

# Using Oracle Database 10 *g* PL/SQL New Features

## Purpose

This module describes the new Oracle Database 10g features introduced in PL/SQL.

## Topics

This module will discuss the following:

- ☒ [Overview](#)
- ☒ [Prerequisites](#)
- ☒ [Examining Bulk Binding Using `INDICES OF` and `VALUES OF` Keywords](#)
- ☒ [Setting Privileges and Parameters for Debugging](#)
- ☒ [Creating a Database Connection](#)
- ☒ [Debugging a PL/SQL Package Body Using JDeveloper](#)

## Viewing Screenshots

## Overview

[Back to List of Topics](#)

In Oracle Database 10 *g* , several enhancements are available in the PL/SQL language. This lesson covers:

1. [Bulk Binding Enhancements](#)
2. [Debugging PL/SQL Using JDeveloper](#)
3. [PL/SQL Compiler Enhancements](#)

## Bulk Binding Enhancements

[Back to List](#)

The Oracle Database 10 *g* extends upon the enhancements introduced in Oracle9 *i* , in the area of Bulk Binding. The `SAVE EXCEPTIONS` syntax was introduced in Oracle9 *i* to capture exceptions while bulk inserting (deleting or updating) rows. While this feature allows the DML to continue, saving any exceptions in the `SQL%BULK_EXCEPTIONS` collection, the performance of the operation is greatly affected. In a certain scenario where the collection being processed is sparse due to the application logic involved, this would be an unnecessary overhead. In Oracle Database 10 *g* , you can overcome these issues with the new `INDICES OF` or `VALUES OF` features.

## Debugging PL/SQL Using JDeveloper

[Back to List](#)

Starting version 9.0.3, JDeveloper supports PL/SQL debugging. This is achieved with the Java Debugging Wire Protocol (JDWP). With Oracle9 *i* Release 2 and higher, the debugger uses the industry standard JPDA (Java Platform Debugger Architecture) whereas for prior releases (Oracle8 *i* or Oracle9 *i* Release 1) the `DBMS_DEBUG` package is used. The debugging UI gives you the ability to single step, step into, and step over PL/SQL objects as well as the ability to set PL/SQL expressions in the Watch and Inspector window, set conditional breakpoints, and view collection data.

## PL/SQL Compiler Enhancements

[Back to List](#)

The PL/SQL compiler is substantially improved in the Oracle Database 10 *g* . The back end code generator is re-engineered. Also, a new parameter, called `PLSQL_OPTIMIZE_LEVEL` , has been introduced to speed up execution of PL/SQL code. This is in addition to Native compilation (which is available since Oracle9 *i* ). Currently the `PLSQL_OPTIMIZE_LEVEL` parameter can be set to " 1 " or " 2 ". 1 is the default, 2 increases the performance.

## Prerequisites

[Back to List](#)

Before starting this module, you should have performed the following:

1. Completed the [Configuring Linux for the Installation of Oracle Database 10g](#) lesson
2. Completed the [Installing the Oracle Database 10g on Linux](#) lesson
3. Completed the [Installing Oracle9i JDeveloper on Linux](#) lesson.
4. Download and unzip the [plsql.zip](#) into your working directory.

## Examining Bulk Binding Using `INDICES OF` and `VALUES OF` Keywords

[Back to List](#)

The following example illustrates the use of `INDICES OF` and `VALUES OF` keywords when using Bulk Binding in PL/SQL.

The `INDICES OF` keyword can be used in a scenario where a collection of records (which is dense) is validated programmatically and invalid records (those not meeting the specified criterion) are removed from the collection, resulting in a sparse collection of valid elements which must then be bulk inserted into a table. By using the `INDICES OF` the exceptions for the missing records are not generated.

The `VALUES OF` keyword can be used in a scenario where a collection of records (sparse or dense) must be copied to one or more collection variables, based upon some condition whereby certain records may or may not be copied, and then inserted into a table. This can be efficiently done with the use of the " `VALUES OF` " syntax and using a pointer array

whose elements are pointers to the selected records within the original collection. This reduces the need to create multiple copies of data. Exception handling is done per pointer record, which means that if two or more pointer records are pointing to the same "original" data, any exceptions reported will indicate the iteration number (pointer element number).

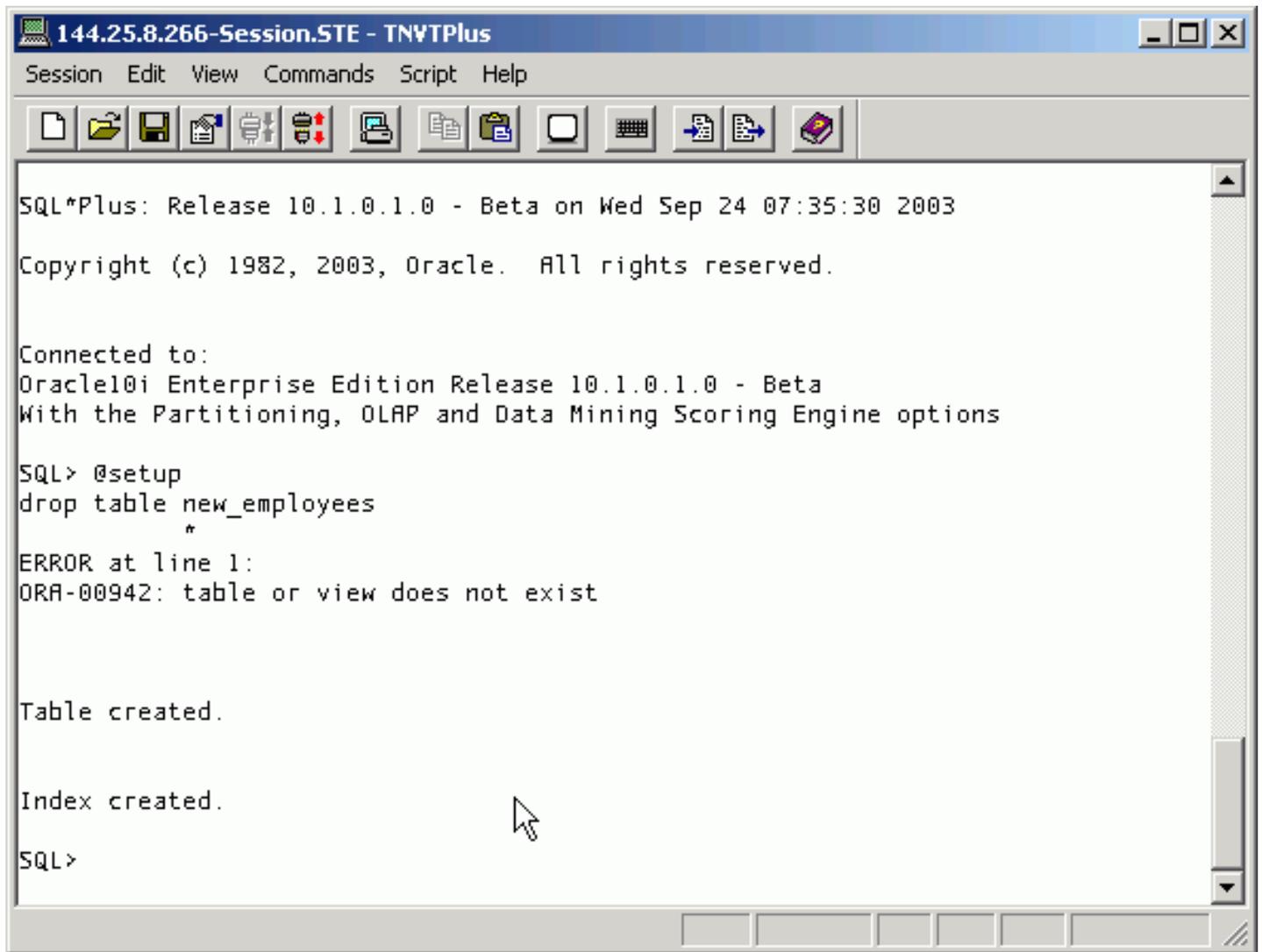
This example uses the `EMPLOYEES` table to build a collection of employees that is deliberately sparse.

1. You will first need to create a `NEW_EMPLOYEES` table from the existing `EMPLOYEES` table (in the `hr` schema) and create a unique index on `EMPLOYEE_ID` column. This will generate an exception when inserting a duplicate record. Execute the following commands from a terminal window:

```
cd wkdir
sqlplus hr/hr
@setup.sql
```

The `setup.sql` script contains the following:

```
drop table new_employees;
create table new_employees as select * from employees where 1=2;
create unique index new_employees_employee_id
  on new_employees (employee_id);
```



```
144.25.8.266-Session.STE - TNVTPlus
Session Edit View Commands Script Help

SQL*Plus: Release 10.1.0.1.0 - Beta on Wed Sep 24 07:35:30 2003

Copyright (c) 1982, 2003, Oracle. All rights reserved.

Connected to:
Oracle10i Enterprise Edition Release 10.1.0.1.0 - Beta
With the Partitioning, OLAP and Data Mining Scoring Engine options

SQL> @setup
drop table new_employees
      *
ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

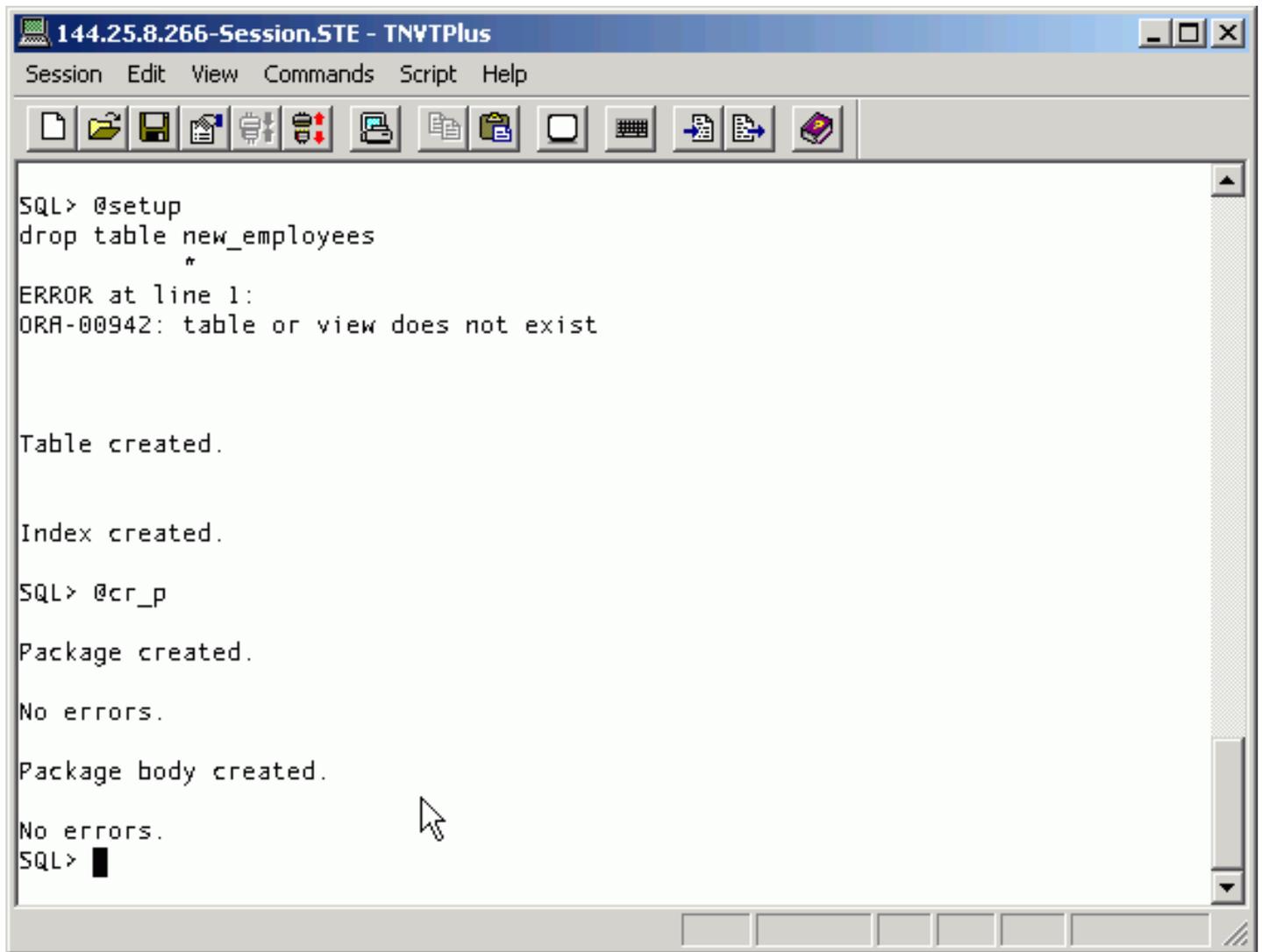
Index created.

SQL>
```

2. Next, you need to create the package `P_BULK_BIND`. This package contains various procedures which create the sparse collection and bulk insert the records into the `NEW_EMPLOYEES` table using three different approaches.

```
@cr_p.sql
```

Click here to see what is contained in the [cr\\_p.sql](#) script.



The screenshot shows a window titled "144.25.8.266-Session.STE - TNVTPPlus". The menu bar includes "Session", "Edit", "View", "Commands", "Script", and "Help". The toolbar contains icons for file operations (new, open, save, print, copy, paste) and execution (run, refresh). The main text area displays the following SQL\*Plus session:

```
SQL> @setup
drop table new_employees
      *
ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

Index created.

SQL> @cr_p
Package created.

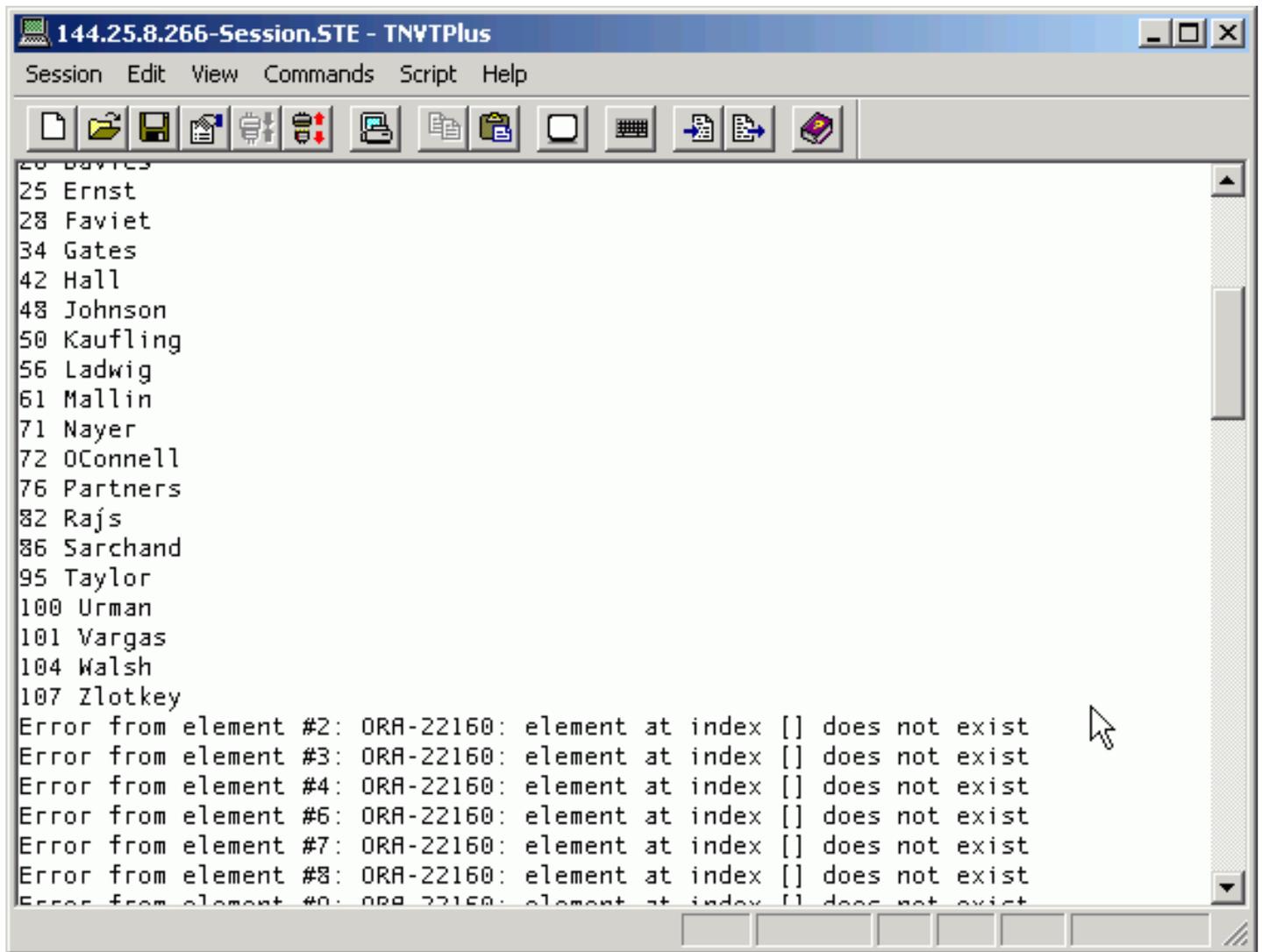
No errors.

Package body created.

No errors.
SQL> █
```

3. To see how the Bulk Binding works in Oracle Database 10g, you will first see how it was done in Oracle9i. The `FORALL` statement uses the `SAVE EXCEPTIONS` syntax to suppress exceptions raised due to missing records, which are later printed in the `EXCEPTION` block. Execute the following commands:

```
set serverout on size 100000
exec p_bulk_bind.Bulk_Insert_Pre_10g (i_make_sparse => true)
```



```

-- this code is located in the Bulk_Insert_Pre_10g procedure
...

```

```
forall j in g_emp_recs.First()..g_emp_recs.Last()
```

```
save exceptions
```

```
insert into new_employees values g_emp_recs(j);
```

```
exception when bulk_errors then
```

```
for j in 1..sql%bulk_exceptions.Count()
```

```
loop
```

```
Dbms_Output.Put_Line ( 'Error from element #' ||
```

```
To_Char(sql%bulk_exceptions(j).error_index) || ': ' ||
```

```
Sqlerrm(SQL%bulk_exceptions(j).error_code) );
```

```
end loop;
```

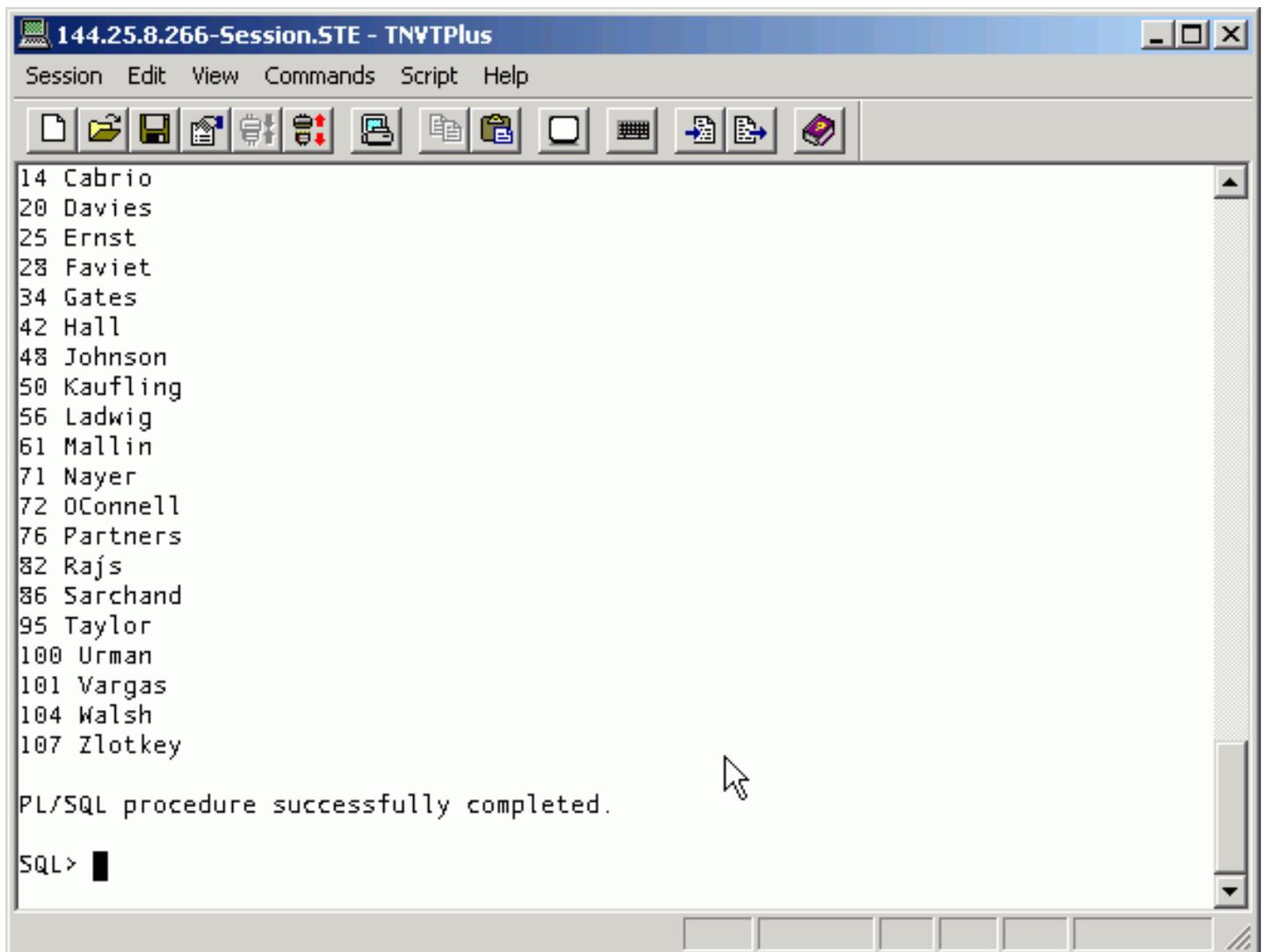
```
...
```

From the output, you see that each missing record has generated an exception ( `ORA-22160` ). As mentioned earlier, even though the remaining records are processed via `SAVE EXCEPTIONS` syntax, this results in a performance degradation if the number of deleted records is fairly high.

4. The next step is to use the Oracle Database 10g `INDICES OF` syntax to ignore the missing records and thereby obtain much better performance. Execute the following commands:

```
exec p_bulk_bind.Bulk_Insert_With_Indices_Of -  
    (i_cause_exception => false)
```

In this case the Boolean input value is used to ignore a block of code responsible for creating an exception condition as you will see in the next step.



The screenshot shows a SQL\*Plus session window titled "144.25.8.266-Session.STE - TNYTPlus". The window has a menu bar with "Session", "Edit", "View", "Commands", "Script", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window displays the following text:

```
14 Cabrio  
20 Davies  
25 Ernst  
28 Faviet  
34 Gates  
42 Hall  
48 Johnson  
50 Kaufling  
56 Ladwig  
61 Mallin  
71 Nayer  
72 OConnell  
76 Partners  
82 Rajs  
86 Sarchand  
95 Taylor  
100 Urman  
101 Vargas  
104 Walsh  
107 Zlotkey  
  
PL/SQL procedure successfully completed.  
  
SQL> █
```

```
-- this code is located in the
Bulk_Insert_With_Indices_Of

-- procedure

...

forall j in
indices of
  g_emp_recs
save exceptions
  insert into new_employees values g_emp_recs(j);

...
```

This time there are no exceptions raised due to the missing records.

5. The above procedure can also be run with the input parameter set to true. This would cause an exception while inserting which must be trapped (by `SAVE EXCEPTIONS`) and later processed. The condition causing the exception is simulated via a procedure call, which nulls the 'email' columns for the employees with the last name of 'Ernst' and 'Urman'. Since the `EMAIL` column has a `NOT NULL` restriction, this causes an exception when the two corresponding records are inserted. The exceptions are later processed from the `SQL%BULK_EXCEPTIONS` collection. Execute the following commands:

```
exec p_bulk_bind.Bulk_Insert_With_Indices_Of -
  (i_cause_exception => true)
```

The screenshot shows a window titled "144.25.8.266-Session.STE - TNVTPlus". The menu bar includes "Session", "Edit", "View", "Commands", "Script", and "Help". The toolbar contains icons for file operations (new, open, save, print, copy, paste) and execution (run, stop, refresh). The main text area displays the following output:

```

48 Johnson
50 Kaufling
56 Ladwig
61 Mallin
71 Nayer
72 OConnell
76 Partners
82 Raj's
86 Sarchand
95 Taylor
100 Urman
101 Vargas
104 Walsh
107 Zlotkey
Error on the 5th iteration
last_name for error element: Ernst
Error was: ORA-01400: cannot insert NULL into ()
Error on the 19th iteration
last_name for error element: Urman
Error was: ORA-01400: cannot insert NULL into ()

PL/SQL procedure successfully completed.

SQL> █

```

```

-- this code is located in the exception area
-- in the
Bulk_Insert_With_Indices_Of
procedure

...

forall j in
indices of
g_emp_recs
-- between g_emp_recs.First() and g_emp_recs.Last()
-- optional

save exceptions
insert into new_employees values g_emp_recs(j);

exception when bulk_errors then
declare
v_iteration pls_integer;

```

```

        n pls_integer;
        k pls_integer;
    begin
        for j in 1..
sql%bulk_exceptions
        .Count()
        loop
            v_iteration := SQL%bulk_exceptions(j).error_index;
            Dbms_Output.Put_Line (
                'Error on the ' || To_Char(v_iteration)
                || 'th iteration' );
        ...
    
```

Note: The error is reported for the iteration number, this is new to the exception handling in Oracle Database 10g.

- Another approach is to use `VALUES OF` clause to create a second collection of pointers to the elements of the original array. Execute the following commands:

```

exec p_bulk_bind.Bulk_Insert_With_Values_Of -
    (i_null_email => true, i_violate_pk => false)

```

The `Point_To_Sparse` procedure is used to create a collection of pointers based on the same logic as before, that is, it points to the elements with the first occurrence of each letter in the last name. It is called from the procedure `Bulk_Insert_With_Values_Of`.

The screenshot shows a window titled "144.25.8.266-Session.STE - TNVTPlus". The menu bar includes "Session", "Edit", "View", "Commands", "Script", and "Help". The toolbar contains icons for file operations, execution, and help. The main text area displays the following output:

```

48 Johnson
50 Kaufling
56 Ladwig
61 Mallin
71 Nayer
72 OConnell
76 Partners
82 Rajs
86 Sarchand
95 Taylor
100 Urman
101 Vargas
104 Walsh
107 Zlotkey
Error on the 5th iteration
last_name for error element: Ernst
Error was: ORA-01400: cannot insert NULL into ()
Error on the 19th iteration
last_name for error element: Urman
Error was: ORA-01400: cannot insert NULL into ()

PL/SQL procedure successfully completed.

SQL> █

```

-- this code is located in the

Bulk\_Insert\_With\_Values\_Of

-- procedure

...

```

forall j in
values of
  g_values_of_tab
save exceptions
  insert into new_employees values g_emp_recs(j);
...

```

The result is the same as step 5, an exception is raised for the 5th and the 19th iteration.

7. To see the effect of reporting the iteration number (or pointer element number) as opposed to the index of the erroneous record, you can run the above procedure with `i_violate_pk => true`. To simulate the error condition, the procedure `Cause_Exception_For_Values_Of` is used to insert a duplicate index entry from the bound collection ( `G_EMP_RECS` ) into the pointer collection ( `G_VALUES_OF_TAB` ). Due to this, the unique key constraint (created in the first step) is violated and the exception is raised upon bulk insert into the `NEW_EMPLOYEES` table. Execute the following commands:

```
exec p_bulk_bind.Bulk_Insert_With_Values_Of -
      (i_null_email => false, i_violate_pk => true)
```

```

144.25.8.266-Session.STE - TNVTPlus
Session Edit View Commands Script Help
last_name for error element: Taylor
Error was: ORA-00001: unique constraint (.) violated
Error on the 40th iteration
last_name for error element: Urman
Error was: ORA-00001: unique constraint (.) violated
Error on the 41th iteration
last_name for error element: Vargas
Error was: ORA-00001: unique constraint (.) violated
Error on the 42th iteration
last_name for error element: Walsh
Error was: ORA-00001: unique constraint (.) violated
Error on the 43th iteration
last_name for error element: Zlotkey
Error was: ORA-00001: unique constraint (.) violated
Error on the 44th iteration
last_name for error element: Ernst
Error was: ORA-00001: unique constraint (.) violated
Error on the 45th iteration
last_name for error element: Urman
Error was: ORA-00001: unique constraint (.) violated

PL/SQL procedure successfully completed.

SQL> █

```

-- this code is located in the

`Bulk_Insert_With_Values_Of`

```

-- procedure

...

exception when bulk_errors then
  declare
    v_iteration pls_integer;
  begin
    for j in 1..sql%bulk_exceptions.Count()
    loop
      v_iteration := SQL%bulk_exceptions(j).error_index;
      Dbms_Output.Put_Line (
        'Error on the ' || To_Char(v_iteration) ||
        'th iteration' );

-- Find the index of the offending element
-- from the iteration number
      Dbms_Output.Put_Line (
        'last_name for error element: ' ||
        g_emp_recs(g_values_of_tab(v_iteration)).last_name );

      Dbms_Output.Put_Line (
        'Error was: ' ||
        Sqlerrm(SQL%bulk_exceptions(j).error_code) );
    end loop;
  end;

...

```

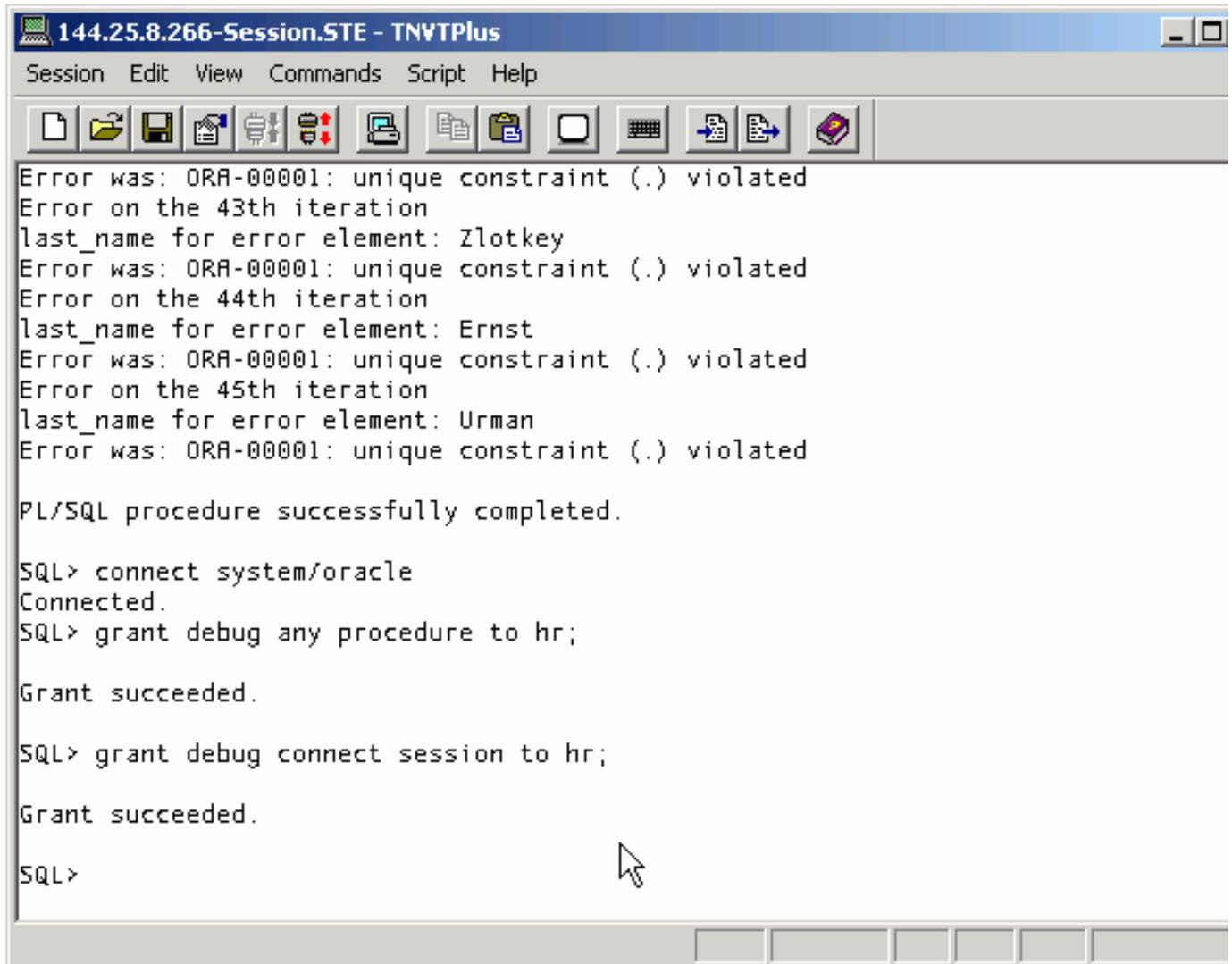
## Setting Privileges and Parameters for Debugging

[Back to List](#)

Before you can use the PL/SQL debug capability within JDeveloper you need make sure the HR user has the appropriate privileges. Perform the following steps:

1. Open a terminal window and execute the following:

```
cd wkdir
sqlplus system/<password>
grant debug any procedure to hr;
grant debug connect session to hr;
```



```
144.25.8.266-Session.STE - TNVTPlus
Session Edit View Commands Script Help
Error was: ORA-00001: unique constraint (.) violated
Error on the 43th iteration
last_name for error element: Zlotkey
Error was: ORA-00001: unique constraint (.) violated
Error on the 44th iteration
last_name for error element: Ernst
Error was: ORA-00001: unique constraint (.) violated
Error on the 45th iteration
last_name for error element: Urman
Error was: ORA-00001: unique constraint (.) violated

PL/SQL procedure successfully completed.

SQL> connect system/oracle
Connected.
SQL> grant debug any procedure to hr;

Grant succeeded.

SQL> grant debug connect session to hr;

Grant succeeded.

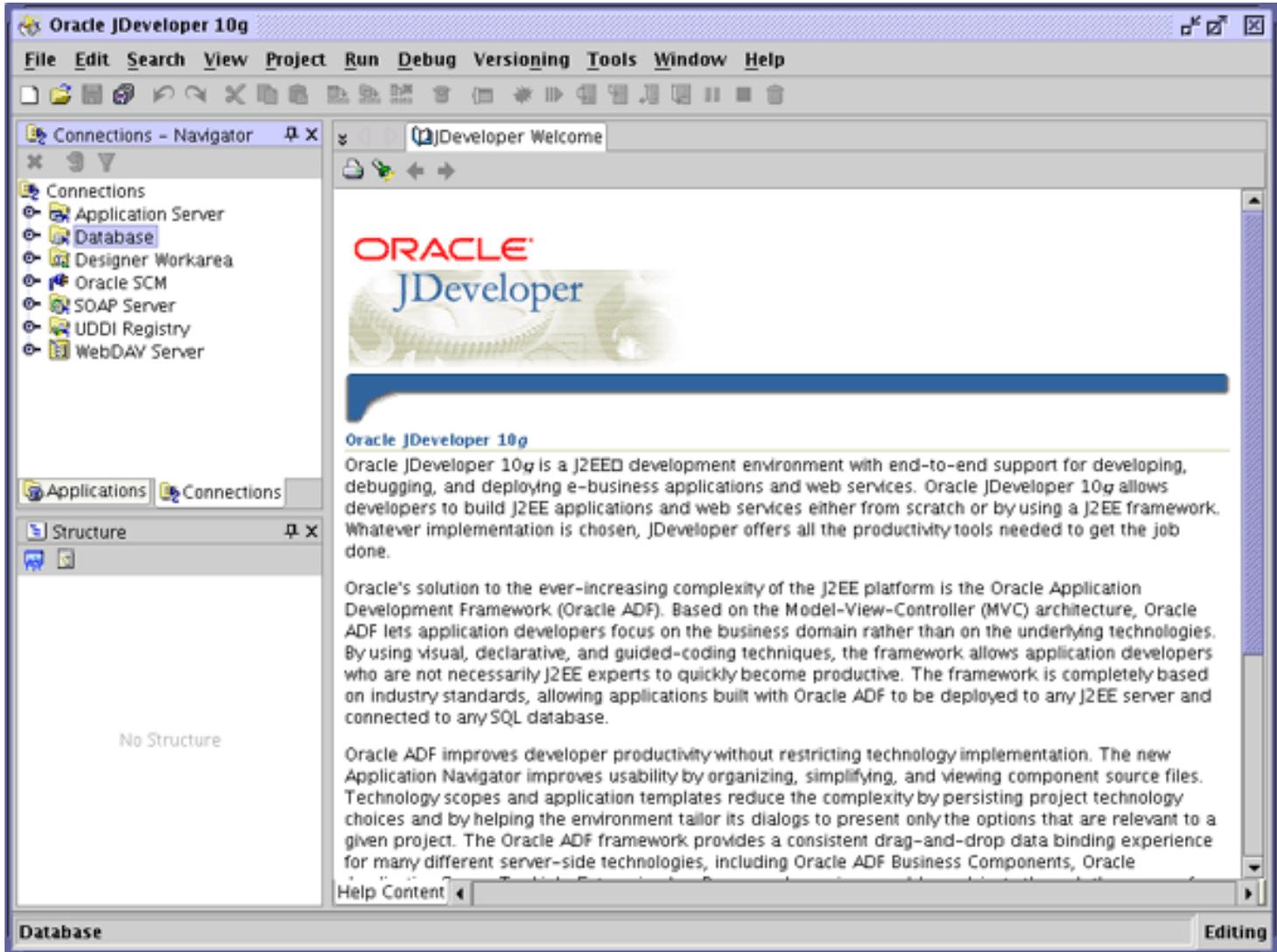
SQL>
```

## Creating a Database Connection

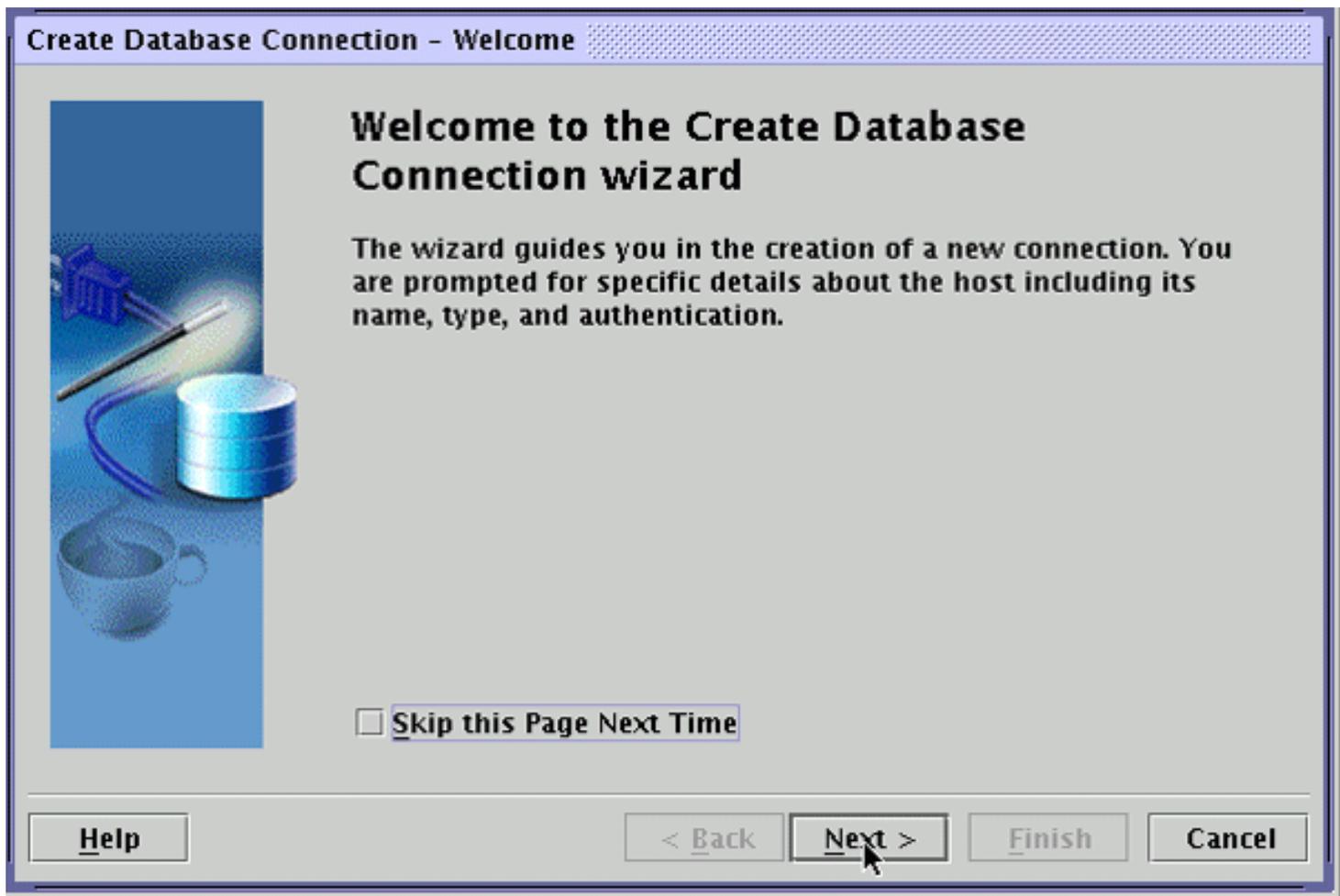
[Back to List](#)

In order to access the PL/SQL, you first need to create a Database Connection. Perform the following steps:

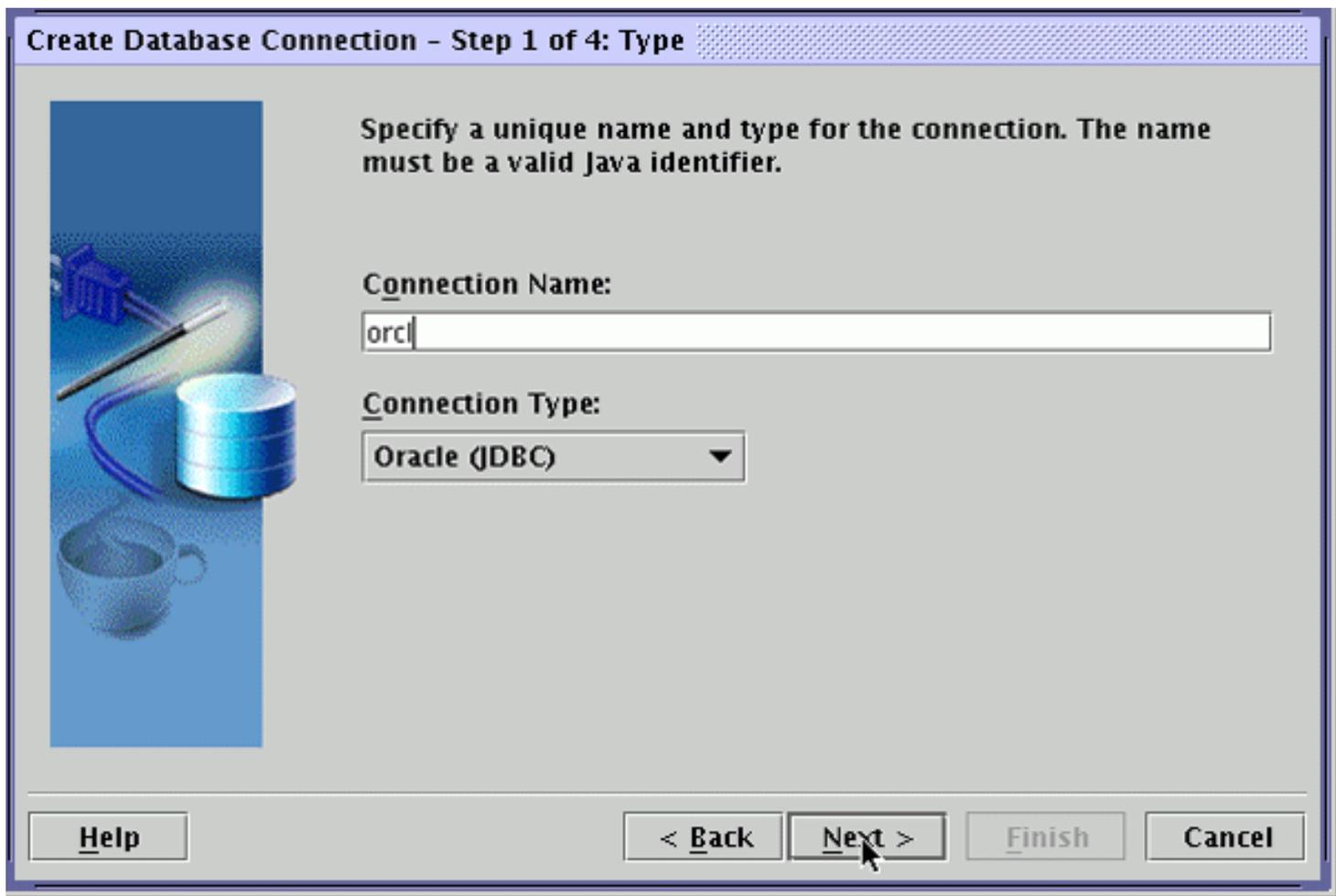
1. Click on the JDeveloper icon you created in your panel.
2. Select the **Connections** tab in the Applications - Navigation window. Then right click on **Database** and select **New Database Connection** .



3. At the Welcome window, click **Next** .



4. Enter **orcl** for the Connection name and click **Next** .



5. Enter **hr** for the username and password and click **Next** .

**Create Database Connection - Step 2 of 4: Authentication**

Specify a username and password to authenticate the connection. To bypass authentication at runtime, select Deploy Password.

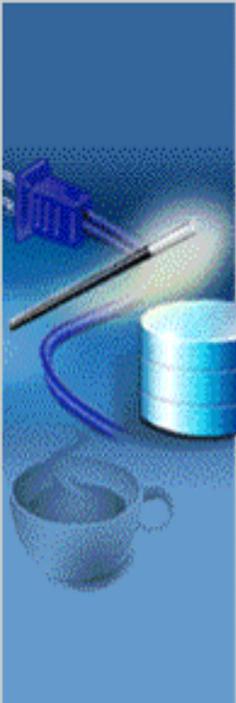
**Username:**  
hr

**Password:**  
\*\*

**Role:**

**Deploy Password**

**Help**      **< Back**      **Next >**      **Finish**      **Cancel**



6. The connection details for your database are shown. Accept the defaults and click **Next** .

**Create Database Connection - Step 3 of 4: Connection**

Specify connection details for the database machine. The database administrator should be able to provide you with this information.

**Driver:** thin

**Host Name:** localhost

**JDBC Port:** 1521

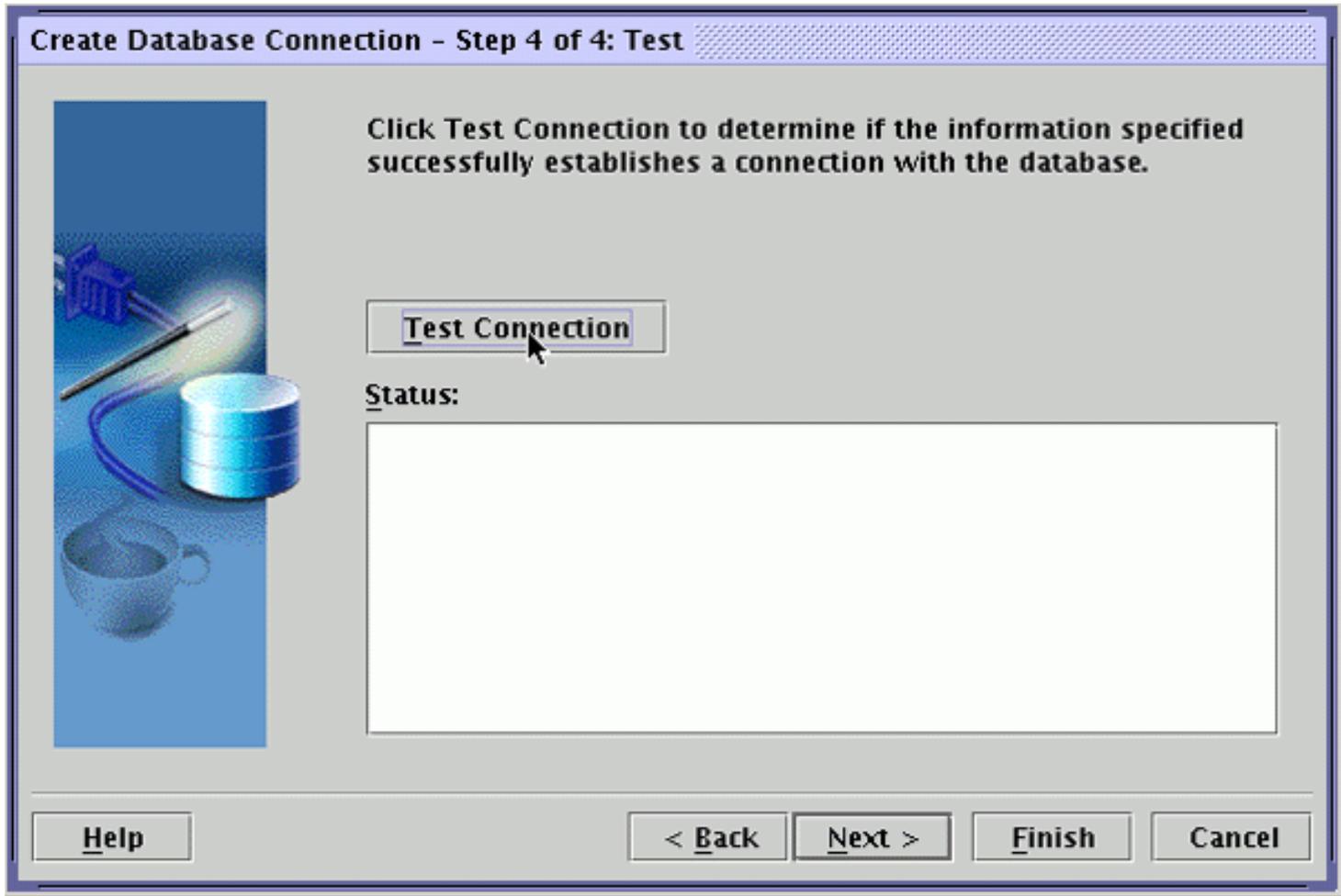
**SID:** ORCL

Enter Custom JDBC URL:

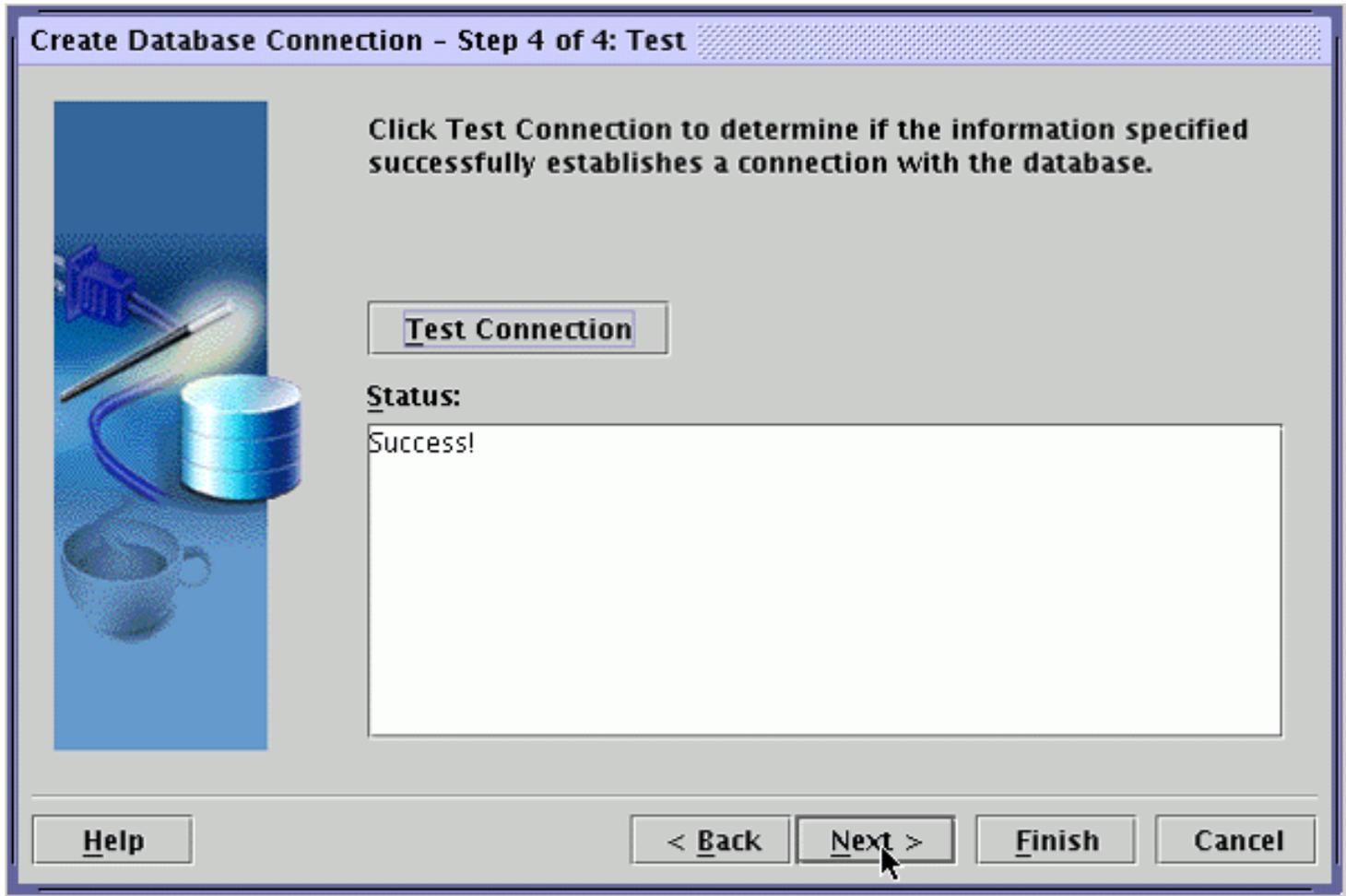
Help < Back Next > Finish Cancel



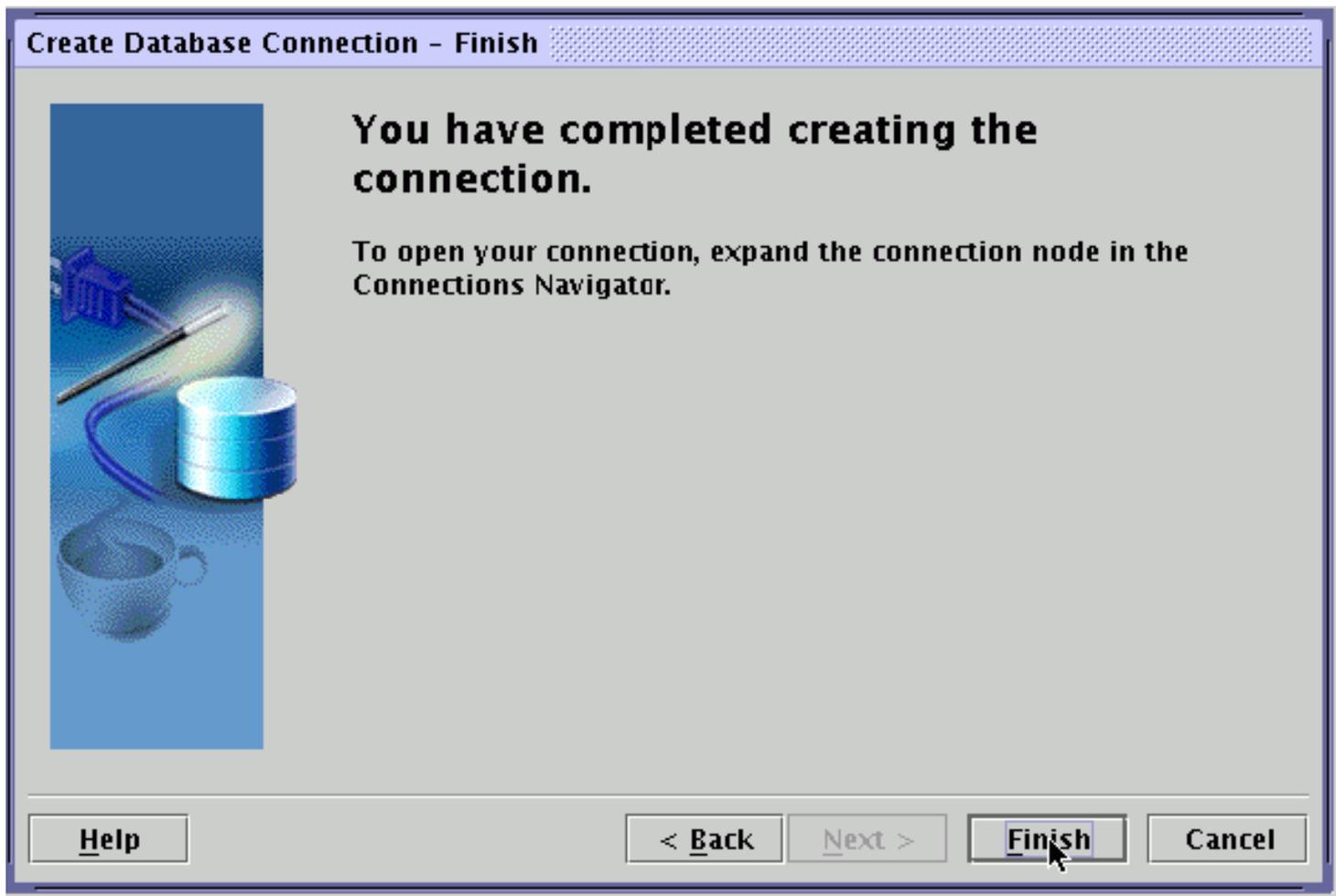
7. Test your connection. Click **Test Connection** .



8. When you receive the Success! message, click **Next** .



9. Click **Finish** to create the connection.

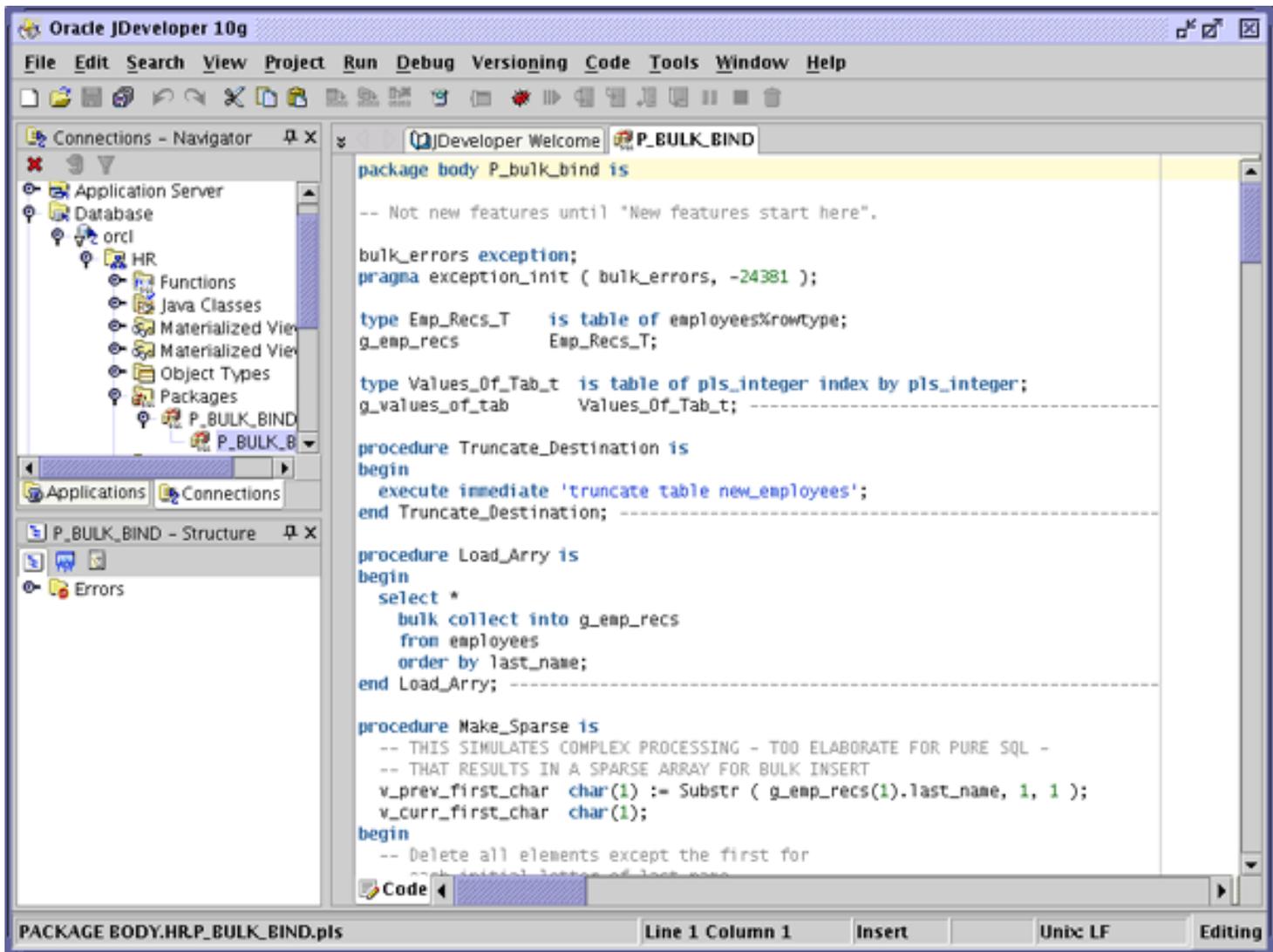


## Debugging a PL/SQL Package Body Using JDeveloper

[Back to List](#)

You will run the PL/SQL package `P_BULK_BIND` you used previously using the JDeveloper PL/SQL Debugger. Perform the following:

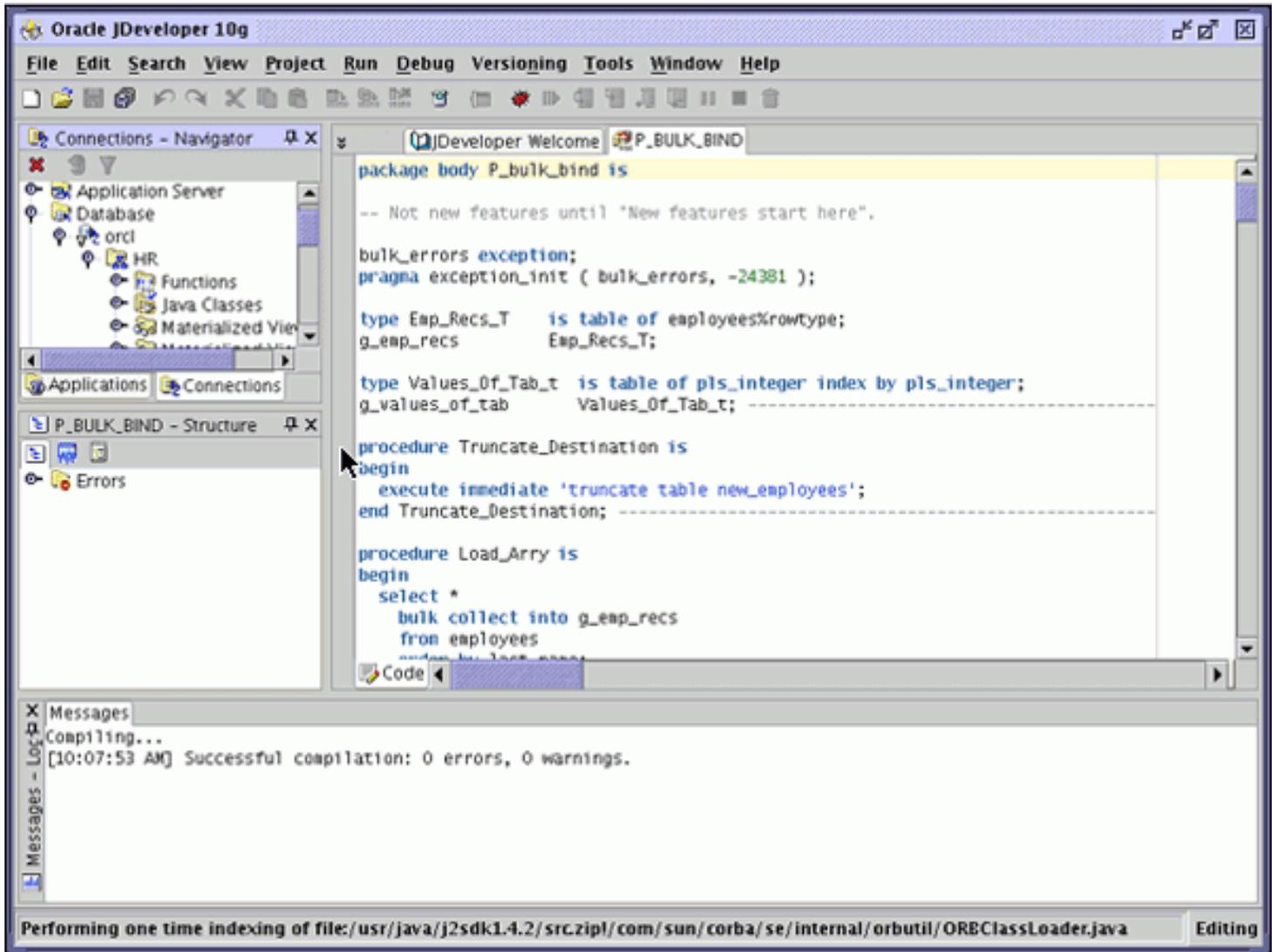
1. From the Connections - Navigator window, expand **Connections -> Database -> orcl -> HR -> Packages** and then right click on the **P\_BULK\_BIND** package body and select **Open** to view the PL/SQL code.



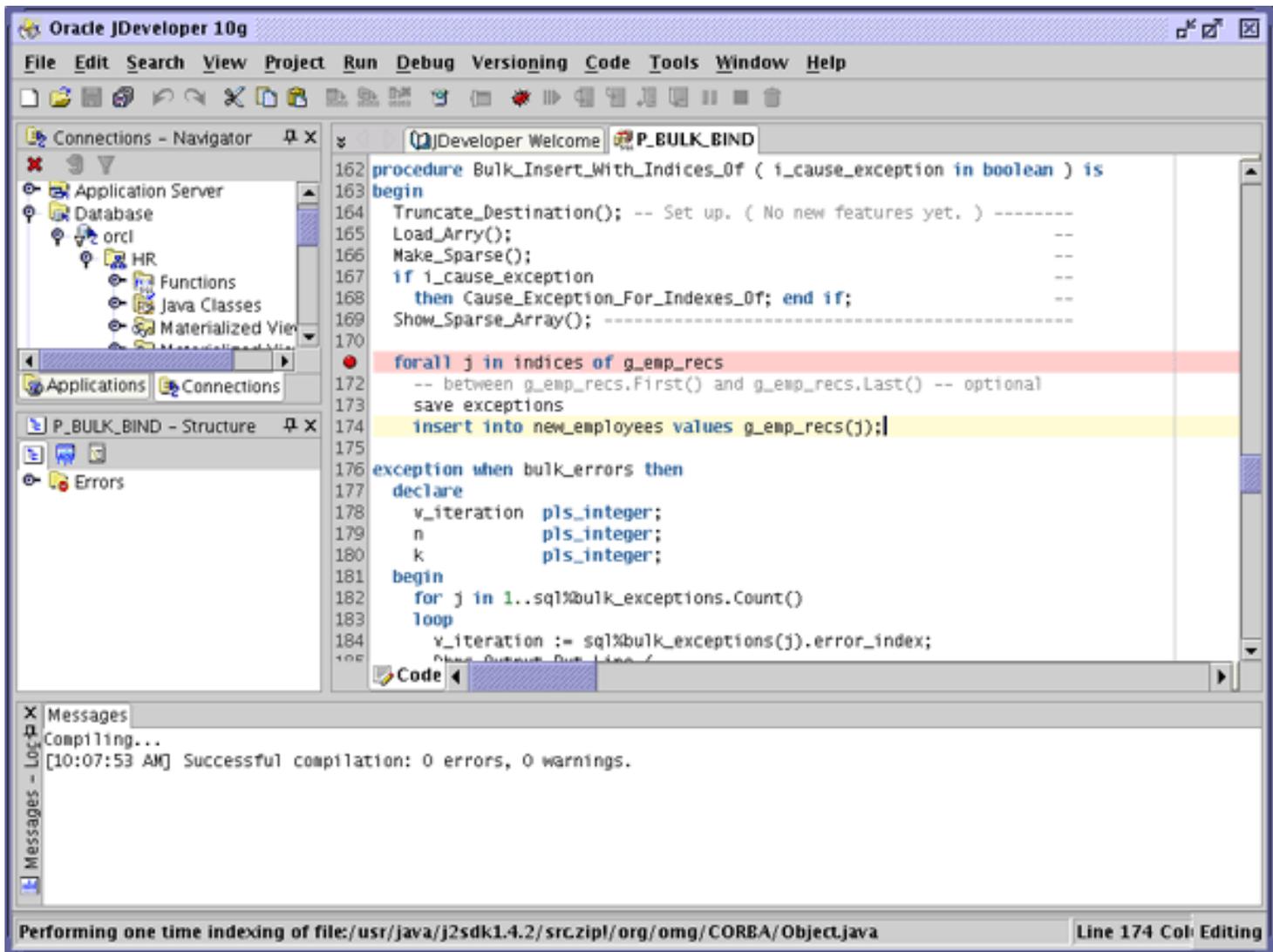
- You need to compile this package body. Right click on the **P\_BULK\_BIND** package body again and select **Make P\_BULK\_BIND**.



3. Right click in the Code Editor gutter and select **Toggle Line Numbers** .



4. To set the breakpoint, scroll down to the statement `forall j in indices of g_emp_rec` (approximately at line 171) and click on the line number.



5. To start debugging, right click in the Code Editor window and select **Debug P\_BULK\_BIND**.

 <b>T<u>o</u>ggle B<u>o</u>okmark</b>	
<b>E<u>x</u>pand T<u>e</u>mplate</b>	Ctrl-Enter
<b>M<u>a</u>ke P_BULK_BIND</b>	Ctrl-M
 <b>R<u>u</u>n P_BULK_BIND</b>	
 <b>D<u>e</u>bug P_BULK_BIND</b>	
<b>T<u>o</u>ggle B<u>r</u>eakpoint</b>	F5
<b>S<u>h</u>ow B<u>r</u>eakpoints Window</b>	
 <b>U<u>n</u>do</b>	Ctrl-Z
 <b>R<u>e</u>do</b>	Ctrl-Y
 <b>C<u>u</u>t</b>	Ctrl-X
 <b>C<u>o</u>py</b>	Ctrl-C
 <b>P<u>a</u>ste</b>	Ctrl-V
<b>S<u>e</u>lect All</b>	Ctrl-A
<b>F<u>i</u>nd in N<u>a</u>avigator</b>	Alt-Home
 <b>C<u>l</u>ose</b>	Shift-F4

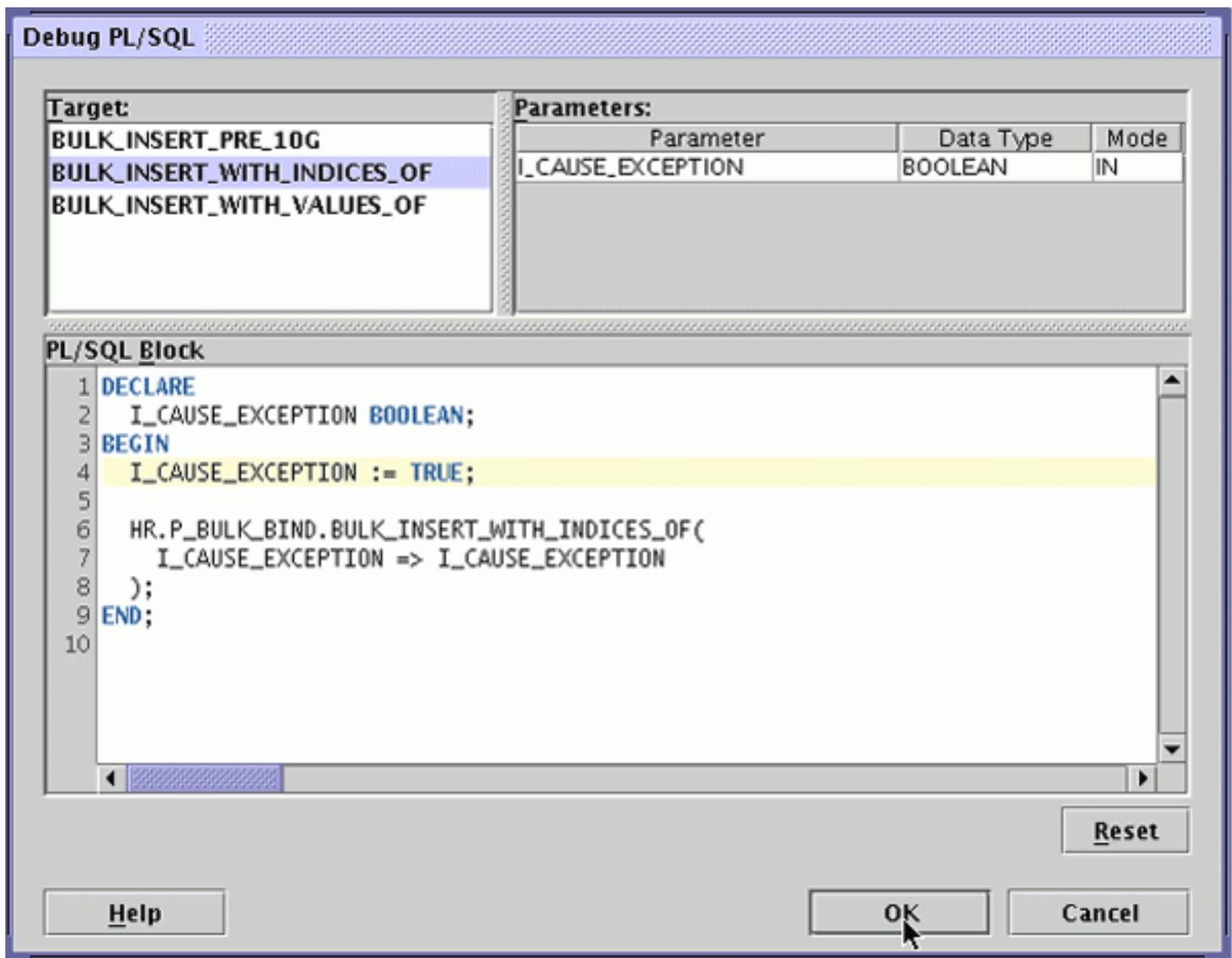
6. This will bring up the Debug PL/SQL window, which displays the three global procedures within the package. Select the procedure `Bulk_Insert_With_Indices_of` . This procedure takes one input parameter, that is a Boolean flag called " `I_CAUSE_EXCEPTION` ". The value for this parameter is to be set to true. In the PL/SQL block area, change

```
I_CAUSE_EXCEPTION := NULL;
```

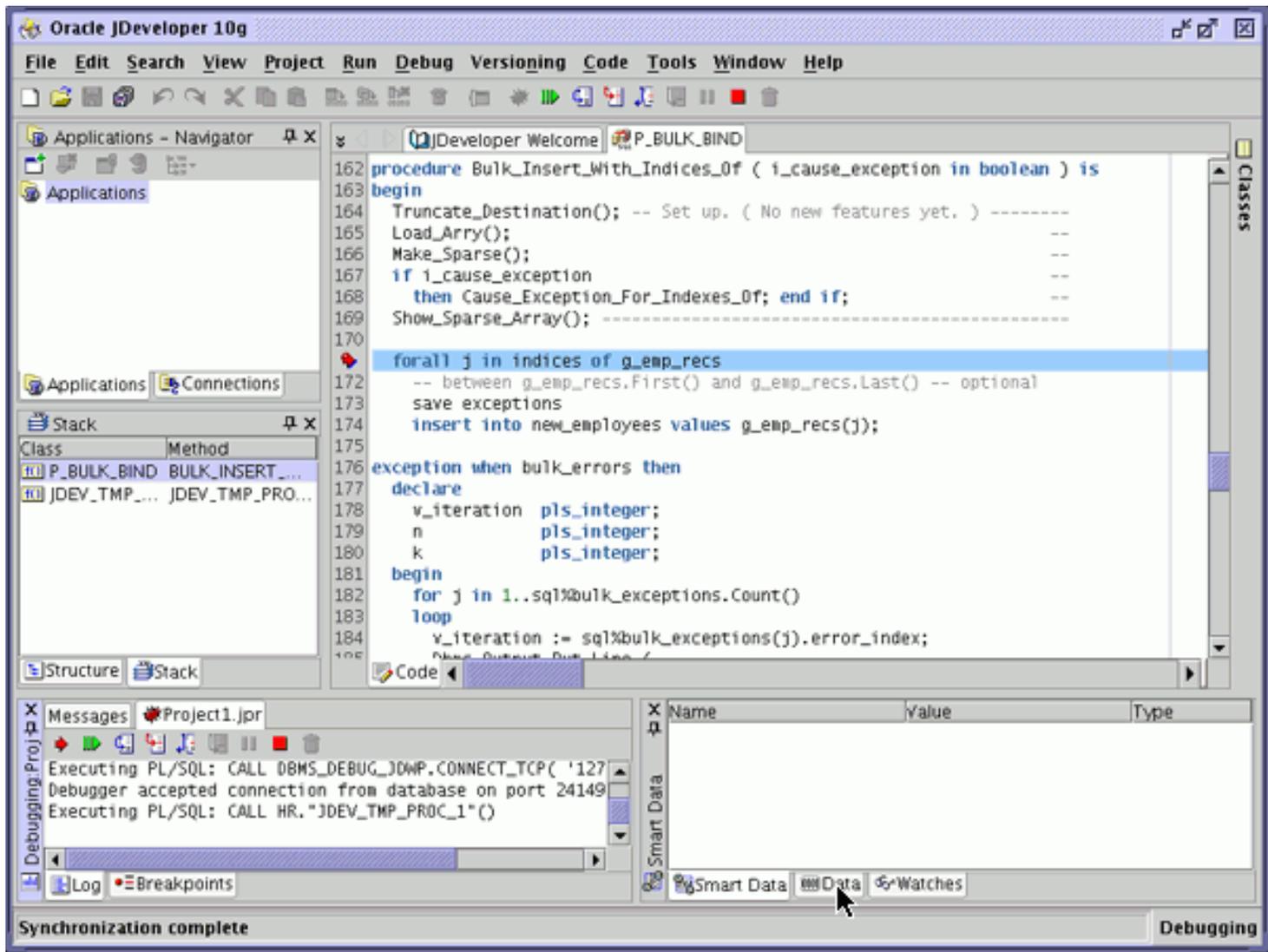
to

```
I_CAUSE_EXCEPTION := TRUE;
```

Then click **OK** .



7. Upon continuing, the debugger executes the statements up to the first breakpoint. At this point, the breakpoint statement (to be executed) next is highlighted. The Stack window displays the procedure being executed. Click the **Data** tab to display the variables and their values, within the current procedure.



8. Expand **Package body variables of P\_BULK\_BIND** . You see two variables - `G_EMP_RECS` and `G_VALUES_OF_TAB` . `G_EMP_RECS` is the PL/SQL table used by the application to bulk insert data into the `NEW_EMPLOYEES` table.

The screenshot shows the Oracle JDeveloper 10g IDE with a PL/SQL procedure being debugged. The main editor displays the code for 'Bulk\_Insert\_With\_Indices\_Of'. The 'Smart Data' window is open, showing the current state of variables: I\_CAUSE\_EXCEPTION (true, BOOLEAN), G\_EMP\_RECS (nested table, EMP\_RECS\_T), and G\_VALUES\_OF\_TAB (indexed table, VALUES\_OF...). The Messages window shows the execution log, including 'Executing PL/SQL: CALL HR.'JDEV\_TMP\_PROC\_1'()'.

```

162 procedure Bulk_Insert_With_Indices_Of ( l_cause_exception in boolean ) is
163 begin
164   Truncate_Destination(); -- Set up. ( No new features yet. ) -----
165   Load_Array();           --
166   Make_Sparse();         --
167   if l_cause_exception
168   then Cause_Exception_For_Indices_Of; end if; --
169   Show_Sparse_Array();  -----
170
171   forall j in indices of g_emp_recs
172     -- between g_emp_recs.First() and g_emp_recs.Last() -- optional
173     save_exceptions
174     insert into new_employees values g_emp_recs(j);
175
176 exception when bulk_errors then
177   declare
178     v_iteration pls_integer;
179     n            pls_integer;
180     k            pls_integer;
181   begin
182     for j in 1..sql%bulk_exceptions.Count()
183     loop

```

Name	Value	Type
I_CAUSE_EXCEPTION	true	BOOLEAN
Package body variable		
G_EMP_RECS	nested table	EMP_RECS_T
G_VALUES_OF_TAB	indexed table	VALUES_OF_...
Package variables of P		

Messages: Project1.jpr  
Connecting to the database orcl.  
Executing PL/SQL: ALTER SESSION SET PLSQL\_DEBUG=TRUE  
Executing PL/SQL: CALL DBMS\_DEBUG\_JDWP.CONNECT\_TCP( '127.0.0.1', 24149 )  
Debugger accepted connection from database on port 24149.  
Executing PL/SQL: CALL HR.'JDEV\_TMP\_PROC\_1'()  
Log Breakpoints

The debuggee process is running. To interact with the Smart Data window, pause the debuggee process. Debugging

- Expand `G_EMP_RECS` -> `_values` -> `[25]` -> `_value` . Notice that `_value` shows the actual record values and the `EMAIL` column is null for this record. To observe the effect, click on the **Resume** button (green button in Main Toolbar).

The screenshot shows the Oracle JDeveloper 10g IDE with a PL/SQL procedure being debugged. The main window displays the code for 'Bulk\_Insert\_With\_Indices\_Of'. The 'Messages' window shows the execution log, and the 'Smart Data' window displays the current state of variables.

**Code Snippet:**

```

162 procedure Bulk_Insert_With_Indices_Of ( i_cause_exception in boolean ) is
163 begin
164   Truncate_Destination(); -- Set up. ( No new Features yet. ) -----
165   Load_Array(); --
166   Make_Sparse(); --
167   if i_cause_exception
168   then Cause_Exception_For_Indices_Of; end if; --
169   Show_Sparse_Array(); -----
170
171   forall j in indices of g_emp_rec
172     -- between g_emp_rec.First() and g_emp_rec.Last() -- optional
173     save_exceptions
174     insert into new_employees values g_emp_rec(j);
175
176   exception when bulk_errors then
177     declare
178       v_iteration pls_integer;
179       n pls_integer;
180       k pls_integer;
181     begin
182       for j in 1..sql%bulk_exceptions.Count()
183     loop

```

**Messages Window:**

```

Connecting to the database orcl.
Executing PL/SQL: ALTER SESSION SET PLSQL_DEBUG=TRUE
Executing PL/SQL: CALL DBMS_DEBUG_JDWP.CONNECT_TCP( '127.0.0.1', 24149 )
Debugger accepted connection from database on port 24149.
Executing PL/SQL: CALL HR."JDEV_TMP_PROC_1"()

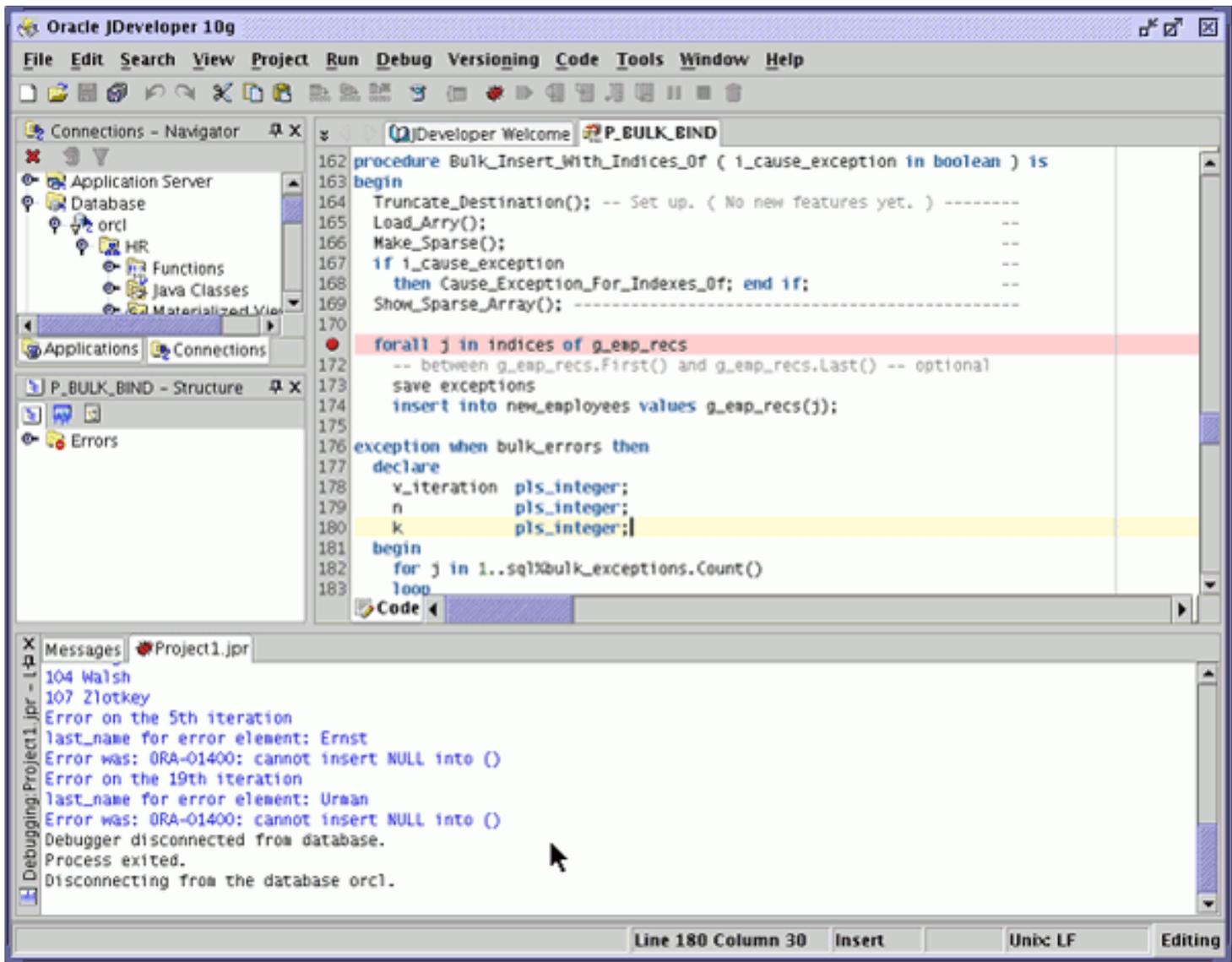
```

**Smart Data Window:**

Name	Value	Type
._key	25	PLS_INT...
._value	'25'	String
._canBeNull	true	boolean
._minValue	0	int
._maxValue	0	int
._value		Rowtype
EMPLOYEE_ID	104	NUMBER...
FIRST_NAME	'Bruce'	VARCHA...
LAST_NAME	'Ernst'	VARCHA...
EMAIL	NULL	VARCHA...
PHONE_NUMBER	500-432-4567	VARCHA...

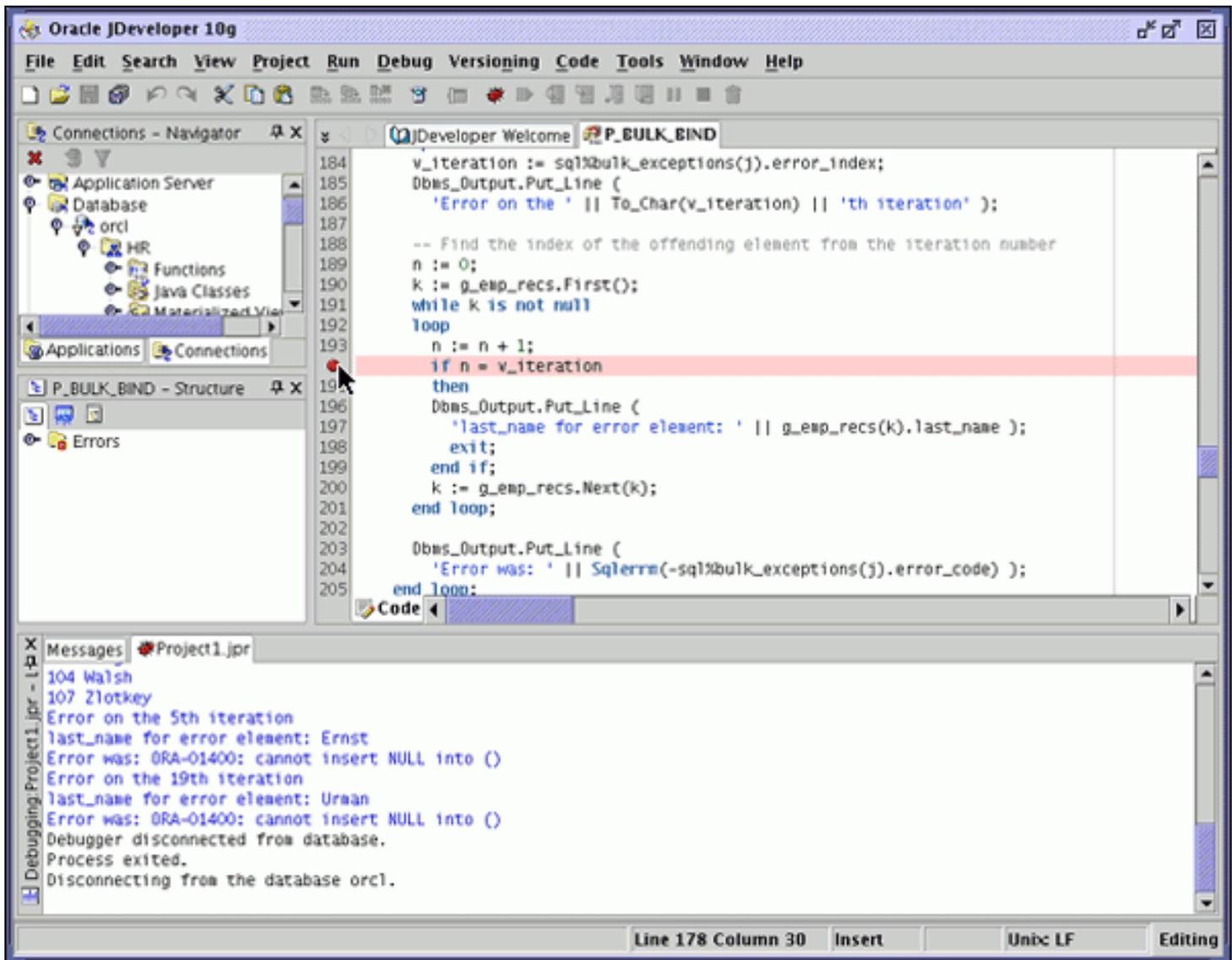
The debuggee process is running. To interact with the Smart Data window, pause the debuggee process. Debugging

10. The procedure is now executed to the end and the "Log" window displays program output at the end of which are the error messages for the 2 records for which insert failed.



11.

To debug the problem, you can start the debugger with a conditional breakpoint. The goal is to halt execution in the exception block for the erroneous record. For this, set a breakpoint at the line `if n = v_iteration`. Click the line number **194**.

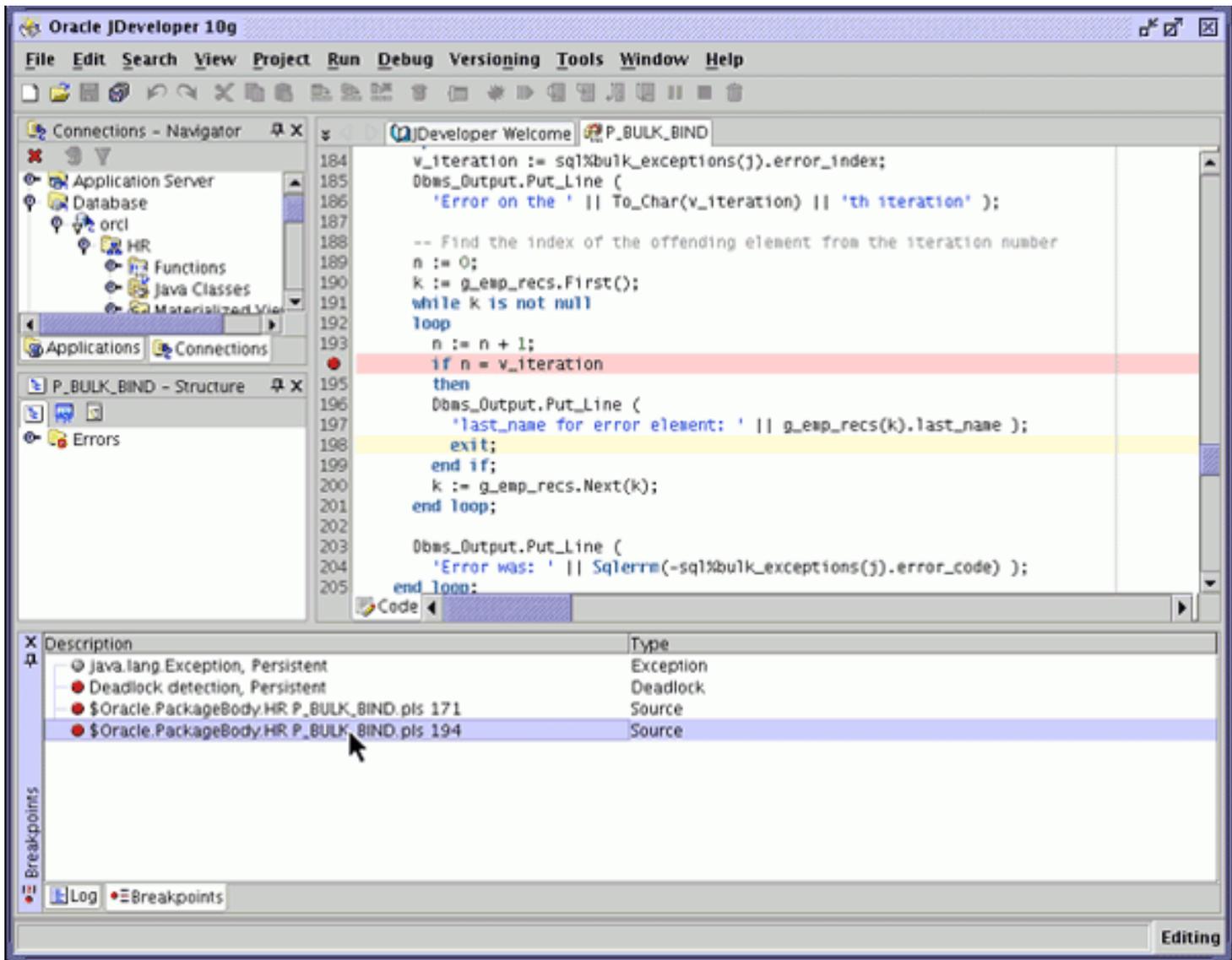


Right click and select **Show BreakPoints Window** .

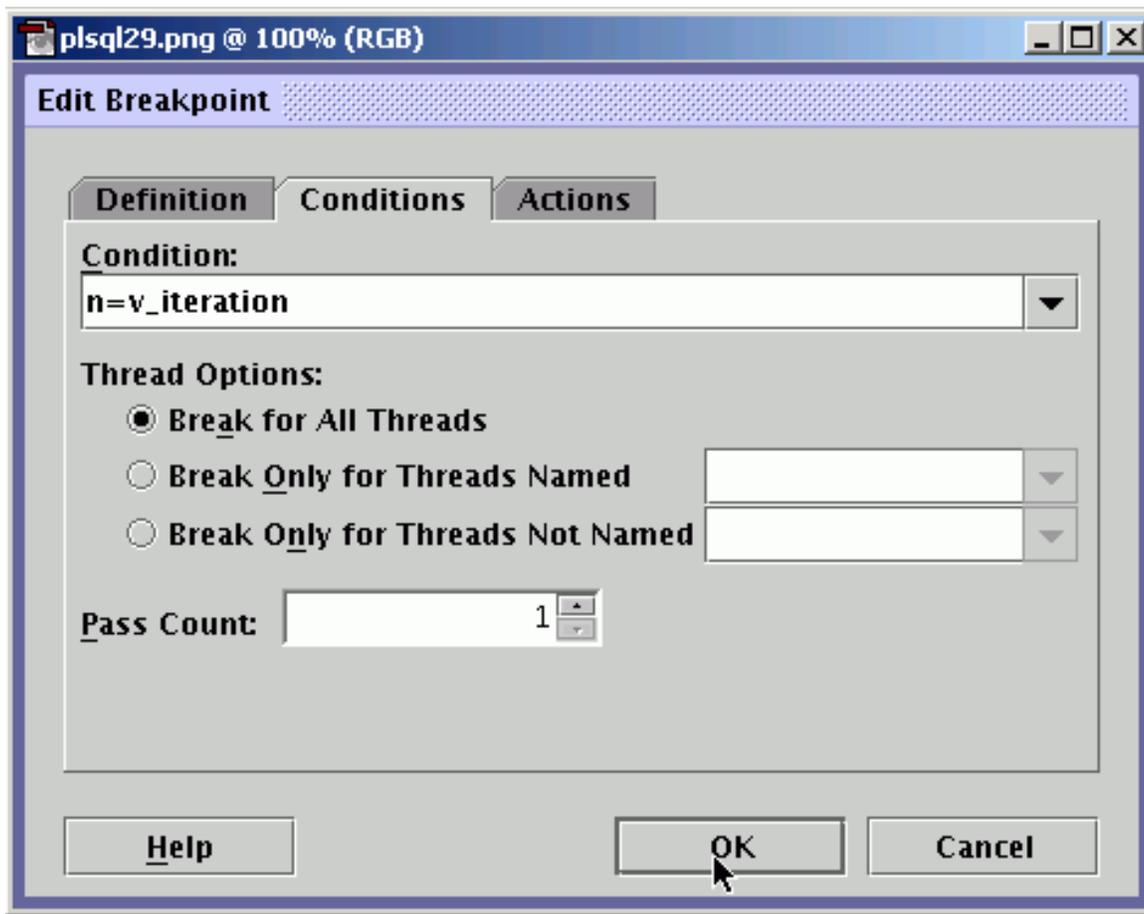
12.

 <b>Toggle Bookmark</b>	
<b>Expand Template</b>	Ctrl-Enter
<b>Make P_BULK_BIND</b>	Ctrl-M
 <b>Run P_BULK_BIND</b>	
 <b>Debug P_BULK_BIND</b>	
<b>Toggle Breakpoint</b>	F5
<b>Show Breakpoints Window</b>	
 <b>Undo</b>	Ctrl-Z
 <b>Redo</b>	Ctrl-Y
 <b>Cut</b>	Ctrl-X
 <b>Copy</b>	Ctrl-C
 <b>Paste</b>	Ctrl-V
<b>Select All</b>	Ctrl-A
<b>Find in Navigator</b>	Alt-Home
 <b>Close</b>	Shift-F4

13. Right click on the breakpoint `$Oracle.PackageBody.HR.P_BULK_BIND.pls 194` , right click and select **Edit** .

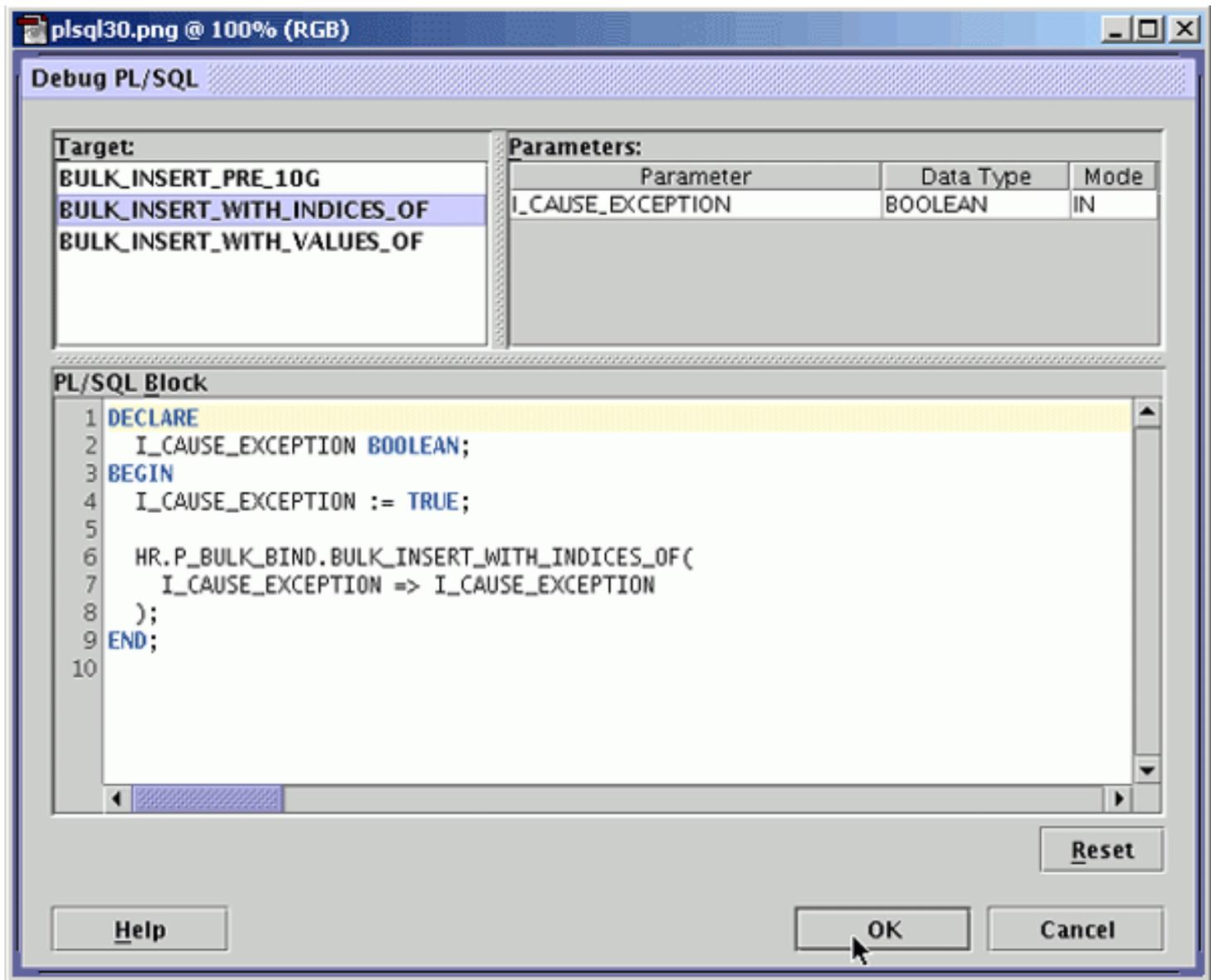


14. Select the **Conditions** tab. In the Condition: field enter the condition `n = v_iteration` . Only when this condition is evaluated to true, the debugger stops at the breakpoint. Then click **OK** .



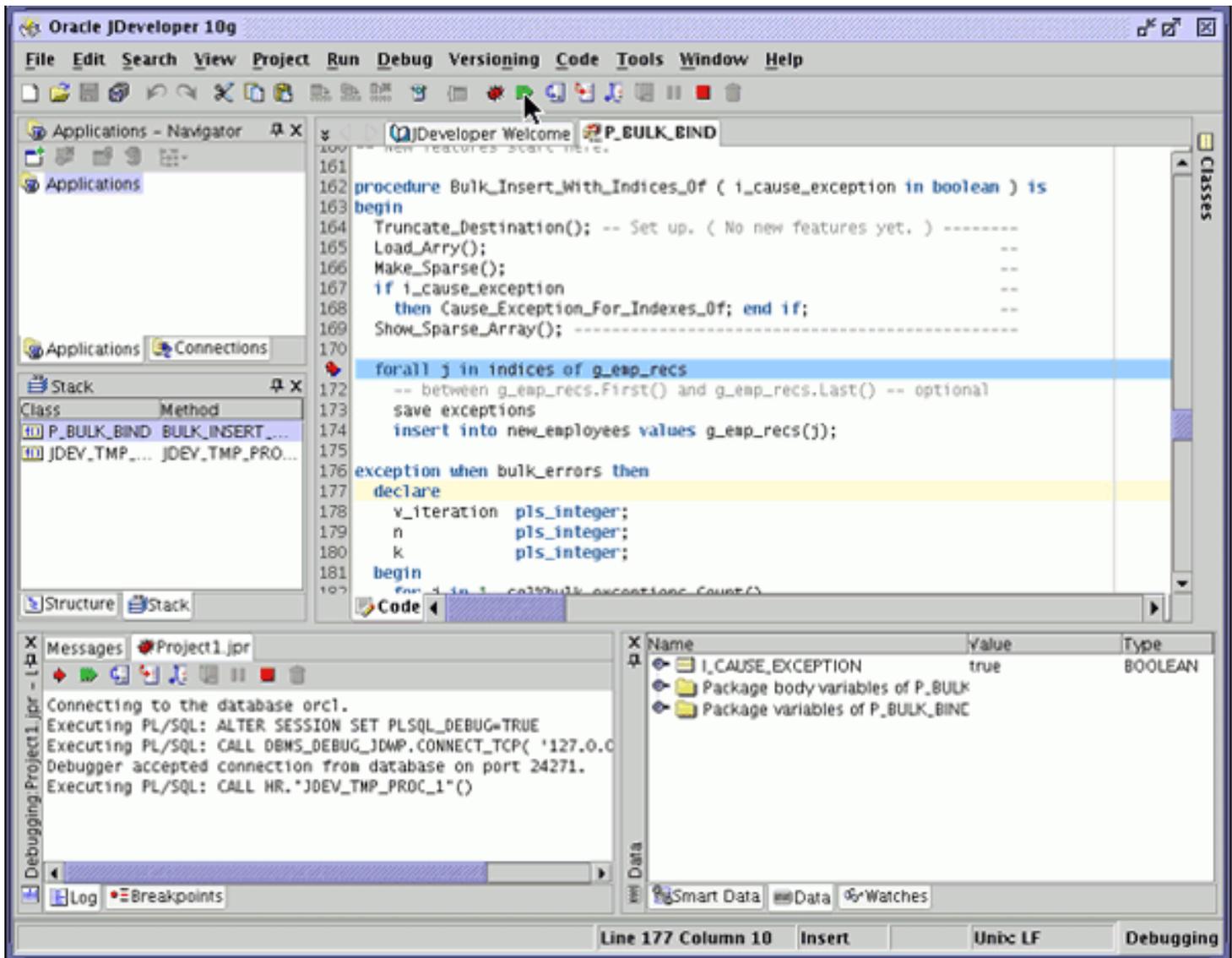
15.

Start the debugger again. Right click in the Code Editor window and select **Debug P\_BULK\_BIND** . At the Debug PL/SQL window, select the procedure **Bulk\_Insert\_With\_Indices\_Of** . **I\_CAUSE\_EXCEPTION** should still be set to **TRUE** . Click **OK** .



The execution halts when the erroneous record is reached (while processing the exceptions). Click **Resume**.

16.



17.

The second breakpoint is reached. The exception is raised for the 5th iteration, which is record index [25]. Upon expanding the record, the 'email' value for this record is null. Click **Resume**.

The screenshot shows the Oracle JDeveloper 10g IDE with a PL/SQL program being debugged. The code is as follows:

```

184     v_iteration := sql%bulk_exceptions(j).error_index;
185     Dbms_Output.Put_Line (
186       'Error on the ' || To_Char(v_iteration) || 'th iteration' );
187
188     -- Find the index of the offending element from the iteration number
189     n := 0;
190     k := g_emp_recs.First();
191     while k is not null
192     loop
193         n := n + 1;
194         if n = v_iteration
195         then
196             Dbms_Output.Put_Line (
197               'last_name for error element: ' || g_emp_recs(k).last_name );
198             exit;
199         end if;
200         k := g_emp_recs.Next(k);
201     end loop;
202
203     Dbms_Output.Put_Line (
204       'Error was: ' || Sqlerrm(-sql%bulk_exceptions(j).error_code) );
205 end loop;

```

The debugger is paused at line 195. The Messages window shows the following execution steps:

```

Connecting to the database orcl.
Executing PL/SQL: ALTER SESSION SET PLSQL_DEBUG=TRUE
Executing PL/SQL: CALL DBMS_DEBUG_JDWP.CONNECT_TCP( '127.0.0.1', 24271)
Debugger accepted connection from database on port 24271.
Executing PL/SQL: CALL HR.'JDEV_TMP_PROC_1'()

```

The Data window shows the current state of variables:

Name	Value	Type
I_CAUSE_EXCEPTION	true	BOOLEAN
V_ITERATION	5	PLS_INT...
_value	*5*	String
_canBeNull	true	boolean
_minValue	0	int
_maxValue	0	int
N	5	PLS_INT...
K	25	PLS_INT...
J	1	PLS_INT...
Package body variables of P_BUL		
Package variables of P_BULK_BIND		

The results are displayed.

18.

The screenshot displays the Oracle JDeveloper 10g IDE interface. The main window shows a PL/SQL program named `P_BULK_BIND` with a breakpoint set at line 194. The code is as follows:

```
184 v_iteration := sql%bulk_exceptions(j).error_index;
185 Dbms_Output.Put_Line (
186 'Error on the ' || To_Char(v_iteration) || 'th iteration' );
187
188 -- Find the index of the offending element from the iteration number
189 n := 0;
190 k := g_emp_recs.First();
191 while k is not null
192 loop
193 n := n + 1;
194 if n = v_iteration
195 then
196 Dbms_Output.Put_Line (
197 'last_name for error element: ' || g_emp_recs(k).last_name );
198 exit;
199 end if;
200 k := g_emp_recs.Next(k);
201 end loop;
202
203 Dbms_Output.Put_Line (
204 'Error was: ' || Sqlerrm(-sql%bulk_exceptions(j).error_code) );
205
```

The Messages window at the bottom shows the following output:

```
Messages
Project1.jpr
107 Zlotkey
Error on the 5th iteration
last_name for error element: Ernst
Error was: ORA-01400: cannot insert NULL into ()
Error on the 19th iteration
last_name for error element: Uraan
Error was: ORA-01400: cannot insert NULL into ()
Debugger disconnected from database.
Process exited.
Disconnecting from the database orcl.
```

The status bar at the bottom indicates the current position is Line 194, Column 1, in Insert mode, with Unix LF line endings, and the editor is in Editing state.