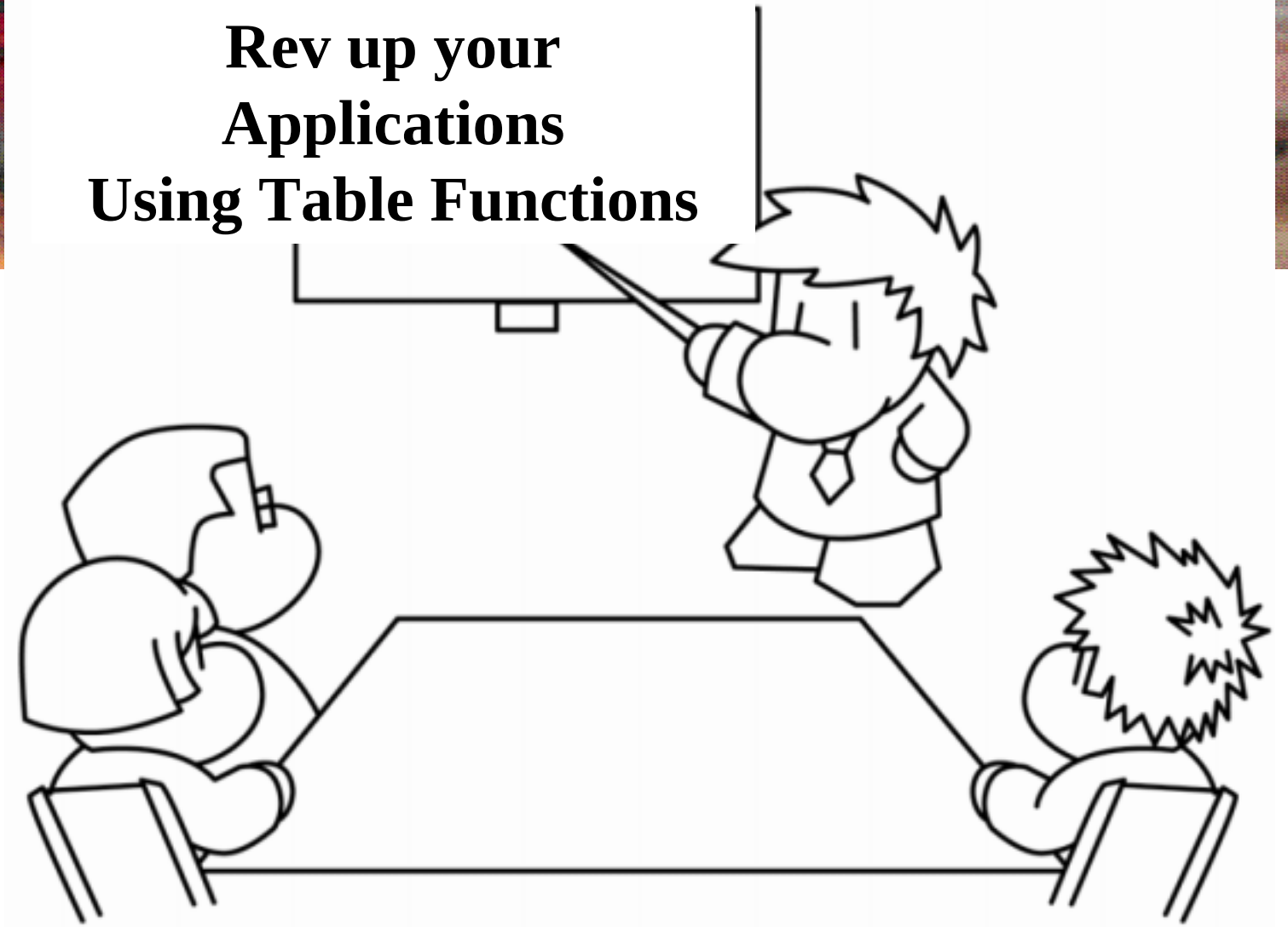
Rev up your Applications Using Table Functions



Dan Hotka

Author/Instructor

Oracle Ace Director

Limitation on Warranty. THERE ARE NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT THERETO, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. PURCHASER SHALL BE SOLELY RESPONSIBLE FOR THE SELECTION, USE, EFFICIENCY AND SUITABILITY OF USE OF INFORMATION CONTAINED HEREIN TO ANY PARTICULAR APPLICATION OR PROBLEM. WWW.DANHOTKA.COM LLC SHALL HAVE NO LIABILITY THEREFOR.

Limitation of Liability. IN NO EVENT SHALL WWW.DANHOTKA.COM LLC BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING, WITHOUT LIMITATION, ANY DAMAGES RELATINGTO LOSS OF DATA, AND ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES OR LOST PROFITS, ARISING OUT OF OR IN ANY WAY RELATED TO YOUR USE OF THE INFORMATION CONTAINED IN THIS MANUAL. IN THE EVENT THAT THE FORGOING IS HELD UNINFORCABLE THE PARTIES AGREE THAT WWW.DANHOTKA.COM LLC'S LIABILITY TO YOU HEREUNDER, IF ANY, SHALL IN NO EVENT EXCEED THE FEE PAID BY THE INJURED PARTY FOR THE MANUAL TO WWW.DANHOTKA.COM. LLC.

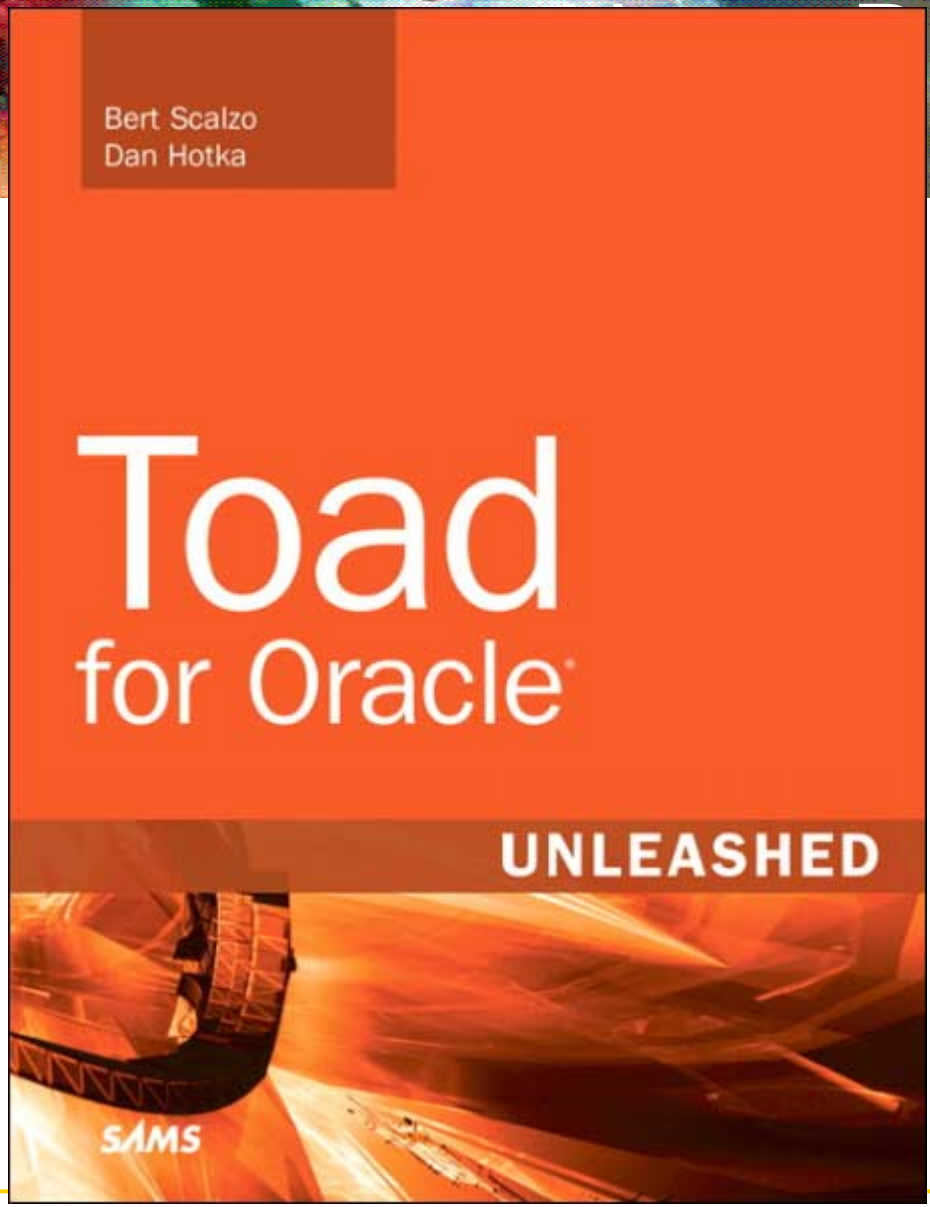
Dan Hotka
Author/Instructor/Oracle ACE Director
CEO
DHotka@Earthlink.net
515 771-3935

Dan is a Training Consultant

- **Dan Hotka**
  - Oracle ACE Director 
    - Oracle ACEs and Oracle ACE Directors are known for their strong credentials as Oracle community enthusiasts and advocates, with candidates nominated by anyone in the Oracle Technology and Applications communities.
    - www.oracle.com/technology/community/oracle_ace/index.html
  - Oracle Authored Expert
    - 32 Years in IT – 27 years working with Oracle
    - 12 books – hundreds of articles
- **DanHotka.Blogspot.com**
  - I also blog on TOADWorld.com

- **Training LiveLessons Video!**
  - Safari Books Online
  - Take the class self-paced
    - www.safaribooksonline.com
  - Purchase the class
    - http://informit.com/hotka
  - SQL Tuning, Intro and Adv PL/SQL,
  - OBIEE Reports and Dashboards, Oracle SQL, and Toad
  - Show your HR or Training Department this info!
- **Flat Fee Training for your company:**
  - 1 Course Fee Price
    - Price includes my portable computer lab!
  - On-site or over the Web!
  - Portable computer lab
- **Training Courses Include:**
  - Oracle Advanced PL/SQL
  - Oracle SQL Tuning
  - TOAD Courses
  - Oracle OBIEE – Reports and Dashboards
  - Discoverer, Intro courses, Cross training!

ook



Discounts on my website Amazon.com

Agenda

- **Pipelined Table Functions**
  - How they work
  - Getting Started
  - Cursors and Pipelined Functions
  - Chaining Pipelined Functions Together
  - Compiler Options for Pipelined Functions

Working with Table Functions

- **How they work:**
  - They return rows as the rows are assembled, while the function is still executing
  - Typical SQL or functions don't return anything until execution has concluded
  - Memory footprint is a fraction because the functions use cursors, ref cursors, and collections but they really don't populate anything, or don't have to populate anything

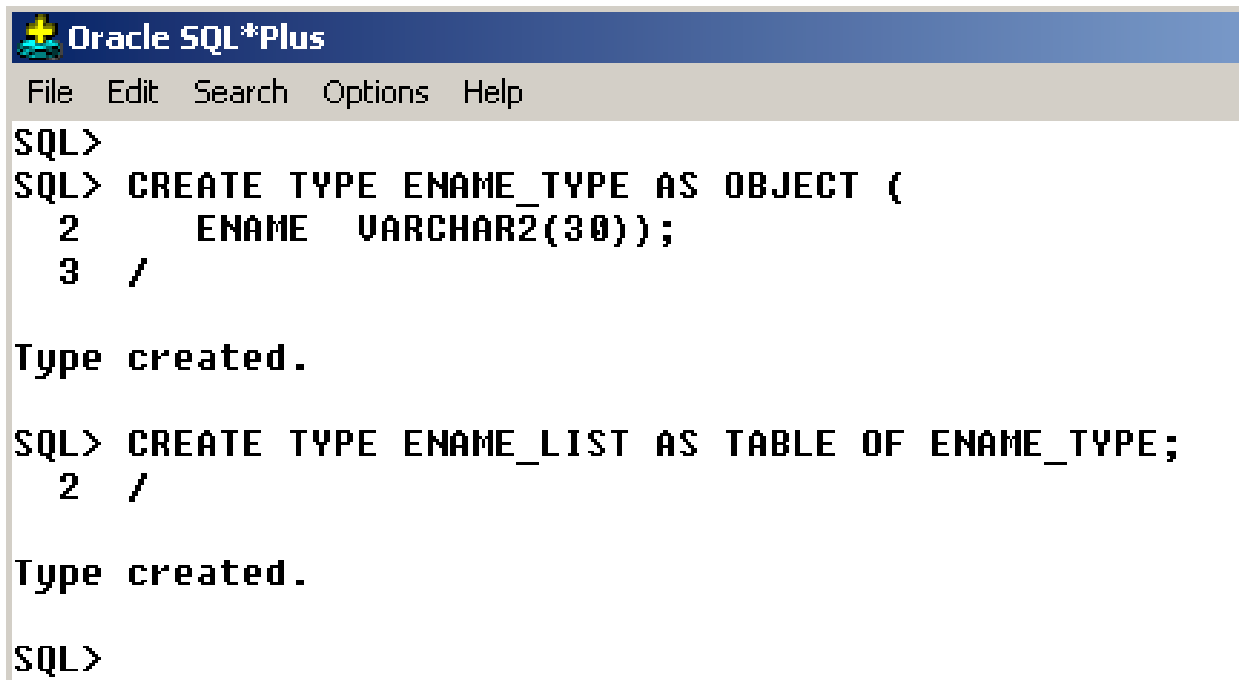
Pipelined Table Functions

- **Pipelined Table functions**
  - Works well with ref cursors
  - Uses keyword PIPELINED on RETURN clause
  - Uses keyword PIPE ROW to present output rows as they are accessed by function

Getting Started

Pipelined Table Functions

Return type must exist before compile time.



Pipelined Table Functions

Oracle SQL*Plus

File Edit Search Options Help

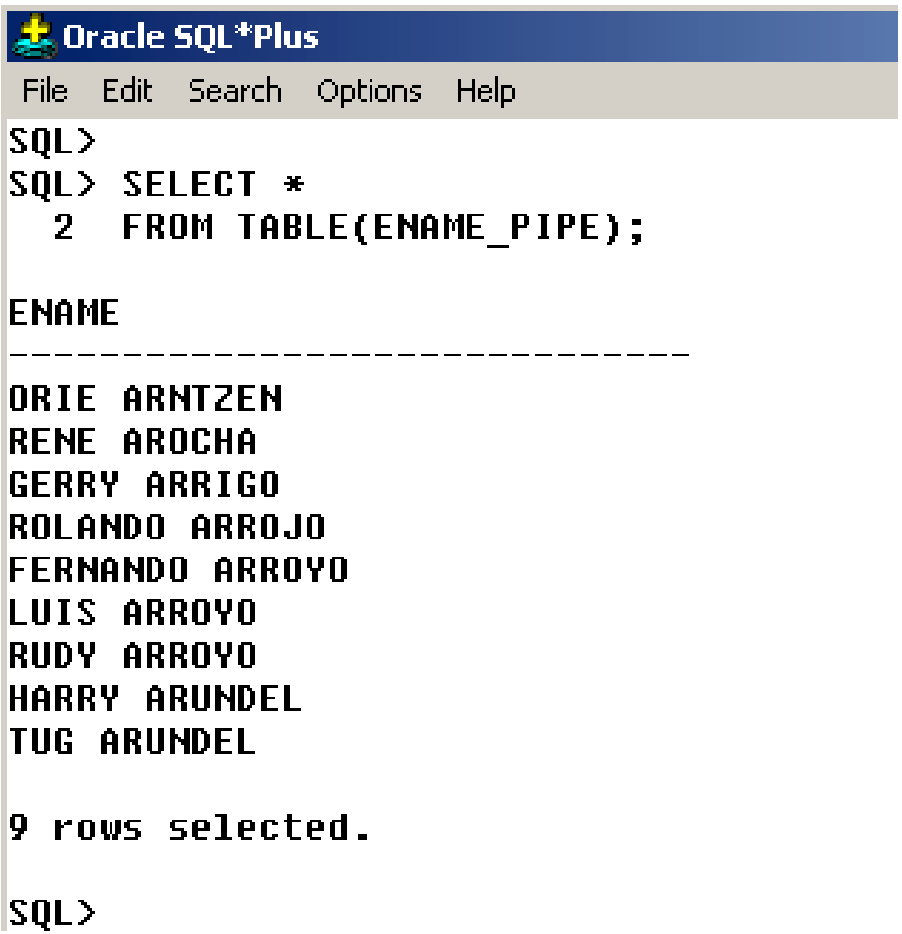
```
SQL>
SQL> CREATE OR REPLACE FUNCTION ENAME_PIPE
  2     RETURN ENAME_LIST PIPELINED
  3  IS
  4     v_EName  ENAME_TYPE;
  5     CURSOR C_EMP_INFO IS
  6        SELECT FIRST_NAME || ' ' || LAST_NAME ENAME
  7        FROM EMP_INFO
  8        WHERE EMPNO < 10;
  9  BEGIN
 10     FOR i In C_EMP_INFO
 11     LOOP
 12        V_EName := ENAME_TYPE(i.ENAME);
 13        PIPE ROW(v_EName);
 14     END LOOP;
 15     RETURN;
 16  END;
 17  /

Function created.

SQL>
```

Code in file: CH11_PIPELINETABLE.txt

Pipelined Table Functions



Cursors and Pipelined Functions

Working with Cursors and Pipelined Table Functions

- These functions allow for rows to be sent directly to the calling application as they are presented to a cursor
- These functions allow for Pipelined Table functions to take over for any implicit cursor that needs speeding up
- Works well with any cursor, ref cursors, and the generic SYS_REFCURSOR

Working with Cursors and Pipelined Table Functions

- **You can pass SQL into Pipelined Table functions**
  - SELECT * from TABLE (RUN_SQL ('SELECT ENAME from EMP');
- **You can Insert Into …Select from a table function**
- **You can use ROWNUM in Where clause**

Source: Oracle Corporation. 2013. "Oracle12c (12.1) PL/SQL Language Reference Guide." Last modified June 2013. E16655_01/appdev.121/e17622.pdf

Working with Cursors and Pipelined Table Functions

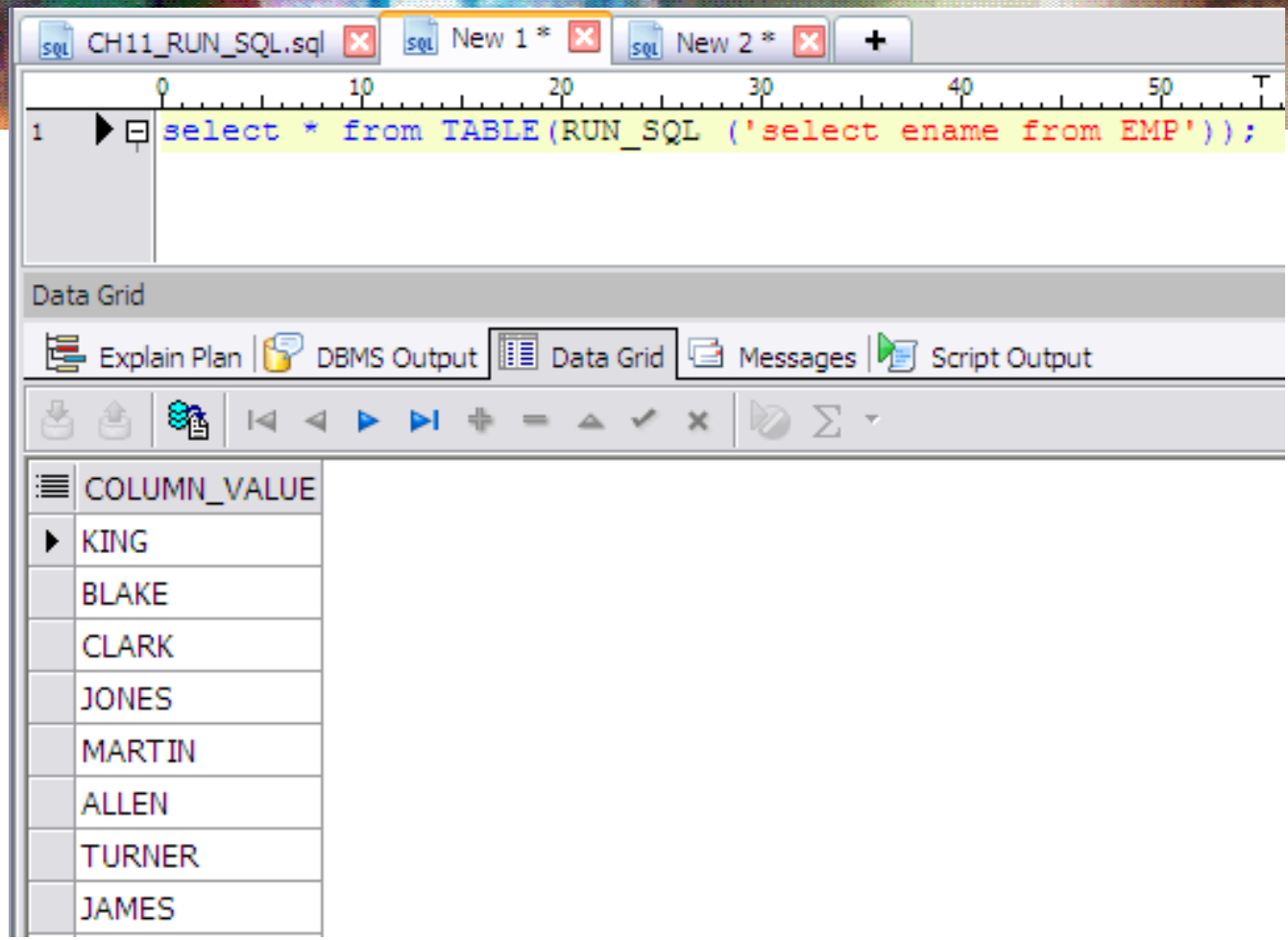
- **RUN_SQL**
  - Some assumptions:
    - Designed to select EMP from ENAME
    - Works like strong cursor
      - Make sure what you select matches up with defined TYPE
    - Can manipulate output
      - It's PL/SQL
  - Can use a package for global declaration of types, cursors, and collections
    - Easier to manage
    - Prevents errors at compile time

Code in file: CH11_RUN_SQL.sql

CH11_RUN_SQL.sql * | New 1 * | f() RUN_SQL | +

```
1  create or replace TYPE emp_ename_table IS TABLE OF varchar2(10);
2
3
4  create or replace FUNCTION RUN_SQL (IN_SQL_TEXT IN VARCHAR2) RETURN emp_ename_table PIPELINED
5  IS
6      type sql_cursor_type is REF CURSOR;
7      sql_cursor sql_cursor_type;
8
9      sql_rec emp_ename_table;
10
11 BEGIN
12     OPEN sql_cursor FOR IN_SQL_TEXT;
13     FETCH sql_cursor BULK COLLECT INTO sql_rec;
14     CLOSE sql_cursor;
15
16     FOR i in 1 .. sql_rec.COUNT
17     LOOP
18        PIPE ROW(sql_rec(i));
19     END LOOP;
20
21     RETURN;
22
23     END RUN_SQL;
24
```

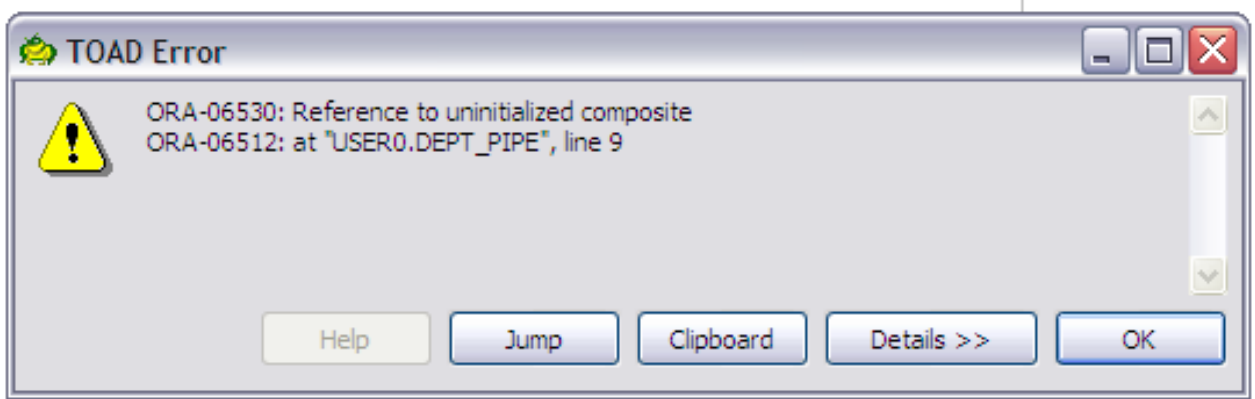
Code in file: CH11_RUN_SQL.sql


CH11_RUN_SQL.sql
New 1 *
New 2 *
0
10
20
30
40
50
1
select * from TABLE(RUN_SQL ('select ename from EMP'));
Data Grid
Explain Plan
DBMS Output
Data Grid
Messages
Script Output
COLUMN_VALUE
KING
BLAKE
CLARK
JONES
MARTIN
ALLEN
TURNER
JAMES

Working with Cursors and Pipelined Table Functions

Clean compiles, but at run time you get this error.
Appears to be a bug – when selecting one column, you don't need to initialize the TYPE variables. When working with multiple columns, you do.

```
   out_row  DEPT_TYPE;
 --  out_row  DEPT_TYPE := DEPT_TYPE(null, null, null);
   CURSOR C_DEPT IS
      SELECT * FROM DEPT;
BEGIN
   out_row.deptno := 0;
   FOR i In C_DEPT
   LOOP
      out_row.deptno := i
      PIPE ROW(out_row);
   END LOOP;
   RETURN;
END;
```



Working with Cursors and Pipelined Table Functions

This works fine.

```
1   CREATE TYPE DEPT_TYPE AS OBJECT (
2        deptno    NUMBER(2),
3        dname  VARCHAR2(14),
4        loc  VARCHAR2(13));
5
6
7   CREATE OR REPLACE FUNCTION dept_PIPE
8      RETURN DEPT_LIST PIPELINED
9   IS
10     out_row  DEPT_TYPE := DEPT_TYPE(null, null, null);
11     CURSOR C_DEPT IS
12        SELECT * FROM DEPT;
13  BEGIN
```

Chaining Pipelined Functions Together

Chaining Pipelined Table Functions

- **You can pass the output of one Pipelined Table function into another**
  - The two functions have to be coordinated so the output of the inner function data types is compatible with the input of the outer/calling function
- **This example sets up a cursor**
  - The cursor is perfect to restrict rows going into the Pipelined Table function

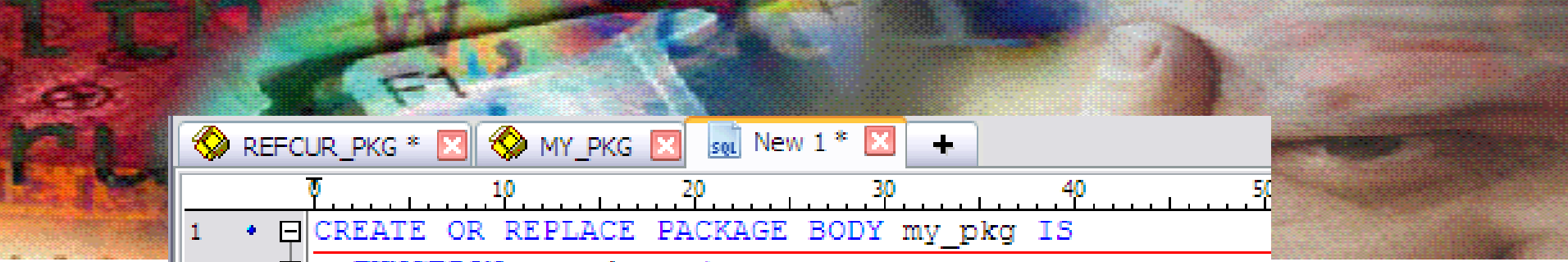
Source: Oracle Corporation. 2013. "Oracle12c (12.1) PL/SQL Language Reference Guide." Last modified June 2013. E16655_01/appdev.121/e17622.pdf

Chaining Pipelined Table Functions Code Example

REFCUR_PKG * | MY_PKG

```
1   CREATE OR REPLACE PACKAGE my_pkg AUTHID DEFINER IS
2     TYPE dept_refcur IS REF CURSOR RETURN dept%ROWTYPE;
3     TYPE deptrec_typ IS RECORD (
4       deptno     NUMBER(2),
5       dname   VARCHAR2(14),
6       loc   VARCHAR2(13)
7     );
8     TYPE outrecset IS TABLE OF deptrec_typ;
9     FUNCTION get_dept (p1 dept_refcur) RETURN outrecset PIPELINED;
10  END my_pkg;
```

Code in file: CH11_GET_DEPT.sql

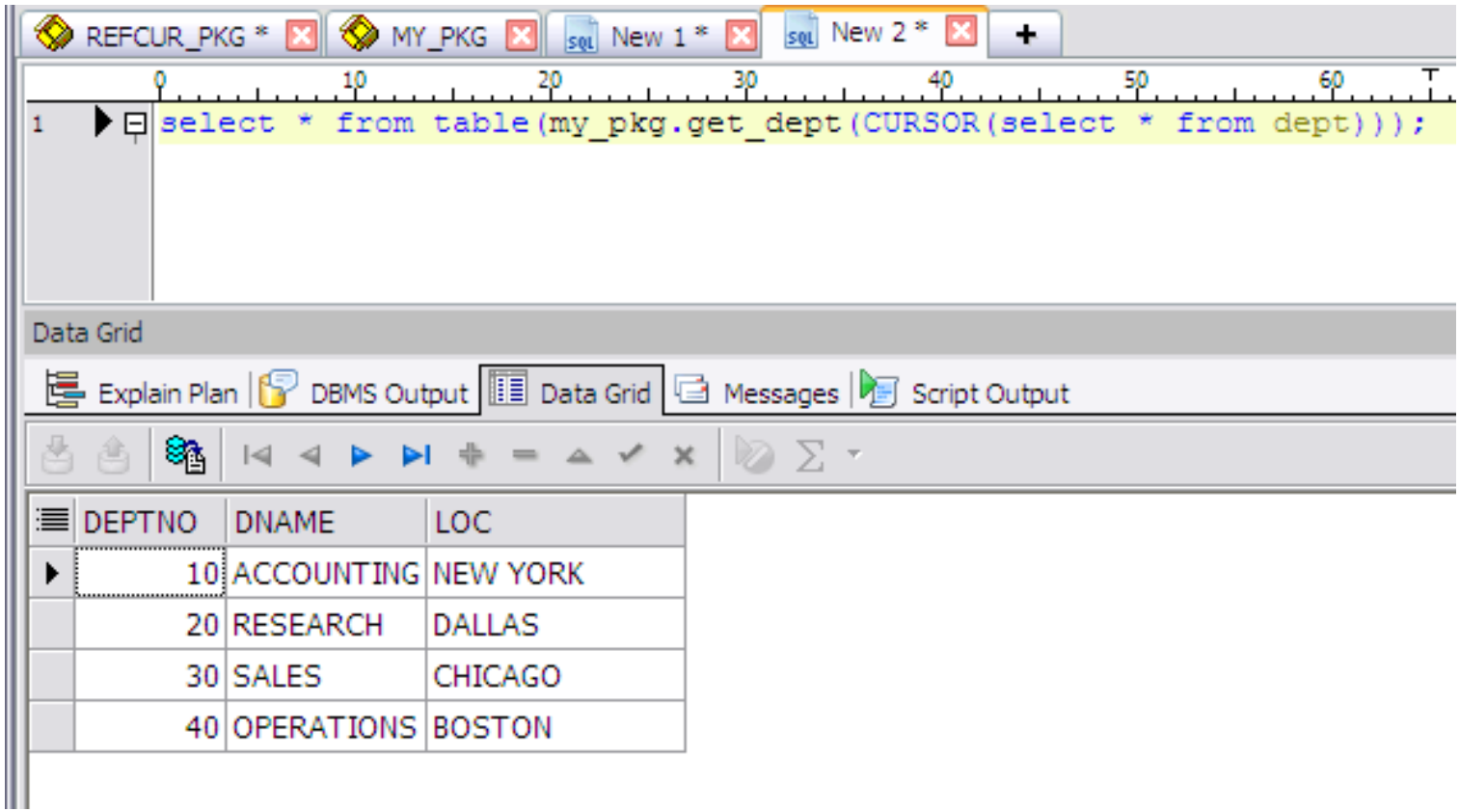


REFCUR_PKG * | MY_PKG | New 1 *

```
1  CREATE OR REPLACE PACKAGE BODY my_pkg IS
2    FUNCTION get_dept (
3      p1 dept_refcur
4    ) RETURN outrecset PIPELINED
5    IS
6      out_rec deptrec_typ;
7      in_rec p1%ROWTYPE;
8    BEGIN
9
10     LOOP
11       FETCH p1 INTO in_rec;
12       EXIT WHEN p1%NOTFOUND;
13       out_rec.deptno := in_rec.deptno;
14       out_rec.dname := in_rec.dname;
15       out_rec.loc := in_rec.loc;
16       PIPE ROW(out_rec);
17     END LOOP;
18     CLOSE p1;
19     RETURN;
20   END get_dept;
21 END my_pkg;
```

Code in file: CH11_GET_DEPT.sql

Chaining Pipelined Table Functions Code Example



Compiler Options for Table Functions

Working with Table Functions

- **Compilers Options**
  - PIPELINED
    - Required
  - PIPE ROW
    - Pushes rows/collection elements to calling application
  - DETERMINISTIC/RESULT_CACHE
    - Yes – these functions are SQL based, not outside-influence based
  - AUTONOMOUS_TRANSACTION
    - If function is running DML, this is a good idea
    - Is a Pragma clause

Working with Table Functions

- **Compilers Options (continued)**
  - PARALLEL_ENABLED ({PARTITION <name> BY [ANY | (HASH | RANGE) <column list>})
    - Good idea if function will access partitioned objects
    - PARTITION NAME
      - Use if there are interdependencies
    - ANY | HASH | RANGE
      - Type of partitioning scheme
      - ANY will randomly allocate source data to the participating processes
        - » Useful when there isn't any related data
      - HASH is useful to group data together by specific customers (for example)
      - RANGE is useful to ensure your data is processed in event order
  - Note: This feature is for Enterprise Edition of Oracle

Working with Table Functions

- **Additional parallel features**
  - Note: This feature is for Enterprise Edition of Oracle
  - Use the PARALLEL hint in your SQL
    - Sets the DOP per SQL
  - ALTER SESSION FORCE PARALLEL (QUERY) PARALLEL <number of processors to assign>;
  - Oracle SQL Tuning features apply to the SQL within these functions as well

Working with Table Functions

- **Restrictions**
  - Does not work with Associative Arrays
  - Does not work with PL/SQL data types
    - Return type cannot be pls_integer, boolean, and so on
- **Downfalls**
  - Not many really
  - Have to specifically code
  - Can lack flexibility
    - Cursors are like strong cursors: have to manage the return data types
  - Can add a level of complexity to applications

Table Functions Tips and Techniques

- **Useful for complex data loads and unloads**
  - One source, two or more targets
- **Useful to encapsulate certain complex calculations**
- **Performance gains can be seen when using these to return data to web-based applications or Java-based applications**
- **Exploit parallel server features**

Oracle12 New Features

- **Oracle12 new compiler feature for functions**
  - Pragma UDF
    - User-defined function
    - Used with functions called from SQL
    - Oracle12 will optimize for use with SQL
      - Should see better performance
      - Not quite sure what it does internally
    - Pragma clauses go anywhere between the IS/AS and the BEGIN
      - I put them before the BEGIN

What have we learned?

- **Pipelined Table Functions**
  - How they work
  - Getting Started
  - Cursors and Pipelined Functions
  - Chaining Pipelined Functions Together
  - Compiler Options for Pipelined Functions