EFFECTIVE USE

of

TABLE FUNCTIONS

By

Edward Kosciuszko

Kosware Inc.

Sequel@Optonline.net

10/7/2007

TABLE FUNCTIONS

Row Sources

- Tables
- Views
- In-line Views
- Nested Table/Varray Column
- Object Constructor

Starting in 9i

Procedurally Defined Rows

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: The Nested Table Column

Via Oracle manuals:

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(t1.people) t2
WHERE t2.department_id = t1.department_id;
```

Most examples contain in-line view:

```
SELECT t1.department_id, t2.*
FROM hr_info t1, TABLE(SELECT people FROM hr_info
                       WHERE department_id = t1.department_id) t2
```

Why not just join and save storage of duplicating key?

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(t1.people) t2
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: VARRAY Example

```
CREATE TYPE TelephoneObj
AS OBJECT (Type VARCHAR2(10), Ph_Number VARCHAR2(20))

 CREATE TYPE TelephoneTab AS VARRAY(6) OF TelephoneObj

 CREATE TABLE contacts
( fname          VARCHAR2(20),
 lname           VARCHAR2(40),
 telephones       TelephoneTab)

INSERT INTO contacts VALUES ('ed','smith',
     TelephoneTab (TelephoneObj ('HOME','973-220-2002'),
   TelephoneObj ('CELL','973-900-2021'))

INSERT INTO contacts VALUES ('jim','jones',
     TelephoneTab (TelephoneObj ('HOME','973-877-1001'),
   TelephoneObj ('CELL','201-762-3321'),
                    TelephoneObj ('WORK','201-887-0001')))
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: VARRAY Example

SELECT fname, lname, type, ph_number
FROM contacts c, TABLE (c.telephones)

FNAME	LNAME	TYPE	PH_NUMBER
ed	smith	HOME	973-220-2002
ed	smith	CELL	973-900-2021
jim	jones	HOME	973-877-1001
jim	jones	CELL	201-762-3321
jim	jones	WORK	201-887-0001

What about outer joins?

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: VARRAY Example

INSERT INTO contacts VALUES ('tom','jones', TelephoneTab ())

SELECT fname, lname, type, ph_number
FROM contacts c, TABLE (c.telephones) (+)

FNAME	LNAME	TYPE	PH_NUMBER
ed	smith	HOME	973-220-2002
ed	smith	CELL	973-900-2021
jim	jones	HOME	973-877-1001
jim	jones	CELL	201-762-3321
jim	jones	WORK	201-887-0001
tom	jones	(null)	(null)

WARNING!
ANSI
"LEFT JOIN"
does not work

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: Object Constructor

CREATE OR REPLACE TYPE IntArray AS VARRAY(25) OF INTEGER

SELECT * FROM TABLE (IntArray(1,2,3,4,5,6,7,8,9))

COLUMN_VALUE
1
2
3
4
5
6
7
8
9

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Looking Back: Object Constructor

Sequence Application

Total sales to date moving backwards in 3 month intervals

```
SELECT trunc(sysdate,'MONTH') –
          numtoyminterval (3*column_value,'MONTH'), sum(quantity_sold)
FROM TABLE (IntArray(1,2,3,4)), sales s
WHERE time_id >= trunc(sysdate,'MONTH') –
                    numtoyminterval (3*column_value,'MONTH')
GROUP BY trunc(sysdate,'MONTH') –
                    numtoyminterval (3*column_value,'MONTH')
```

- Sequential
- Random
- Whatever…

TRUNC(SYSDAT...	SUM(QUANTI...
01-NOV-99	5834093
01-FEB-00	4387667
01-MAY-00	3083209
01-AUG-00	1787246

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Now!: PL/SQL Function

SQL Defined Types

Returns VARRAY or Nested Table of either:

1. **Simple element of native datatype (e.g. NUMBER, VARCHAR2(20)**
2. **Structured element defined via CREATE TYPE**

```
FUNCTION <name> [(<parameters>)]   RETURN <array_type> IS
        <local temp array of array_type>
BEGIN
        <logic to populate the array>
        RETURN <local temp array>;
END;
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

SQL Defined Type Example

Converting string of choices into array

```
BEGIN
        FOR irec IN (SELECT cust_first_name, cust_last_name FROM customers
                WHERE cust_id IN
                        (SELECT column_value
                         FROM TABLE (Get_Selections ('40,120, 200,')) ) ) LOOP

                dbms_output.put_line (irec.cust_first_name||' '||irec.cust_last_name);

        END LOOP;
END;
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

SQL Defined Type Example

```
CREATE OR REPLACE TYPE SelectionsTab IS TABLE of VARCHAR2(10)

CREATE OR REPLACE FUNCTION Get_Selections (i_choices VARCHAR2)
                        RETURN SelectionsTab IS
                arg_start_position      INTEGER := 1;
                comma_position          INTEGER;
                temp_tab                SelectionsTab := SelectionsTab ();
                n_choices               INTEGER := 0;
                arg_length              INTEGER;
BEGIN

                LOOP
                        comma_position := INSTR(i_choices,',',arg_start_position, 1);
                        EXIT WHEN comma_position = 0;

                        n_choices := n_choices + 1;
                        temp_tab.EXTEND;

                        arg_length := comma_position - arg_start_position;

                        temp_tab(n_choices) := SUBSTR(i_choices, arg_start_position, arg_length);

                        arg_start_position := comma_position + 1;
                END LOOP;

                RETURN temp_tab;

END;
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS - Segue

Alternatives to Decompose String of Values

IN_Str = '40,120, 200,'

Common Usage

```
SELECT substr(str,1,instr(str,',',1)-1)
FROM     (SELECT substr(:IN_Str, instr(:IN_Str,',',-2,LEVEL)+1) str   FROM Dual
          CONNECT BY LEVEL <= length(:IN_Str) - length(replace(:IN_Str,',','')))
```

Better

```
SELECT regexp_substr(:IN_Str,'[^,]+',1,level) FROM dual
CONNECT BY LEVEL <= length(:IN_Str) - length(replace(:IN_Str,',',''))
```

SQL_TEXT	CPU_TIME
SELECT /* ED02 */ column_value FROM TABLE (Get_Selections ('40,120, 200, 250, 300,301,400,5...	5372
SELECT /* ED02 */ regexp_substr('40,120, 200, 250, 300,301,400,500,','[^,]+',1,level) FROM dual C...	1614
SELECT /* ED02 */ substr(str,1,instr(str,',',1)-1) FROM (SELECT substr('40,120, 200, 250, 300,301,...	3044

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Pipelined

- First Rows vs. All Rows
- Return Rows ASAP
- RETURN exits only
- No temp storage

```
FUNCTION <name> [(<parameters>)]   RETURN <array_type> PIPELINED IS
        [<simple element> <Oracle datatype> | <temp_object> <object_type>]
BEGIN
        [<temp_object is initialized>]
        <logic to populate to produce row for output>
        PIPE ROW (<simple element> | <temp_object> | <object_type(values)>) ;
        RETURN;
 END;
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Pipelined Example – Simple Element

```
CREATE TYPE NumTab IS TABLE OF NUMBER;


CREATE FUNCTION IntegerList (End_Int INTEGER)
                              RETURN NumTab PIPELINED IS
        ret_arg    NUMBER;
BEGIN
        FOR i IN 1..End_Int LOOP
                ret_arg := i;
                PIPE ROW (ret_arg);
        END LOOP;
        RETURN;
END;
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Pipelined Example – Structured Element

```
CREATE TYPE NumObject IS OBJECT (num_col NUMBER)

CREATE TYPE NumTab IS TABLE OF NumObject;


CREATE FUNCTION IntegerList (End_Int INTEGER) RETURN NumTab PIPELINED IS
        ret_arg    NumObject :=  NumObject(null);  -- initialize
BEGIN
        FOR i IN 1..End_Int LOOP
                ret_arg.num_col := i;
                PIPE ROW (ret_arg); -- object instance used
        END LOOP;
        RETURN;
END;
```

Previous example would fail with above CREATE TYPEs since PIPE ROW would require NumObject argument

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Pipelined Example – Structured Element

```
CREATE FUNCTION IntegerList (End_Int INTEGER)
                              RETURN NumTab PIPELINED IS
        ret_arg               NUMBER; -- simple, not NumObject type
BEGIN
        FOR i IN 1..End_Int LOOP

                ret_arg := i;
                PIPE ROW (NumObject(ret_arg));

        END LOOP;

        RETURN;
END;
```

Simple variable must be cast as NumObject

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

PL/SQL Defined Types

```
CREATE PACKAGE plsql_types IS
        TYPE SalesObjType IS RECORD (prod_id NUMBER(6,0), cust_id NUMBER);
        TYPE SalesTableType IS TABLE OF SalesObjType;
        FUNCTION Test  RETURN SalesTableType;
END;

CREATE PACKAGE BODY plsql_types IS

        FUNCTION Test  RETURN SalesTableType IS
                temp_tab  SalesTableType:= SalesTableType();
        BEGIN
                RETURN temp_tab;
        END test;
END;
```

OBJECT types not allowed

Package compiles, but function cannot be used as table function:
ORA-0092: invalid datatype

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

PL/SQL Defined Types

Solution: Convert to PIPELINED Function

```
CREATE PACKAGE plsql_types IS

        TYPE SalesObjType IS RECORD (prod_id NUMBER(6,0), cust_id NUMBER);
        TYPE SalesTableType IS TABLE OF SalesObjType;
        FUNCTION Test  RETURN SalesTableType PIPELINED;

END;

CREATE PACKAGE BODY plsql_types IS

        FUNCTION Test  RETURN SalesTableType PIPELINED IS
                temp_rec    SalesObjType ;
        BEGIN
                temp_rec := NULL;
                PIPE ROW (temp_rec);
                temp_rec.prod_id := 1;
                temp_rec.cust_id := 1002;
                PIPE ROW (temp_rec);

                RETURN;
        END test;
END;
```

TABLE defined in package must be PIPELINED

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Execution Plans

New Operations

- **COLLECTION INTERATOR**
  TABLE Operator exists
- **PICKLER FETCH**
  Output via PL/SQL Function
- **CONSTRUCTOR FETCH**
  Output via Object Constructor

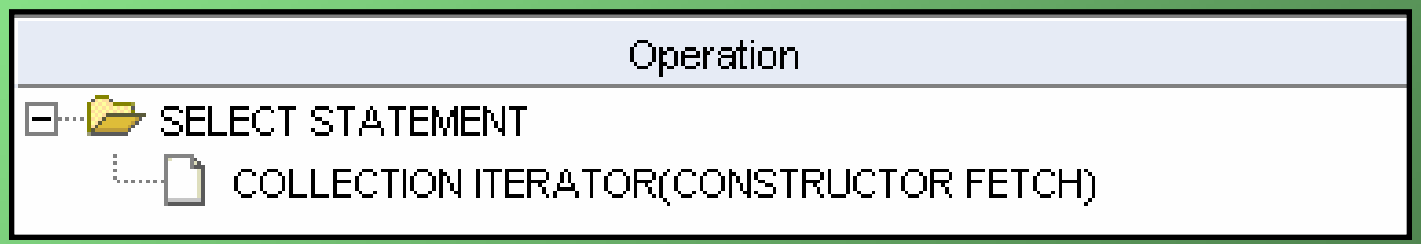
Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Execution Plans

SELECT * FROM TABLE (Get_PLSQL_Routines('SH'))

SELECT * FROM TABLE (Tab_Numbers(1,2,3))



Edward Kosciuszko
SeQueL@Optonline.net

Monitoring SQL – V$SQL

What you execute…

```
DECLARE
        temp_tab IntArray := IntArray(85, 105,250);
        cnt      INTEGER;
BEGIN
        SELECT count(*) INTO cnt FROM sales
        WHERE prod_id IN
                (SELECT column_value FROM TABLE (temp_tab));
        dbms_output.put_line ('count = '||cnt);
 END;
```

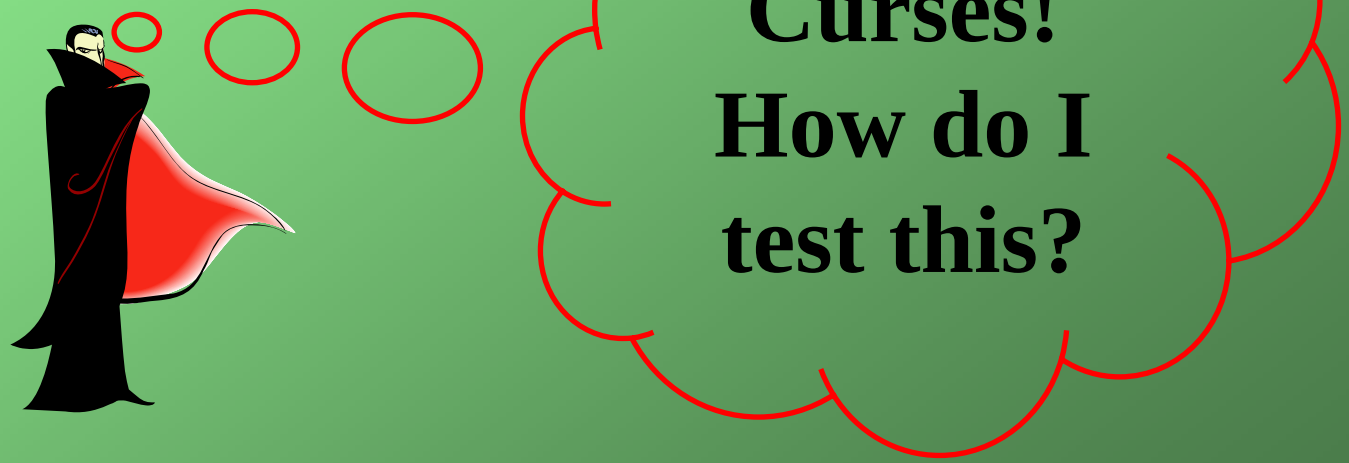

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Monitoring SQL – V$SQL

What V$SQL shows…

```
SELECT COUNT(*) FROM SALES WHERE PROD_ID IN
                (SELECT COLUMN_VALUE FROM TABLE (:B1 ))
```



Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Monitoring SQL – V$SQL

```
EXPLAIN PLAN FOR
SELECT COUNT(*) FROM SALES WHERE PROD_ID IN
                (SELECT COLUMN_VALUE FROM TABLE (:B1 ))
```

ORA-22905: cannot access rows from a nested-table item

SOLUTION: Cast the bind variable

```
EXPLAIN PLAN FOR
SELECT COUNT(*) FROM SALES WHERE PROD_ID IN
      (SELECT COLUMN_VALUE FROM TABLE (cast(:B1 as IntArray)
))
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Cardinality Hint

```
SELECT trunc(sysdate,'MONTH') - numtoyminterval (3*column_value,'MONTH'),
                    sum(quantity_sold)
FROM TABLE (IntArray(1,2,3,4)), sales s
WHERE time_id >= trunc(sysdate,'MONTH') –
                    numtoyminterval (3*column_value,'MONTH')
GROUP BY trunc(sysdate,'MONTH') - numtoyminterval(3*column_value,'MONTH')
```

9i Execution Plan

Operation	Cardinality	Partition Start	Partition Stop	Partition Id	ACCESS PREDICATES
SELECT STATEMENT	831309678				
SORT(GROUP BY)	831309678				
MERGE JOIN	831309678				
SORT(JOIN)	16360				
COLLECTION ITERATOR(CONSTRUCTOR FETCH)					
SORT(JOIN)	1016271				"S"."TIME_ID">=TRUNC(SYSDATE@!,'f...
PARTITION RANGE(ALL)		1	16	6	
TABLE ACCESS(FULL) SH.SALES	1016271	1	16	6	

16,360 rows?

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Cardinality Hint

CARDINALITY(<object>, <rows>) hint

4*50,814(Sales Rows) = 203,254 rows

Adding CARDINALITY hint to SQL

9i Execution Plan

Operation	Cardinality	Partition Start	Partition Stop	Partition Id	ACCESS PREDICATES
SELECT STATEMENT	203254				
SORT(GROUP BY)	203254				
NESTED LOOPS	203254				
COLLECTION ITERATOR(CONSTRUCTOR FETCH)					
PARTITION RANGE(ITERATOR)		KEY	16	4	
TABLE ACCESS(FULL) SH.SALES	50814	KEY	16	4	

Execution Statistics

SQL_TEXT	CPU	CLOCK	BUFFER_GETS	DISK_READS
SELECT trunc(sysdate - numtoyminterv...	3.218	5.982	2412	2499
SELECT /*+ cardinality(n,4) */ trunc(sys...	4.203	4.239	3010	0

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Cardinality Hint

```
CREATE TYPE NumbersType AS OBJECT (Num1 NUMBER, Num2 NUMBER)

CREATE TYPE NumbersTable AS TABLE OF NumbersType

DECLARE
        num_list NumbersTable:=NumbersTable(
                                NumbersType (1,2),
                                NumbersType (7,0019),
                                NumbersType (3,1999));
BEGIN
        FOR irec IN (SELECT /*+ cardinality(x, cardinality(num_list)) */
                        num1, num2 FROM TABLE (num_list) x)  LOOP

        END LOOP;

END;
```

CARDINALITY function

CARDINALITY hint

But this doesn't work because expression considered in comment (hint comment).

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Cardinality Hint

```
CREATE TYPE NumbersType AS OBJECT (Num1 NUMBER, Num2 NUMBER)
CREATE TYPE NumbersTable AS TABLE OF NumbersType

DECLARE
        num_list NumbersTable:=NumbersTable(  NumbersType (1,211),
                                              NumbersType (2,20019),
                                              NumbersType (3,1999));
        sql_text VARCHAR2(100) := 'SELECT /* cardinality(x,'||cardinality(num_list)||
              ') */ num1, num2 FROM customers c, TABLE (cast(:num_list as numberstable)) x '||
              ' where c.cust_city_id = num1';
BEGIN
    OPEN tmp_cur FOR sql_text USING IN  num_list;
    LOOP
        FETCH tmp_cur into temp1, temp2;
        EXIT WHEN tmp_cur%NOTFOUND;
    END LOOP;
    CLOSE tmp_cur;
END;
```

Must concatenate actual value.

Must bind nested table. Otherwise "invalid identifier".

TABLE FUNCTIONS

Warnings

Table Function Output

- Sorting/Grouping ?
- WHERE clause filters?
- Joins?

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Result Cache - 11g

- Retain query results for subsequent executions.
- Subquery factoring at inter-session level.

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Result Cache - 11g

```
CREATE PACKAGE CachedTest IS

…       FUNCTION EmpCnt (DeptnoIN NUMBER) RETURN DeptStatsTab
                RESULT_CACHE;

END CachedTest;

CREATE PACKAGE BODY CachedTest IS

        FUNCTION EmpCnt (DeptnoIN NUMBER) RETURN DeptStatsTab
                RESULT_CACHE RELIES_ON (emp, dept) IS

        …
```

- Oracle manuals appears to indicate must be performed in package?!
- "RETURN parameter of (or containing) object type" disallowed
- PIPELINED is disallowed

OK…I give up. How?

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Result Cache - 11g

RESULT_CACHE hint ?

Create stand-alone function, EmpCnt.

SELECT /*+ RESULT_CACHE */ FROM TABLE (EmpCnt(10))

Make function DETERMINISTIC and …

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Applications

- Complex SQL Requirement
- PL/SQL Coding Tool
- Simplify Client-Side Code
- Security – Auditing
- Oracle Bug Work-Arounds
- Ease of maintenance

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Complex SQL Requirement

Remember how the nested column is automatically in sync w/ table?

```
SELECT t1.department_id, t2.* FROM hr_info t1, TABLE(t1.people) t2
```

How about this?

```
SELECT* FROM table_of_strings t1, TABLE(get_selections(t1.text))
```

Recall the following is invalid:

```
FROM dept d, (SELECT ename FROM emp e
              WHERE e.deptno = d.deptno)
```

NOTE: No work w/ ANSI join

ID	TEXT	COLUMN_VALUE
1	1111,222,3333,	1111
1	1111,222,3333,	222
1	1111,222,3333,	3333
2	2211,112,3399,	2211
2	2211,112,3399,	112
2	2211,112,3399,	3399
3	aaa1,akaa,kgfkg,	aaa1
3	aaa1,akaa,kgfkg,	akaa
3	aaa1,akaa,kgfkg,	kgfkg

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Complex SQL Requirement

PROBLEM: Need to join to view which is many-table join and aggregated. Analytical functions are no solution.

```
CREATE VIEW sales_stats AS
        SELECT prod_id, cust_id, sum(amount_sold) FROM sales
        GROUP BY prod_id, cust_id

Following

        SELECT * FROM customers c, sales_stats s
        WHERE c.cust_id = s.cust_id
```


Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Complex SQL Requirement

```
CREATE TYPE SalesStatsObj IS OBJECT (prod_id NUMBER, tot_sold NUMBER)

CREATE TYPE SalesStatsTab IS TABLE OF SalesStatsObj


CREATE FUNCTION sales_stats (CustID_IN NUMBER)
                                RETURN SalesStatsTab IS

   temp_tab SalesStatsTab:=SalesStatsTab();

BEGIN

   SELECT SalesStatsObj(prod_id, sum(amount_sold))
   BULK COLLECT INTO temp_tab
   FROM sales
   WHERE cust_id = CustID_IN
   GROUP BY prod_id;

   RETURN temp_tab;

END;
```

Now join aggregates only what is required !

```
SELECT * FROM customers c,
       TABLE (sales_stats(c.cust_id) )
```

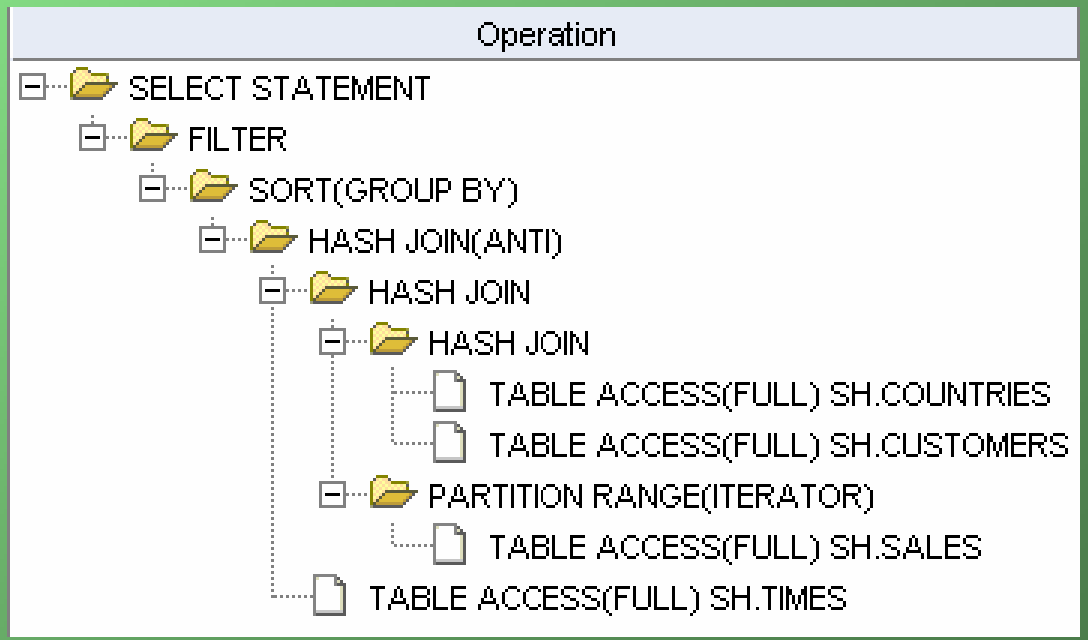

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

PL/SQL Coding Tool

```
SELECT c.cust_id, cust_last_name, sum(amount_sold) FROM customers c, sales s
WHERE c.cust_id = s.cust_id
AND country_id IN (SELECT country_id FROM countries WHERE country_region = 'Europe')
AND time_id NOT IN (SELECT time_id FROM times WHERE fiscal_year = 1994)
GROUP BY c.cust_id, cust_last_name HAVING count(*) > 0
```

Are you sure which step is 1st and which is next?



Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

PL/SQL Coding Tool

```
ALTER TABLE PLAN_TABLE ADD step INTEGER;

CREATE TYPE id_rec_typ AS OBJECT (id integer, parent_id integer, position integer, step integer);
CREATE  TYPE id_tbl_typ AS TABLE OF id_rec_typ;

CREATE OR REPLACE PROCEDURE GENERATE_STEPS  AS

        id_table id_tbl_typ := id_tbl_typ();
        <plus other variables>

        PROCEDURE process_children (par_id integer, next_step IN OUT integer) IS

        BEGIN

                FOR irec IN (Select * From Table (Cast(id_table As id_tbl_typ))
                        Where PARENT_ID = par_id Order by position) LOOP

                        <recursively process steps>

                END LOOP;
        END process_children;

BEGIN

    <logic update PLAN_TABLE steps>

END;
```

My tool within Generate_Steps procedure.

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

PL/SQL Coding Tool

Now…

1. **EXPLAIN PLAN FOR …**
2. **begin Generate_Steps; end;**
3. **SELECT …FROM plan_table**

STEP	LPAD('',LEVEL*8)||OPERATION||NVL2(OPTIONS,'('||OPTIONS||')',NULL)||NVL2(OBJECT_OV
11	SELECT STATEMENT Cost = 1118 CPU Cost = 613491972 IO Cost =1041
10	FILTER Filtered via: COUNT(*)>0
9	SORT (GROUP BY) Temp Space = 8,209,000 Cost = 1118 CPU Co
8	HASH JOIN (ANTI) Temp Space = 9,323,000 Accessed via: "
6	HASH JOIN Accessed via: "C"."CUST_ID"="S"."CUST_ID" Co
3	HASH JOIN Accessed via: "C"."COUNTRY_ID"="COUNTRIE
1	TABLE ACCESS (FULL) of Sh.Countries Filtered via
2	TABLE ACCESS (FULL) of Sh.Customers Cost = 73
5	PARTITION RANGE (ITERATOR)
4	TABLE ACCESS (FULL) of Sh.Sales Cost = 270 CP
7	TABLE ACCESS (FULL) of Sh.Times Filtered via: "TIMES"."FIS

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Ease of maintenance

Original problem: Optimize ORDER BY

```
ORDER BY DECODE( UPPER(:Sort1), 'DESC',
              (DECODE( UPPER(:Sort2), 'ORGANIZATION_CODE', ORG.ORGANIZATION_CODE,
                       'NAME', ORG.NAME, 'LAST_NAME', STUD.LAST_NAME,
                       'STUDENT_CODE', STUD.STUDENT_CODE,
                       'STUDENT_GRADE', STUD.GRADE)) DESC,
              DECODE( UPPER(:Sort2), 'ORGANIZATION_CODE', ORG.ORGANIZATION_CODE,
                       'NAME', ORG.NAME, 'LAST_NAME', STUD.LAST_NAME, 'STUDENT_CODE',
                       STUD.STUDENT_CODE, 'STUDENT_GRADE', STUD.GRADE) ASC
```

Developer allowed user to determine dynamically, how to sort data.

On 3 tier platform, and application permitted “next set of rows”.

REAL BAD PERFORMANCE !!

Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Ease of maintenance

Example Dynamic Sort Solution

```
SELECT
FROM (SELECT rownum rnum, cust_id, cust_last_name, cust_year_of_birth,
                          prod_id, quantity_sold
                FROM (SELECT c.cust_id,  cust_first_name, cust_last_name, cust_year_of_birth,
                                                          prod_id, quantity_sold
                          FROM customers c, sales s
                          WHERE c.cust_id = s.cust_id
                          AND s.channel = 2
                          ORDER BY decode (:choice, 'LAST NAME', cust_last_name,
                                                                'YEAR BORN', cust_year_of_birth) )
                WHERE rownum <= :EndRow)
WHERE rnum >= :StartRow
```

- Create table function w/ multiple SQL
- Each SQL represents sort option
- Tune each individual SQL

Edward Kosciuszko
SeQueL@Optonline.net

Optimizing Sort

```
SELECT *
FROM (SELECT /*+ use_nl(c,s) */ rownum rnum, c.cust_id, c.cust_last_name, s.quantity_sold
            FROM (SELECT /* index(customers, CUSTOMERS_LAST_NAME_INDEX) */
                        cust_id,  cust_first_name, cust_last_name, cust_year_of_birth
                        FROM customers
                        ORDER BY cust_last_name ASC) c, sales s
            WHERE c.cust_id = s.cust_id
            AND s.channel_id = 2
            AND rownum <= :EndRow)
WHERE rnum >= :StartRow
```

- Sort individual table 1st as in-line view
- Force Nested Loop join to ensure sort order

SQL_TEXT	ROWS_P...	CPU(secs)	CLOCK(secs)	BUFFER_GETS
SELECT /* NEW */ ...	40	0.051	0.056	2624
SELECT /* OLD */ ...	40	0.332	0.334	5964

Original problem improved 4 to 105 X's depending on sort

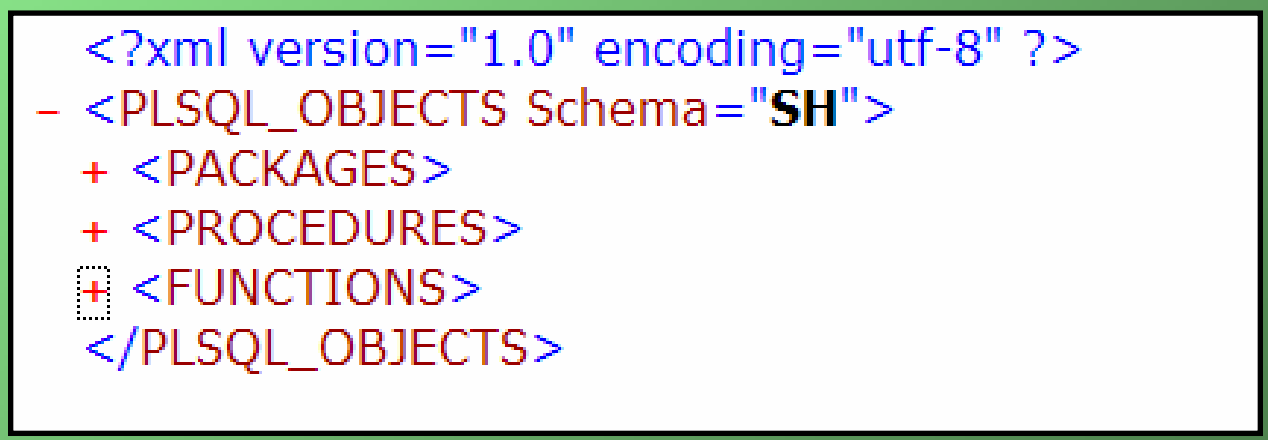
Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Simplify Client-Side Code

Application Requirement: Display navigator tree of all PL/SQL objects in specified schema in web browser via XML output.

SELECT * FROM TABLE (Get_PLSQL_Routines('SH'))



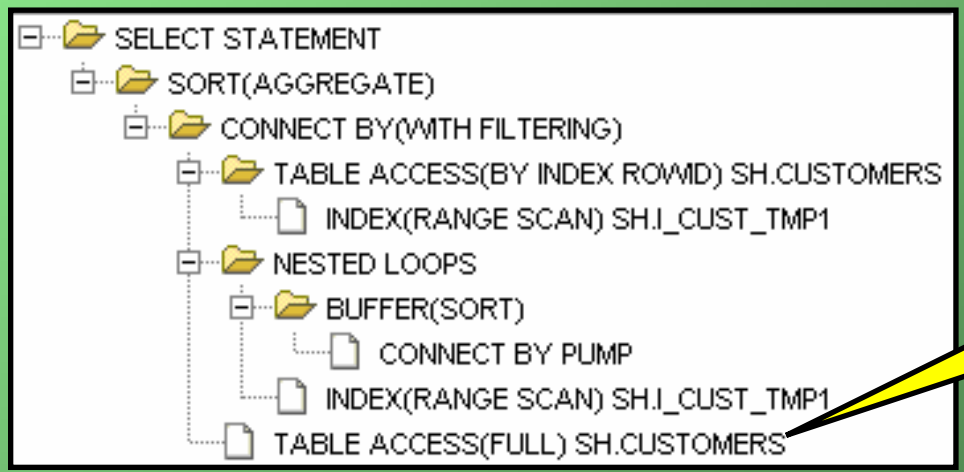
Edward Kosciuszko
SeQueL@Optonline.net

Oracle Bug Work-Arounds

Oracle 10g Error: Full Table Scan for CONNECT BY

```
UPDATE Customers SET Temp_ID = ROWNUM

SELECT  count(*) FROM Customers START WITH Temp_ID = 55000
CONNECT BY PRIOR Temp_ID + 1 = Temp_ID
```



Imagine on Oracle Grid ?

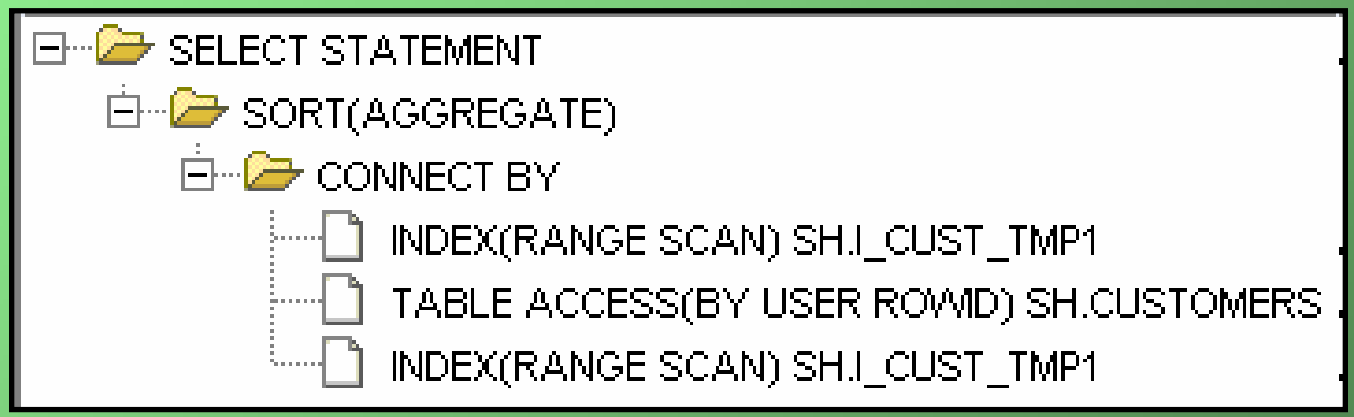
Edward Kosciuszko
SeQueL@Optonline.net

TABLE FUNCTIONS

Oracle Bug Work-Arounds

ALTER SYSTEM SET "_old_connect_by_enabled"=FALSE

- **Reverts to 9i optimizer**
- **10g CONNECT BY features unavailable**



Solution: Recursive Table Function

Edward Kosciuszko
SeQueL@Optonline.net