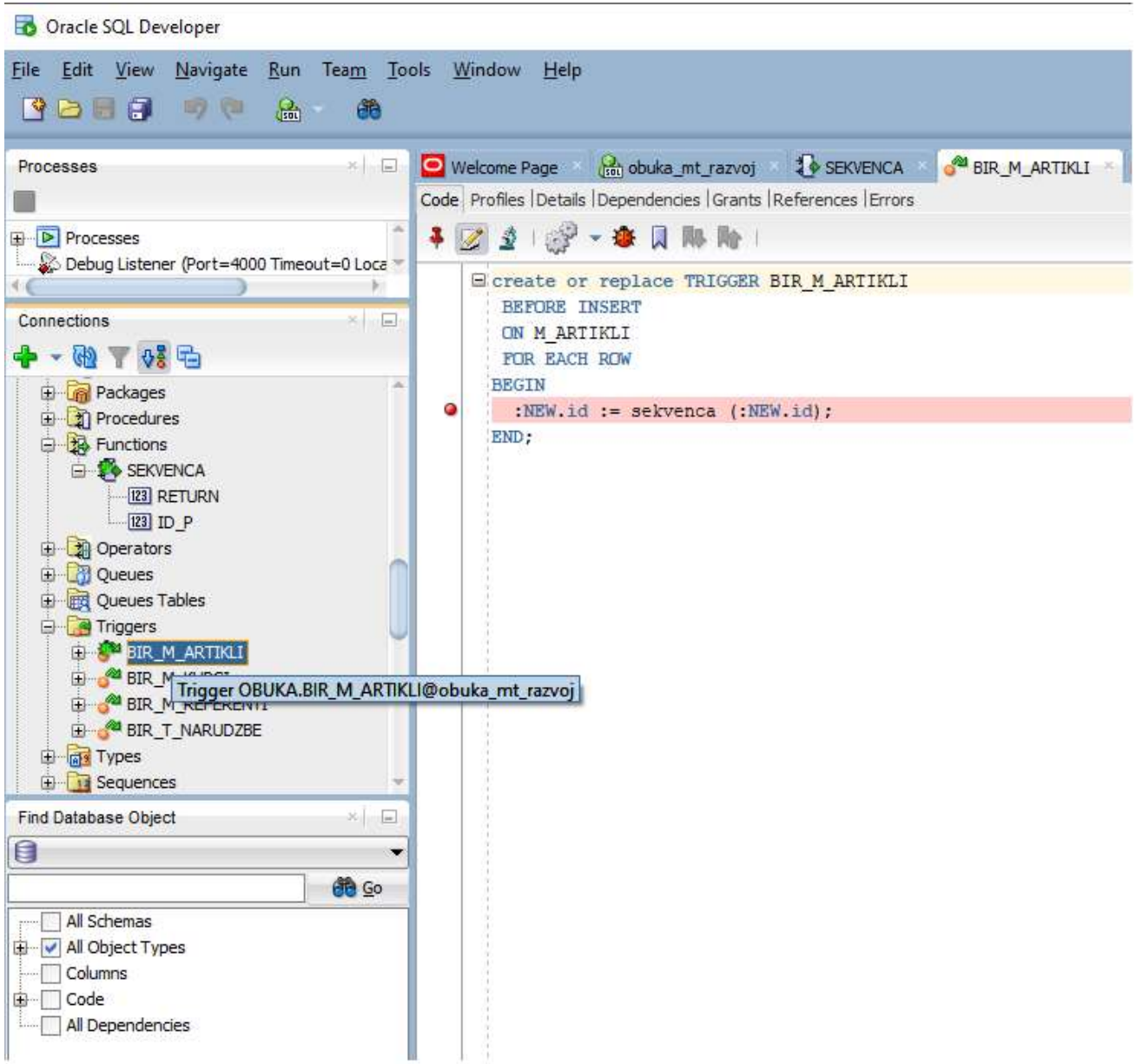
SQL Developer debugging

- 3. … unijeti IP adresu lokalnog računala, zapamtiti port, OK



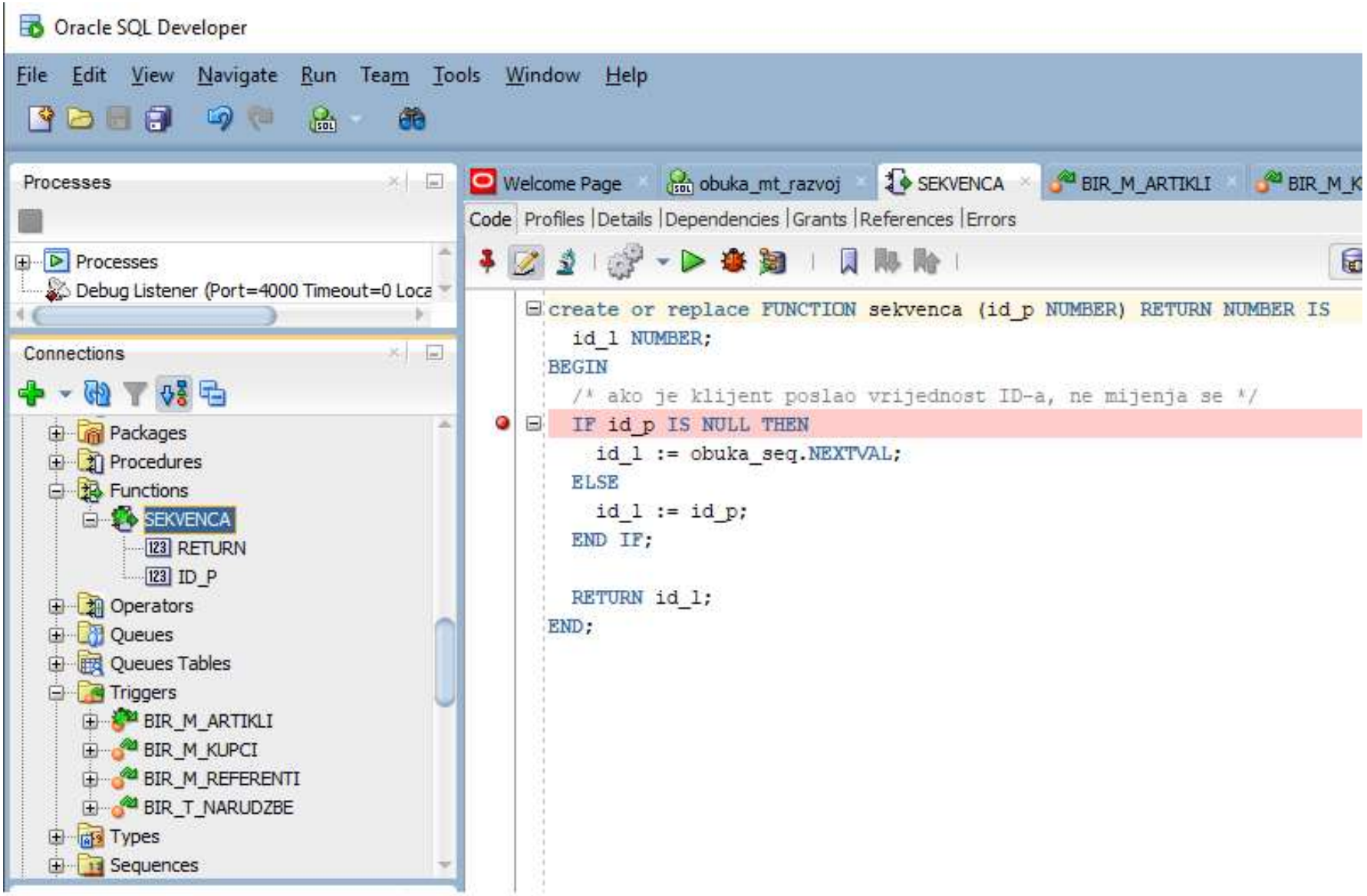
SQL Developer debugging

- 4. U Sql Developeru postavljamo Break Point u triggeru.



SQL Developer debugging

- 4. … U Sql Developeru postavljamo Break Point u funkciji.



SQL Developer debugging

- 5. Otvaramo dva puta SQL+.

Jedan SQL+ "glumi" Forms sesiju, ulazimo kao user OBUKA.

U drugi SQL+ ulazimo kao neki jaki user (u ovom slučaju SYS) i dajemo (za ovaj slučaj) naredbu:

```
select sid, serial#
  from v$plsql_debuggable_sessions
 where username = 'OBUKA'
   and program like 'sqlplus%'
 order by 1;
       SID    SERIAL#
---------- ----------
       412      10137
```

SQL Developer debugging

- 6. U drugom SQL+ izvršavamo

DBMS_DEBUG_JDWP.CONNECT_TCP kojemu proslijedimo vrijednosti za IP adresu i TCP port lokalnog računala, te

SID i SERIAL# sesije u kojoj se izvodi prvi SQL+ (ili Forms):

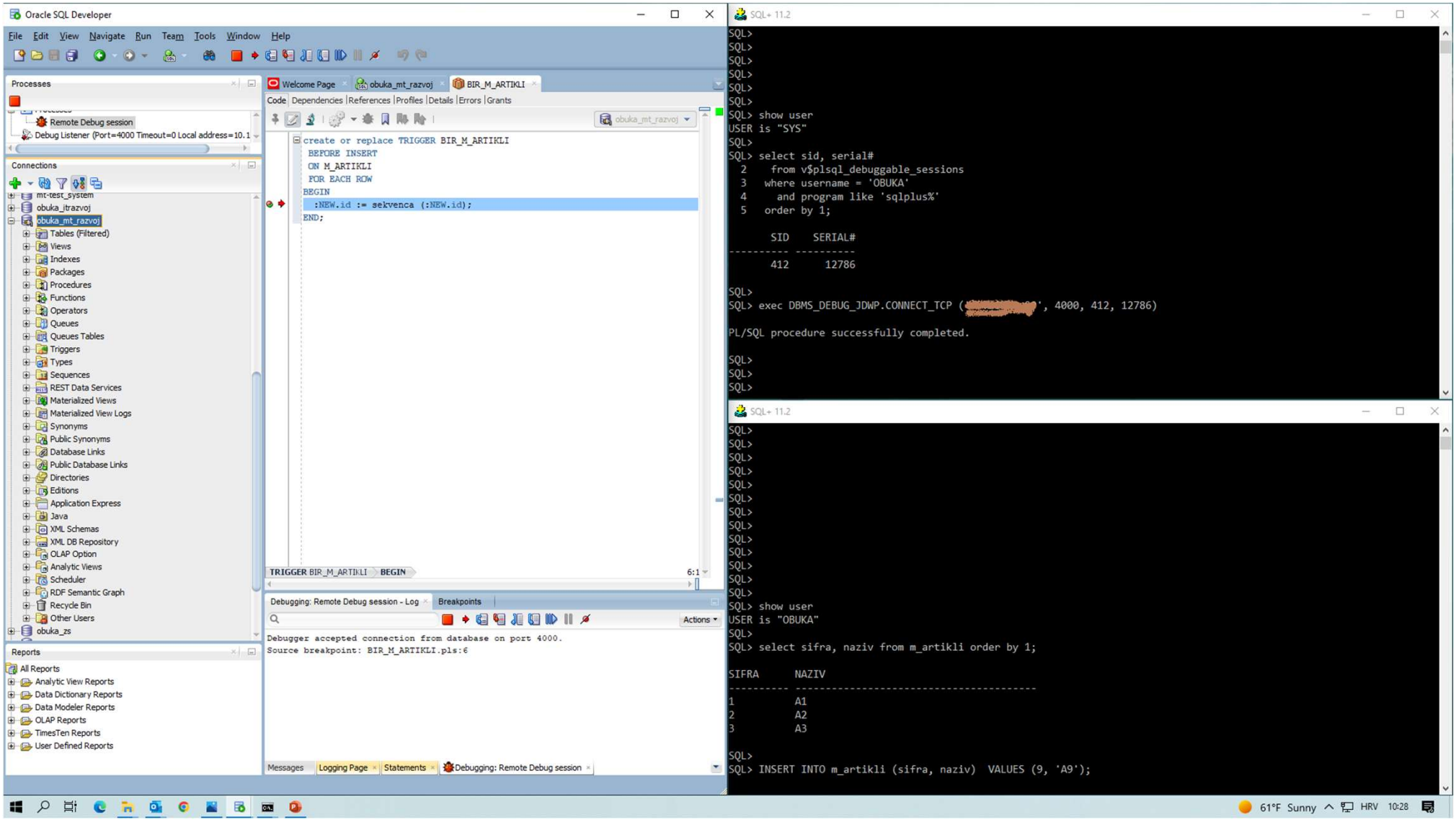
```
exec DBMS_DEBUG_JDWP.CONNECT_TCP
     ('x.x.x.x', 4000, 412, 10137)
```

7. U prvom SQL+ (koji glumi Forms aplikaciju) unesemo novi redak u tablicu M_ARTIKLI, čime će se na bazi pokrenuti trigger BIR_M_ARTIKLI, koji poziva funkciju SEKVENCA:

```
INSERT INTO m_artikli (sifra, naziv)
  VALUES (9, 'A9');
```

SQL Developer debugging

- 7. … Kad klijent pokrene proceduru i ona dođe do Break Pointa, kontrolu preuzima remote debugging proces.



Zaključak

- **Forms 12c debugger omogućava remote debugging.**
- Za razliku od Forms 6i debuggera,
  Forms 12c debugger ne prikazuje prolaz kroz Forms triggere.
- Kao i Forms Forms 6i debugger, niti Forms 12c Debugger ne prikazuje prolaz kroz triggere na bazi.
  Ali, Forms 12c debugger ne prikazuje prolaz niti kroz procedure / funkcije na bazi (iako imaju Break Point).
- Možemo koristiti SQL Developer za remote debugging koda na bazi. **Ako je Oracle baza 12.2 ili veća, SQL Developer remote debugging radi bez mijenjanja koda na bazi.**
- **Ne možemo paralelno koristiti**
  Forms Builder remote debugging i
  SQL Developer remote debugging.