27.03.2026, APEX World 2026, Ede

Table Functions at it's best

Oliver Lemm

code of change

Hyand

Oliver Lemm

Deputy Head of Business Unit APEX / Low Code Architect, Project Manager & Developer

APEX connect

DOAG Konferenz + Ausstellung

ODTUG Kscope

IT-Tage 2026

ā'pěks world (#orclapex)

SOUG swiss oracle user group

Internationally connected delivery power.

Hyand

Germany

- Ratingen
- Brunswick
- Hamburg
- Dortmund
- Frankfurt
- Munich
- Berlin

Poland

- Wrocław

Lithuania

- Vilnius
- Kaunas

Romania

- Cluj-Napoca

India

- Pune

25+
Customers with > 1 billion € revenue

110+
Million Euros revenue (2024)

14
Locations in 5 countries

750+
Employees

24
Nationalities

Agenda

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

Motivation_

getting data
in SQL context
combined
with PL/SQL logic

using metadata
for
building Queries

Alternatives_

SQL Query on Table

- advantage
  - for simple queries fastest way to develop
- disadvantages
  - Logic not reuseable
  - structure of tables must be known
  - complex scenarios mixing up query and application logic
  - No “metadata” use inside “standard SQL”

SQL | Output | Statistics

```
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
```

	DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06.1981	2450,00
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT		17.11.1981	5000,00
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01.1982	1300,00
4	40	OPERATIONS	BOSTON						
5	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	12.01.1983	1100,00
6	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12.1981	3000,00
7	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04.1981	2975,00
8	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	09.12.1982	3000,00
9	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12.1980	800,00
10	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02.1981	1600,00
11	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05.1981	2850,00
12	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12.1981	950,00
13	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09.1981	1250,00
14	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	08.09.1981	1500,00
15	30	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02.1981	1250,00

SQL Query on Views

- advantages
  - data relation encapsulated view
  - reuseable
  - complexity partly manageable

- disadvantages
  - complex scenarios mixing up query and application logic
  - plsql and sql mixing can be difficult to read
  - no Metadata

SQL Output Statistics

```
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
```

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
1	10	ACCOUNTING	NEW YORK	7782	CLARK
2	10	ACCOUNTING	NEW YORK	7839	KING
3	10	ACCOUNTING	NEW YORK	7934	MILLER
4	40	OPERATIONS	BOSTON		
5	20	RESEARCH	DALLAS	7876	ADAM
6	20	RESEARCH	DALLAS	7902	FORD
7	20	RESEARCH	DALLAS	7566	JONES
8	20	RESEARCH	DALLAS	7788	SCOTT

Materialized Views

- advantages
  - relations over many tables persistently saved
  - bring data from different sources together
  - good for interface usage
  - can be timesaving during query

- disadvantages
  - only readable
  - another data structure
  - no choice for huge data (not normalized)
  - 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	HIRED
1	10	ACCOUNTING	NEW YORK	7782	CLARK	MANAGER	7839	09.06
2	10	ACCOUNTING	NEW YORK	7839	KING	PRESIDENT		17.11
3	10	ACCOUNTING	NEW YORK	7934	MILLER	CLERK	7782	23.01
4	40	OPERATIONS	BOSTON					
5	20	RESEARCH	DALLAS	7876	ADAMS	CLERK	7788	12.01
6	20	RESEARCH	DALLAS	7902	FORD	ANALYST	7566	03.12
7	20	RESEARCH	DALLAS	7566	JONES	MANAGER	7839	02.04
8	20	RESEARCH	DALLAS	7788	SCOTT	ANALYST	7566	09.12
9	20	RESEARCH	DALLAS	7369	SMITH	CLERK	7902	17.12
10	30	SALES	CHICAGO	7499	ALLEN	SALESMAN	7698	20.02
11	30	SALES	CHICAGO	7698	BLAKE	MANAGER	7839	01.05
12	30	SALES	CHICAGO	7900	JAMES	CLERK	7698	03.12
13	30	SALES	CHICAGO	7654	MARTIN	SALESMAN	7698	28.09
14	30	SALES	CHICAGO	7844	TURNER	SALESMAN	7698	08.09
15	30	SALES	CHICAGO	7521	WARD	SALESMAN	7698	22.02

Temporary Table

- advantages
  - bring data from different sources together
  - no need to clear (only persistent in session)
  - good for temporary usage

- disadvantages
  - needs to be fulfilled in every session
  - another data structure
  - no choice for huge data
  - no “live” data
  - 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;
/
```

SQL Query with using Analytic Functions

- advantage
  - fastest way aggregating data in SQL
  - parallelism

- disadvantages
  - Logic not reuseable
  - structure of tables must be known
  - Limited to predefined functions

AVG *	BIT_AND_AGG *	BIT_OR_AGG *	BIT_XOR_AGG *
CLUSTER_DETAILS	CLUSTER_DISTANCE	CLUSTER_ID	CLUSTER_SET
COUNT *	COVAR_POP *	COVAR_SAMP *	CUME_DIST
FEATURE_DETAILS	FEATURE_ID	FEATURE_SET	FEATURE_VALUE
FIRST_VALUE *	KURTOSIS_POP *	KURTOSIS_SAMP *	LAG
LAST_VALUE *	LEAD	LISTAGG	MATCH_RECOGNIZ
MEDIAN	MIN *	NTH_VALUE *	NTILE
PERCENTILE_CONT	PERCENTILE_DISC	PREDICTION	PREDICTION_COS
PREDICTION_COST	PREDICTION_DETAILS	PREDICTION_PROBABILITY	PREDICTION_SET
RATIO_TO_REPORT	REGR_ (Linear Regression) Functions *	ROW_NUMBER	SKEWNESS_POP *
STDDEV *	STDDEV_POP *	STDDEV_SAMP *	SUM *
VAR_SAMP *	VARIANCE *	String Aggregation	Top-N Queries

SQL Macros

- Scalar Expressions
  - select, where and having
- Table Expressions
  - from

- advantages
  - define your own “analytic function”
  - logic reuseable

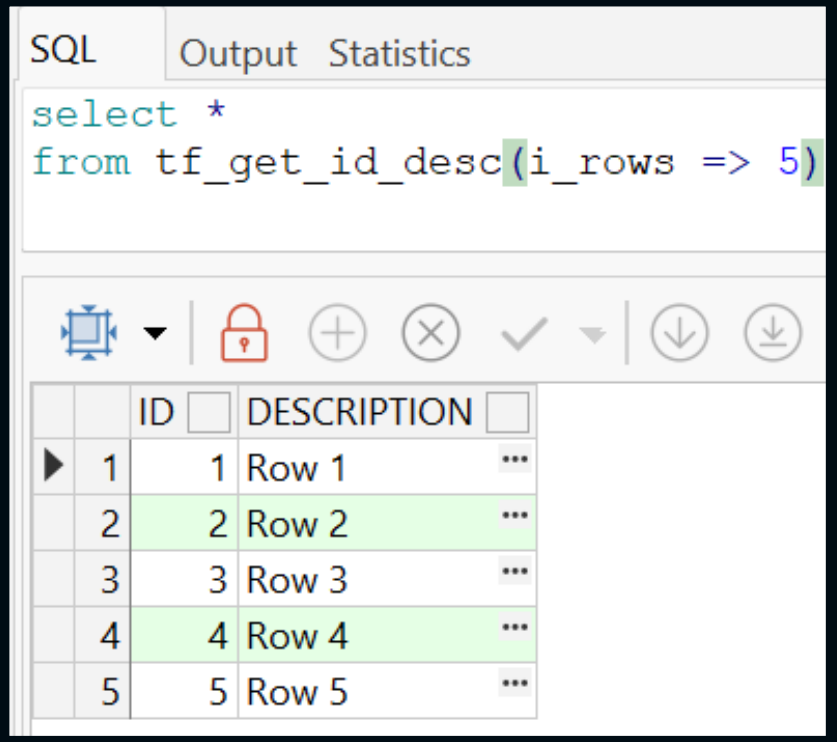
- disadvantages
  - for single column values -> scalar

```
create or replace function f_calculate_vat(pi_value numbe
  return varchar2 sql_macro(scalar)
is
begin
  return q'{ pi_value * 0.19 }';
end; /
```

```
create or replace function f_salary_by_dep
return varchar2 sql_macro(table)
is
begin
  return q'{
    select deptno, sum(sal) as salary_total
    from   emp
    group by deptno
  }';
end;
/
```

“standard” Table Function

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



Pipeline Table Function

- advantages
  - query “complex” livedata
  - faster than table function
  - less memory (PGA) usage
  - combine data and logic
  - readable code
- disadvantages
  - only readable data
  - another data structure
  - “small” overhead in code
  - Index not in use

```
select * from ptf_id_descriptions(i_rows => 5)
```

	ID	DESCRIPTION
1	1	Row 1
2	2	Row 2
3	3	Row 3
4	4	Row 4
5	5	Row 5

Focus on Pipeline Table Functions used in SQL context

Basics_

“standard” 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;
/
```

“standard” Table Function

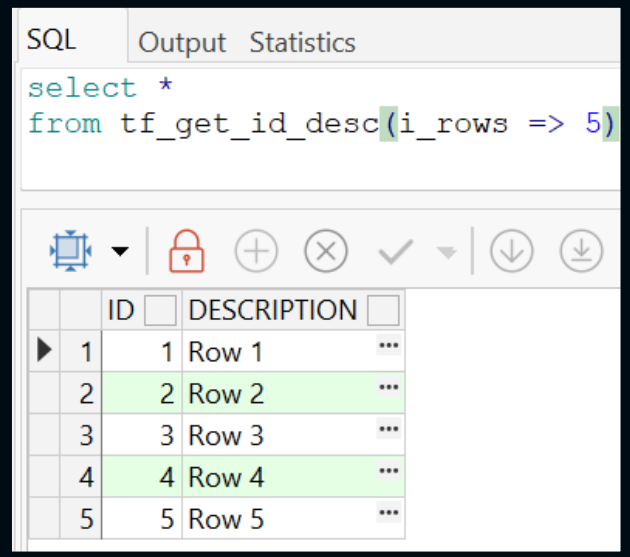
- since Oracle 12.2 the query does not need table before your table function

```
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)
); /
```



“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(I, 'Row ' || i));
  end loop;

  return;
end;
/
```

Type as Object or Record?

- standalone type “as object”
  - for reuse in different packages

```
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)
)
;
/
```

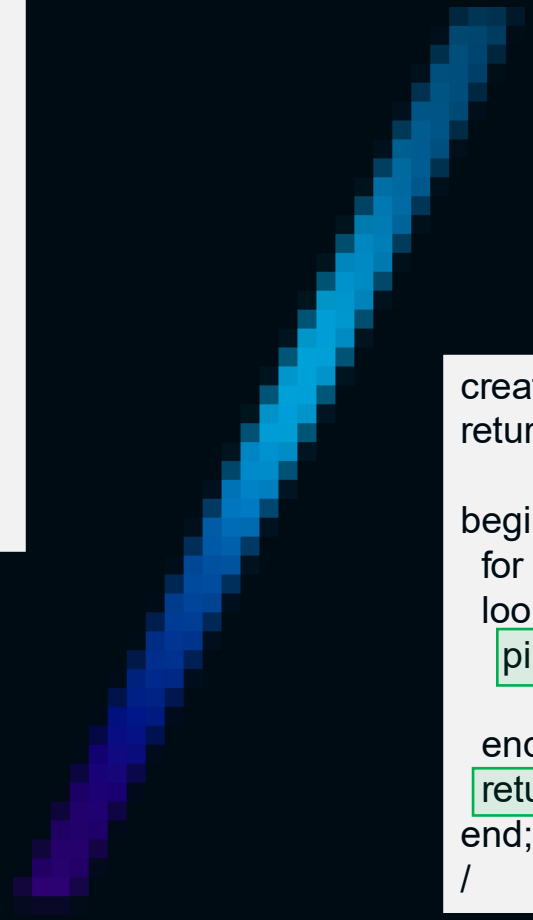
- creating type as part of a package “is record”
  - for use only in this package or table function
  - shadow types are created by oracle

```
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));
```

- define a standard in your project

Table Function

```
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

```
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(I, 'Row ' || i));

  end loop;
 return;
end;
/
```

Best Practices (1)

- use CREATE or REPLACE
- use FORCE for object types

```
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;
/
```

- NAMING Guidelines
  - types => …_t
  - table function => tf_...
  - pipeline table function => ptf_...

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

Best Practices (2)

- define table function inside PACKAGEs
  - no single function (or procedure)
- use named Syntax
  - Parameters can’t be mixed up
  - Adding new attributes inside type won’t affect code negative

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

Performance_

Context Switch

- Every switch between SQL and PL/SQL Interpreter costs

```
3  function ptf_emp_manager_context_switch return emps_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 in an attribute 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_commision_n
4   (
5     i_salaray    in emp.sal%type
6    ,i_commision in emp.comm%type
7   ) return number deterministic as
8   begin
9     return sys.diutil.bool_to_int(i_salaray < i_commision);
10  end f_salary_below_commision_n;
```

SQL Output Statistics

```
select e.ename
      ,e.sal
      ,e.comm
      ,table_functions_pkg.f_salary_below_commision_n(i_salaray    => e.sal
                                                     ,i_commision => e.comm)
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	32.67
COLLECTION ITERATOR PICKLER FETCH	TABLE_FUNCTIONS_PKG	PTF_EMP_MANAGER	29	8.168	32.67

```
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)
```

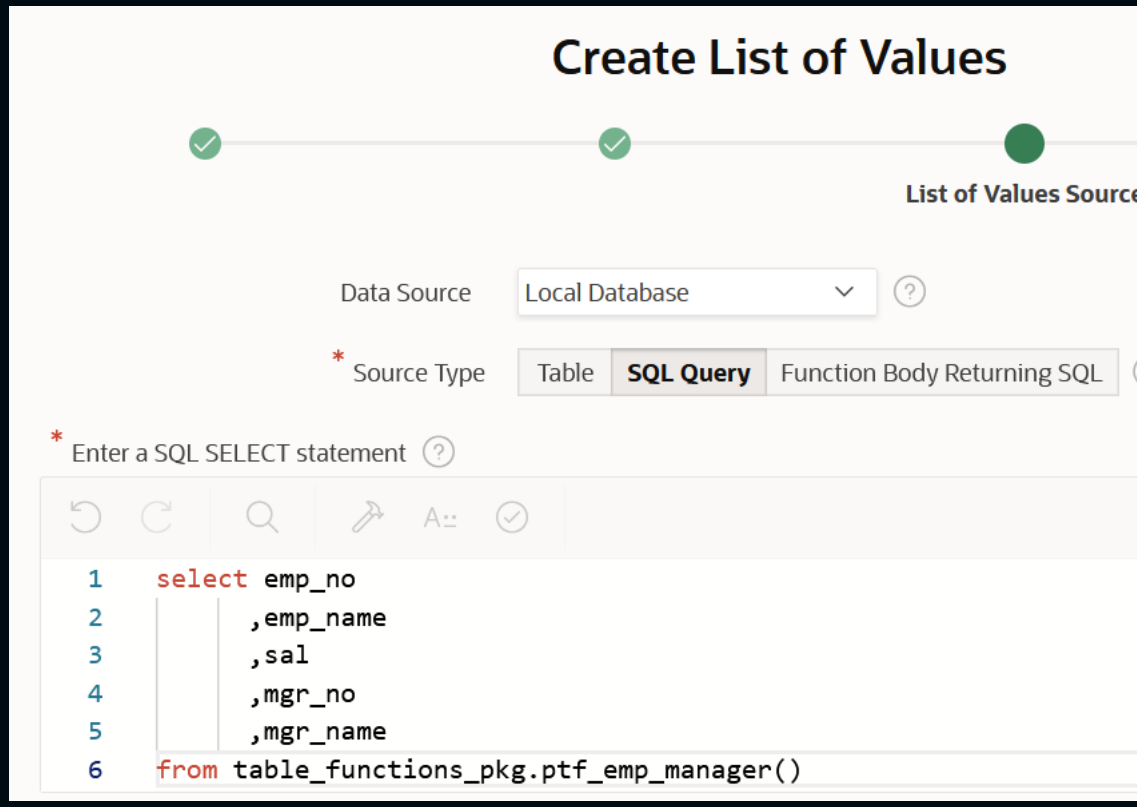
	EMP_NO	EMP_NAME	SAL	MGR_NO	MGR_NAME
1	7839	KING	5000		
2	7698	BLAKE	2850	7839	KING

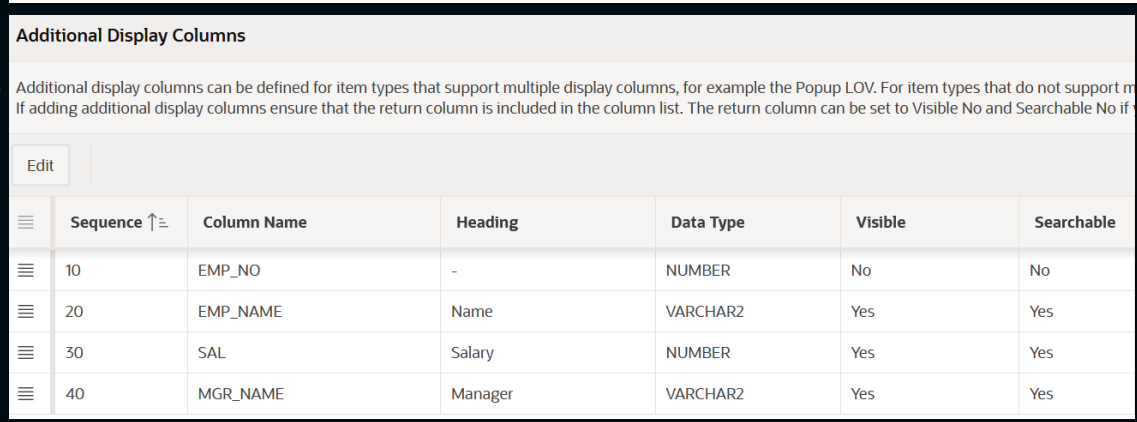
```
56    exception
57      when no_data_needed then
58        dbms_output.put_line('Try not to query more rows and filter afterwards.' || sqlerrm);
```

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



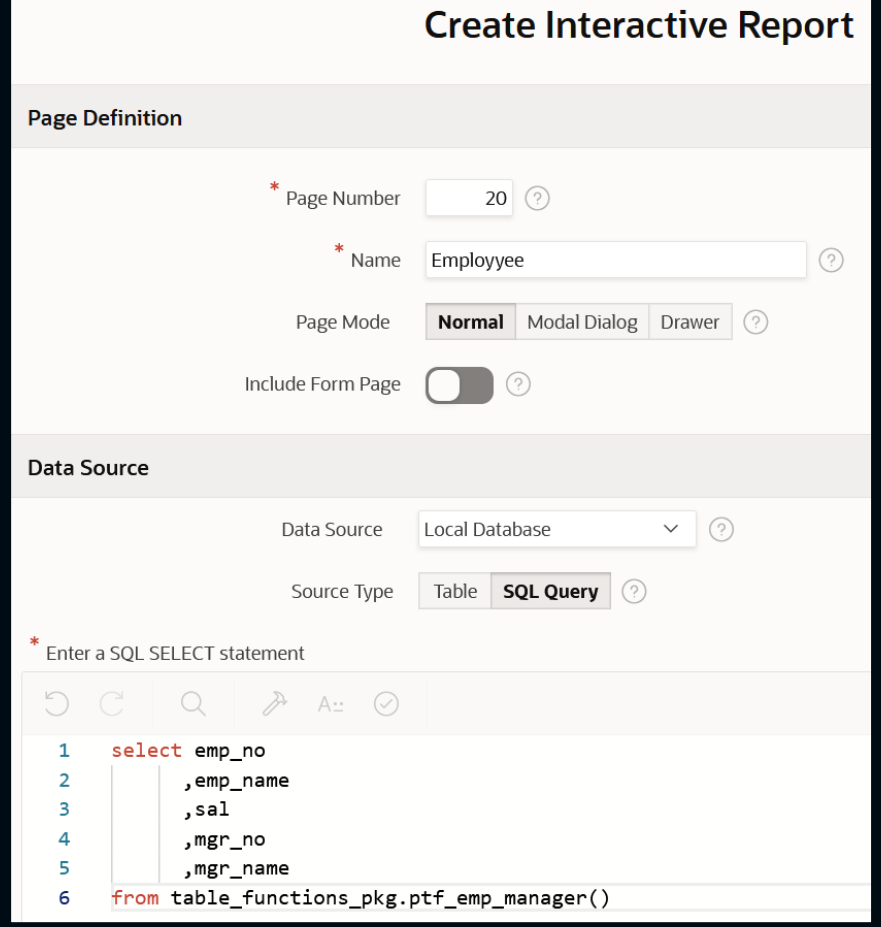


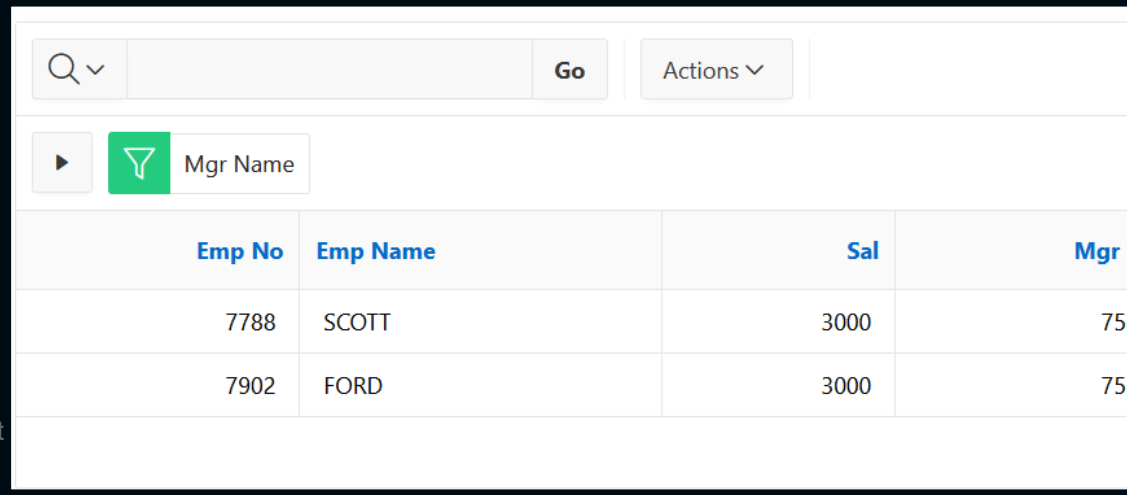
Sequence	Column Name	Heading	Data Type	Visible	Searchable
10	EMP_NO	-	NUMBER	No	No
20	EMP_NAME	Name	VARCHAR2	Yes	Yes
30	SAL	Salary	NUMBER	Yes	Yes
40	MGR_NAME	Manager	VARCHAR2	Yes	Yes

APEX

Interactive Report

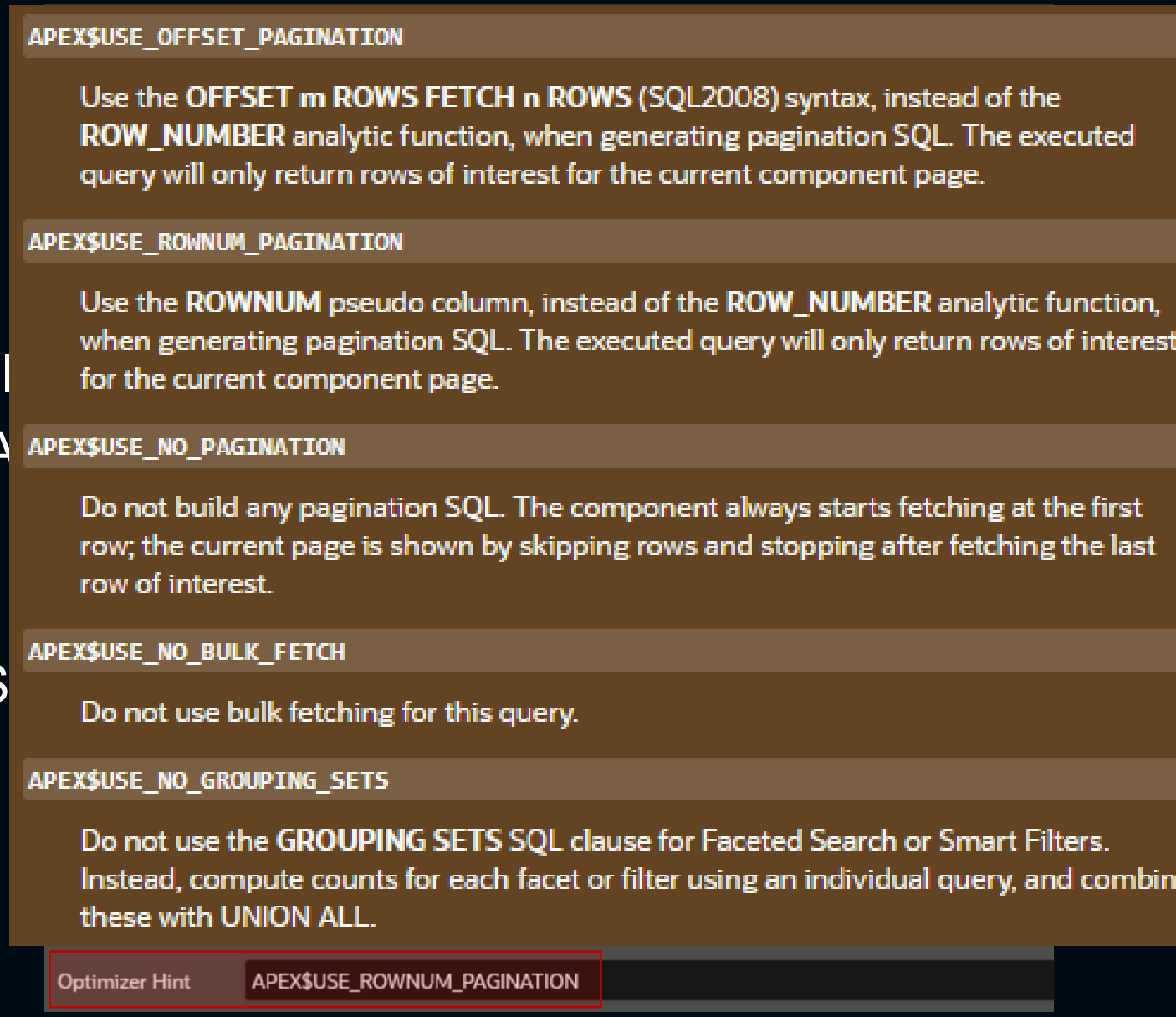
- Source Type SQL Query
- Filtering
  - adds WHERE clause
  - fires every time filter is adjusted
- Combine Pipeline Table function with APEX Collections for IR with heavy usage





Interactive Report

- Optimizer Hints “in APEX”
  - APEX$USE_OFFSET_PAGINATI
  - APEX$USE_ROWNUM_PAGINA
  - APEX$USE_NO_PAGINATION
  - APEX$USE_NO_BULK_FETCH
  - APEX$USE_NO_GROUPING_S



Conclusion_

1. Scenario 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),
```

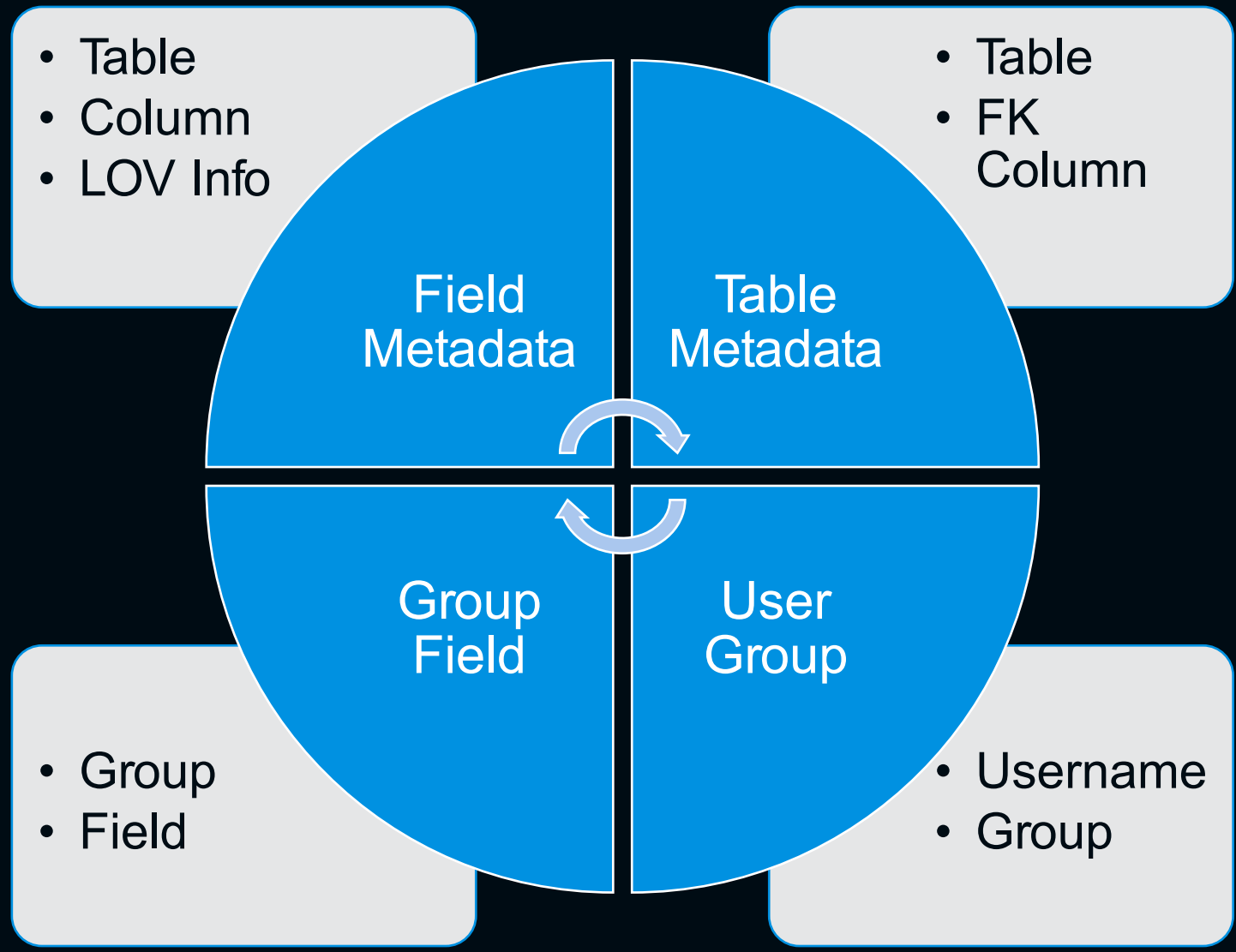
SQL | Output | Statistics

```
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

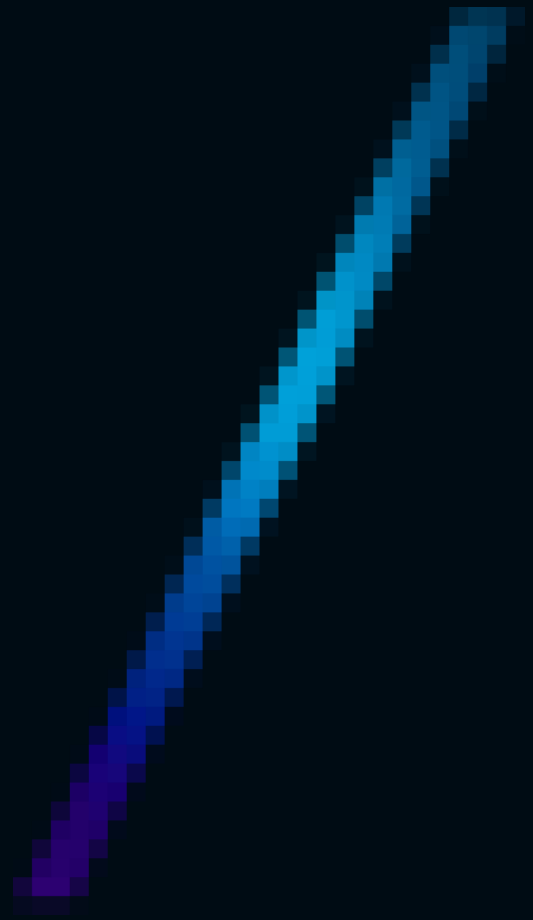
2. Scenario Metadata & User rights Compare Changes

- 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

@OliverLemm

@oliverlemm.bsky.social

NL.OUG
NL ORACLE USER GROUP

Please fill in your evaluations