

wednesday 26th of march 2025, Database Synergy Day 2025

Table Functions at it's best

Oliver Lemm

code of change



about me

Deputy Head of Business Unit APEX / Low Code @ Hyand
Architect, Projekt Manager & Developer



High delivery reliability and international network of locations

Hamburg
Wolfsburg
2 x Braunschweig
2 x Berlin
Dortmund
Ratingen
Cologne
Frankfurt
Munich
Wrocław
Kaunas
Vilnius

Cluj-Napoca

Pune

> 900

Employees

(of which 160 in shoring
locations)

> 150

Clients

> 10

Industries

16

Locations

Agenda

1. Basics
2. Alternatives
3. Performance
4. Error Handling
5. APEX
6. Conclusion

Basics

Table Function

- define a type for a single row
- define a type as table
- define a function returning the table type
- defined for PL/SQL usage

```
create or replace type id_description_t force as object  
(  
    id            number,  
    description   varchar2(50)  
); /
```

```
create or replace force type id_descriptions_t as table of id_description_t;  
/
```

```
create or replace function tf_get_id_desc(i_rows in number)  
return id_descriptions_t as  
    l_table id_descriptions_t := id_descriptions_t();  
begin  
    for i in 1 .. i_rows  
    loop  
        l_table.extend;  
        l_table(l_table.last) := id_description_t(i, 'Row ' || i);  
    end loop;  
    return l_table;  
end;  
/
```

Table Function

- since Oracle 12.2 the query does not need table before your table function

select * from **table**(tf_get_id_desc(i_rows => 5))

except for functions without parameters which are called like:

select * from table(tf_get_id_desc)

```
create or replace type id_description_t force as object
(
  id          number,
  description  varchar2(50)
); /
```

SQL		Output	Statistics
		<pre>select * from tf_get_id_desc(i_rows => 5)</pre>	
		ID	DESCRIPTION
		1	Row 1
		2	Row 2
		3	Row 3
		4	Row 4
		5	Row 5

Best Practices

- use CREATE or REPLACE

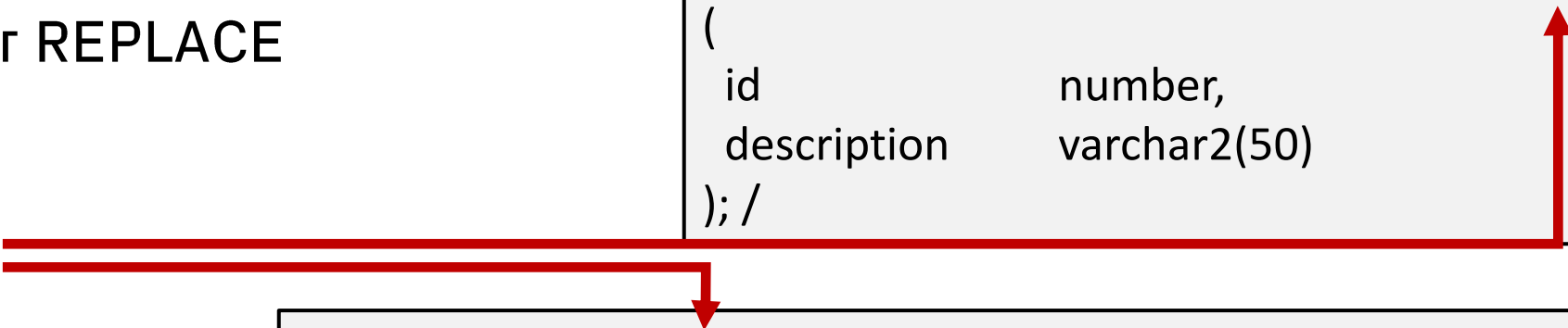
- use FORCE

- NAMING Guidelines

types => ..._t

table function => tf_...

```
create or replace type id_description_t force as object  
(  
  id          number,  
  description varchar2(50)  
); /
```



```
create or replace force type id_descriptions_t as table of id_description_t;  
/
```



```
create or replace function tf_get_id_desc(i_rows in number)  
return id_descriptions_t as  
  
  ...  
end;  
/
```

Best Practices

- define table function inside PACKAGES

- use named Syntax

Parameters can't be mixed up

Adding new attributes inside type won't affect
code negative

```
37  function ptf_emp_manager return emps_t
38      is
39
40  begin
41      for employee in (select e.empno
42                        ,e.ename
43                        ,e.sal
44                        ,mgr.empno mgr_no
45                        ,mgr.ename mgr_name
46                      from emp e
47                      left join emp mgr on mgr.empno = e.mgr)
48  loop
49      pipe row(new emp_t(emp_no => employee.empno
50                        ,emp_name => employee.ename
51                        ,sal      => employee.sal
52                        ,mgr_no   => employee.mgr_no
53                        ,mgr_name => employee.mgr_name));
54  end loop;
55
56  return;
```



Pipeline Table Function

- less code
- better performance
- subsequent processing before all rows processed
- “empty” return
- defined for SQL usage

```
create or replace function ptf_id_descriptions(i_rows in number)
return id_descriptions_t
  pipelined as
begin

  for i in 1 .. i_rows
  loop
    pipe row(id_description_t(1, 'Row ' || i));
  end loop;

  return;
end;
/
```

Type in PL/SQL or standalone Object

- standalone type —————→
for reuse in different areas
- creating type part of a package —————→
for use only in this package or table function
- define a standard in your project

```
create or replace type emp_t force as object
(
  emp_no    number,
  emp_name  varchar2(50 char),
  sal       number,
  mgr_no    number,
  mgr_name  varchar2(50 char)
)
;
```

```
1 create or replace package table_functions_pkg is
2
3   type emp_t is record(
4     emp_no    number
5     , emp_name varchar2(50 char)
6     , sal      number
7     , mgr_no   number
8     , mgr_name varchar2(50 char));
```

Focus on Pipeline Table Functions using in SQL context

Alternatives

Tables

- advantages
 1. for simple queries fastest way to develop

- disadvantages
 1. no reuse
 2. structure of tables must be known
 3. complex scenarios mixing up query and application logic

SQL Output Statistics											
<pre> select d.deptno ,d.dname ,d.loc ,e.empno ,e.ename ,e.job ,e.mgr ,e.hiredate ,e.sal ,e.comm from dept d left join emp e on e.deptno = d.deptno order by d.dname ,e.ename </pre>											
	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE			
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.1981			
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT		17.11.1981			
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.1982			
4	40	OPERATIONS	BOSTON								
5	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	12.01.1983			
6	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12.1981			
7	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04.1981			
8	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	09.12.1982			
9	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.1980			
10	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02.1981			
11	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05.1981			
12	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12.1981			
13	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09.1981			
14	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	08.09.1981			
15	30	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02.1981			

Views

- advantages

1. data relation encapsulated view
2. reuseable
3. complexity partly manageable

- disadvantages

1. complex scenarios mixing up query and application logic
2. plsql and sql mixing can be difficult to read

SQL	Output	Statistics
<pre>create or replace force view dept_emp_v as select d.deptno ,d.dname ,d.loc ,e.empno ,e.ename ,e.job ,e.mgr ,e.hiredate ,e.sal ,e.comm from dept d left join emp e on e.deptno = d.deptno order by d.dname ,e.ename</pre>		

SQL

Output Statistics

```
select d.deptno,
       d.dname,
       d.loc,
       d.empno,
       d.ename,
       d.job,
       d.mgr,
       d.hiredate,
       d.sal,
       d.comm
from dept_emp_v d
```

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE
▶ 1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09-08-79
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT		17-11-79
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23-07-79
4	40	OPERATIONS	BOSTON					
5	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	12-08-79
6	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03-12-73
7	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02-08-73
8	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	09-04-77
9	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	05-07-76

Materialized Views

- advantages
 1. relations over many tables persistently saved
 2. bring data from different sources together
 3. good for interface usage
 4. can be timesaving during query
- disadvantages
 1. only readable
 2. another data structure
 3. no choice for huge data
 4. no “live” data

```

Dialog Editor
create materialized view dept_emp_mv
build immediate
refresh force
on demand
as
select d.deptno
      ,d.dname
      ,d.loc
      ,e.empno
      ,e.ename
      ,e.job
      ,e.mgr
      ,e.hiredate
      ,e.sal
      ,e.comm
from dept d
left join emp e on e.deptno = d.deptno
order by d.dname
      ,e.ename
    
```

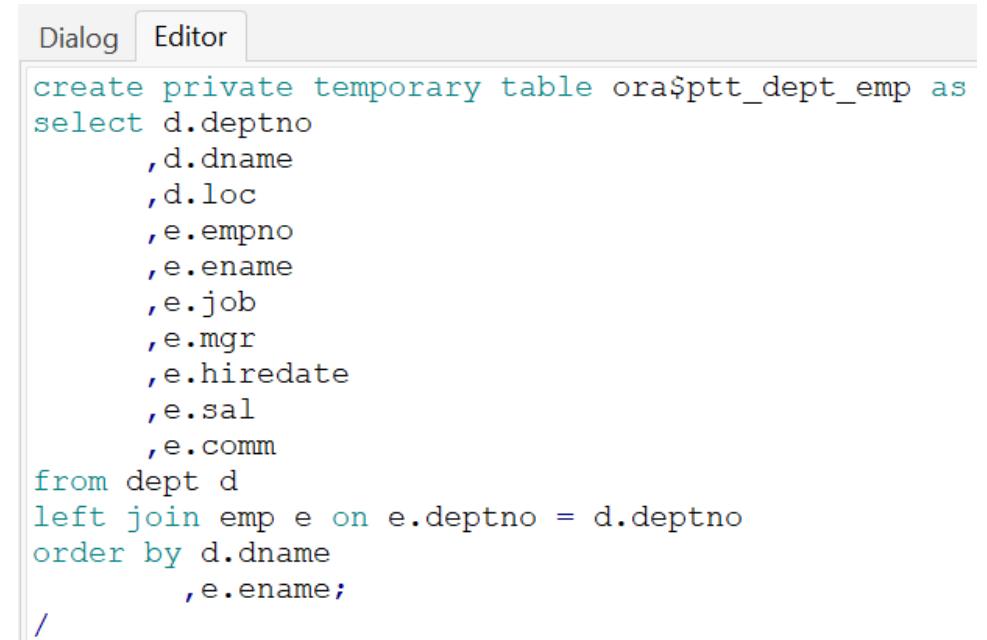
SQL Output Statistics

```
select * from dept_emp_mv t
```

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.1
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT		17.11.1
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.1
4	40	OPERATIONS	BOSTON					
5	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	12.01.1
6	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12.1
7	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04.1
8	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	09.12.1
9	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.1
10	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02.1
11	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05.1
12	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12.1
13	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09.1
14	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	08.09.1
15	20	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02.1

Temporary Table

- advantages
 1. bring data from different sources together
 2. no need to clear (only persistent in session)
 3. good for temporary use
- disadvantages
 1. needs to be fulfilled in every session
 2. another data structure
 3. no choice for huge data
 4. no “live” data
 5. Not useable combined with APEX



```
Dialog Editor
create private temporary table ora$ptt_dept_emp as
select d.deptno
      ,d.dname
      ,d.loc
      ,e.empno
      ,e.ename
      ,e.job
      ,e.mgr
      ,e.hiredate
      ,e.sal
      ,e.comm
from dept d
left join emp e on e.deptno = d.deptno
order by d.dname
        ,e.ename;
/
```

Table Function

- advantages
 1. query “complex” livedata
 2. combine data and logic
 3. readable code
- disadvantages
 1. only readable data
 2. another data structure
 3. overhead in code
 4. Index not in use

```
create or replace function tf_get_id_desc(i_rows in number)
return id_descriptions_t as
  l_table id_descriptions_t := id_descriptions_t();
begin
  for i in 1 .. i_rows
  loop
    l_table.extend;
    l_table(l_table.last) := id_description_t(i, 'Row ' || i);
  end loop;
  return l_table;
end;
/
```

Pipeline Table Function

- advantages

1. query “complex” livedata
2. faster than table function
3. less memory (PGA) usage
4. combine data and logic
5. readable code

- disadvantages

1. only readable data
2. another data structure
3. “small” overhead in code
4. Index not in use

```
create or replace function ptf_id_descriptions(i_rows in number)
return id_descriptions_t
  pipelined as
begin

  for i in 1 .. i_rows
  loop
    pipe row(id_description_t(1, 'Row ' || i));
  end loop;

  return;
end;
/
```

Performance

Context Switch

- Every switch between SQL and PL/SQL Interpreter costs

```

3  function ptf_emp_manager_context_switch return emp_t
4  pipelined is
5      l_mgr_name emp.ename%type;
6  begin
7      for employee in (select e.empno
8                          ,e.ename
9                          ,e.mgr
10                         from emp e)
11  loop
12      select e.ename
13      into l_mgr_name
14      from emp e
15      where e.empno = employee.mgr;
16
17      pipe row(new emp_t(employee.empno
18                          ,employee.ename
19                          ,employee.mgr
20                          ,l_mgr_name));
21  end loop;
22
23  return;
24  end ptf emp manager context switch;

```

- If possible call further information in one Query

```

30  for employee in (select e.empno
31                      ,e.ename
32                      ,mgr.empno mgr_no
33                      ,mgr.ename mgr_name
34                      from emp e
35                      left join emp mgr on mgr.empno = e.mgr)
36  loop
37      pipe row(new emp_t(employee.empno
38                          ,employee.ename
39                          ,employee.mgr_no
40                          ,employee.mgr_name));
41  end loop;

```

Deterministic

- Deterministic is an parameter for PL/SQL functions
- Return Value relies only on Input
⇒ same Input results in same return
- Caching works only in SQL

```
3 function f_salary_below_commission_n
4 (
5     i_salary    in emp.sal%type
6     ,i_commission in emp.comm%type
7 ) return number deterministic as
8 begin
9     return sys.diutil.bool_to_int(i_salary < i_commission);
10 end f_salary_below_commission_n;
```

SQL

Output

Statistics

```
select e.ename
       ,e.sal
       ,e.comm
       ,table_functions_pkg.f_salary_below_commission_n(i_salary => e.sal
                                                         ,i_commission => e.comm) sal_below_com
from emp e
where e.comm is not null
order by ename
```



	ENAME	SAL	COMM	SAL_BELOW_COM
1	ALLEN	1600,00	300,00	0
2	MARTIN	1250,00	1400,00	1
3	TURNER	1500,00	0,00	0
4	WARD	1250,00	500,00	0

Combination and Optimizer

- Never combine table function and pipeline table function you are losing both advantages and you get all disadvantages
- Table Functions are Black Boxes for Oracle Optimizer
- Optimizer estimates block size in execution plan
 - ⇒ Don't use table function for small data with performance needs

```
select emp_no
       ,emp_name
       ,sal
       ,mgr_no
       ,mgr_name
from table_functions_pkg.ptf_emp_manager()
```

Optimizer goal All rows

Tree HTML Text XML

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			29	8.168	32.672
COLLECTION ITERATOR PICKLER FETCH	TABLE_FUNCTIONS_PKG	PTF_EMP_MANAGER	29	8.168	32.672

```
select e.empno
       ,e.ename
       ,e.sal
       ,mgr.empno mgr_no
       ,mgr.ename mgr_name
from emp e
left join emp mgr on mgr.empno = e.mgr
```

Optimizer goal All rows

Tree HTML Text XML

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			6	14	392
HASH JOIN OUTER			6	14	392
NESTED LOOPS OUTER			6	14	392
STATISTICS COLLECTOR					
TABLE ACCESS FULL	DEMO	EMP	3	14	252
TABLE ACCESS BY INDEX ROWID	DEMO	EMP	3	1	10
INDEX UNIQUE SCAN	DEMO	EMP_PK			
TABLE ACCESS FULL	DEMO	EMP	3	14	140

Result Cache

- not supported



Error Handling

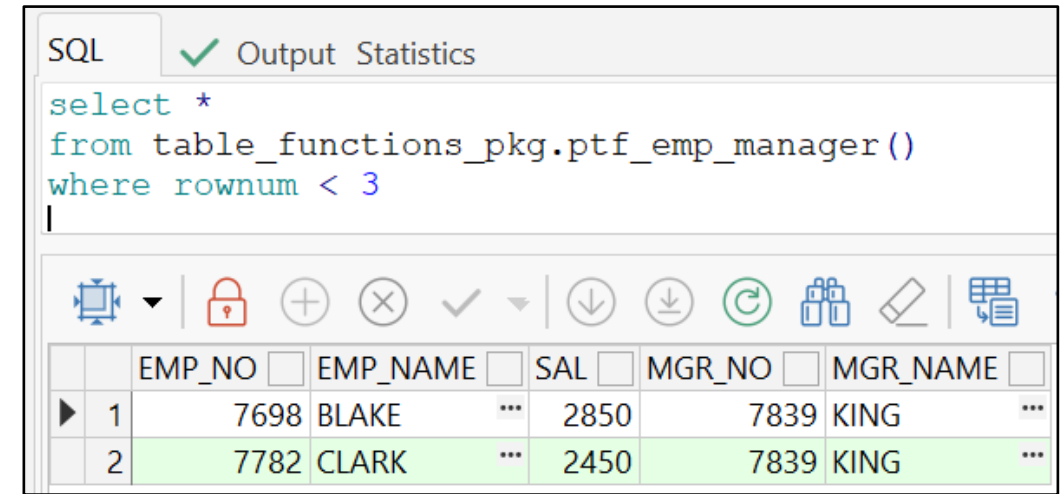
No data needed

- Problem

1. query data with table function
2. filter data afterwards in where clause

- Solution

1. filter with in-Parameter
2. use exception handling

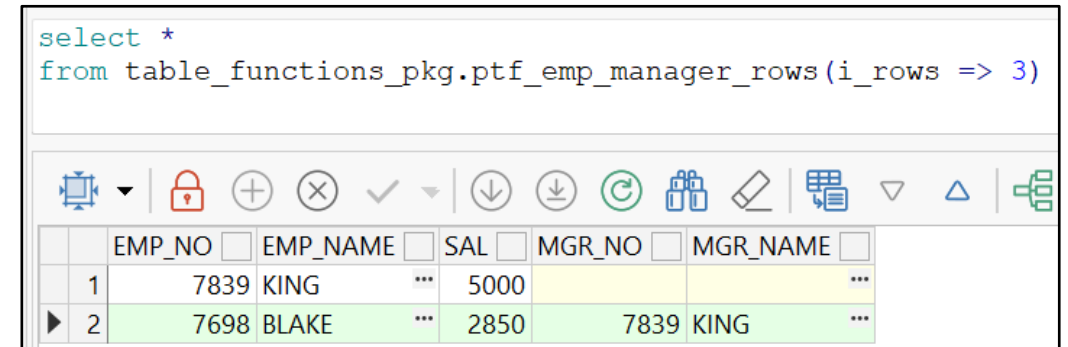


SQL ✓ Output Statistics

```
select *  
from table_functions_pkg.ptf_emp_manager()  
where rownum < 3  
|
```

		EMP_NO	EMP_NAME	SAL	MGR_NO	MGR_NAME
▶	1	7698	BLAKE	2850	7839	KING
	2	7782	CLARK	2450	7839	KING

ORA-06548: no more rows needed



```
select *  
from table_functions_pkg.ptf_emp_manager_rows(i_rows => 3)
```

		EMP_NO	EMP_NAME	SAL	MGR_NO	MGR_NAME
	1	7839	KING	5000		
▶	2	7698	BLAKE	2850	7839	KING

```
56 exception  
57 when no_data_needed then  
58 dbms_output.put_line('Try not to query more rows and filter afterwards.' || sqlerrm);
```

APEX

List of Value

- Source Type SQL Query needed
- Sort in your Table Function Query
- Additional columns for Popup LOV
- attention: Filtering works by adding WHERE

Create List of Values

✓ ✓ ● **List of Values Source**

Data Source ?

* Source Type ☐ Table ☒ **SQL Query** ☐ Function Body Returning SQL ?

* Enter a SQL SELECT statement ?

↶ ↷ 🔍 ↗ A:: ✓

```
1 select emp_no
2      ,emp_name
3      ,sal
4      ,mgr_no
5      ,mgr_name
6 from table_functions_pkg.ptf_emp_manager()
```

Additional Display Columns

Additional display columns can be defined for item types that support multiple display columns, for example the Popup LOV. For item types that do not support multiple display columns, the return column is included in the column list. The return column can be set to Visible No and Searchable No if needed.

Edit

≡	Sequence ↑≡	Column Name	Heading	Data Type	Visible	Searchable
≡	10	EMP_NO	-	NUMBER	No	No
≡	20	EMP_NAME	Name	VARCHAR2	Yes	Yes
≡	30	SAL	Salary	NUMBER	Yes	Yes
≡	40	MGR_NAME	Manager	VARCHAR2	Yes	Yes

Interactive Report

- Source Type SQL Query
- Filtering
 1. adds WHERE clause
 2. fires every time filter is adjusted
- Combine Pipeline Table function with APEX Collections for IR with heavy usage

Create Interactive Report

Page Definition

* Page Number ?

* Name ?

Page Mode **Normal** Modal Dialog Drawer ?

Include Form Page ☐ ?

Data Source

Data Source ?

Source Type **Table** **SQL Query** ?

* Enter a SQL SELECT statement

↶

↷

🔍

🔧

A::

✓

```
1 select emp_no
2      ,emp_name
3      ,sal
4      ,mgr_no
5      ,mgr_name
6 from table_functions_pkg.ptf_emp_manager()
```

Q

▼

Go

Actions

▼

▶

▼

Mgr Name

Emp No	Emp Name	Sal	Mgr No	Mgr Name
7788	SCOTT	3000	7566	JONES
7902	FORD	3000	7566	JONES

1 - 2

Conclusion

Conclusion

1. Scenario to use Historical + App Data

- Historical Data
- Application Data
- Results in a user-friendly Output who changed what attribute when

```
create table HISTORISIERUNGEN
(
  hist_id                NUMBER not null,
  hist_tabellenname      VARCHAR2(32 CHAR) not null,
  hist_spaltenname       VARCHAR2(32 CHAR) not null,
  hist_feldwert_alt      VARCHAR2(4000 CHAR),
  hist_feldwert_neu      VARCHAR2(4000 CHAR),
  hist_aenderungstyp     VARCHAR2(8 CHAR),
  hist_geaendert_am      DATE not null,
  hist_geaendert_von     VARCHAR2(255 CHAR) not null,
  hist_fost_key          VARCHAR2(1 CHAR),
  hist_fond_id           NUMBER,
  hist_ants_id           NUMBER,
  hist_apex_page         NUMBER,
  hist_sicht             VARCHAR2(1 CHAR),
  hist_fond_fondsprofil_version NUMBER,
  hist_app_session_id    VARCHAR2(32 CHAR),
)
```

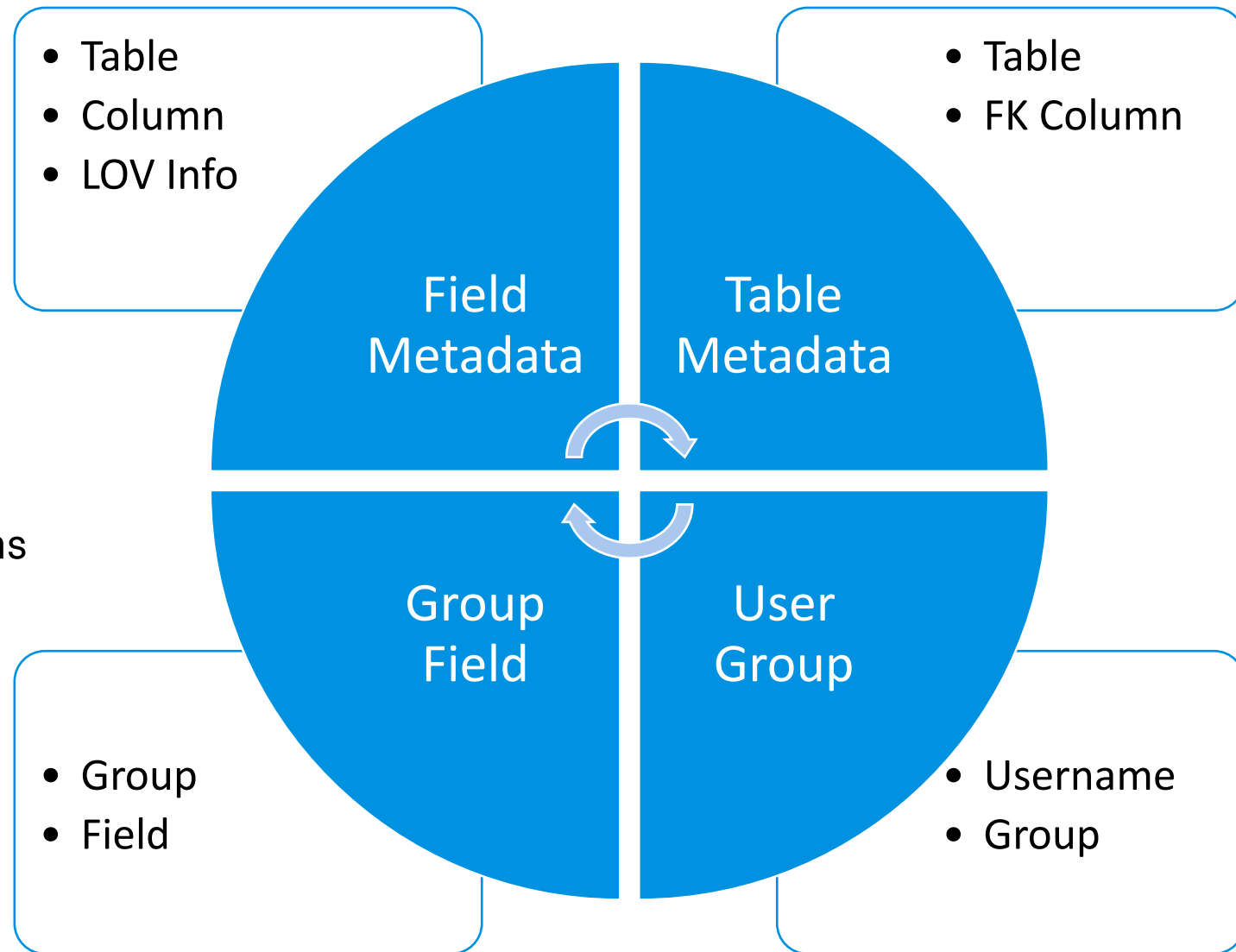
SQL Output Statistics

```
select i.page_id
      ,i.page_name
      ,i.region
      ,i.label
      ,i.display_as
from apex_application_page_items i
join apex_application_page_regions r on r.region_id = i.region_id
                                   and r.application_id = i.application_id
where i.application_id = 200
order by i.page_id
        ,r.display_sequence
        ,i.display_sequence
```

	PAGE_ID	PAGE_NAME	REGION	LABEL	DISPLAY_AS
1	0	Global Page	Suchergebnisse	Suche	Text Field
2	1	Projektmeldung	Einstellungen	Abrechnungstyp	Radio Group
3	1	Projektmeldung	Projektinformationen Links	Rechnungsempfänger (Kunde)	Popup LOV
4	1	Projektmeldung	Upload		File Upload
5	1	Projektmeldung	Volumen pro Jahr		Hidden

2. Scenario to use Metadata & User rights

- Metadata for your Tables
- Relation between Tables
- User and Group Relations
- Relation between Groups and Data columns



Use when

- Read data
- mixing up PL/SQL and SQL
- SQL getting to complex
- Amount of context switches is not too high

Don't use when

- In simple PL/SQL context
- when amount of data is very low (100- rows) or very high (10k+ rows)
- Components are able to work with PTF (no extra WHERE)

Some useful links

- Steven Feuerstein - <https://blogs.oracle.com/connect/post/pipelined-table-functions>
- Steven Feuerstein - <https://www.youtube.com/watch?v=YNAdK3jqmEg>
- Oracle Base - <https://oracle-base.com/articles/misc/pipelined-table-functions>
- Jürgen Sieben - <https://www.informatik-aktuell.de/entwicklung/programmiersprachen/oracle-pl/sql-deterministische-funktionen-oder-result-cache.html>

Say Hy_

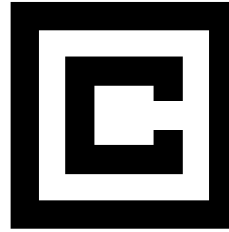
+49 (0) 2102 30 961-0

oliver.lemm@hyand.com

X [@OliverLemm](#)



[@oliverlemm.bsky.social](#)



© 2024 – The developed thoughts and ideas are the intellectual property of Hyand and are subject of copyright law. Reproduction, transfer to third parties or use – even of parts – is only permitted with the express of Hyand.