



Control Complexity with Collections

Travis R. Rogers

Truppenbi LLC

travis@jerseyrogers.com

Session Goals

1. Define PL/SQL Collections.
2. Define Oracle Objects (collections).
3. Effects on the Database.
4. Use and Interaction.
5. Debugging hints.
6. Complex system, simple interface.

About Performance

Prelude

The 80/20 Rule

- 20% of the time ultimate performance must be achieved
- 80% of the time decent performance is acceptable.

Another 80/20 Rule

- 20% of the effort is required to get decent performance.
- 80% of the effort is required to get ultimate performance.

Why is this important?

- Use the 20% you need for 80% of your job and the other 80% only when required.
- Don't get overwhelmed by all the functionality that exists in order to support 20% of the requirements.
- We'll use this logic later.

What is most important?

SQL

Collections – A History

Part 1

Another (brief) Digression

The "RECORD" will be a significant part of this conversation even though it is not a collection.

In the beginning...

- The PL/SQL Table
 - Index By Table
 - Associative Array
- The Record
 - User Defined Type (UDT)
 - User Defined Record

Then...

- PL/SQL Tables containing Records.
- Subtype

Finally...

- VArrays
- Nested Tables
- Objects
 - Abstract Data Type (ADT)

So Today there are...

- Associative Arrays – A PL/SQL table enhanced with:
 - The ability to contain Records.
 - The ability to be indexed by a string
- Nested Tables – A less capable Associative Array???
- VArray
- Records
- SubTypes

And of course the Object

- Can be **LIKE**:
 - Subtype
 - Record
 - Nested Table
 - VArray
- Can contain:
 - Scalar types
 - Other Objects
- Slices, Dices and Peels

DB Effects

Part 2

DB Effects

MEMORY!!!!!!

Objects vs. Collections

Part 3

Object vs. Collection

- Objects do more
- Objects require more memory
- Objects are fast but are slower than collections
- Objects cannot have or be an Associative Array.
- Objects can be persisted as-is.

Object vs. Collection

- Collections can be used in all bulk operations but Objects can't.
- Objects allow multiple constructors, instance methods, static methods and comparison methods.
- There is so much functionality in and around Objects that they may be ignored.

Object vs. Collection

- Objects and collections can be used in PLSQL.
- Objects cannot have or be an Associative Array.
- Objects are full fledged database citizens which can be seen by PL/SQL and SQL but a collection can only be seen by PL/SQL.
- Yes, I'm sure...let's take a look.

Simple Examples

Part 4

No Collections in SQL



The screenshot shows a window titled "SQL Window - SELECT object_name FROM user_objects WHERE object_type ...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following query:

```
1 SELECT object_name
2 FROM user_objects
3 WHERE object_type = 'TYPE';
4
```

Below the query is a toolbar with various icons for execution and formatting. The "Output" tab is active, showing a table with one column header: "OBJECT_NAME". The table is currently empty.

At the bottom of the window, a status bar displays the time "3:30" and the message "0 rows selected in 0.031 seconds".

No Collections in SQL

```
Program Window - Edit source of package PKG_OF_TYPES@DB  
pkg_of_types  
A Z  
+ Declaration  
1 CREATE OR REPLACE PACKAGE pkg_of_types AS  
2   TYPE tr_date_list IS TABLE OF DATE;  
3  
4   SUBTYPE tr_num IS NUMBER(1);  
5  
6   TYPE tr_num_list IS TABLE OF tr_num;  
7  
8   TYPE tr_record IS RECORD (firstname VARCHAR2(20)  
9                               ,lastname VARCHAR2(20)  
10                              ,rank      NUMBER(1));  
11  
12   TYPE tr_rec_list IS TABLE OF tr_record;  
13 END;
```

& 13:5 File saved successfully

No Collections in SQL



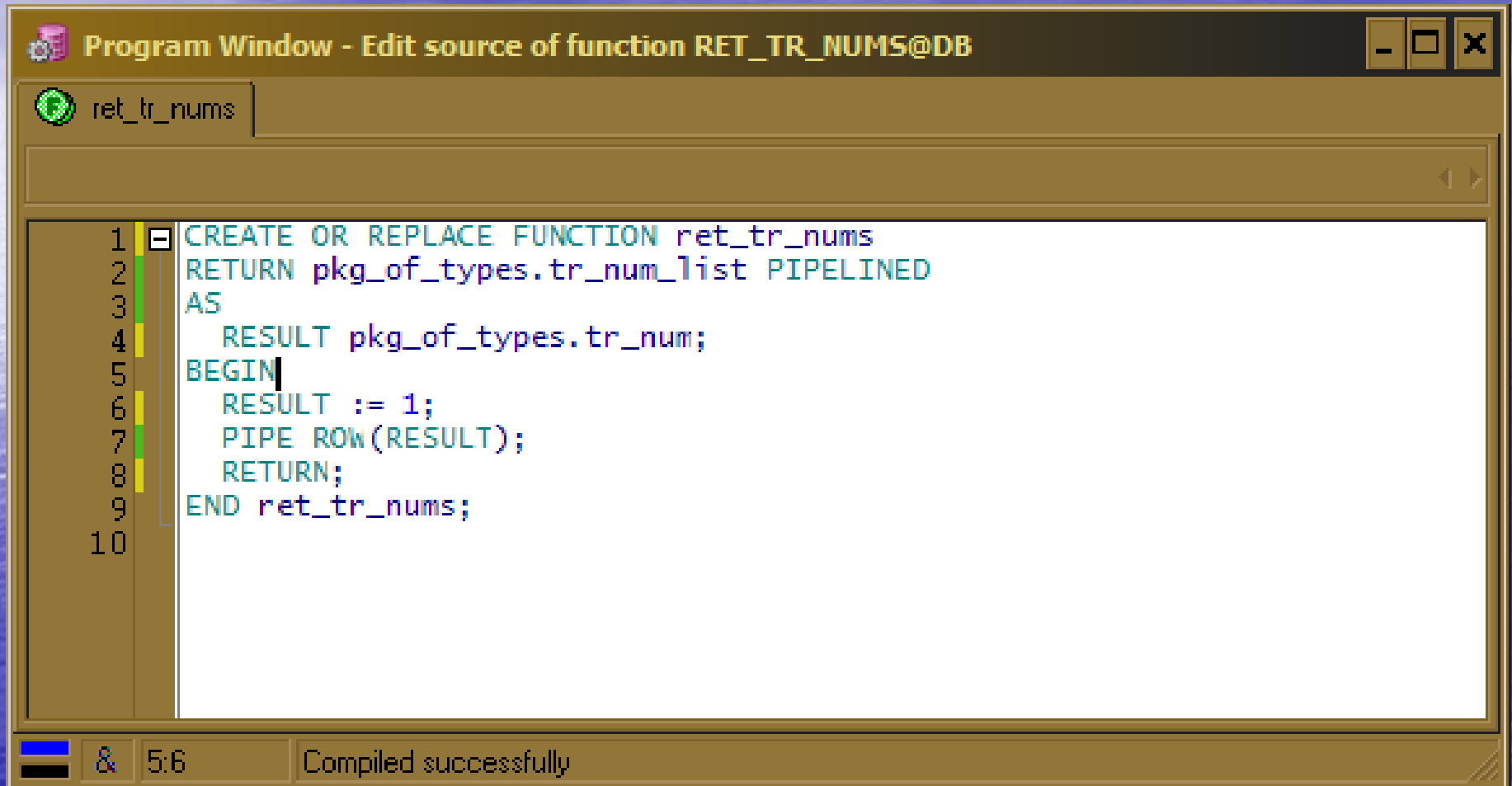
The screenshot shows a window titled "SQL Window - SELECT object_name FROM user_objects WHERE object_type ...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following query:

```
1 SELECT object_name
2 FROM user_objects
3 WHERE object_type = 'TYPE';
4
```

Below the query is a toolbar with various icons for execution and formatting. The "Output" tab is active, showing a table with one column header: "OBJECT_NAME". The table is currently empty.

At the bottom of the window, a status bar displays the time "3:30" and the message "0 rows selected in 0.031 seconds".

No Collections in SQL

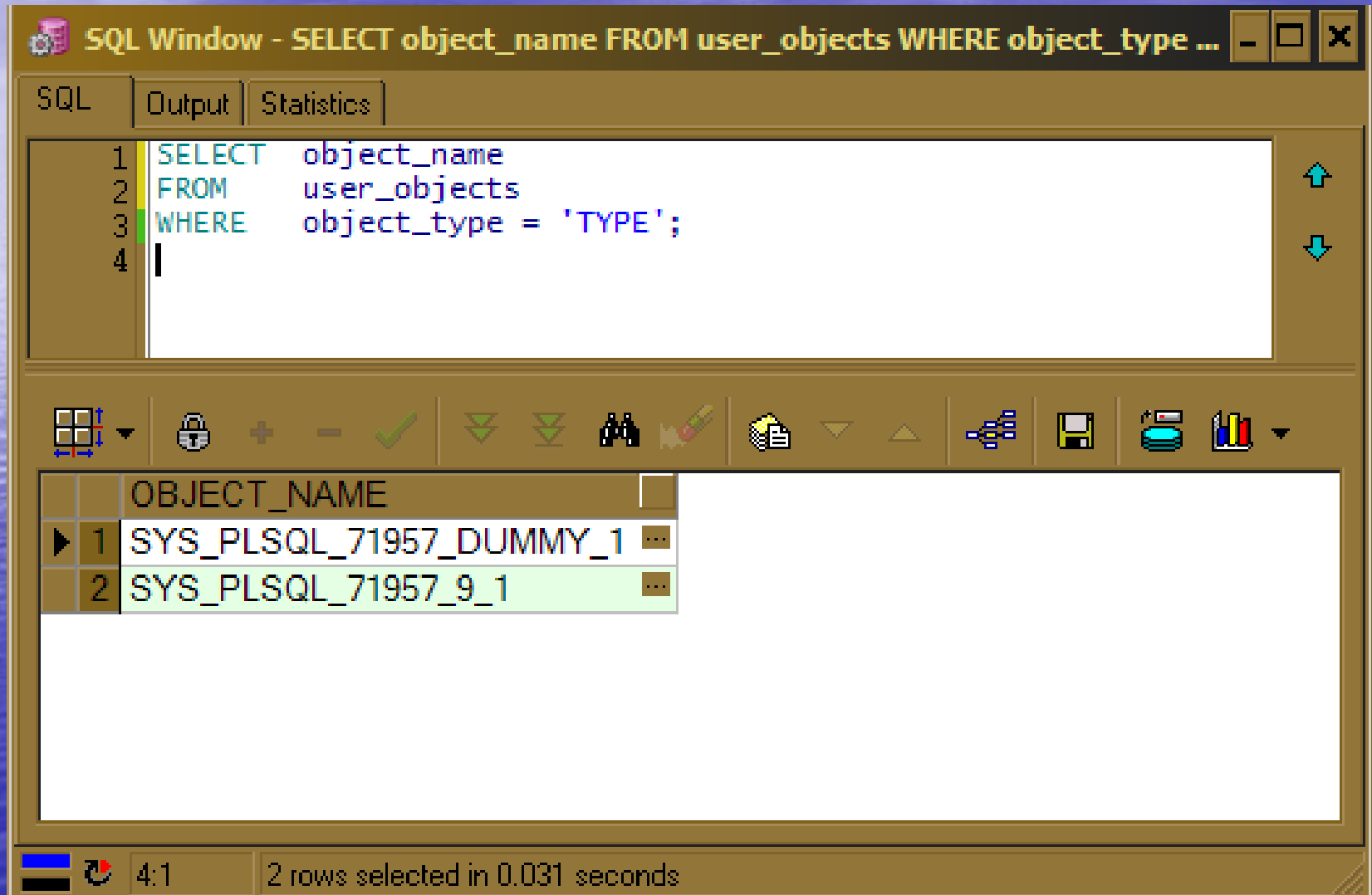


```
Program Window - Edit source of function RET_TR_NUMS@DB
ret_tr_nums

1 CREATE OR REPLACE FUNCTION ret_tr_nums
2 RETURN pkg_of_types.tr_num_list PIPELINED
3 AS
4     RESULT pkg_of_types.tr_num;
5 BEGIN
6     RESULT := 1;
7     PIPE ROW(RESULT);
8     RETURN;
9 END ret_tr_nums;
10

& 5:6 Compiled successfully
```


No Collections in SQL



The screenshot shows an SQL window with the following content:

SQL Window - SELECT object_name FROM user_objects WHERE object_type ...

SQL | Output | Statistics

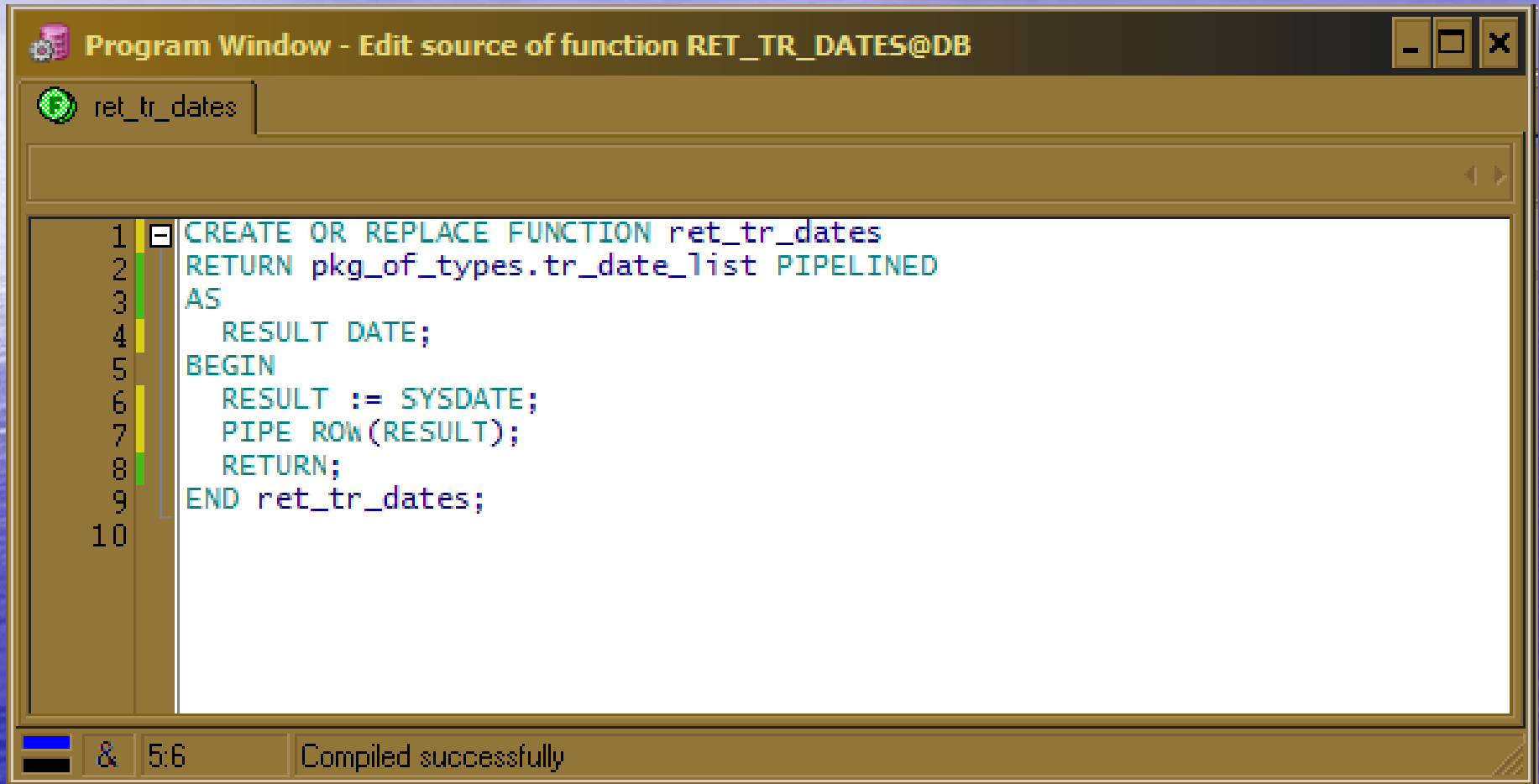
```
1 SELECT object_name
2 FROM user_objects
3 WHERE object_type = 'TYPE';
4 |
```

OBJECT_NAME

1	SYS_PLSQL_71957_DUMMY_1
2	SYS_PLSQL_71957_9_1

4:1 | 2 rows selected in 0.031 seconds

No Collections in SQL

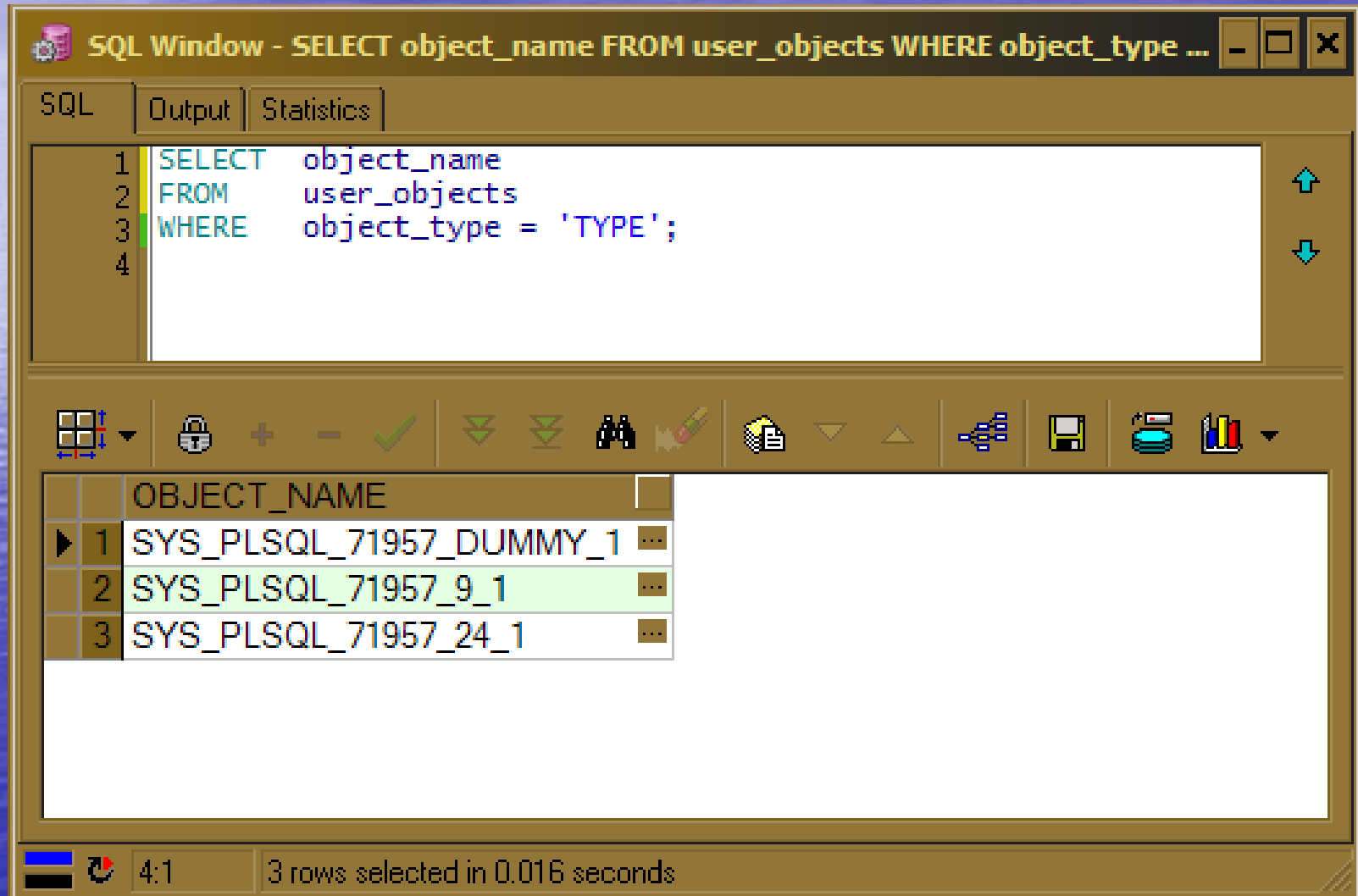


The screenshot shows a window titled "Program Window - Edit source of function RET_TR_DATES@DB". The window contains a tab for "ret_tr_dates" and a text area with the following SQL code:

```
1 CREATE OR REPLACE FUNCTION ret_tr_dates
2 RETURN pkg_of_types.tr_date_list PIPELINED
3 AS
4     RESULT DATE;
5 BEGIN
6     RESULT := SYSDATE;
7     PIPE ROW(RESULT);
8     RETURN;
9 END ret_tr_dates;
10
```

At the bottom of the window, there is a status bar with a blue icon, the text "& 5:6", and the message "Compiled successfully".

No Collections in SQL



The screenshot shows a window titled "SQL Window - SELECT object_name FROM user_objects WHERE object_type ...". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, displaying the following query:

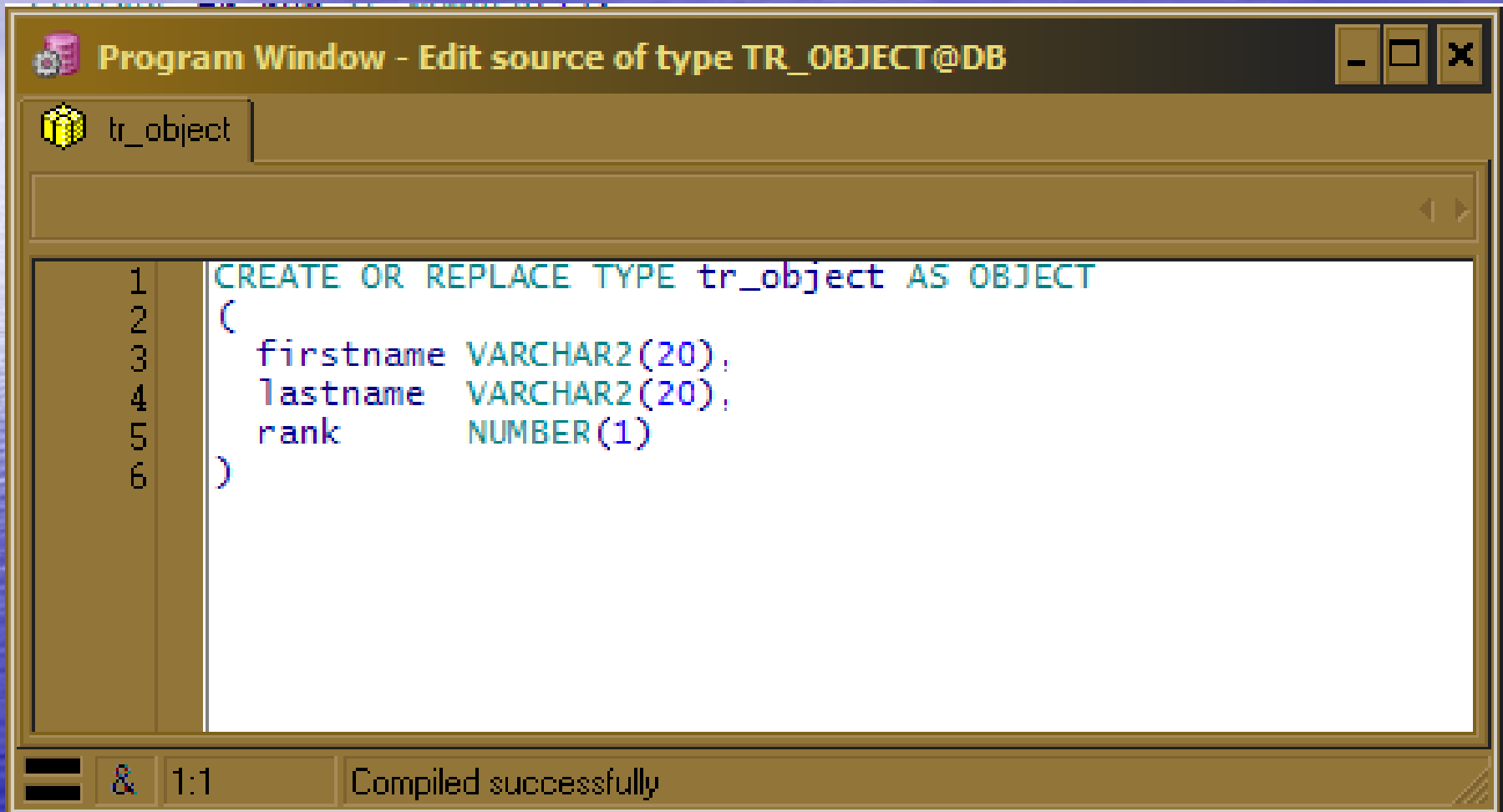
```
1 SELECT object_name
2 FROM user_objects
3 WHERE object_type = 'TYPE';
4
```

Below the query is a toolbar with various icons for execution, refresh, and other database operations. The "Output" tab is active, showing a table with the following data:

	OBJECT_NAME	
▶ 1	SYS_PLSQL_71957_DUMMY_1	...
2	SYS_PLSQL_71957_9_1	...
3	SYS_PLSQL_71957_24_1	...

At the bottom of the window, a status bar shows "4:1" and "3 rows selected in 0.016 seconds".

No Collections in SQL

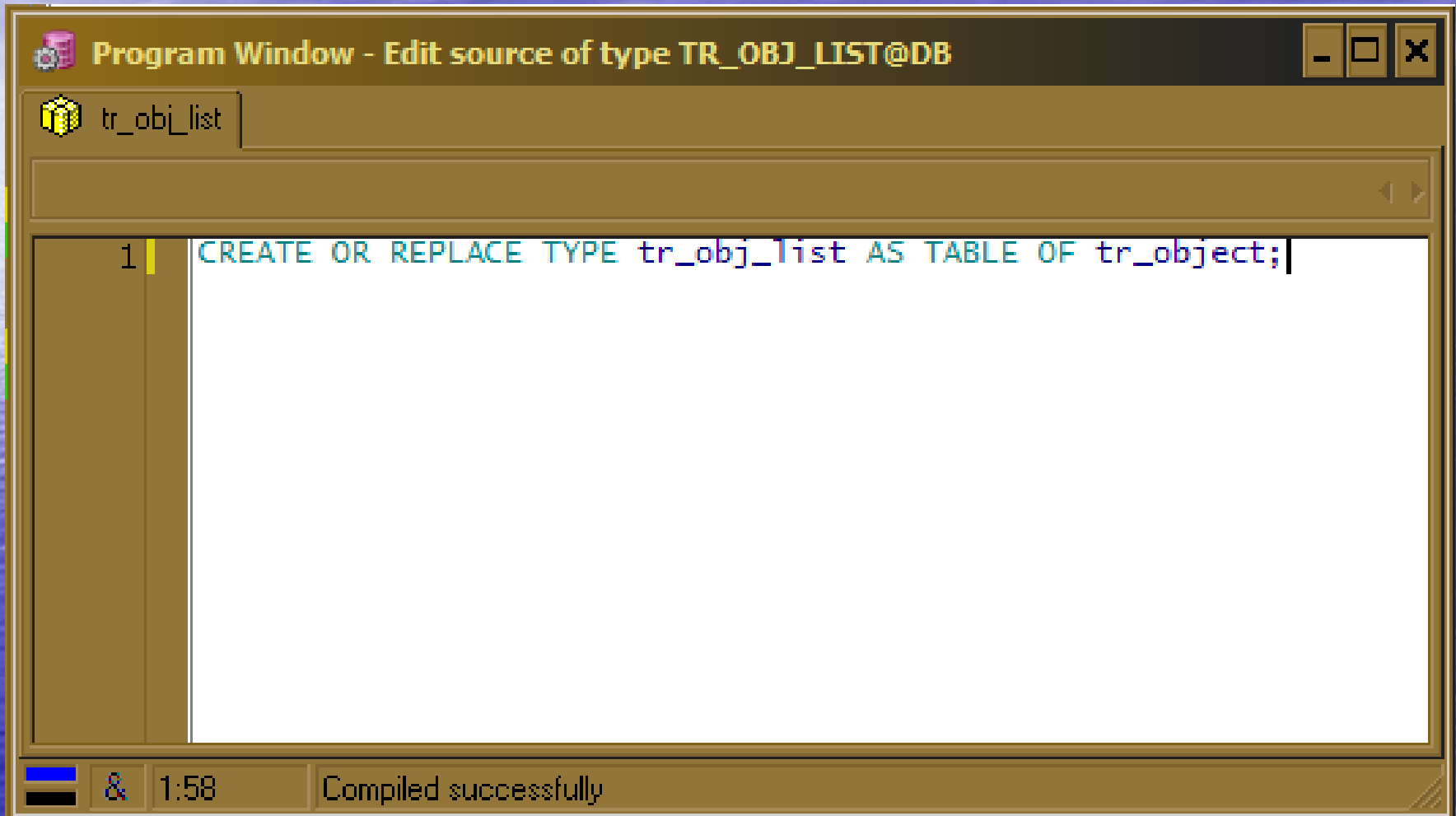


The screenshot shows a window titled "Program Window - Edit source of type TR_OBJECT@DB". The window contains a tab labeled "tr_object" and a text editor with the following SQL code:

```
1 CREATE OR REPLACE TYPE tr_object AS OBJECT
2 (
3     firstname VARCHAR2(20),
4     lastname  VARCHAR2(20),
5     rank      NUMBER(1)
6 )
```

The status bar at the bottom indicates "& 1:1" and "Compiled successfully".

No Collections in SQL

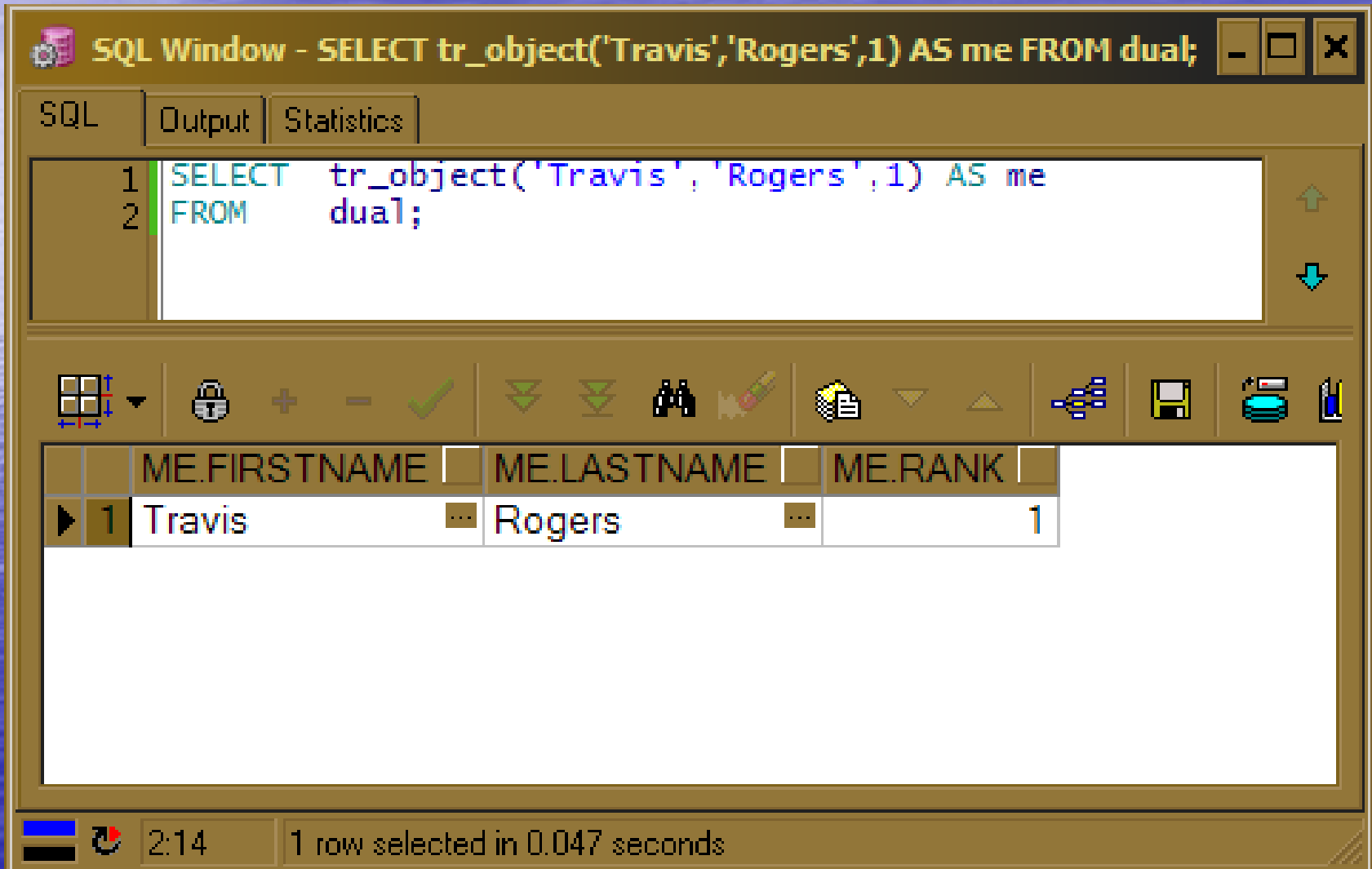


The screenshot shows a window titled "Program Window - Edit source of type TR_OBJ_LIST@DB". The window contains a single line of SQL code: "CREATE OR REPLACE TYPE tr_obj_list AS TABLE OF tr_object;". The status bar at the bottom indicates that the code was compiled successfully.

```
1 CREATE OR REPLACE TYPE tr_obj_list AS TABLE OF tr_object;
```

& 1:58 Compiled successfully

No Collections in SQL



The screenshot shows an SQL window titled "SQL Window - SELECT tr_object('Travis','Rogers',1) AS me FROM dual;". The window has three tabs: "SQL", "Output", and "Statistics". The "SQL" tab is active, showing the following query:

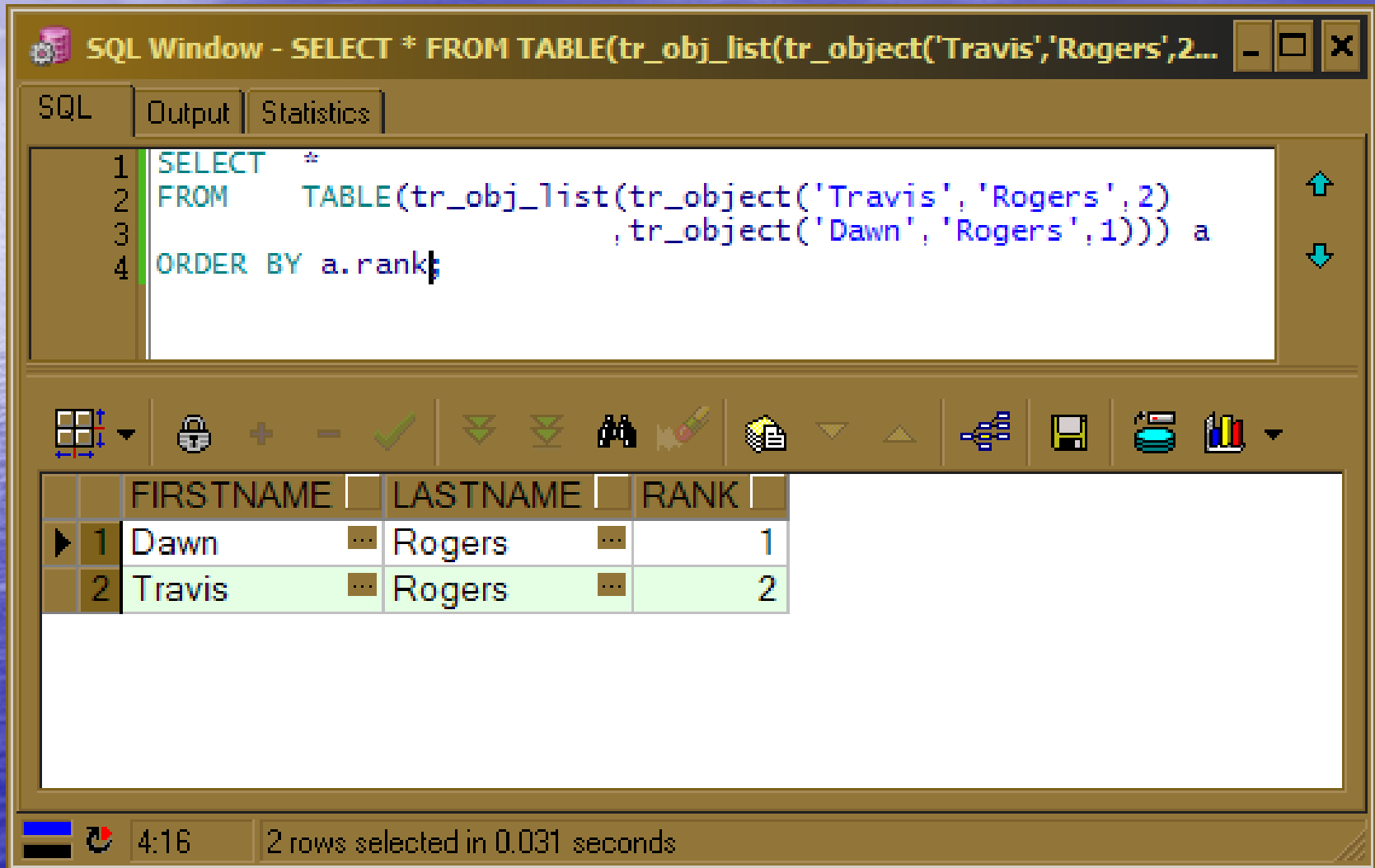
```
1 SELECT tr_object('Travis','Rogers',1) AS me
2 FROM dual;
```

Below the query editor is a toolbar with various icons for execution, refresh, and other database operations. The result set is displayed in a table with the following columns: ME.FIRSTNAME, ME.LASTNAME, and ME.RANK. The table contains one row of data:

	ME.FIRSTNAME	ME.LASTNAME	ME.RANK
1	Travis	Rogers	1

At the bottom of the window, there is a status bar showing a refresh icon, the time "2:14", and the message "1 row selected in 0.047 seconds".

No Collections in SQL



The screenshot shows an SQL Window with the following content:

SQL Window - SELECT * FROM TABLE(tr_obj_list(tr_object('Travis','Rogers',2... - □ ×

SQL Output Statistics

```
1 SELECT *
2 FROM TABLE(tr_obj_list(tr_object('Travis', 'Rogers', 2)
3 , tr_object('Dawn', 'Rogers', 1))) a
4 ORDER BY a.rank;
```

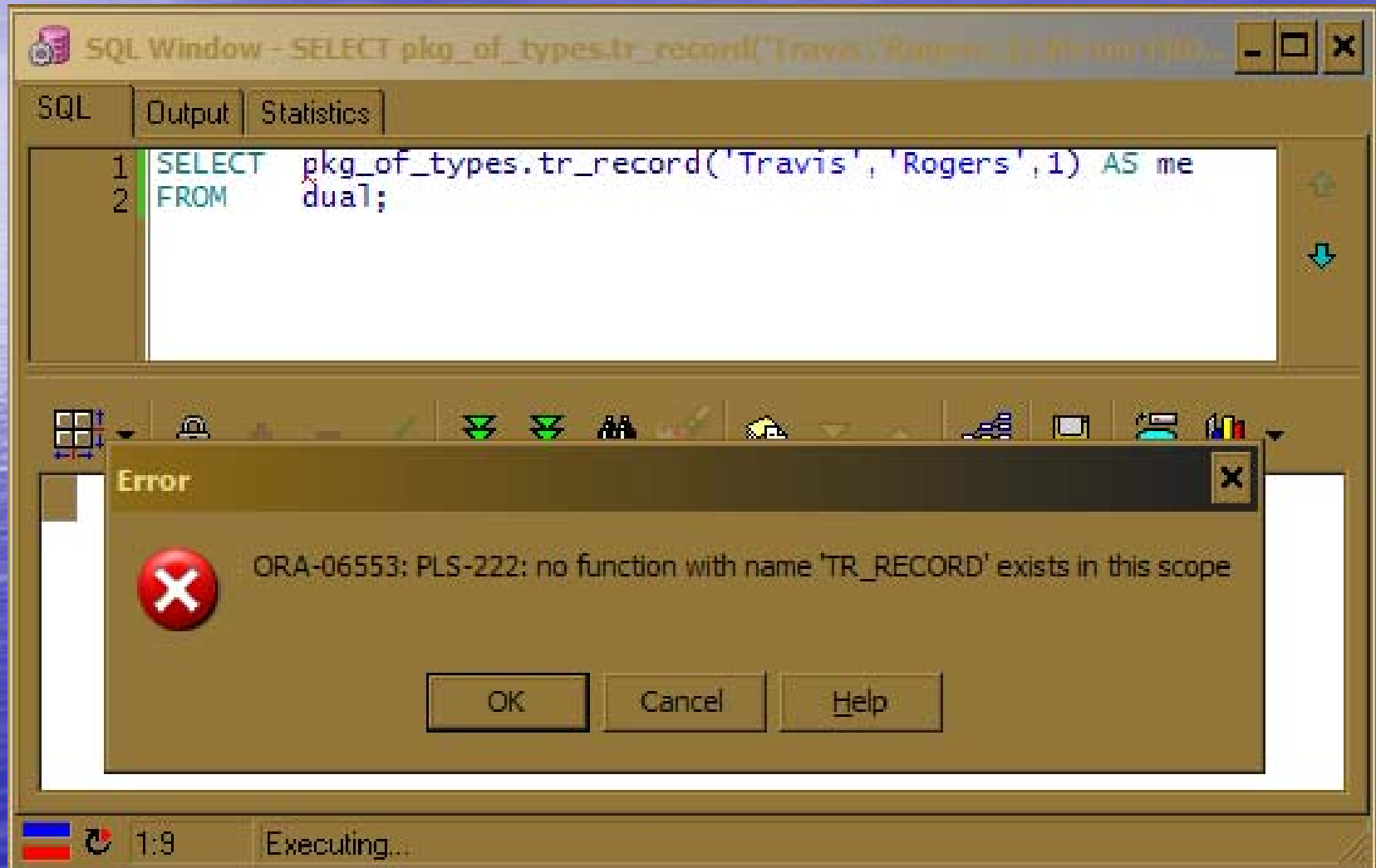
↑ ↓

Grid, Lock, +, -, ✓, ↓, ↓, People, Eraser, Grid, ▲, ▼, Print, Save, Refresh, Chart

	FIRSTNAME	LASTNAME	RANK
▶ 1	Dawn	Rogers	1
2	Travis	Rogers	2

4:16 2 rows selected in 0.031 seconds

No Collections in SQL



The image shows a screenshot of an SQL window with the following content:

SQL Window - SELECT pkg_of_types.tr_record('Travis','Rogers',1) AS me

SQL Output Statistics

```
1 SELECT pkg_of_types.tr_record('Travis','Rogers',1) AS me
2 FROM dual;
```

Error

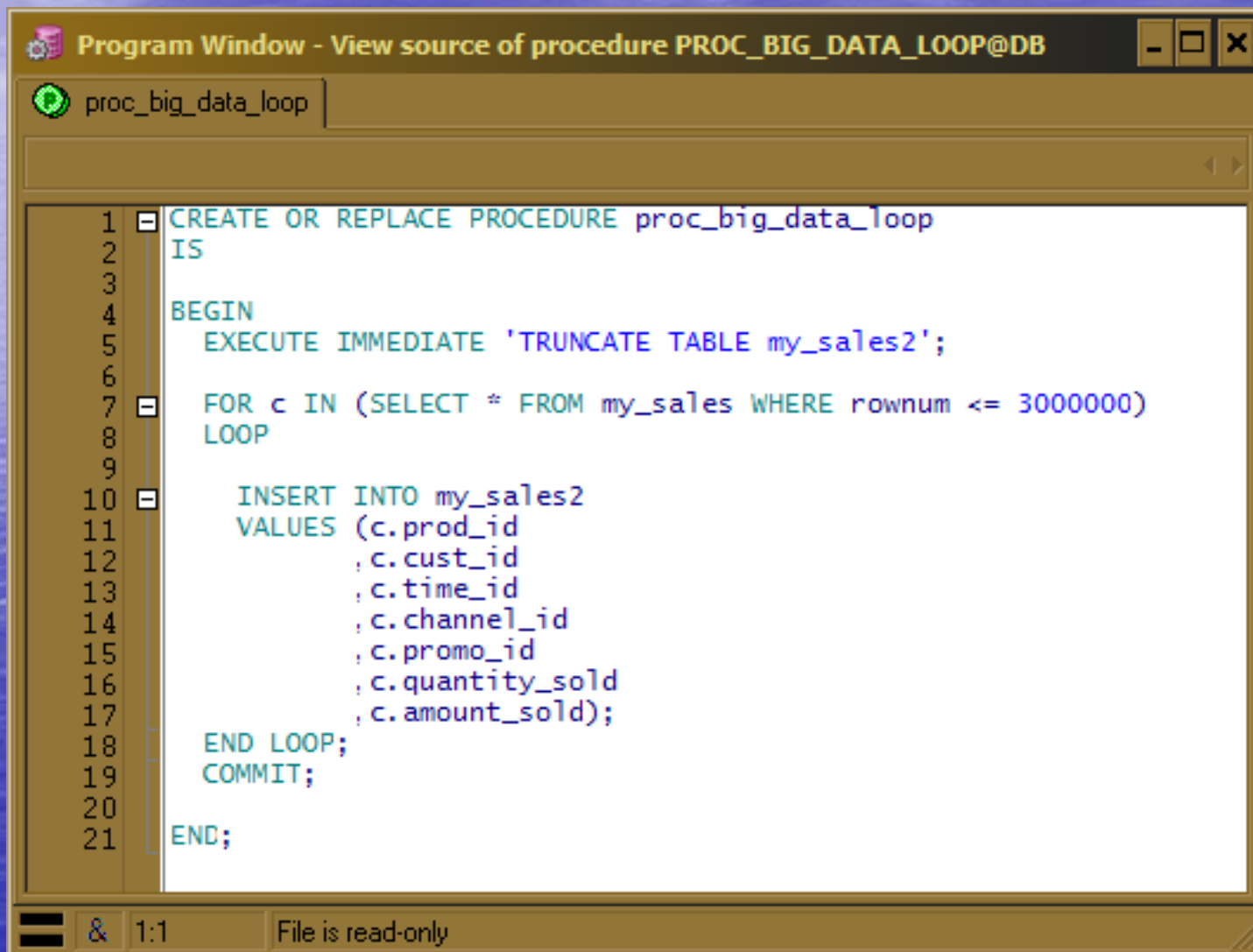
ORA-06553: PLS-222: no function with name 'TR_RECORD' exists in this scope

OK Cancel Help

1:9 Executing...

The screenshot illustrates a common issue in SQL where a function name is misspelled or does not exist in the current scope. The error message 'ORA-06553: PLS-222: no function with name 'TR_RECORD' exists in this scope' is displayed in a dialog box, indicating that the function 'TR_RECORD' is not recognized by the database. The SQL window shows the query being executed, and the status bar at the bottom indicates that the query is still executing.

Bulk Operations - Loop

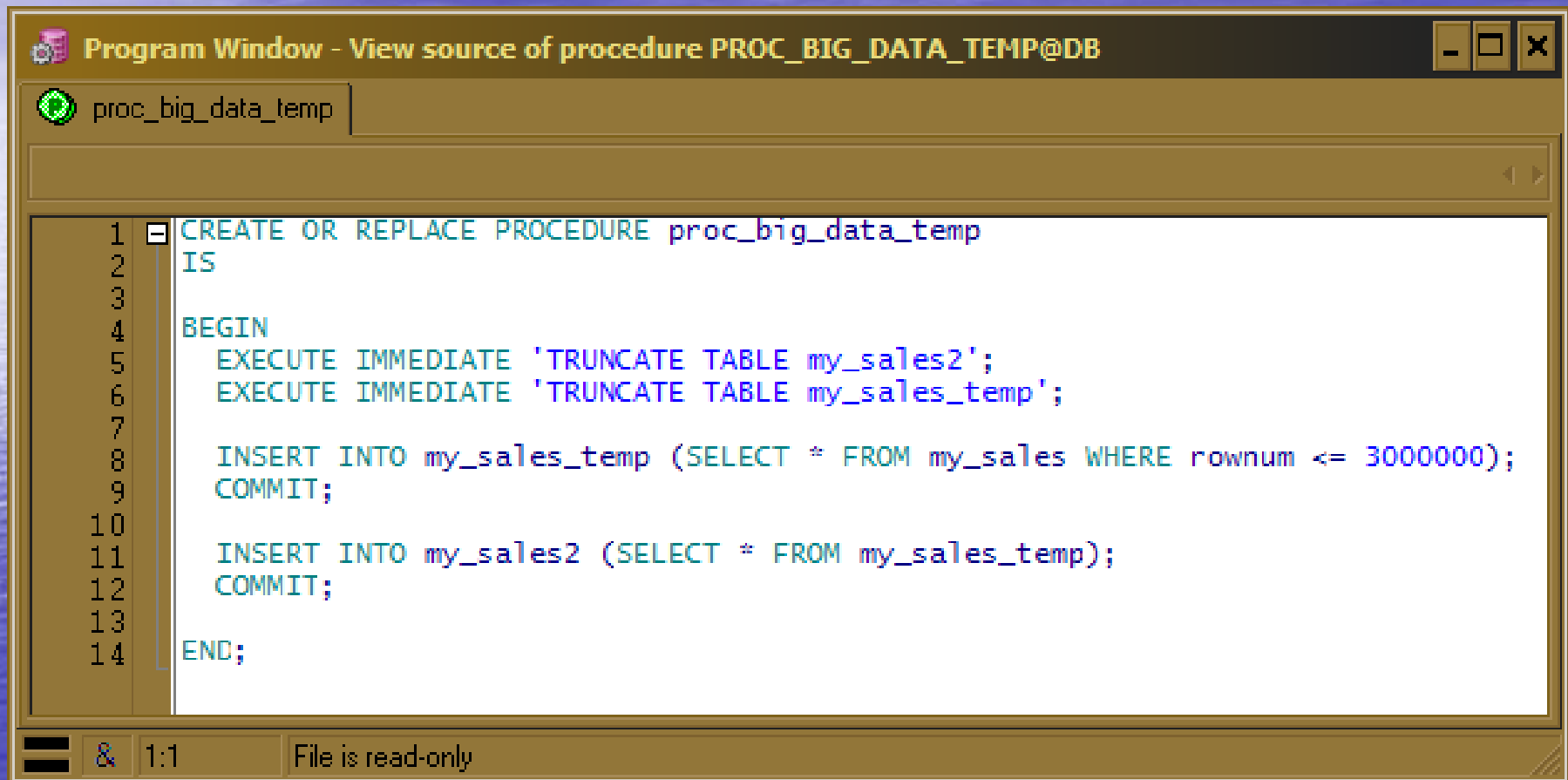


```
Program Window - View source of procedure PROC_BIG_DATA_LOOP@DB
proc_big_data_loop

1 CREATE OR REPLACE PROCEDURE proc_big_data_loop
2 IS
3
4 BEGIN
5     EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
6
7     FOR c IN (SELECT * FROM my_sales WHERE rownum <= 3000000)
8     LOOP
9
10        INSERT INTO my_sales2
11        VALUES (c.prod_id
12                ,c.cust_id
13                ,c.time_id
14                ,c.channel_id
15                ,c.promo_id
16                ,c.quantity_sold
17                ,c.amount_sold);
18    END LOOP;
19    COMMIT;
20
21 END;
```

& 1:1 File is read-only

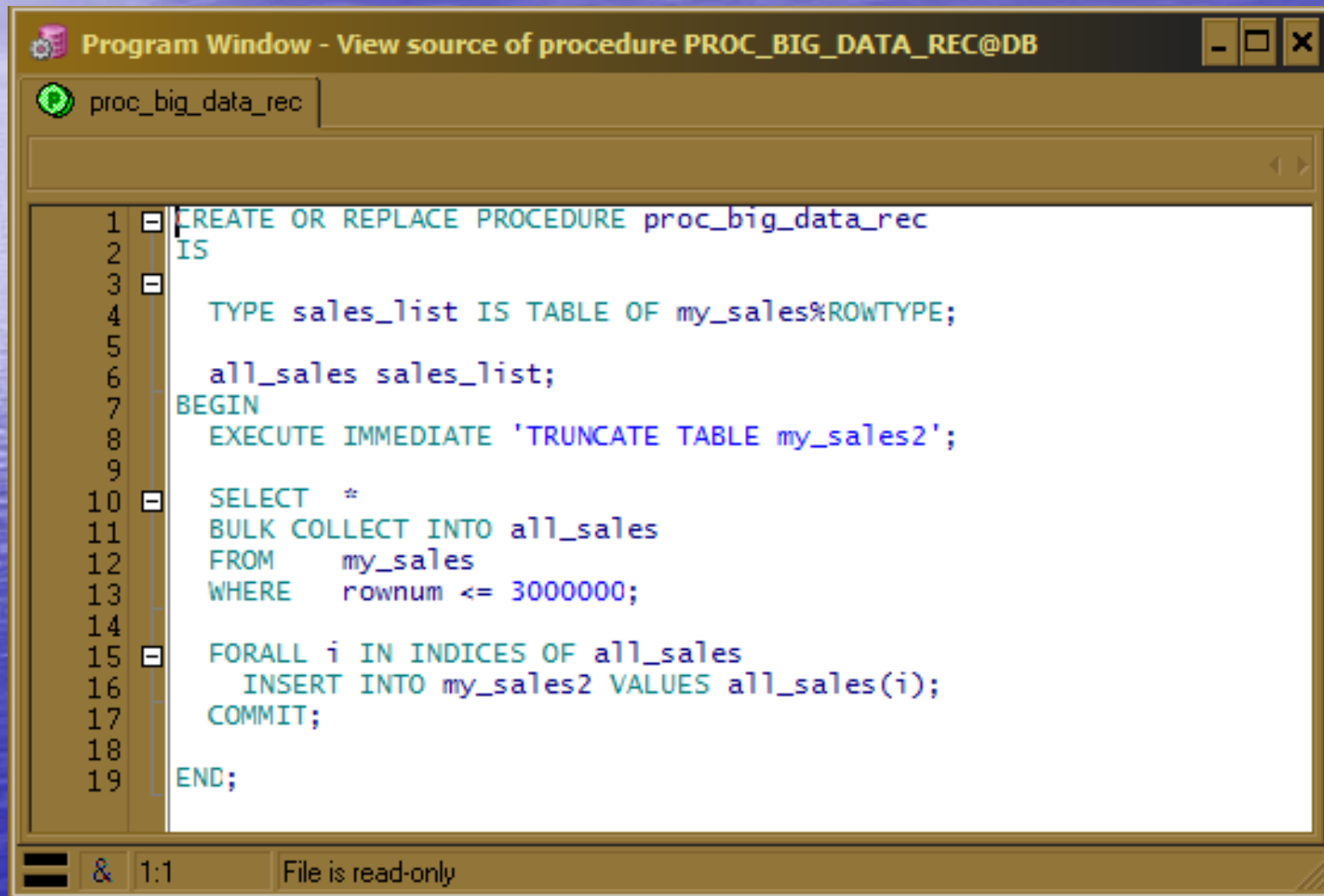
Bulk Operations – Table2Table



```
Program Window - View source of procedure PROC_BIG_DATA_TEMP@DB
proc_big_data_temp
1 CREATE OR REPLACE PROCEDURE proc_big_data_temp
2 IS
3
4 BEGIN
5     EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
6     EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales_temp';
7
8     INSERT INTO my_sales_temp (SELECT * FROM my_sales WHERE rownum <= 3000000);
9     COMMIT;
10
11     INSERT INTO my_sales2 (SELECT * FROM my_sales_temp);
12     COMMIT;
13
14 END;
```

& 1:1 File is read-only

Bulk Operations – Nested Table

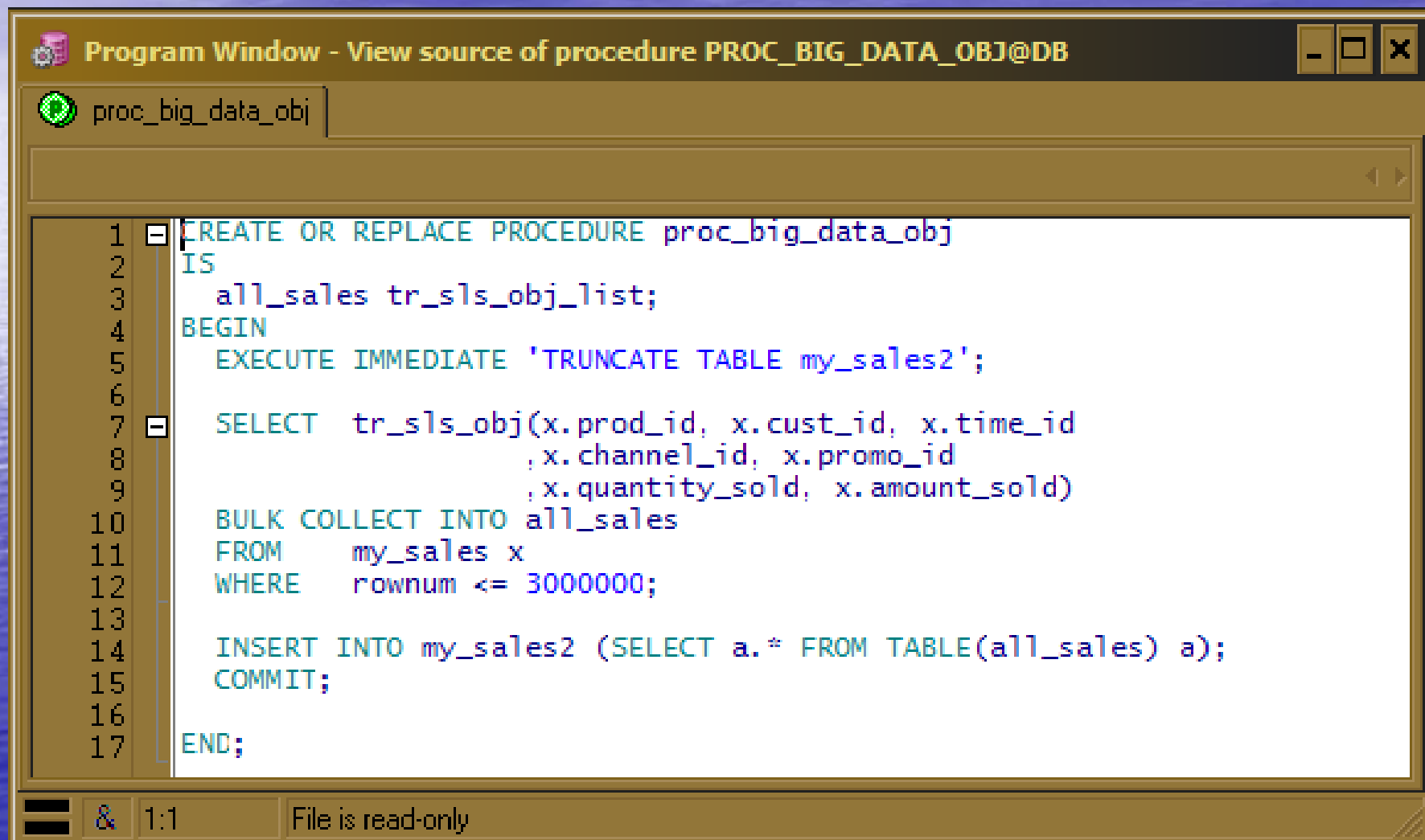


```
Program Window - View source of procedure PROC_BIG_DATA_REC@DB
proc_big_data_rec

1 CREATE OR REPLACE PROCEDURE proc_big_data_rec
2 IS
3
4     TYPE sales_list IS TABLE OF my_sales%ROWTYPE;
5
6     all_sales sales_list;
7 BEGIN
8     EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
9
10    SELECT *
11    BULK COLLECT INTO all_sales
12    FROM    my_sales
13    WHERE   rownum <= 3000000;
14
15    FORALL i IN INDICES OF all_sales
16        INSERT INTO my_sales2 VALUES all_sales(i);
17    COMMIT;
18
19 END;
```

& 1:1 File is read-only

Bulk Operations - Object

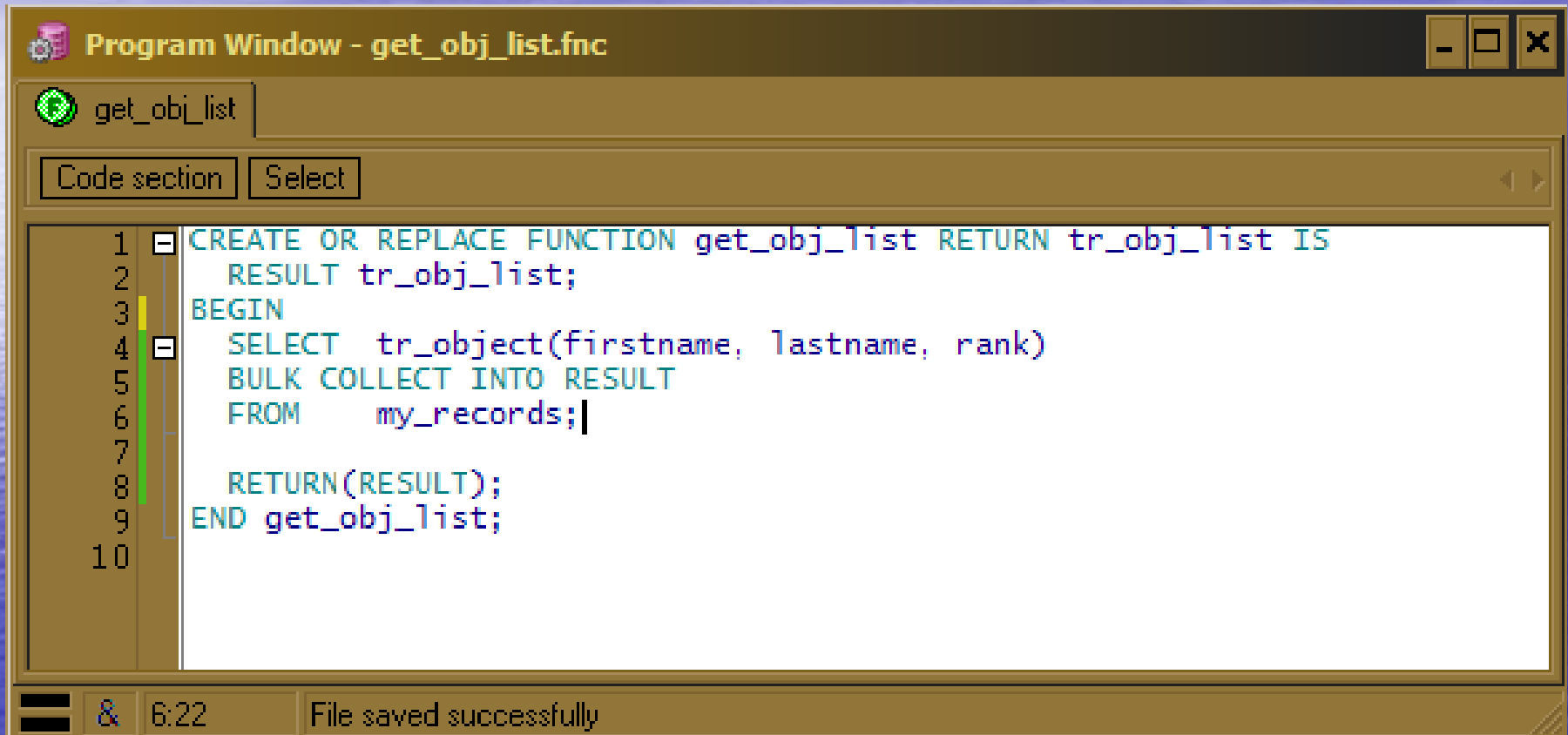


The screenshot shows a window titled "Program Window - View source of procedure PROC_BIG_DATA_OBJ@DB". The window contains a text editor with the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE proc_big_data_obj
2 IS
3   all_sales tr_sls_obj_list;
4 BEGIN
5   EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
6
7   SELECT  tr_sls_obj(x.prod_id, x.cust_id, x.time_id
8             ,x.channel_id, x.promo_id
9             ,x.quantity_sold, x.amount_sold)
10  BULK COLLECT INTO all_sales
11  FROM    my_sales x
12  WHERE   rownum <= 3000000;
13
14  INSERT INTO my_sales2 (SELECT a.* FROM TABLE(all_sales) a);
15  COMMIT;
16
17 END;
```

At the bottom of the window, there is a status bar with a menu icon, an ampersand (&), the text "1:1", and the message "File is read-only".

External Usage



The screenshot shows a window titled "Program Window - get_obj_list.fnc". The window contains a code editor with the following SQL code:

```
1 CREATE OR REPLACE FUNCTION get_obj_list RETURN tr_obj_list IS
2   RESULT tr_obj_list;
3 BEGIN
4   SELECT tr_object(firstname, lastname, rank)
5   BULK COLLECT INTO RESULT
6   FROM   my_records;
7
8   RETURN(RESULT);
9 END get_obj_list;
10
```

At the bottom of the window, there is a status bar showing the time "6:22" and the message "File saved successfully".

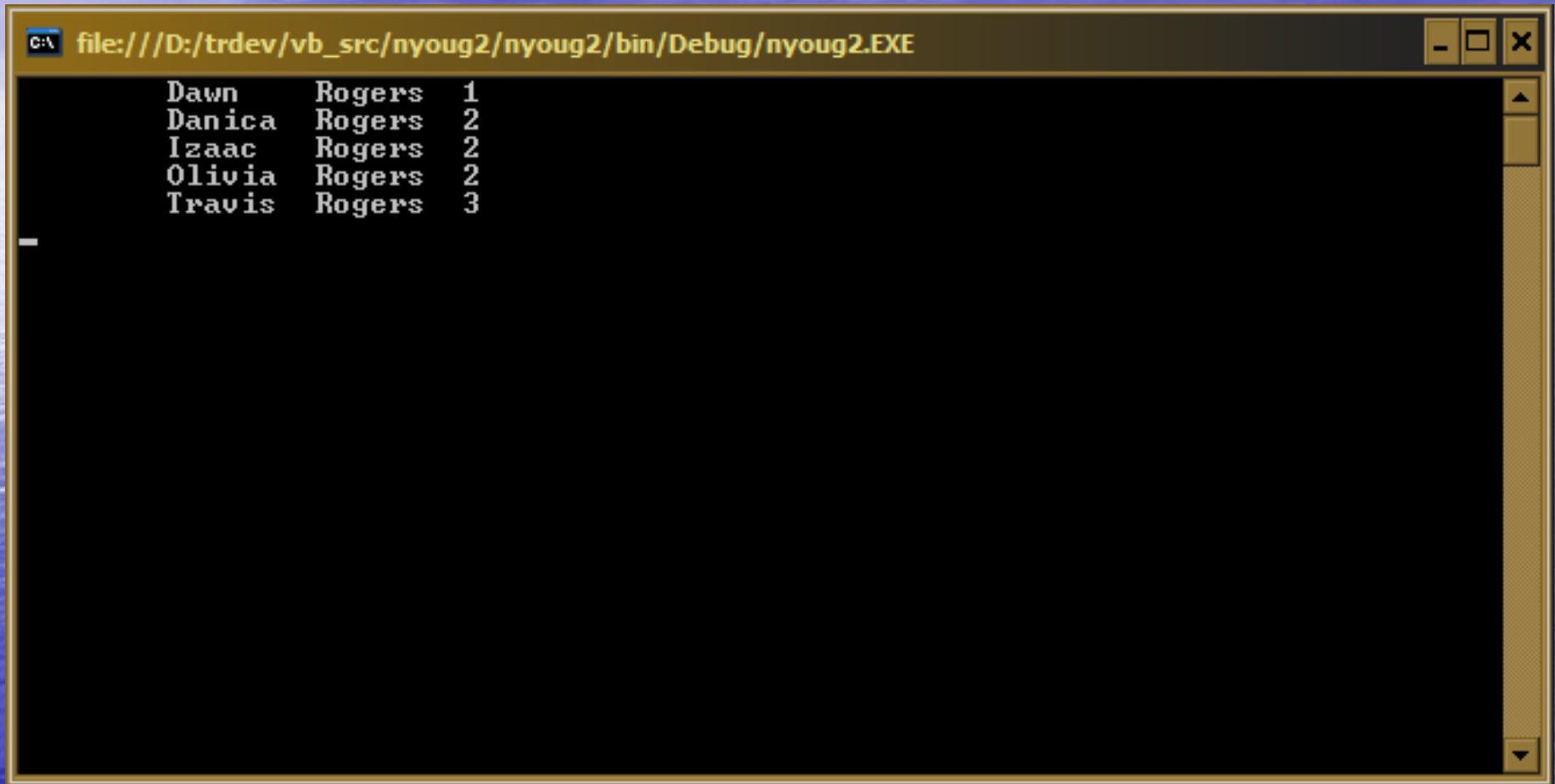
External Usage – VB.Net

```
Module1.vb*
Module1 (Declarations)
Imports System
Imports System.Data
Imports System.Data.OracleClient

Module Module1
    Public Sub Main()
        Dim connectionString As String = "Data Source=db;User Id=nyoug2;Password=ny2"
        Dim queryString As String = "SELECT a.firstname, a.lastname, a.rank FROM TABLE(get_obj_list()) a"

        Using connection As New OracleConnection(connectionString)
            Dim command As OracleCommand = connection.CreateCommand()
            command.CommandText = queryString
            Try
                connection.Open()
                Dim dataReader As OracleDataReader = command.ExecuteReader()
                Do While dataReader.Read()
                    Console.WriteLine(vbTab & "{0}" & vbTab & "{1}" & vbTab & "{2}" _
                        , dataReader(0), dataReader(1), dataReader(2))
                Loop
                dataReader.Close()
            Catch ex As Exception
                Console.WriteLine(ex.Message)
            End Try
        End Using
    End Sub
End Module
```

External Usage – VB.Net



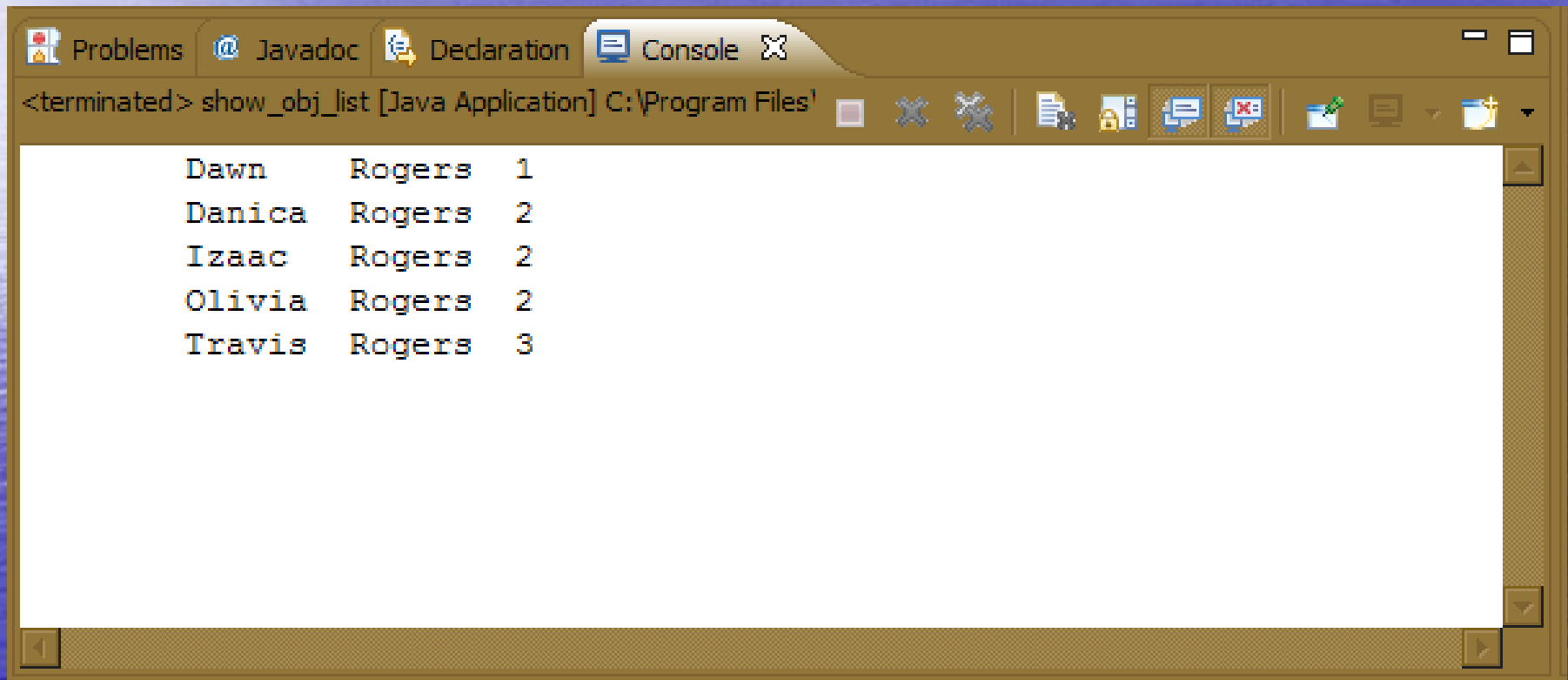
```
C:\ file:///D:/trdev/vb_src/nyoug2/nyoug2/bin/Debug/nyoug2.EXE
Dawn    Rogers  1
Danica  Rogers  2
Izaac   Rogers  2
Olivia  Rogers  2
Travis  Rogers  3
```

Dawn	Rogers	1
Danica	Rogers	2
Izaac	Rogers	2
Olivia	Rogers	2
Travis	Rogers	3

External Usage – Java

```
show_obj_list.java X
1 package nypug2;
2
3 import java.sql.*;
4
5 public class show_obj_list
6 {
7     public static void main(String[] args)
8     {
9         try
10        {
11            Connection con=null;
12            Class.forName("oracle.jdbc.driver.OracleDriver");
13            con=DriverManager.getConnection("jdbc:oracle:thin:@192.168.123.101:1522:db","nyoug2","ny2");
14            Statement s=con.createStatement();
15            ResultSet rs = s.executeQuery( "SELECT a.* FROM TABLE(get_obj_list()) a" );
16            while( rs.next() )
17            {
18                System.out.println("\t" + rs.getString(1)+ "\t" + rs.getString(2) +
19                                    "\t" + rs.getInt(3));
20            }
21            rs.close() ;
22            s.close() ;
23            con.close() ;
24        }
25        catch(Exception e)
26        {
27            e.printStackTrace();
28        }
29    }
30 }
```

External Usage – Java



The screenshot shows an IDE console window with the following content:

```
<terminated> show_obj_list [Java Application] C:\Program Files!  
  
Dawn    Rogers  1  
Danica  Rogers  2  
Izaac   Rogers  2  
Olivia  Rogers  2  
Travis  Rogers  3
```

The console window has a title bar with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active. The window title is '<terminated> show_obj_list [Java Application] C:\Program Files!'. The console output is displayed in a monospaced font.

Controlling Complexity

Part 5

Apply 20%, 80% of the time

- If a type is never going to be used by SQL then use the PL/SQL functionality.
- If there is $> 20\%$ chance that a type will be used by SQL then use an Object.
- Use a Nested Table for both SQL and PL/SQL.

Allow others their 20%

- Don't store Objects.
- API's should
 - return Collections
 - AND fill an IN OUT cursor.
- Use the TABLE function to simplify usage.
Use custom constructors in Objects.
- Objects should simplify and/or flatten the interface to complex data storage.

Scenario – The Opportunity

- Hardware store with both Web and Brick-and-Mortar presence.
- Lot's of historical data.
- A desire to do use this data to
 - Project future sales.
 - Drive large promotional campaigns.
 - Drive personalized promotions.

Scenario – The Opportunity

- A user interface allowing the dynamic creation and exploration of data and application of different algorithms.
- Profiles should be save-able and re-usable and a history should be kept of their use.
- Should be able to compare actual results later to projected results.
- Data points and algorithms should be easily added in the future.

Scenario – The Data

- Customer Data - Name, address, phone, age
- Sales Data - Detail of each item sold, when and to which customer.
- Environmental data by date and region
 - Weather
 - Housing market details
 - Mortgage and Home Equity rates and availability
 - Local and national Economic data

Scenario – Inputs

- Data points to be considered
- Timelines to be considered per data point (these could be fixed or relative to sales dates)
- Algorithms to apply (sum, average, moving average, correlation to sales and etc)

The background is a smooth blue gradient, transitioning from a lighter blue at the top to a darker blue at the bottom. A bright sun flare is visible on the left side, creating a white and yellow glow that fades into the blue background.

Has your head exploded yet?

Scenario – The Good News

- The data is all there and nicely organized, indexed and etc.
- Our primary responsibilities are
 - Get the data
 - Apply the appropriate algorithms
 - Present the results via an API to other developers who then use them (reports, portals, client server, export to other tools, and on and on...)

Scenario – The Simplicity

- Rules
- Profiles
 - Customer
 - Environmental
- Results (KPI)

What is our responsibility?

Results

And how did this turn into a
discussion of OOAD?

Postlude

This is not another digression!

- Object oriented design and coding can become overly complex but it has its place even in a database.
- Traditional relational data access is more complex than the general API user's capabilities. (and is getting more complex)
- Oracle provides the capability and it can be powerful so use it.

This is not another digression!

- Oracle provides the capability and it can be powerful so use it.
- But remember the 80/20 rule and don't abuse it.

Travis R. Rogers

Truppenbi LLC

travis@jerseyrogers.com