

***Kako debugirati Forms programe
ili (alternativan naziv)
Kako debugirati (engl. debugging)
Forms programe***

Zlatko Sirotić, univ.spec.inf.

ISTRA TECH d.o.o.

Pula

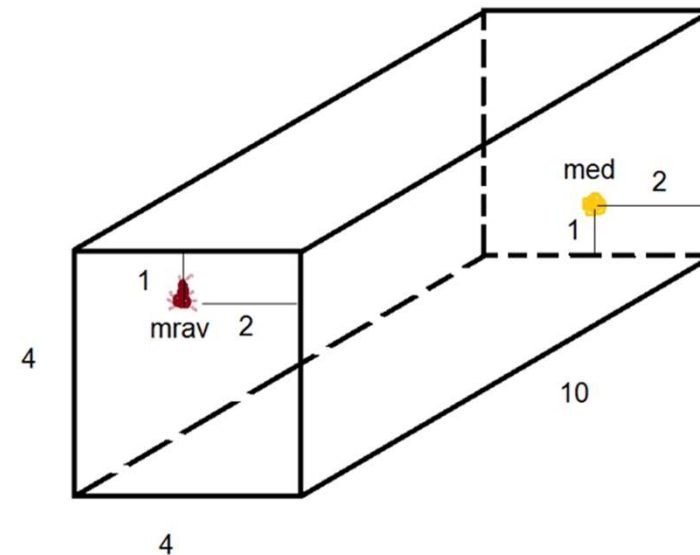
- ISTRA TECH je novo ime (od 2015.) poduzeća **Istra informatički inženjering**, osnovanog 1990. godine.
- Radim na informatičkim poslovima od 1984. godine.
- Oracle softverske alate (baza, Designer CASE, Forms 4GL, Reports, Java) koristim oko 25 godina.
- Objavljivao sam stručne radove na kongresima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

- 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.
- "Odišlo 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.istrattech.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 😊

Different MVC Approaches

- Your main choices:

Alternative	Client	Middle	Data
1	Thin	Fat	Thin
2	Fat	Fat	Thin
3	Fat	Thin	Thin
4	Fat	Thin	Fat
5	Thin	Fat	Fat
6	Thin	Thin	Fat
7	Fat	Fat	Fat

Fat = Lots of code in this tier
Thin = Little code in this tier
(no number 8...)

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.

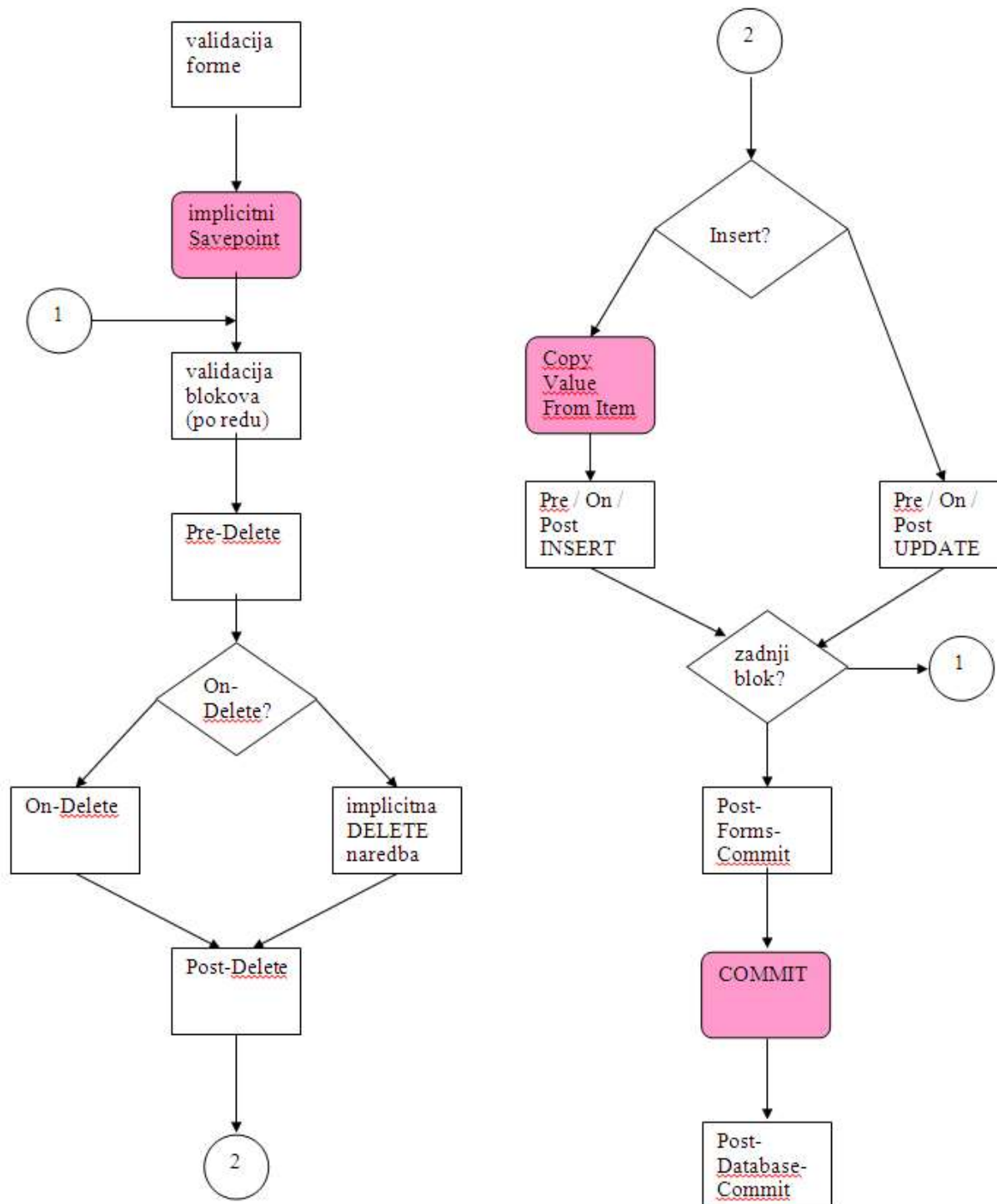
Oracle Forms

- 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

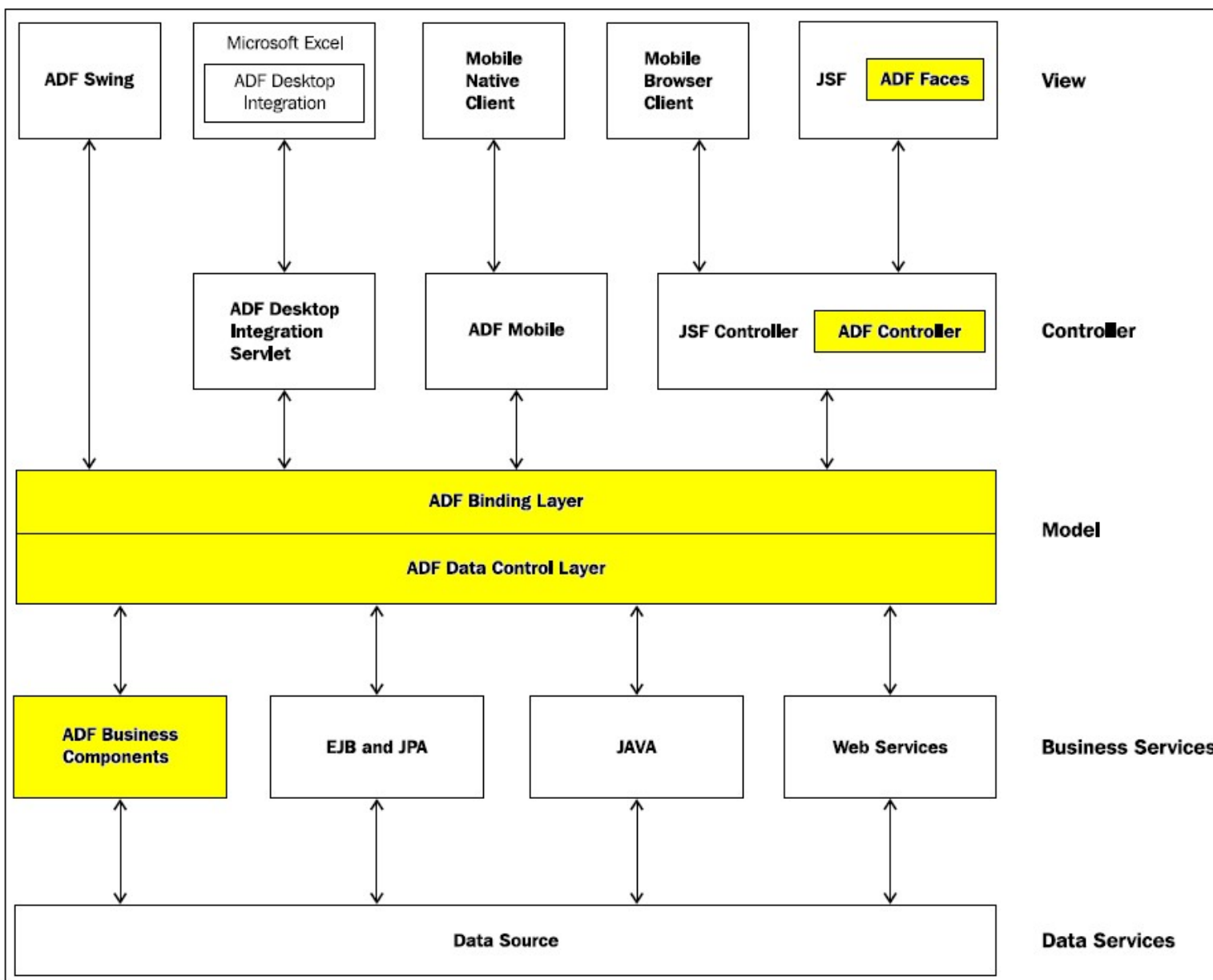
- Među najvažnijim, ali i najsloženijim Forms procesima, jesu COMMIT i POST procesi.
- COMMIT proces može se jednostavno (i točno) prikazati kao:
COMMIT proces =
 POST proces +
 COMMIT naredba na bazi +
 Forms okidač Post-Database-Commit
- Suština je u tome da se ažuriranje redaka (unos / izmjena / brisanje) 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).



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 pokrivaio 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



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;  
*/
```

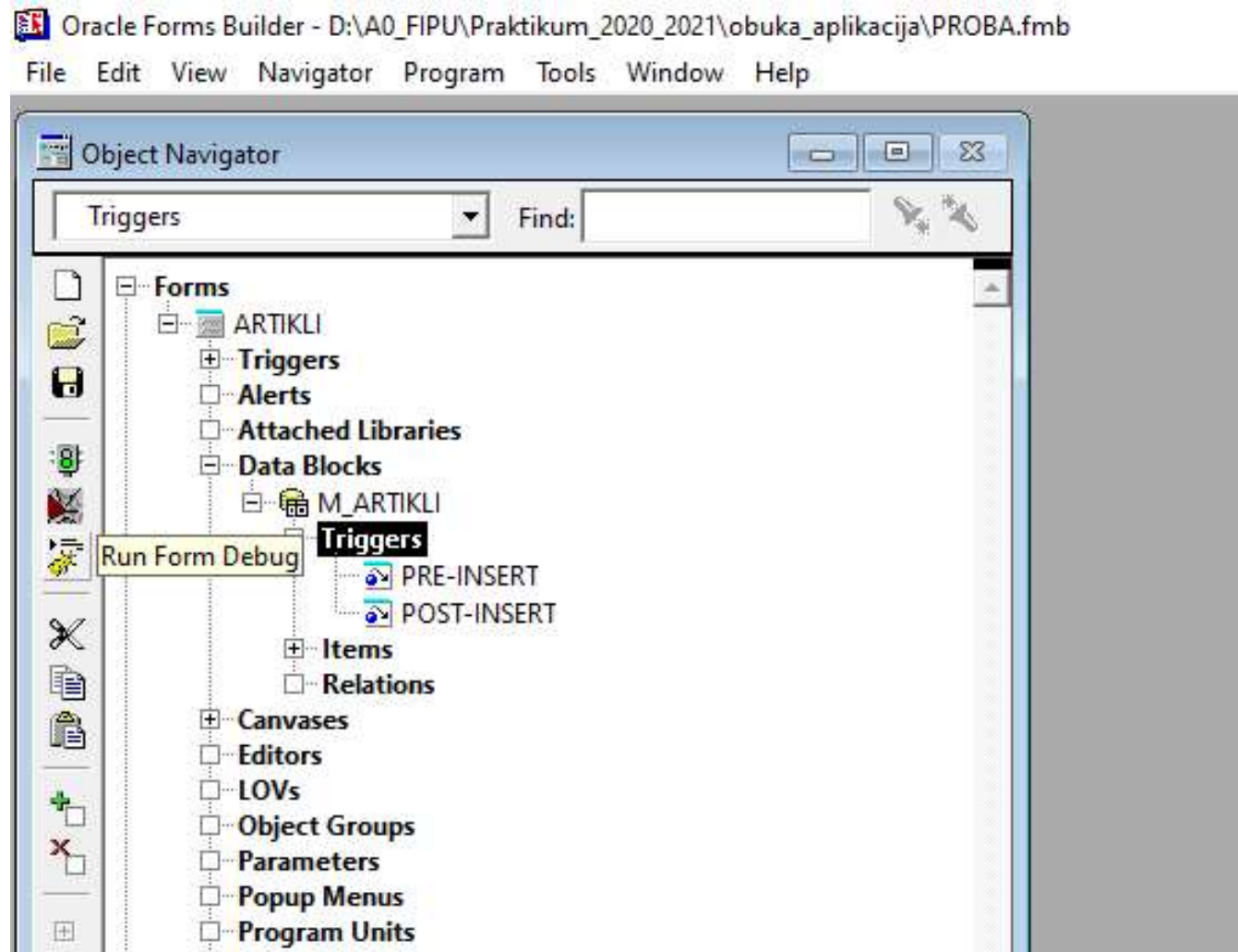
Forms 6i debugging

- 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.

The screenshot displays the Oracle Forms Builder interface on the left and the Oracle Forms Runtime debugger on the right. The Object Navigator in the Builder shows a tree structure for the 'ARTIKLI' form, with the 'PRE-INSERT (M_ARTIKLI)' trigger selected. The Forms Debugger window shows the client program unit 'PRE-INSERT (M_ARTIKLI) (Block)' with the following PL/SQL code:

```
00001 BEGIN
00002 MESSAGE ('In pre-insert trigger');
00003 PAUSE;
00004
00005 /*
00006 DECLARE
00007   i INTEGER := 0;
00008 BEGIN
00009   i := sekvenca (NULL);
```

The debugger also shows a tree view of the 'ARTIKLI' form structure, including 'Triggers', 'Attached Libraries', 'Blocks', 'M_ARTIKLI', 'Triggers', 'Items', 'Parameter Lists and Values', and 'Program Units'. The 'PRE-INSERT (M_ARTIKLI)' trigger is highlighted. At the bottom of the debugger, the PL/SQL prompt shows the command to set a break point:

```
PL/SQL> .break .
Breakpoint #1 installed at line 2 of PRE-INSERT (M_ARTIKLI)
PL/SQL>
```

Forms 6i debugging

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

Oracle Forms Builder - D:\A0_FIPU\Praktikum_2020_2021\obuka_aplikacija\PROBA.fmb
File Edit View Navigator Program Tools Window Help

Object Navigator

Triggers

Forms

- ARTIKLI
 - Triggers
 - Alerts
 - Attached Libraries
 - Data Blocks
 - M_ARTIK
 - Triggers
 - Items
 - Relations
 - Canvases
 - Editors
 - LOVs
 - Object Groups
 - Parameters
 - Popup Menus
 - Program Units
 - Property Classes
 - Record Groups
 - Reports
 - Visual Attributes
 - Windows
- Menus
- PL/SQL Libraries
- Object Libraries
 - OLB01
 - Library Tabs
- Built-in Packages
- Database Objects

Oracle Forms Runtime

Akcije Matični podaci Dokumenti Window Pomoć

Spremi

Artikli

Šifra	Naziv
1	A1
10	A10
2	A2
3	A3
4	A4
5	A5
6	A6

Forms 6i debugging

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

The screenshot displays the Oracle Forms Builder interface with the Forms Debugger active. The Object Navigator on the left shows the 'Triggers' folder expanded for the 'ARTIKLI' form, with the 'PRE-INSERT (M_ARTIKLI)' trigger selected. The main window shows the trigger's PL/SQL code, and the Forms Debugger window is open, showing the execution flow. The debugger is currently stopped at line 2, which contains the message command: `B(01) MESSAGE ('In pre-insert trigger');`. The stack trace below the code shows the current execution context: [2] Block PRE-INSERT (M_ARTIKLI) Line 2.

```
Oracle Forms Builder - D:\A0_FIPU\Praktikum_2020_2021\obuka_aplikacija\PROBA.fmb
File Edit View Navigator Program Tools Window Help

Object Navigator
Triggers
Forms
  ARTIKLI
    Triggers
    Alerts
    Attached Libra
    Data Blocks
      M_ARTIK
        Triggers
          PRE-INSERT (M_ARTIKLI)
          PRE-INSERT (M_ARTIKLI)
        Items
        Relation
      Canvases
      Editors
      LOVs
      Object Groups
      Parameters
      Popup Menus
      Program Units
      Property Classe
      Record Groups
      Reports
      Visual Attribut
      Windows
    Menus
    PL/SQL Libraries
    Object Libraries
      OLB01
        Library Tabs
    Built-in Packages
    Database Objects

Oracle Forms Runtime
Akcije Matični podaci Dokumenti View Navigator Program Debug Pomoć

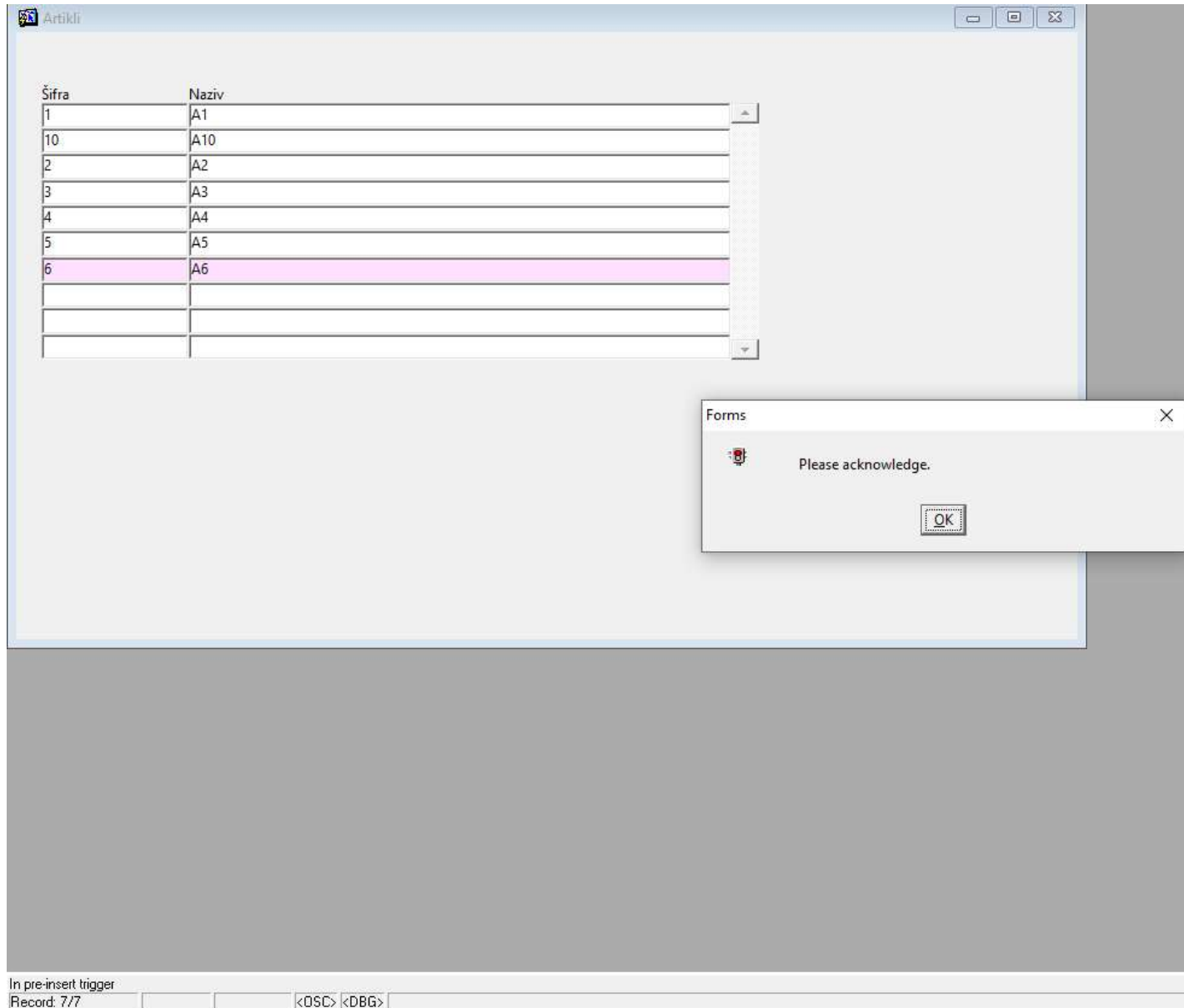
Artikli
Šifra
1
10
2
3
4
5
6

Forms Debugger
Step Into Client Program Unit: PRE-INSERT (M_ARTIKLI) (Block)
00001 BEGIN
00002 B(01) MESSAGE ('In pre-insert trigger');
00003 PAUSE;
00004
00005 /*
00006 DECLARE
00007   i INTEGER := 0;
00008 BEGIN
00009   i := sekvenca (NULL);

Modules
Global Variables
System Variables
Command Line Parameters
Built-in Packages
Debug Actions
Stack
  [0] Program Unit <Unknown> Line 2
  [1] Program Unit <Unknown> Line 544
  [2] Block PRE-INSERT (M_ARTIKLI) Line 2
Database Objects
```


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).

The screenshot shows the Oracle Forms Builder environment. The main window displays the 'Object Navigator' on the left, showing a tree view of the form objects. The 'Triggers' folder is expanded, showing the 'PRE-INSERT (M_ARTIKLI) (Block)' trigger. The 'Forms Debugger' window is open, showing the PL/SQL code for the trigger. The code is as follows:

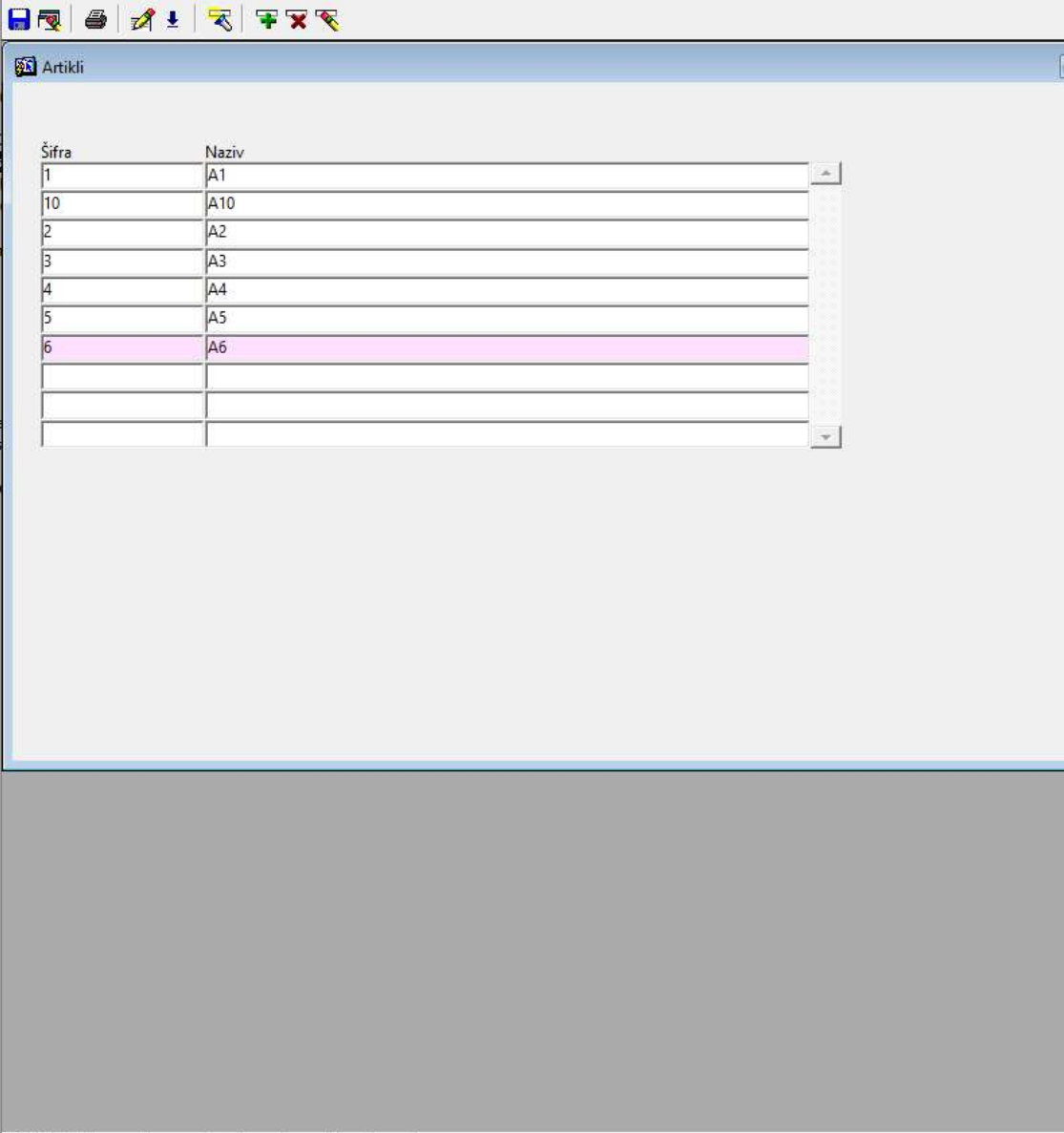
```
00007  i INTEGER := 0;  
00008  BEGIN  
00009    i := sekvenca (NULL);  
00010    MESSAGE ('i: ' || i);  
00011    PAUSE;  
00012  END;  
00013  */  
00014  END;
```

The 'Stack' window in the debugger shows the current execution context:

- [0] Program Unit <Unknown> Line 2
- [1] Program Unit <Unknown> Line 544
- [2] Block PRE-INSERT (M_ARTIKLI) Line 14

Forms 6i debugging

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



Šifra	Naziv
1	A1
10	A10
2	A2
3	A3
4	A4
5	A5
6	A6

FRM-40400: Transaction complete: 1 records applied and saved.

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:

Oracle Forms Runtime

Akcije Matični podaci Dokumenti View Navigator Program Debug Pomoć

Artikli

Šifra	Naziv
1	A1
10	A10
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7

Forms Debugger

Step Over Program Unit: PRE-INSERT (M_ARTIKLI) (Block)

```
00001 BEGIN
00002 B (01) MESSAGE ('In pre-insert trigger');
00003 PAUSE;
00004
00005 DECLARE
00006   i INTEGER := 0;
00007 BEGIN
00008   i := sekvenca (NULL);
00009   MESSAGE ('i: ' || i);
00010   PAUSE;
00011 END;
```

Modules

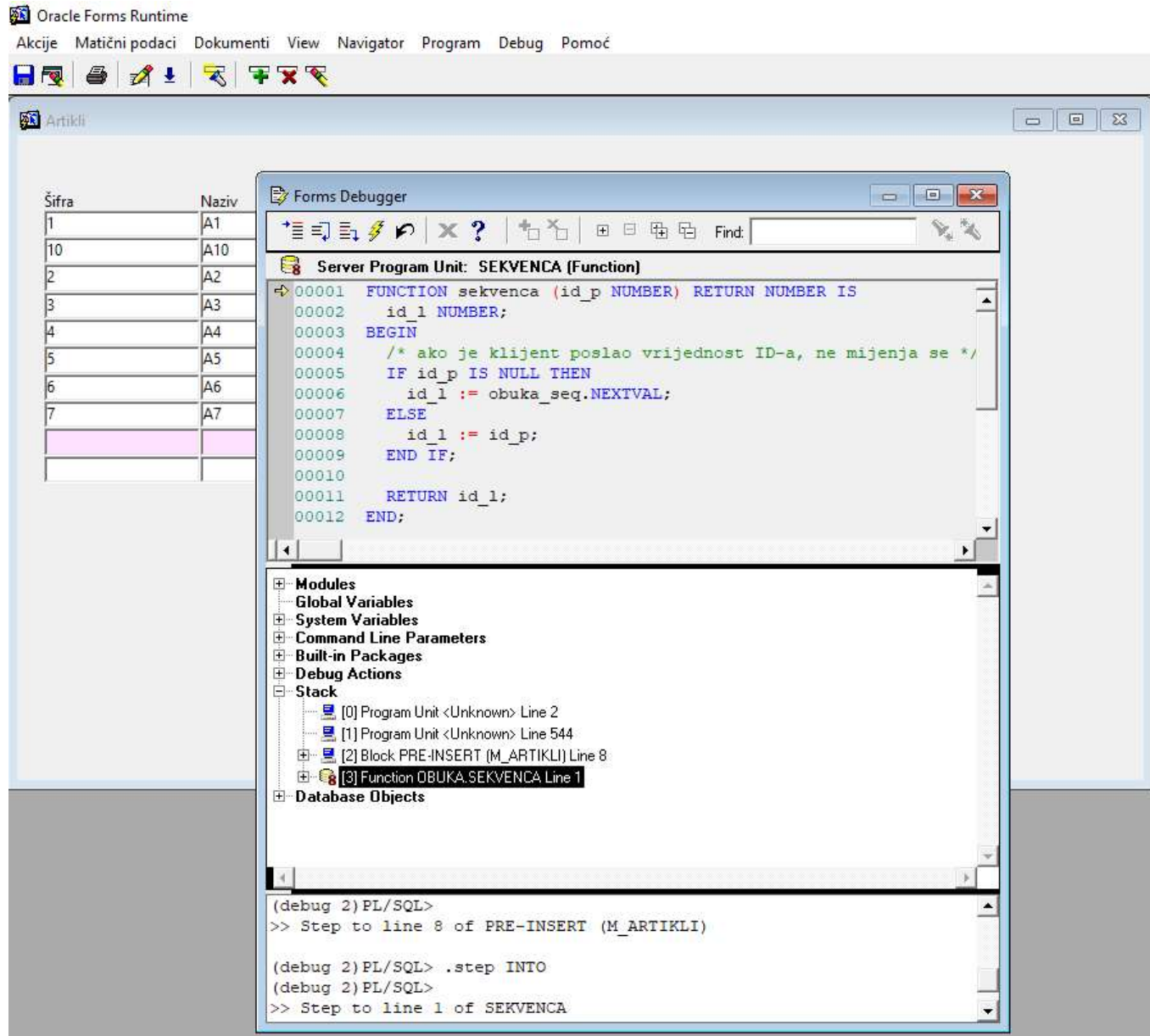
- Global Variables
- System Variables
- Command Line Parameters
- Built-in Packages
- Debug Actions
- Stack
 - [0] Program Unit <Unknown> Line 2
 - [1] Program Unit <Unknown> Line 544
 - [2] Block PRE-INSERT (M_ARTIKLI) Line 8
- Database Objects

(debug 2) PL/SQL>
>> Step to line 6 of PRE-INSERT (M_ARTIKLI)

(debug 2) PL/SQL> .step INTO
(debug 2) PL/SQL>

Forms 6i debugging

- Vidimo da se Forms Debugger postavio na početak funkcije:



Oracle Forms Runtime

Akcije Matični podaci Dokumenti View Navigator Program Debug Pomoć

Artikli

Šifra	Naziv
1	A1
10	A10
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7

Forms Debugger

Server Program Unit: SEKVENCA (Function)

```

00001 FUNCTION sekvenca (id_p NUMBER) RETURN NUMBER IS
00002   id_l NUMBER;
00003 BEGIN
00004   /* ako je klijent poslao vrijednost ID-a, ne mijenja se */
00005   IF id_p IS NULL THEN
00006     id_l := obuka_seq.NEXTVAL;
00007   ELSE
00008     id_l := id_p;
00009   END IF;
00010
00011   RETURN id_l;
00012 END;

```

Modules

- Global Variables
- System Variables
- Command Line Parameters
- Built-in Packages
- Debug Actions
- Stack
 - [0] Program Unit <Unknown> Line 2
 - [1] Program Unit <Unknown> Line 544
 - [2] Block PRE-INSERT (M_ARTIKLI) Line 8
 - [3] Function OBUKA.SEKVENCA Line 1
- Database Objects

```

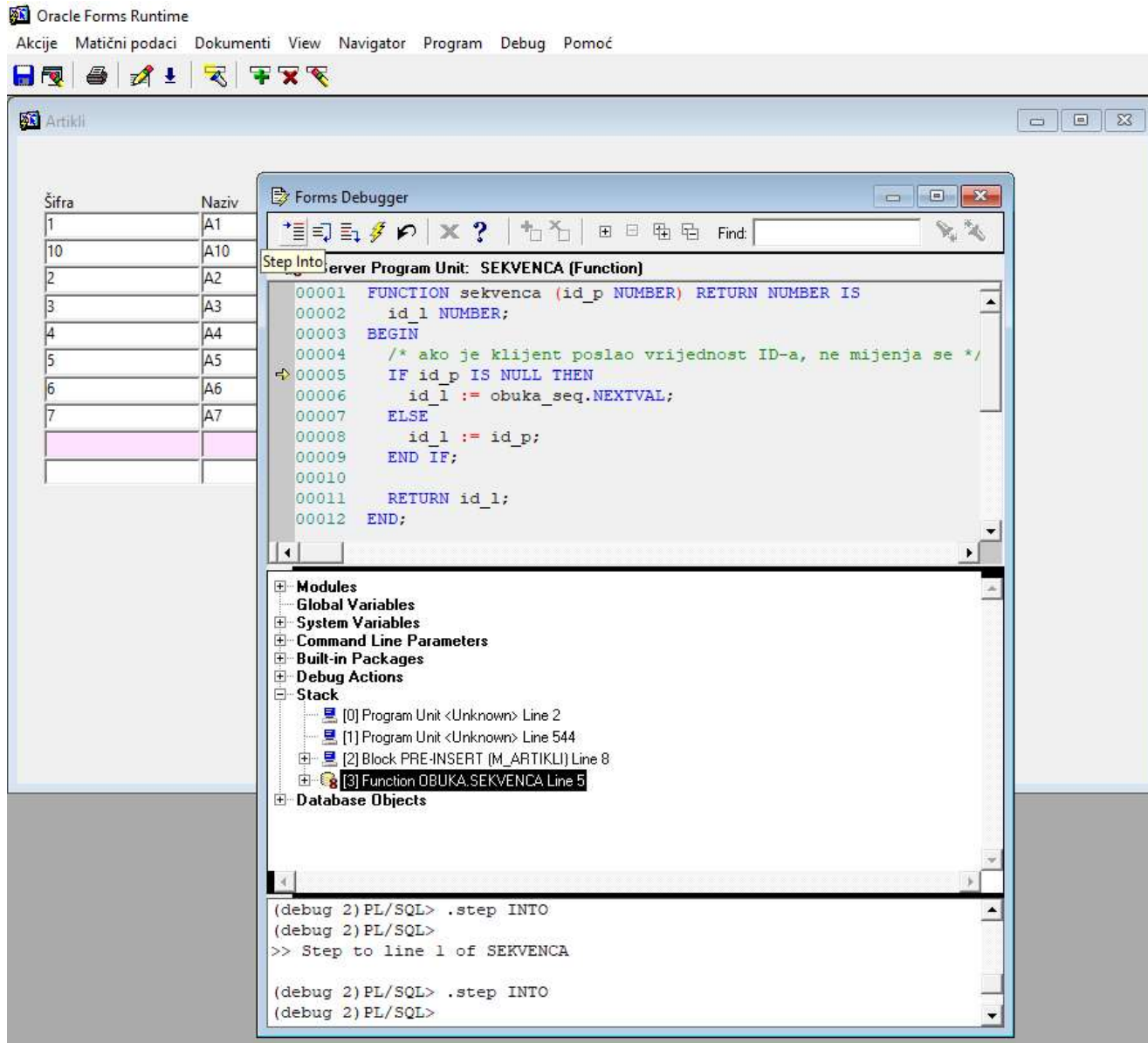
(debug 2) PL/SQL>
>> Step to line 8 of PRE-INSERT (M_ARTIKLI)

(debug 2) PL/SQL> .step INTO
(debug 2) PL/SQL>
>> Step to line 1 of SEKVENCA

```


Forms 6i debugging

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



Oracle Forms Runtime

Akcije Matični podaci Dokumenti View Navigator Program Debug Pomoć

Artikli

Šifra	Naziv
1	A1
10	A10
2	A2
3	A3
4	A4
5	A5
6	A6
7	A7

Forms Debugger

Step Into

Server Program Unit: SEKVENCA (Function)

```

00001 FUNCTION sekvenca (id_p NUMBER) RETURN NUMBER IS
00002   id_l NUMBER;
00003 BEGIN
00004   /* ako je klijent poslao vrijednost ID-a, ne mijenja se */
00005   IF id_p IS NULL THEN
00006     id_l := obuka_seq.NEXTVAL;
00007   ELSE
00008     id_l := id_p;
00009   END IF;
00010
00011   RETURN id_l;
00012 END;

```

Modules

- Global Variables
- System Variables
- Command Line Parameters
- Built-in Packages
- Debug Actions
- Stack
 - [0] Program Unit <Unknown> Line 2
 - [1] Program Unit <Unknown> Line 544
 - [2] Block PRE-INSERT (M_ARTIKLI) Line 8
 - [3] Function OBUKA.SEKVENCA Line 5
- Database Objects

```

(debug 2) PL/SQL> .step INTO
(debug 2) PL/SQL>
>> Step to line 1 of SEKVENCA

(debug 2) PL/SQL> .step INTO
(debug 2) PL/SQL>

```

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

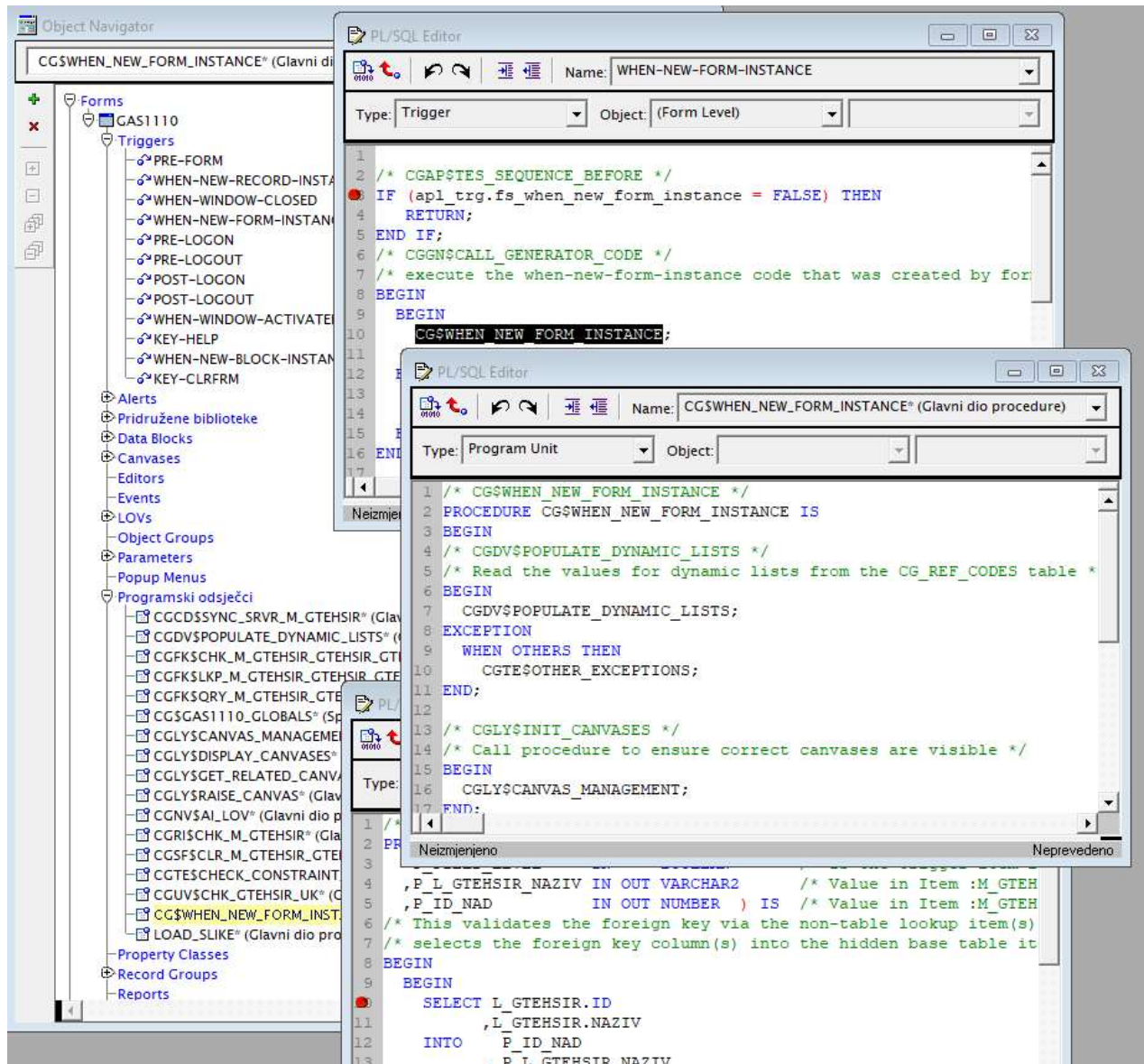
4. Once executed, it will prompt HOST and PORT (it has choices for the Debugger) then click button "OK"
5. At Form builder, choose **Debug -> "Attach Debug"**
6. It will prompt to enter Host and Port.
Enter the values as Prompted by step 4 then click button "OK"
7. Run the form and follow the break points, 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

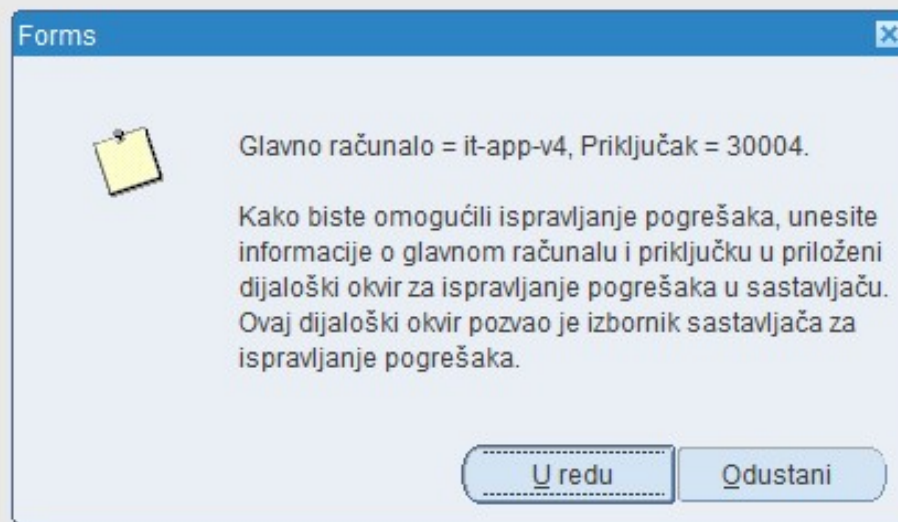
- 1. Open Forms Builder
- 2. Open a Form which you plan to debug. Add a breakpoint ...



The screenshot displays the Oracle Forms 12c development environment. On the left, the Object Navigator shows a tree structure for the form 'GAS1110', with 'Triggers' expanded to show various events like 'WHEN-NEW-FORM-INSTANCE'. Two PL/SQL Editor windows are open. The top window shows the trigger code for 'WHEN-NEW-FORM-INSTANCE', with a red breakpoint marker on line 3: 'IF (apl trig.fs_when_new_form_instance = FALSE) THEN'. The bottom window shows the code for the procedure 'CG\$WHEN_NEW_FORM_INSTANCE', with a red breakpoint marker on line 1: '/* CG\$WHEN_NEW_FORM_INSTANCE */'. The Object Navigator also shows a list of 'Programski odsječki' (Program Units) including 'CGD\$SYNC_SRV_M_GTEHSIR', 'CGD\$POPULATE_DYNAMIC_LISTS', and 'CG\$WHEN_NEW_FORM_INST'.

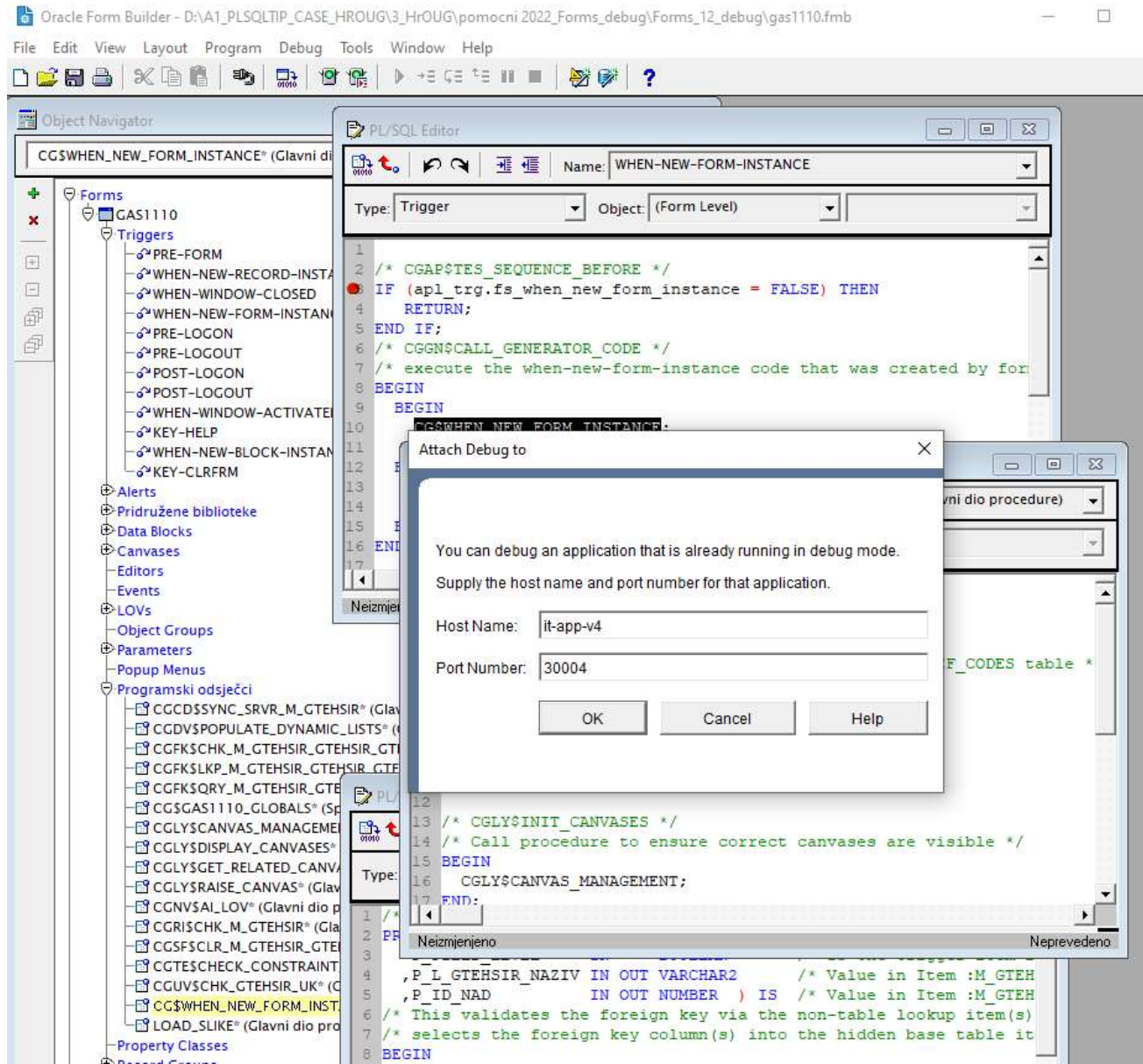
Forms 12c debugging

- 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 debugirati (zamrzne se i čeka ...).



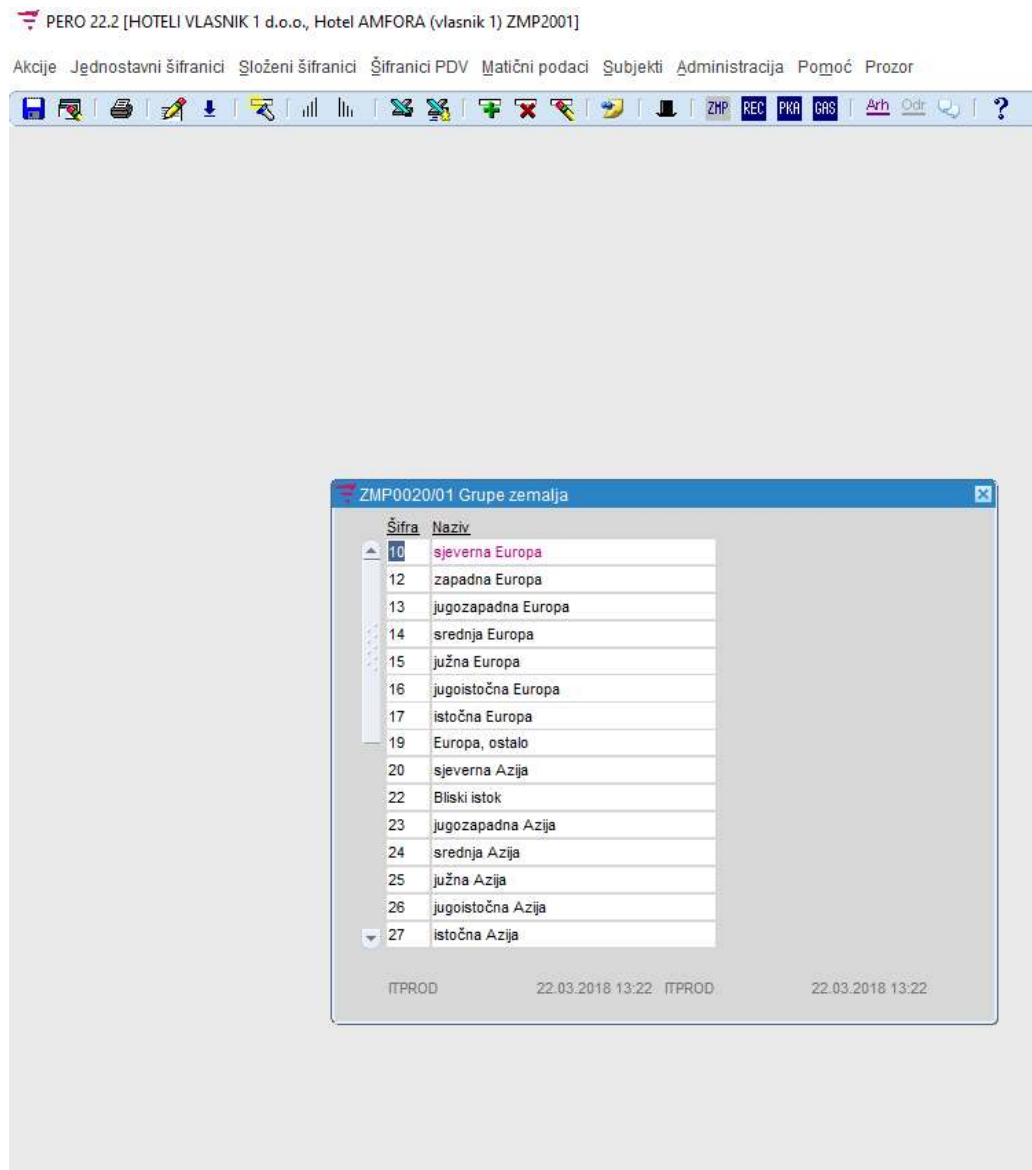
Forms 12c debugging

- 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

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

The screenshot displays the Oracle Forms 12c development environment. On the left, the Object Navigator shows the project structure for 'GAS1110'. The main window is the PL/SQL Editor, which is open to the 'WHEN-NEW-FORM-INSTANCE' trigger. The code in the editor includes a conditional check for a flag and a call to a procedure named 'CG\$WHEN_NEW_FORM_INSTANCE'. A second PL/SQL Editor window is open, showing the code for the 'CG\$WHEN_NEW_FORM_INSTANCE' procedure, which includes logic for dynamic lists and canvas management.

On the right, a data table is visible, showing a list of items with columns for 'Šifra' (Code) and 'Naziv' (Name). The table is titled 'GAS8120/08 Tehnološke grupe sirovina'. The following table represents the data shown in the screenshot:

Šifra	Naziv
000	OSTALO
100	MESO
1001	meso teleće
103	JUHE KONCENTRAT
105	SUHOMESNATI PROIZVODI
110	RIBE, ŠKOLJKE I RAKOVI
115	VOĆE
120	POVRĆE
125	TJESTENINA I ŽITARICE, RIŽ
130	KRUH, PECIVA, KOLAČI, KEKSI
135	ČOKOLADA, PUDING, KAKAO
140	MARMELADA, DŽEM I MED
145	ULJE, MARGARIN, MASLAC
150	MLUEKO I MLUEČNI PROIZVODI
155	JAJA, MAJONEZA I SL.
160	ZAČINI, MIRODUE I DODACI
165	ČAJEVI, KAVA I SUROGATI
170	VINA I PJENUŠCI
171	PJENUŠCI I ŠAMPANJCI
175	ŽESTOKA ALKOHOLNA PIĆA
176	BITTERI I LIKERI
177	KONJAK, BRANDY, VINJAK
178	GIN, VODKA, TEQUILA, MEZCAL
180	PIVO
185	MINERALNA I SODA VODA

□ Sada počinje debugging proces.

The screenshot displays the Oracle Forms Builder interface during a debugging session. The main window is the PL/SQL Editor, showing the code for a trigger named 'WHEN-NEW-FORM-INSTANCE' and a procedure named 'CGFK\$LKP_M_GTEHSIR_GTEHSIR_GTE'. The trigger code is as follows:

```

1  /* CGAP$TIES_SEQUENCE_BEFORE */
2  IF (apl_trg.fs_when_new_form_instance = FALSE) THEN
3  RETURN;
4  END IF;

```

The procedure code is as follows:

```

1  /* CGFK$LKP_M_GTEHSIR_GTEHSIR_GTE */
2  PROCEDURE CGFK$LKP_M_GTEHSIR_GTEHSIR_GTE (
3  P_FIELD_LEVEL      IN      BOOLEAN      /* Is the trigger item 1
4  ,P_L_GTEHSIR_NAZIV IN OUT VARCHAR2    /* Value in Item :M_GTEH
5  ,P_ID_NAD           IN OUT NUMBER     ) IS /* Value in Item :M_GTEH
6  /* This validates the foreign key via the non-table lookup item(s)
7  /* selects the foreign key column(s) into the hidden base table it
8  BEGIN
9  BEGIN
10 SELECT L_GTEHSIR.ID
11       ,L_GTEHSIR.NAZIV
12 INTO   P_ID_NAD
13       , P_L_GTEHSIR_NAZIV
14 FROM   M_GTEHSIR L_GTEHSIR
15 WHERE  L_GTEHSIR.SIFRA = :M_GTEHSIR.L_GTEHSIR_SIFRA;
16 EXCEPTION
17 WHEN TOO_MANY_ROWS THEN

```

The Object Navigator on the left shows the form structure for 'GAS1110', including various triggers and program units. The Debug Console at the bottom shows the stack of active frames:

```

Stack
-----
Stack Frames
GAS1110.CGFK$LKP_M_GTEHSIR_GTEHSIR_GTE...
GAS1110.WHEN-VALIDATE-ITEM (M_GTE...
zmpa000.GAS::5

```

On the right side, a data table is visible, titled 'GAS1110/01 Tehnološke grupe sirovina'. It lists various items with their codes and names:

Šifra	Naziv
000	OSTALO
100	MESO
1001	meso teleće
103	JUHE KONCENTRAT
105	SUHOMESNATI PROIZVODI
110	RIBE, ŠKOLJKE I RAKOVI
115	VOĆE
120	POVRĆE
125	TJESTENINA I ŽITARICE, RIŽA, JE
130	KRUH, PECIVA, KOLAČI, KEKSI
135	ČOKOLADA, PUDING, KAKAO U I
140	MARMELADA, DŽEM I MED
145	ULJE, MARGARIN, MASLAC
150	MLJEKO I MLJEČNI PROIZVODI
155	JAJA, MAJONEZA I SL.
160	ZAČINI, MIRODUE I DODACI JELIM
165	ČAJEVI, KAVA I SUROGATI KAVI
170	VINA I PJENUŠCI
171	PJENUŠCI I ŠAMPANJCI
175	ŽESTOKA ALKOHOLNA PIĆA
176	BITTERI I LIKERI
177	KONJAK, BRANDY, VINJAK
178	GIN, VODKA, TEQUILA, MEZCAL
180	PIVO
185	MINERALNA I SODA VODA

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).

The screenshot illustrates a debugging scenario in Oracle Forms 12c. On the left, the Oracle Form Builder interface is shown with the PL/SQL Editor open to a trigger named 'WHEN-NEW-FORM-INSTANCE'. The trigger code is as follows:

```

1
2 /* CGAP$TES_SEQUENCE_BEFORE */
3 IF (apl_trg.fs_when_new_form_instance = FALSE) THEN
4     RETURN;
5 END IF;
6 /* CCGN$CALL_GENERATOR_CODE */
7 /* execute the when-new-form-instance code that was created by for
8 BEGIN
9 BEGIN
10 CG$WHEN_NEW_FORM_INSTANCE;
11
12
13
14
15
16 END

```

Below the trigger editor, the PL/SQL Editor shows a procedure named 'CG\$WHEN_NEW_FORM_INSTANCE' with the following code:

```

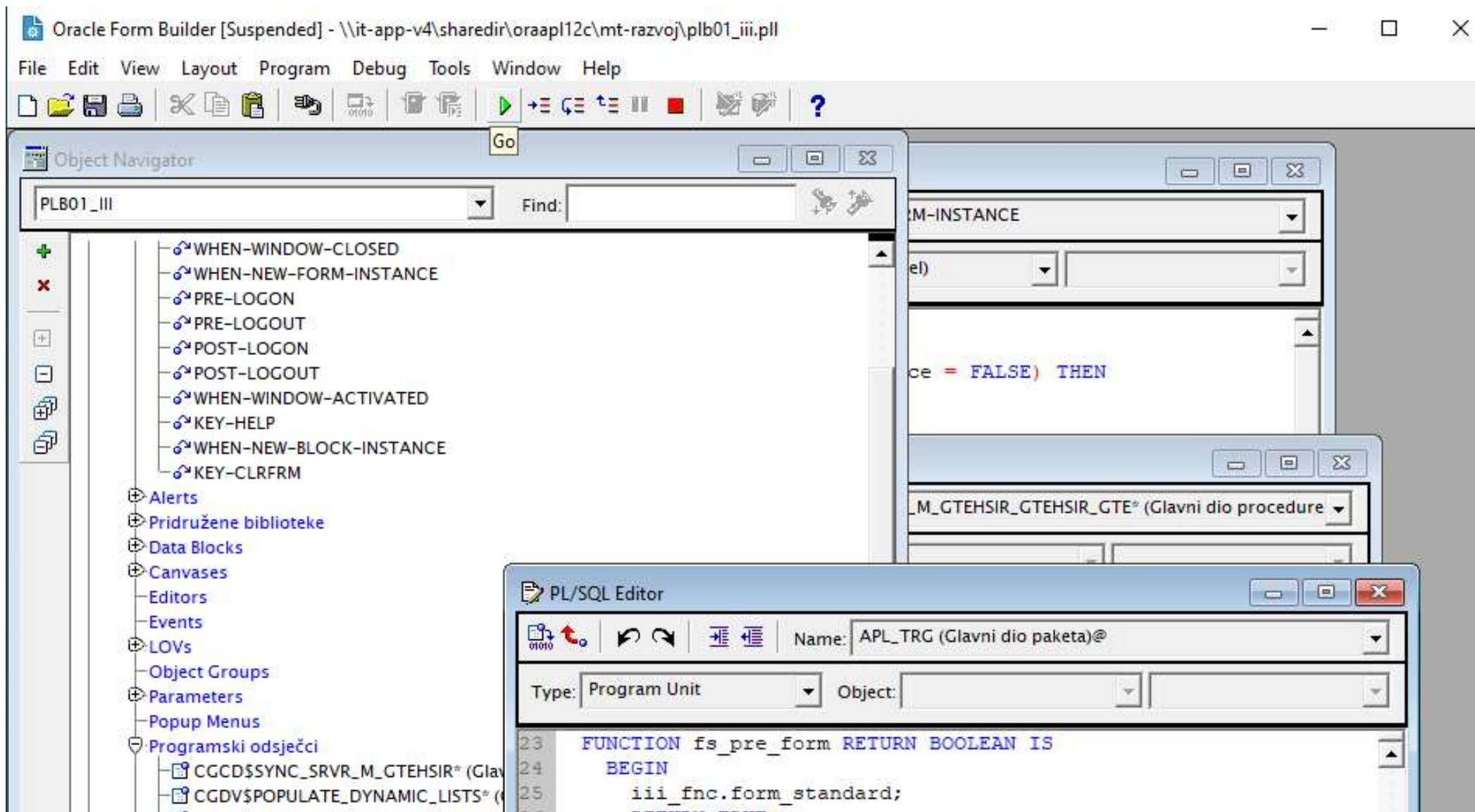
1 /* CG$WHEN_NEW_FORM_INSTANCE */
2 PROCEDURE CG$WHEN_NEW_FORM_INSTANCE IS
3 BEGIN
4 /* CGDV$POPULATE_DYNAMIC_LISTS */
5 /* Read the values for dynamic lists from the CG_REF_CODES table */
6 BEGIN
7     CGDV$POPULATE_DYNAMIC_LISTS;
8 EXCEPTION
9     WHEN OTHERS THEN
10        CGTE$OTHER_EXCEPTIONS;
11 END;
12
13 /* CGLY$INIT_CANVASES */
14 /* Call procedure to ensure correct canvases are visible */
15 BEGIN
16     CGLY$CANVAS_MANAGEMENT;
17 END;

```

On the right, a web application window titled 'PERO 22.2 [HOTELI VLASNIK 1 d.o.o., Hotel AMFORA (vlasnik 1) ZMP2001]' is shown. It displays a table of raw material groups. A modal window titled 'GAS8120/08 Tehnološke grupe sirovina' is open, showing a list of items with columns 'Šifra' and 'Naziv'. The list includes items like '1001 meso teleće', '176 BITTERI I LIKERI', '165 ČAJEVI, KAVA I SUROGATI KAVE', etc.

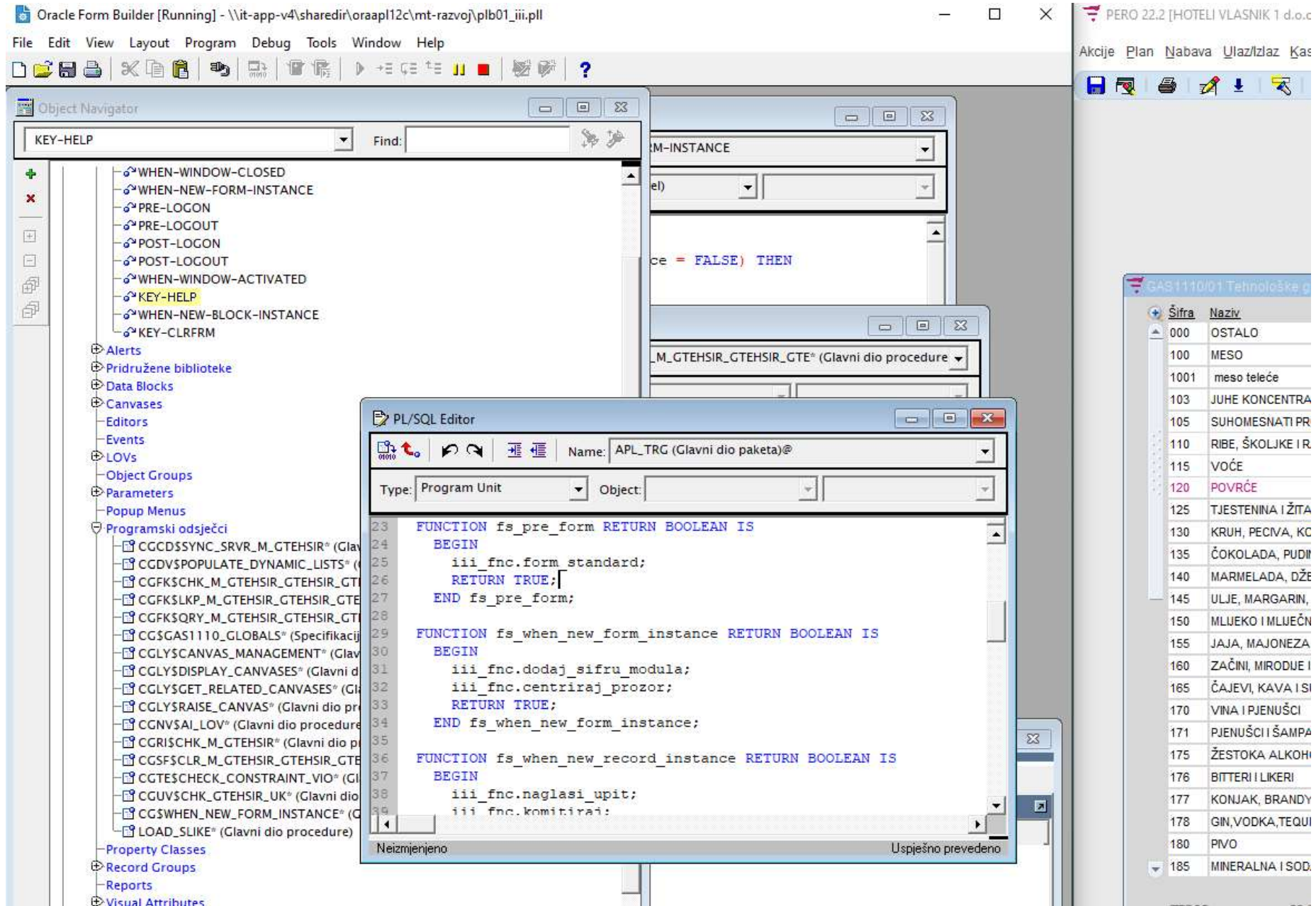
Forms 12c debugging

□ Tipkom Go prekidamo debugiranje.



Forms 12c debugging

□ Aplikacija nastavlja s radom, do sljedećeg Break Pointa.



The screenshot shows the Oracle Form Builder interface with the Object Navigator on the left, displaying a tree view of objects including 'KEY-HELP'. The main window shows the form's data blocks and triggers. A PL/SQL Editor window is open, displaying the following code:

```

23 FUNCTION fs_pre_form RETURN BOOLEAN IS
24 BEGIN
25     iii_fnc.form standard;
26     RETURN TRUE;
27 END fs_pre_form;
28
29 FUNCTION fs_when_new_form_instance RETURN BOOLEAN IS
30 BEGIN
31     iii_fnc.dodaj_sifru_modula;
32     iii_fnc.centriraj_prozor;
33     RETURN TRUE;
34 END fs_when_new_form_instance;
35
36 FUNCTION fs_when_new_record_instance RETURN BOOLEAN IS
37 BEGIN
38     iii_fnc.naglas_i_upit;
39     iii_fnc.komitiraj;

```

The PL/SQL Editor window also shows the status 'Neizmjenjeno' and 'Uspješno prevedeno'. On the right side, there is a window titled 'PERO 22.2 [HOTELI VLASNIK 1 d.o.o.]' displaying a list of items with columns 'Šifra' and 'Naziv':

Šifra	Naziv
000	OSTALO
100	MESO
1001	meso teleće
103	JUHE KONCENTRA
105	SUHOMESNATI PR
110	RIBE, ŠKOLJKE I R
115	VOĆE
120	POVRĆE
125	TJESTENINA I ŽITA
130	KRUH, PECIVA, KC
135	ČOKOLADA, PUDI
140	MARMELADA, DŽE
145	ULJE, MARGARIN,
150	MLJEKO I MLJEČN
155	JAJA, MAJONEZA
160	ZAČINI, MIRODJE I
165	ČAJEVI, KAVA I SI
170	VINA I PJENUŠCI
171	PJENUŠCI I ŠAMPA
175	ŽESTOKA ALKOHI
176	BITTER I LIKERI
177	KONJAK, BRANDY
178	GIN, VODKA, TEQUI
180	PIVO
185	MINERALNA I SOD.

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

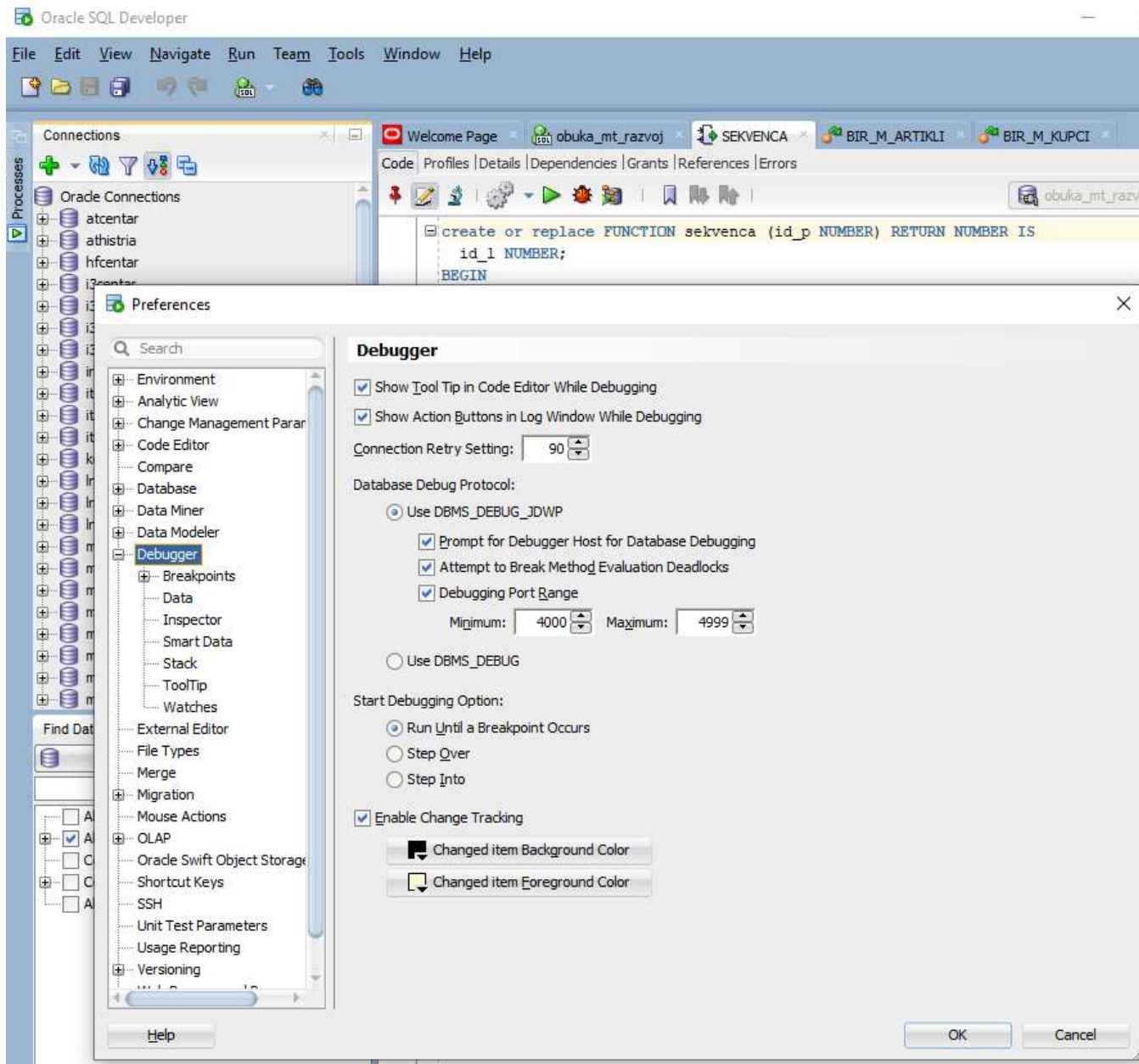
5. Izvršiti npr. za Forms sesije (prije i poslije pokretanja određenog Forms modula - da lakše nađemo željenu sesiju):

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

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

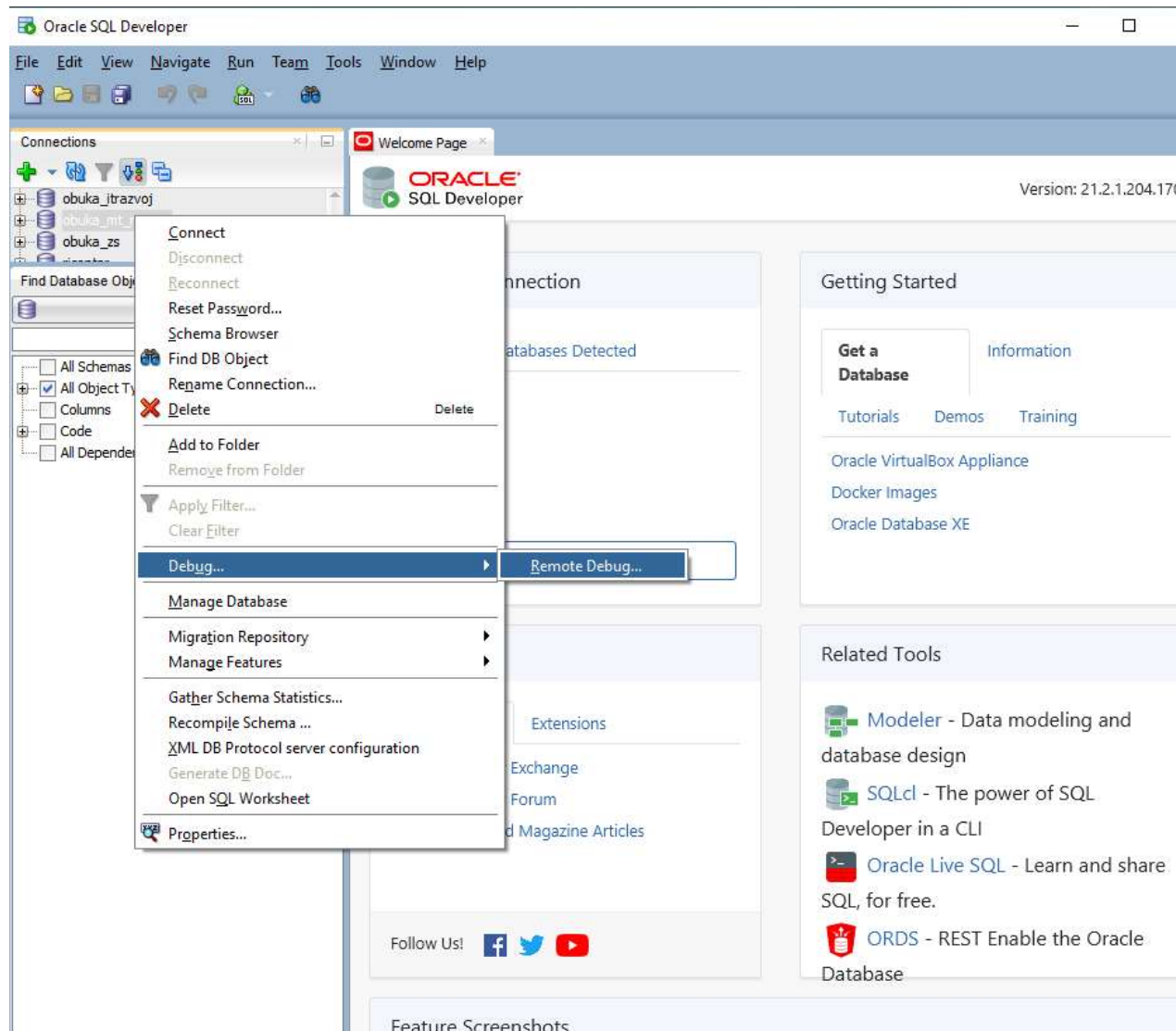
7. Kad klijent pokrene proceduru i ona dođe do Break Pointa, kontrolu preuzima remote debugging proces.

□ 1. ... postaviti Debugging Port Range

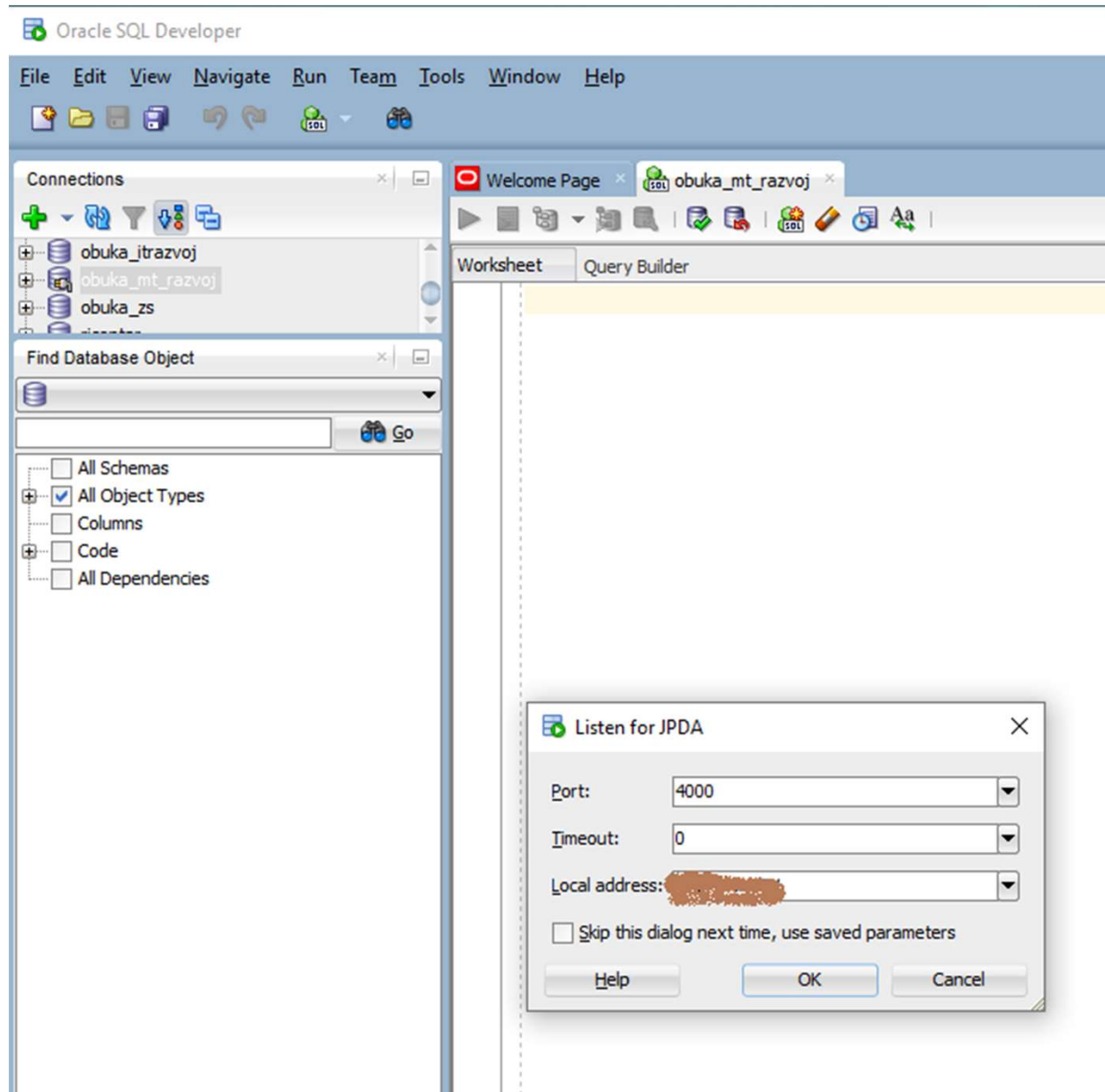


SQL Developer debugging

- 2. ALTER ... COMPILE **DEBUG**; (može kroz SQL Developer)
- 3. U Sql Developeru pokrenuti remote debugging:
 - desna tipka miša na konekciju ... Debug -> Remote Debug

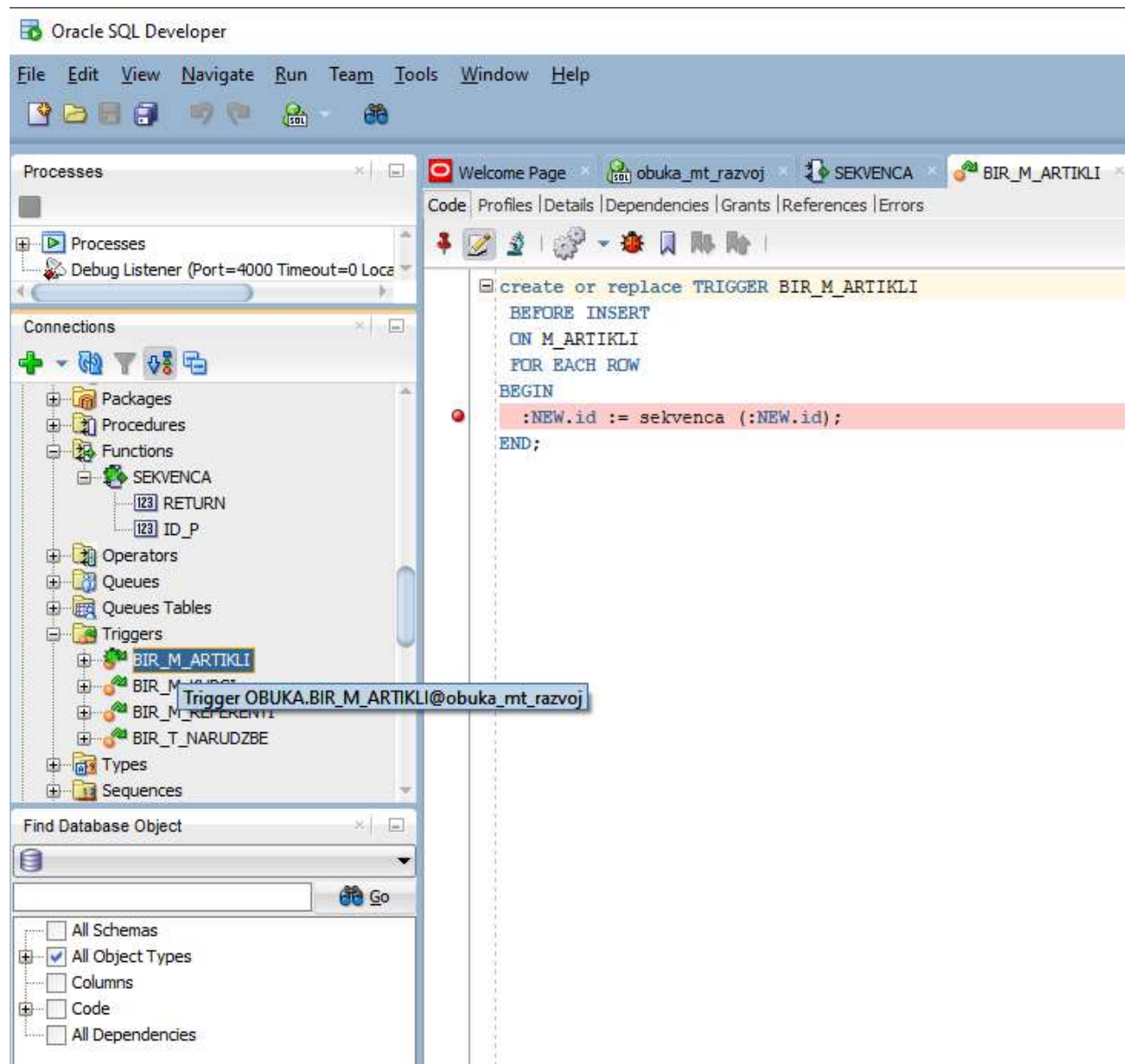


- 3. ... unijeti IP adresu lokalnog računala, zapamtiti port, OK



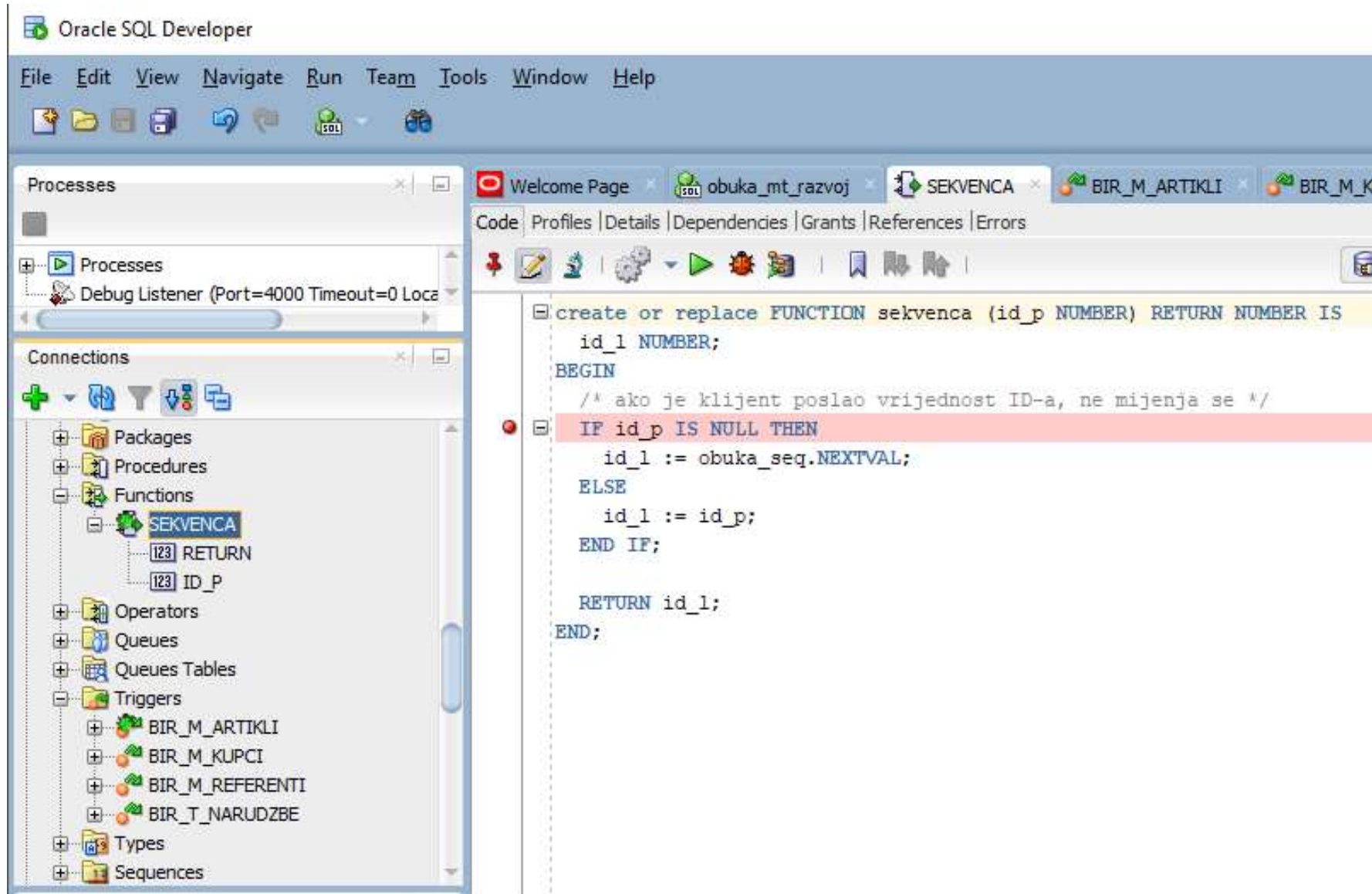
SQL Developer debugging

- 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$sql_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 prosljedimo 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');
```


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.**