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
No Collections in SQL

CREATE OR REPLACE PACKAGE pkg_of_types AS
  TYPE tr_date_list IS TABLE OF DATE;
  SUBTYPE tr_num IS NUMBER(1);
  TYPE tr_num_list IS TABLE OF tr_num;
  TYPE tr_record IS RECORD (firstname VARCHAR2(20), lastname VARCHAR2(20), rank NUMBER(1));
  TYPE tr_rec_list IS TABLE OF tr_record;
END;
No Collections in SQL

```sql
SELECT object_name FROM user_objects WHERE object_type = 'TYPE';
```

Output:

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

0 rows selected in 0.031 seconds
No Collections in SQL

```
CREATE OR REPLACE FUNCTION ret_tr_nums
RETURN pkg_of_types.tr_num_list PIPELINED
AS
  RESULT pkg_of_types.tr_num;
BEGIN
  RESULT := 1;
  PIPE ROW (RESULT);
  RETURN;
END ret_tr_nums;
```
No Collections in SQL

```
SELECT object_name
FROM user_objects
WHERE object_type = 'TYPE';
```
No Collections in SQL

```sql
CREATE OR REPLACE FUNCTION ret_tr_dates
RETURN pkg_of_types.tr_date_list PIPELINED
AS
  RESULT DATE;
BEGIN
  RESULT := SYSDATE;
  PIPE ROW(Result);
  RETURN;
END ret_tr_dates;
```
No Collections in SQL

SQL Window - SELECT object_name FROM user_objects WHERE object_type = 'TYPE';

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_PLSQL_71957_DUMMY_1</td>
</tr>
<tr>
<td>SYS_PLSQL_71957_9_1</td>
</tr>
<tr>
<td>SYS_PLSQL_71957_24_1</td>
</tr>
</tbody>
</table>

3 rows selected in 0.016 seconds
No Collections in SQL

Program Window - Edit source of type TR_OBJECT@DB

tr_object

CREATE OR REPLACE TYPE tr_object AS OBJECT
(
  firstname VARCHAR2(20),
  lastname  VARCHAR2(20),
  rank      NUMBER(1)
)

Compiled successfully
No Collections in SQL

CREATE OR REPLACE TYPE tr_obj_list AS TABLE OF tr_object;
No Collections in SQL
No Collections in SQL

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

<table>
<thead>
<tr>
<th></th>
<th>FIRSTNAME</th>
<th>LASTNAME</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dawn</td>
<td>Rogers</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Travis</td>
<td>Rogers</td>
<td>2</td>
</tr>
</tbody>
</table>

2 rows selected in 0.031 seconds
No Collections in SQL

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

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

```
CREATE OR REPLACE PROCEDURE proc_big_data_loop IS

BEGIN
    EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';

    FOR c IN (SELECT * FROM my_sales WHERE rownum <= 3000000)
    LOOP
        INSERT INTO my_sales2
        VALUES (c.prod_id,
                c.cust_id,
                c.time_id,
                c.channel_id,
                c.promo_id,
                c.quantity_sold,
                c.amount_sold);
    END LOOP;
    COMMIT;

END;
```
Bulk Operations - Table2Table

```
CREATE OR REPLACE PROCEDURE proc_big_data_temp
IS
BEGIN
  EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
  EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales_temp';
  INSERT INTO my_sales_temp (SELECT * FROM my_sales WHERE rownum <= 3000000);
  COMMIT;
  INSERT INTO my_sales2 (SELECT * FROM my_sales_temp);
  COMMIT;
END;
```
Bulk Operations - Nested Table

```sql
CREATE OR REPLACE PROCEDURE proc_big_data_rec
IS

  TYPE sales_list IS TABLE OF my_sales%ROWTYPE;
  all_sales sales_list;
BEGIN
  EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';

  SELECT *
  BULK COLLECT INTO all_sales
  FROM my_sales
  WHERE rownum <= 3000000;

  FORALL i IN INDICES OF all_sales
    INSERT INTO my_sales2 VALUES all_sales(i);
  COMMIT;
END;
```

CREATE OR REPLACE PROCEDURE proc_big_data_obj
IS
    all_sales tr_sls_obj_list;
BEGIN
    EXECUTE IMMEDIATE 'TRUNCATE TABLE my_sales2';
    SELECT tr_sls_obj(x.prod_id, x.cust_id, x.time_id,
                      x.channel_id, x.promo_id,
                      x.quantity_sold, x.amount_sold)
    BULK COLLECT INTO all_sales
    FROM my_sales x
    WHERE rownum <= 3000000;
    INSERT INTO my_sales2 (SELECT a.* FROM TABLE(all_sales) a);
    COMMIT;
END;
CREATE OR REPLACE FUNCTION get_obj_list RETURN tr_obj_list IS
RESULT tr_obj_list;
BEGIN
SELECT tr_object(firstname, lastname, rank)
BULK COLLECT INTO RESULT
FROM my_records;

RETURN(RESULT);
END get_obj_list;
Imports System
Imports System.Data
Imports System.Data.OleDb

Module Module1
    Public Sub Main()
        Dim connectionString As String = "Data Source=db;User Id=nyong2;Password=ny2"
        Dim queryString As String = "SELECT a.firstname, a.lastname, a.rank FROM TABLE(get_obj_list()) a"

        Using connection As New OleDbConnection(connectionString)
            Dim command As OleDbCommand = connection.CreateCommand()
            command.CommandText = queryString

            Try
                connection.Open()
                Dim dataReader As OleDbDataReader = 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

Dawn Rogers 1
Danica Rogers 2
Izaac Rogers 2
Olivia Rogers 2
Travis Rogers 3
package nypug2;

import java.sql.*;

public class show_obj_list
{
    public static void main(String[] args)
    {
        try
        {
            Connection con=null;
            Class.forName("oracle.jdbc.driver.OracleDriver");
            con=DriverManager.getConnection("jdbc:oracle:thin:@192.168.123.101:1522:db","nypug2","ny2");
            Statement s=con.createStatement();
            ResultSet rs = s.executeQuery("SELECT a.* FROM TABLE(get_obj_list()) a");
            while( rs.next() )
            {
                System.out.println("\t" + rs.getString(1) + "\t" + rs.getString(2) + "\t" + rs.getInt(3));
            }
            rs.close();
            s.close();
            con.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
External Usage – Java
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)
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.