

The Educational Conference For Oracle Technology Users

ODTUG  
Kscope23  
aurora, co    june 25-29



Interested in volunteering?  
Scan the code to sign up



Don't Forget To  
Fill Out Your Evals

# Programming with utPLSQL



Philipp Salvisberg  
28<sup>th</sup> June 2023

# Philipp Salvisberg

## Data Engineering Principal

- Database Centric Development
- Model Driven Software Development
- Open-Source Development

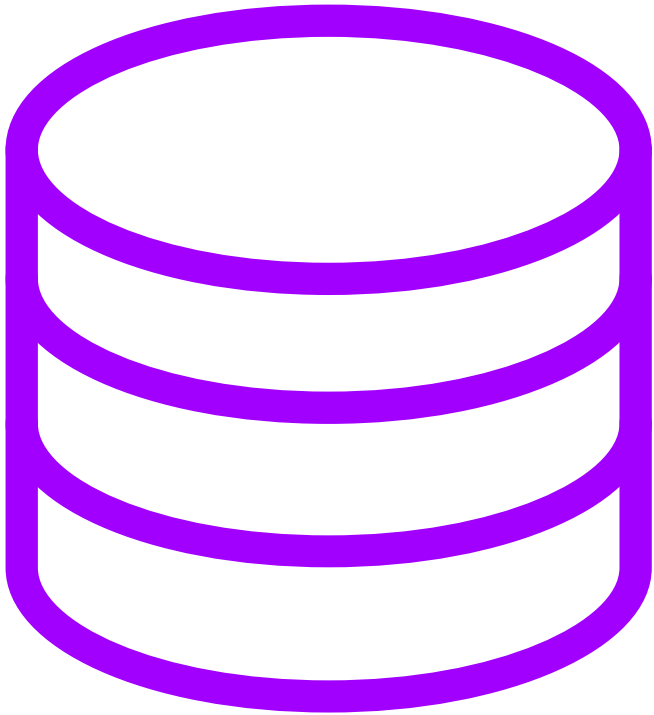
[philipp.salvisberg@accenture.com](mailto:philipp.salvisberg@accenture.com)

<https://www.salvis.com/blog>



# Introduction

# Testing Scope in Database Development



Every component of an application that is deployed in the database.

# Table (Check Constraint, Virtual Column)

```
create table t (  
  id integer generated always as identity  
    not null constraint t_pk primary key,  
  phone_number varchar2(30 char) not null  
    constraint t_phone_number_ck check (  
      regexp_like(phone_number,  
        '^(\+?(\d{1,3}))?'  
        || '([-.\s]*(\d{3})[-.\s]*)?'  
        || '((\d{3})[-.\s]*)'  
        || '(\d{2,4})'  
        || '([-.\s]*(\d+))?$'  
      )  
    )  
);
```

insert into t  
(phone\_number)  
values  
('+41 79 558 35 22');

-- country  
-- 1st group  
-- 2nd group, 1st sub  
-- 2nd group, 2nd sub  
-- 2nd group, 3rd sub

Source: <https://regexpr.com/38pvy>

# View




select \* from deptsal;

```
create or replace view deptsal as
  select d.deptno,
         d.dname,
         coalesce(sum(e.sal), 0) as sum_sal,
         coalesce(count(e.empno), 0) as num_emps,
         coalesce(round(avg(e.sal), 2), 0) as avg_sal
  from dept d
 left join emp e
    on e.deptno = d.deptno
 group by d.deptno, d.dname;
```

Source: <https://github.com/PhilippSalvisberg/utplsql-red-stack-demo/blob/b-view-test/src/main/view/deptsal.sql>

# Materialized View

```
create materialized view deptsal refresh fast on commit as
  select deptno,
         dname,
         sum_sal,
         num_emps,
         round(avg_sal, 2) as avg_sal,
         'EMP' as row_source,      -- required for fast refresh after insert
         rowid as row_id          -- required for fast refresh after any DML
  from deptsal_emp_mv
union all
  select deptno,
         dname,
         0,
         0,
         0,
         'DEPT' as row_source,    -- required for fast refresh after insert
         rowid as row_id          -- required for fast refresh after any DML
  from deptsal_dept_mv
where emp_deptno is null;
```



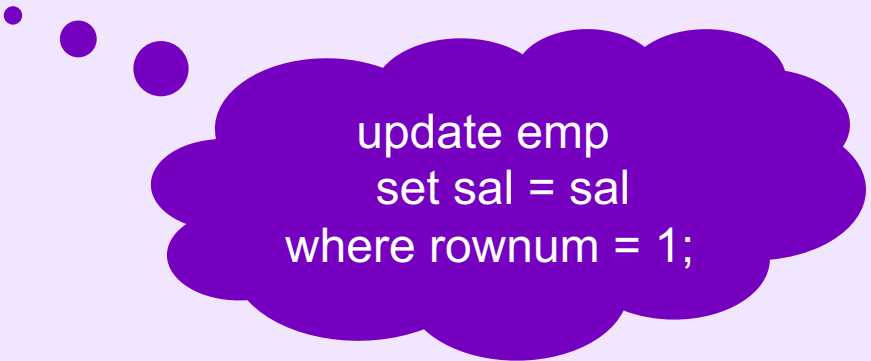
update emp  
set sal = sal + 100  
where deptno = 10;  
  
commit;  
  
select \* from deptsal;

Source: <https://github.com/PhilippSalvisberg/utplsql-red-stack-demo/blob/c-mview-test/src/main/mview/deptsal.sql>



# Trigger

```
create or replace trigger emp_as_iud
  after insert or update or delete on emp
begin
  etl.refresh_deptsal;
end;
/
```



update emp  
set sal = sal  
where rownum = 1;

Source: [https://github.com/PhilippSalvisberg/utplsql-red-stack-demo/blob/a-diy-test-5-trigger/src/main/trigger/emp\\_as\\_iud.sql](https://github.com/PhilippSalvisberg/utplsql-red-stack-demo/blob/a-diy-test-5-trigger/src/main/trigger/emp_as_iud.sql)

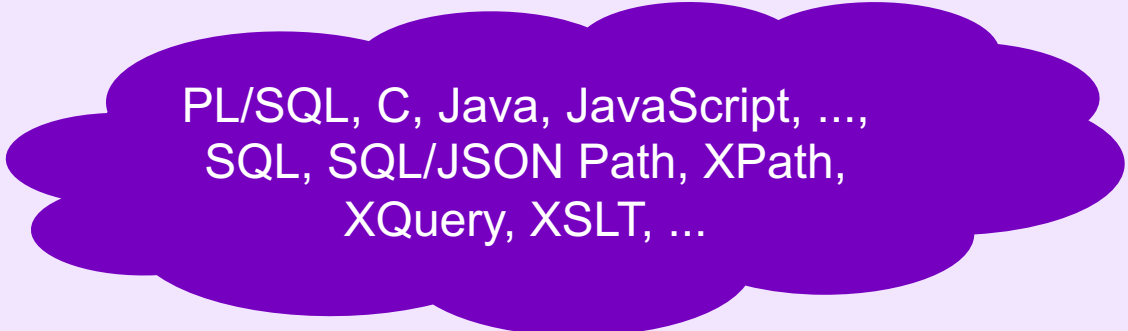
# Package (Procedure, Function, Type)

```
create or replace package etl is  
    procedure refresh_deptsal;  
end etl;  
/
```



exec etl.refresh\_deptsal;

The diagram consists of a purple thought bubble containing the text 'exec etl.refresh\_deptsal;'. To the left of the bubble, three small purple circles of increasing size lead to the bubble, indicating a call to the procedure defined in the package.



PL/SQL, C, Java, JavaScript, ...,  
SQL, SQL/JSON Path, XPath,  
XQuery, XSLT, ...

The diagram consists of a large purple thought bubble containing a list of languages and query types. To the left of the bubble, three small purple circles of increasing size lead to the bubble, indicating the scope of languages and query types supported by the package.

Source: <https://github.com/PhilippSalvisberg/utplsql-red-stack-demo/blob/a-diy-test-3-success/src/main/package/etl.pks>

# Test Automation

"The use of software separate from the software being tested to **control the execution of tests** and the comparison of actual outcomes with predicted outcomes."

Source: [https://en.wikipedia.org/wiki/Test\\_automation](https://en.wikipedia.org/wiki/Test_automation)

# utPLSQL

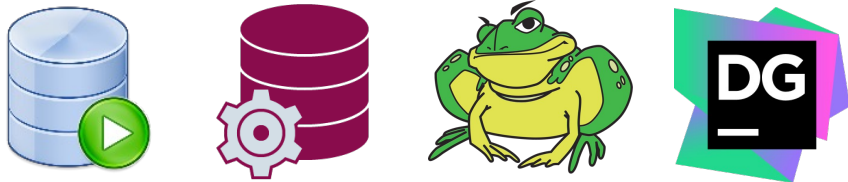
## Core Testing Framework

- Schema in the database
- No repository
- Annotation based tests



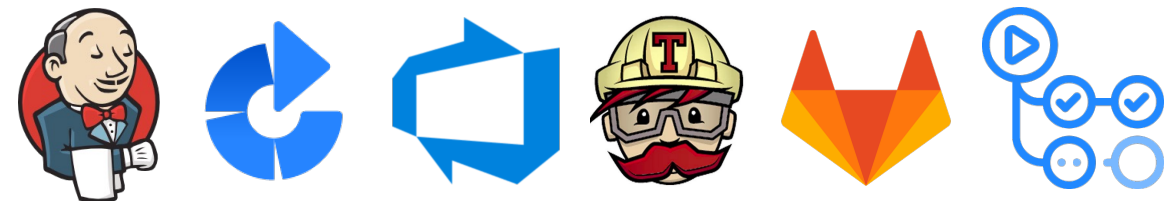
## Development

- Realtime Reporter
- Code Coverage, Code Templates, etc.



## Test Automation

- Command Line Client
- Maven Plugin
- Various Reporters



# Test Declaration

```
create or replace package test_suite as
  --%suite

  --%test
  procedure test_case;
end test_suite;
```

--%displayname(<description>)  
**--%test(<description>)**  
--%tags(<tag>[,...])  
--%throws(<exception>[,...])  
--%beforeall  
--%afterall  
--%beforeeach  
--%aftereach  
--%beforetest([...])  
--%aftertest([...])  
--%rollback(manual)  
--%disabled(<reason>)

**--%suite(<description>)**  
--%suitepath(<path>)  
--%tags(<tag>[,...])  
--%displayname(<description>)  
--%beforeall([...])  
--%afterall([...])  
--%beforeeach([...])  
--%aftereach([...])  
--%rollback(manual)  
--%disabled(<reason>)  
--%context  
--%endcontext

# Test Implementation

```
create or replace package body test_suite as
  procedure test_case is
    c_actual      sys_refcursor;
    c_expected sys_refcursor;
  begin
    -- arrange
    insert into dept (deptno, dname, loc) values (-10, 'utPLSQL', 'Winterthur');
    -- act
    insert into emp (empno, ename, job, hiredate, sal, deptno)
    values (-1, 'Jacek', 'Developer', trunc(sysdate), 4700, -10);
    -- assert
    open c_actual for select deptno, dname, sum_sal from deptsal where deptno = -10;
    open c_expected for select -10 as deptno, 'utPLSQL' as dname, 4200 as sum_sal from dual;
    ut.expect(c_actual).to_equal(c_expected).join_by('DEPTNO');
  end test_case;
end test_suite;
```

## Matcher:

be\_between, be\_empty, be\_false, be\_greater\_than,  
be\_greater\_or\_equal, be\_less\_or\_equal, be\_less\_than,  
be\_like, be\_not\_null, be\_null, be\_true, contain, **equal**,  
have\_count, match, be\_within, be\_within\_pct, ...

Extended options for refcursor, object  
type, JSON, nested table and varray:

- include(<items>)
- exclude(<items>)
- unordered
- **join\_by(<items>)**

# Test Run

```
set serveroutput on size unlimited
exec ut.run('test_suite')
```

```
test_suite
  test_case [.024 sec] (FAILED - 1)
```

Failures:

1) test\_case

**Actual:** refcursor [ count = 1 ] was **expected** to equal: refcursor [ count = 1 ]

Diff:

Rows: [ 1 differences ]

**PK** <DEPTNO>-10</DEPTNO> - **Actual:** <SUM\_SAL>4700</SUM\_SAL>

**PK** <DEPTNO>-10</DEPTNO> - **Expected:** <SUM\_SAL>4200</SUM\_SAL>

at "REDSTACK.TEST\_SUITE.TEST\_CASE", line 14 ut.expect(c\_actual).to\_equal(c\_expected).join\_by('DEPTNO');

Finished in .027162 seconds

1 tests, 1 failed, 0 errored, 0 disabled, 0 warning(s)

List of ...

schema

[schema.]**package**[.procedure]

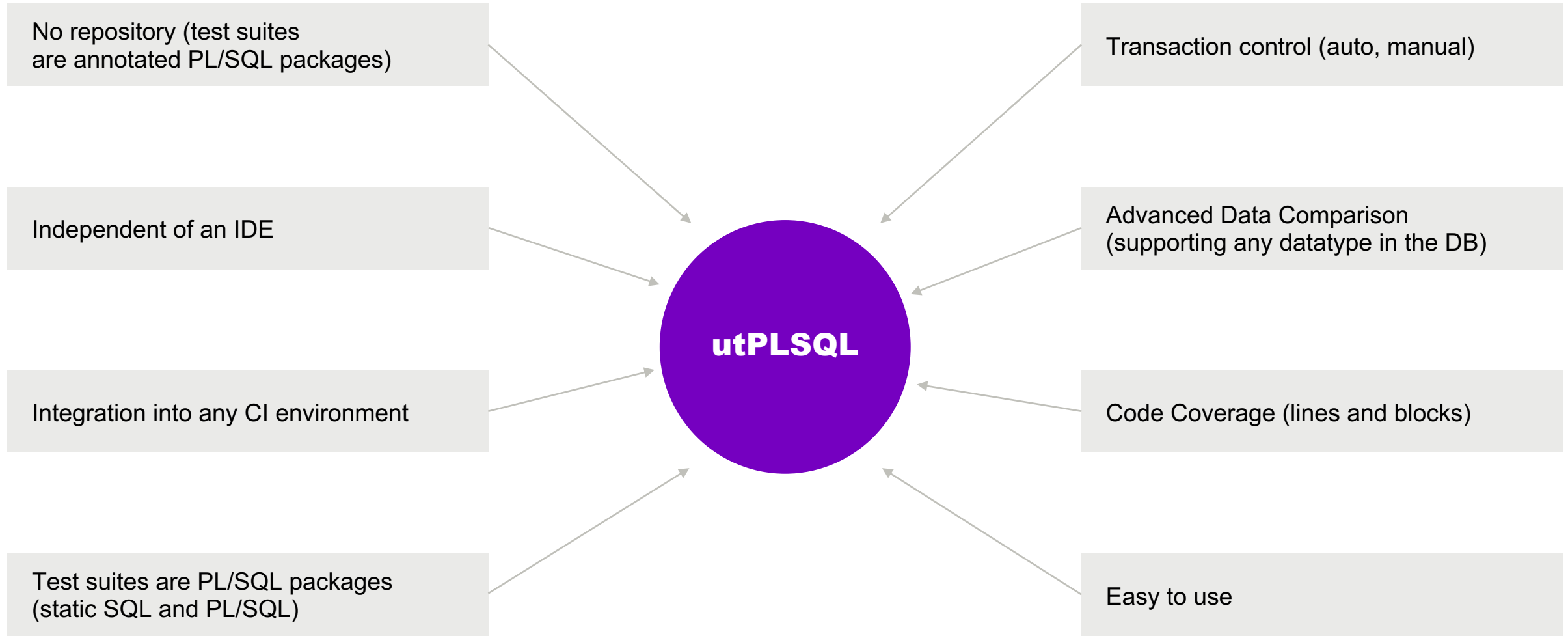
[schema]:suitepath[.context][.procedure]

Optionally extended by ...

, a\_tags => 'includeTag, -excludeTag[, ...]'



# Why?



# Installation

# Install utPLSQL Core Testing Framework

```
source — zsh — 100x30
Synonym UT_COVERALLS_REPORTER created.

Synonym UT_COVERAGE_COBERTURA_REPORTER created.

Synonym UT_DEBUG_REPORTER created.

Synonym UT_REPORTERS created.

Synonym UT_REPORTER_BASE created.

Synonym UT_OUTPUT_REPORTER_BASE created.

Synonym DBMSPCC_BLOCKS created.

Synonym DBMSPCC_RUNS created.

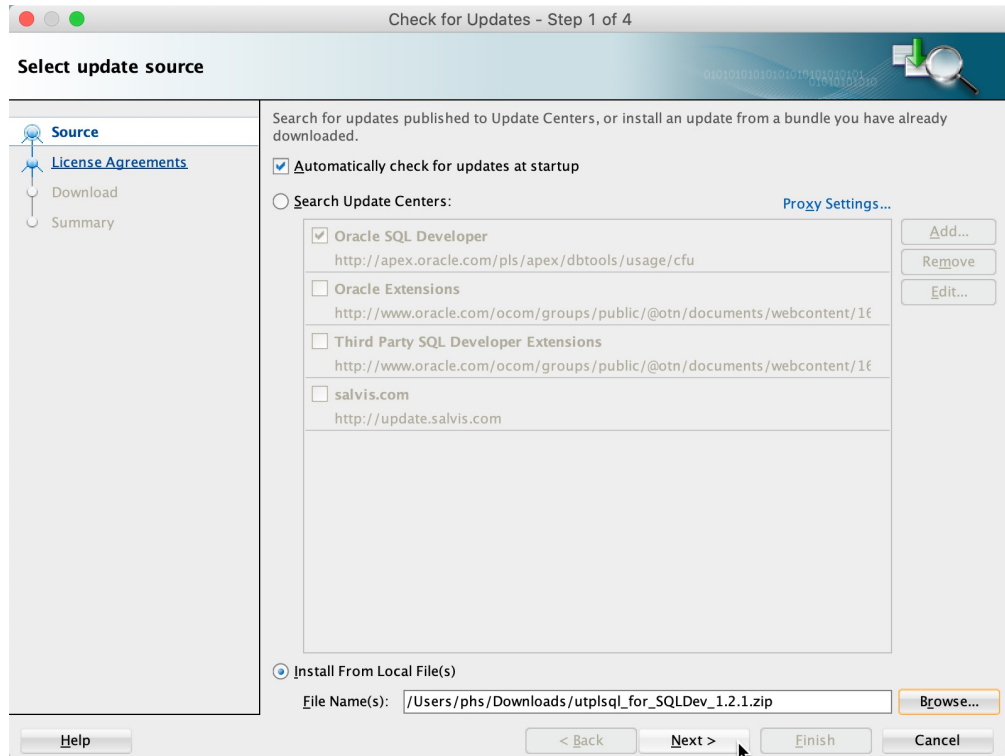
Synonym DBMSPCC_UNITS created.

Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
sql sys/oracle@xepdb2 as sysdba @install_headless.sql 11.63s user 0.80s system 57% cpu 21.796 total
phs@macphs21 source %
```

## Instructions

1. Download utPLSQL.zip from <https://github.com/utPLSQL/utPLSQL/releases>
2. Unzip utPLSQL.zip
3. cd source
4. sqlplus / as sysdba @install\_headless.sql
  - User UT3
  - Password XNtxj8eEgA6X6b6f
  - Tablespace USERS

# Install utPLSQL for SQL Developer From File

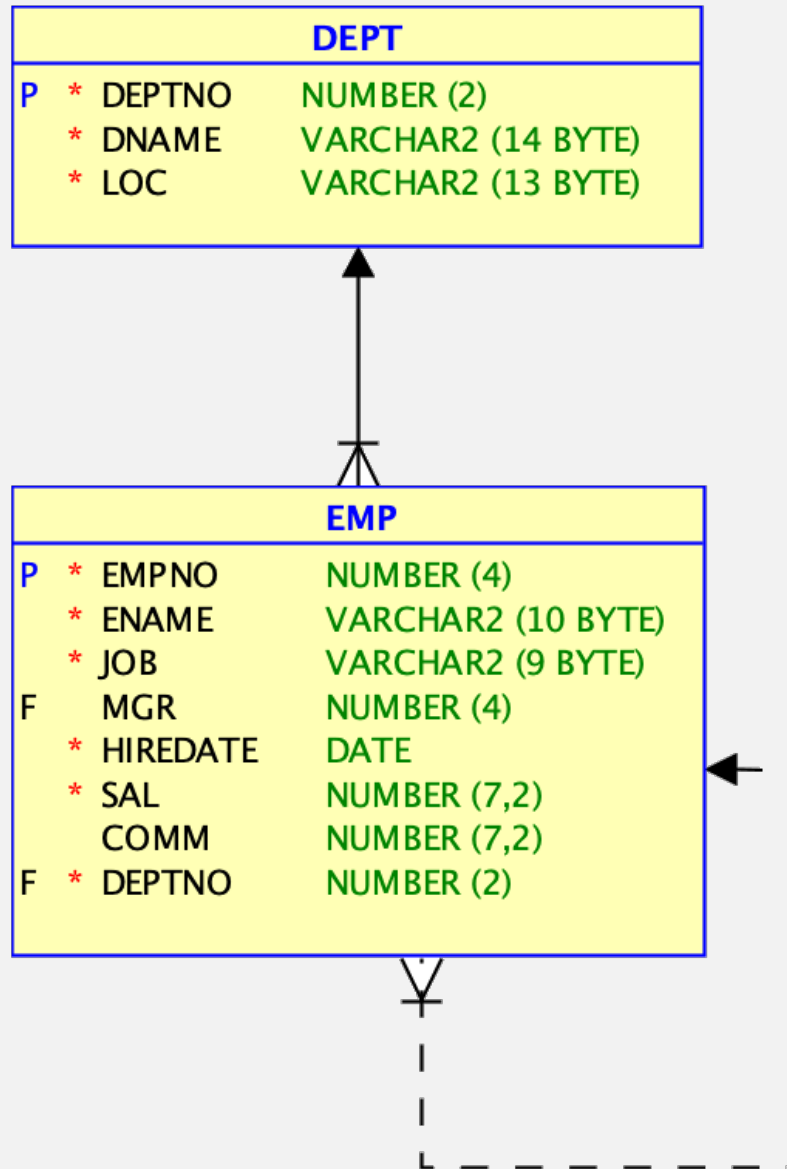


## Instructions

1. Download utplsqli\_for\_SQLDev\_\*.zip from <https://github.com/utPLSQL/utPLSQL-SQLDeveloper/releases>
2. Start SQL Developer
3. Select “Check for Updates...” in the help menu
4. Use the “Install from Local File” option to install the previously downloaded “utplsqli\_for\_SQLDev\_\*.zip” file
  - User must have read/write access to SQL Developer installation directory
  - Run as Administrator, if required
5. Restart SQL Developer

You can also configure an Update Center, see <https://github.com/PhilippSalvisberg/sqldev-update>

# Build and Run Tests

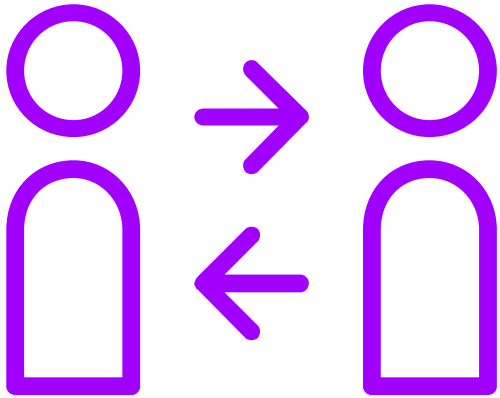


## Demo Case Study

As a HR manager, I need a table with the key figures

- salary total,
- number of employees and
- average salary per department to assess fairness.

# Review



Solution is more complex than necessary

We could use a view instead

Table deptsal is refreshed too often, e.g. on rollback or if more than one DML is used in a transaction

Providing a table is not mandatory, even if the HR manager has explicitly requested one

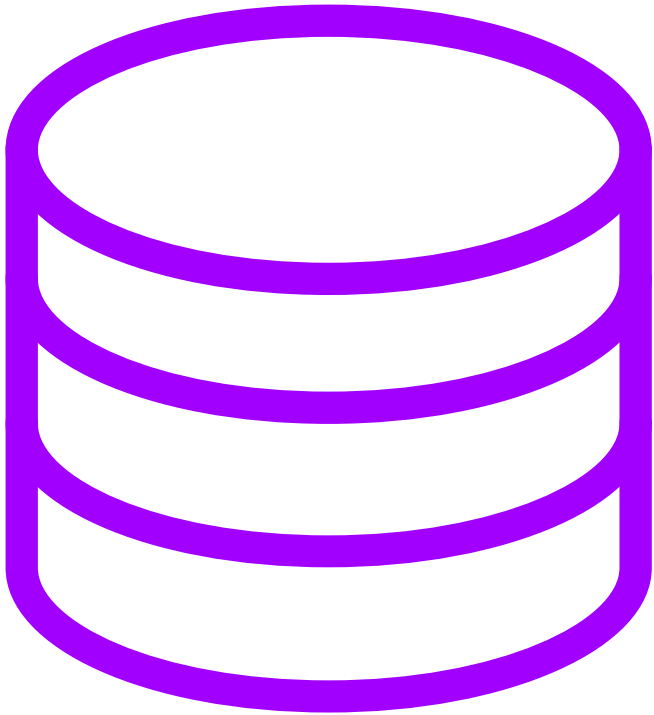
A materialized view is not yet required, data volume is small, a regular view should be fast

This would significantly reduce our code base and simplify maintenance

# **Unit Tests versus Database Tests**



# Database Testing Realities



- State (Table, Package, Session Context)
- Dependencies with State

# What About Test Doubles



- Dummies
- Stubs
- Spies
- Mocks
- Fakes

Might require a  
dedicated  
test schema

Missing  
frameworks

Good option for  
3<sup>rd</sup> party system  
access

Source: Heidi Moneymaker, <https://www.instagram.com/p/BILm4tCBzyJ/>

# Code Coverage

# Code Coverage – Definition

"A measure used to describe the **degree** to which the source code **of a program is executed** when a particular test suite runs."

Source: [https://en.wikipedia.org/wiki/Code\\_coverage](https://en.wikipedia.org/wiki/Code_coverage)

# Line Coverage

```
create or replace function f(a in integer) return integer is
begin
    if a is null then
        return 0;
    else
        return a * a;
    end if;
end f;
/
```

Two test cases for  
100% coverage

# Code Block Coverage (12.2 and Higher)

```
create or replace function f(a in integer) return integer is
begin
    if a is null then return 0; else return a * a; end if;
end f;
/
```

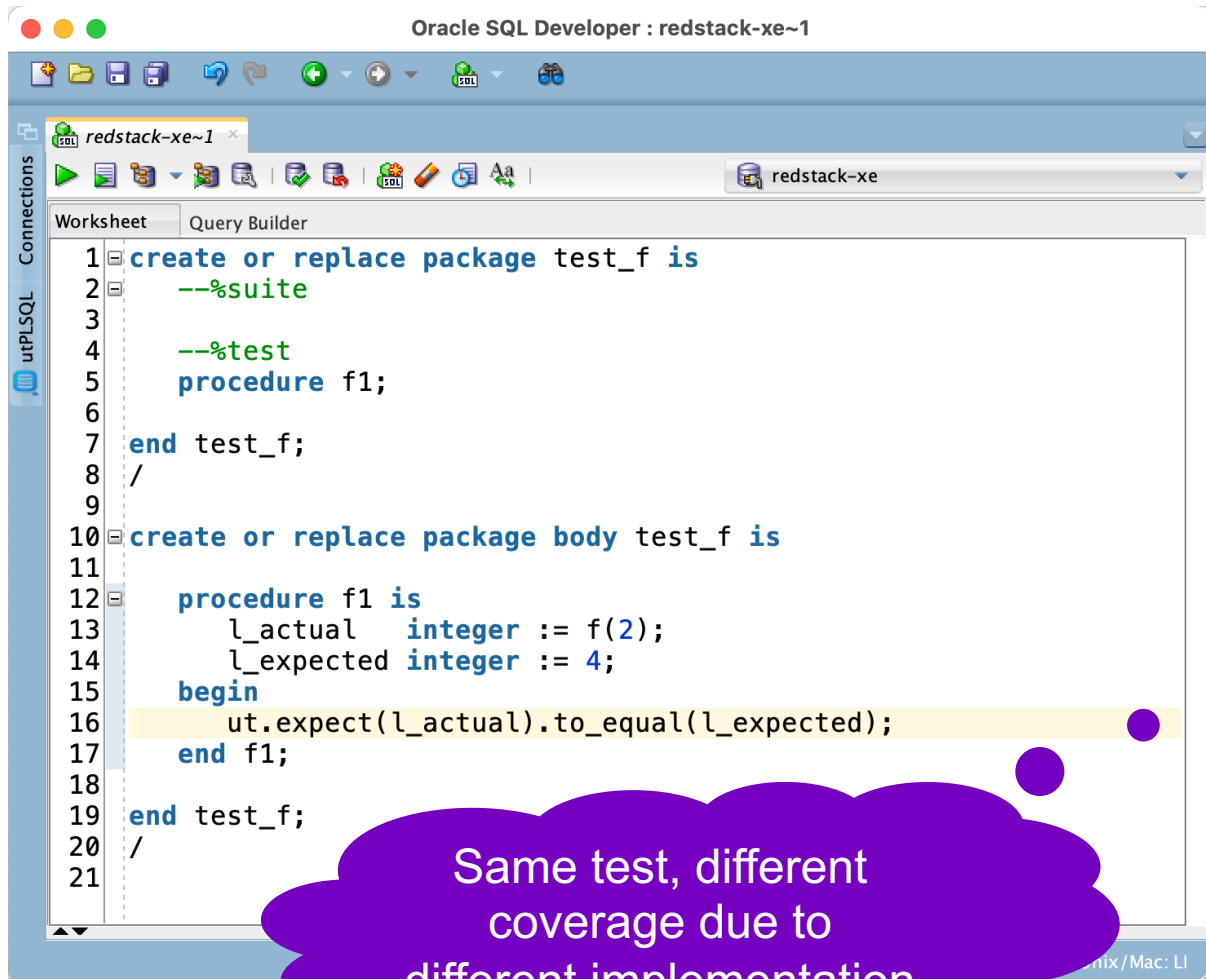
Two test cases for  
100% coverage

```
create or replace function f(a in integer) return integer is
begin
    return nvl(a * a, 0);
end f;
/
```

One test case for  
100% coverage

# utPLSQL – Line & Code Block Coverage

1 of 2  
code blocks  
covered



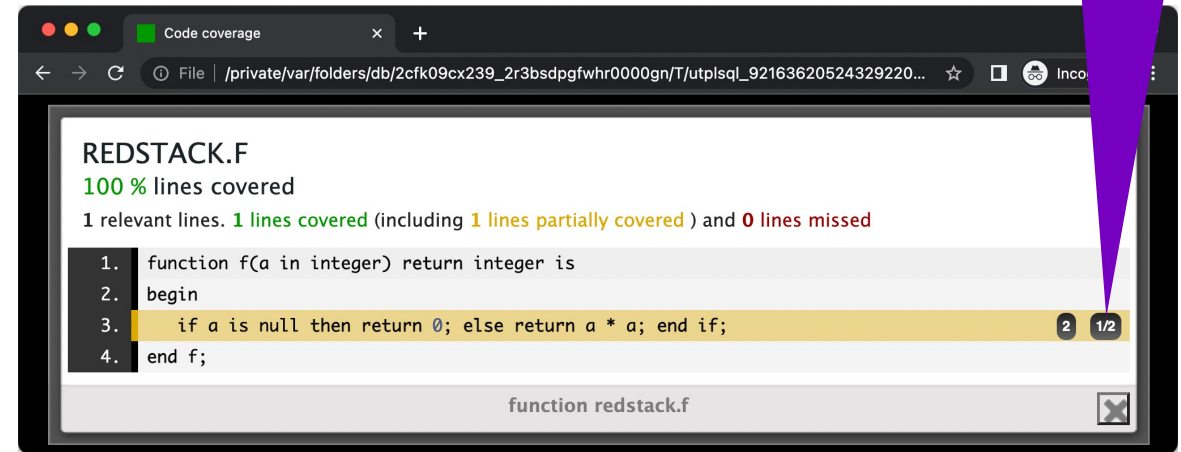
Oracle SQL Developer : redstack-xe~1

redstack-xe~1

Worksheet

```
1 create or replace package test_f is
2   --%suite
3
4   --%test
5   procedure f1;
6
7 end test_f;
8 /
9
10 create or replace package body test_f is
11
12   procedure f1 is
13     l_actual integer := f(2);
14     l_expected integer := 4;
15   begin
16     ut.expect(l_actual).to_equal(l_expected);
17   end f1;
18
19 end test_f;
20 /
21
```

Same test, different  
coverage due to  
different implementation



Code coverage

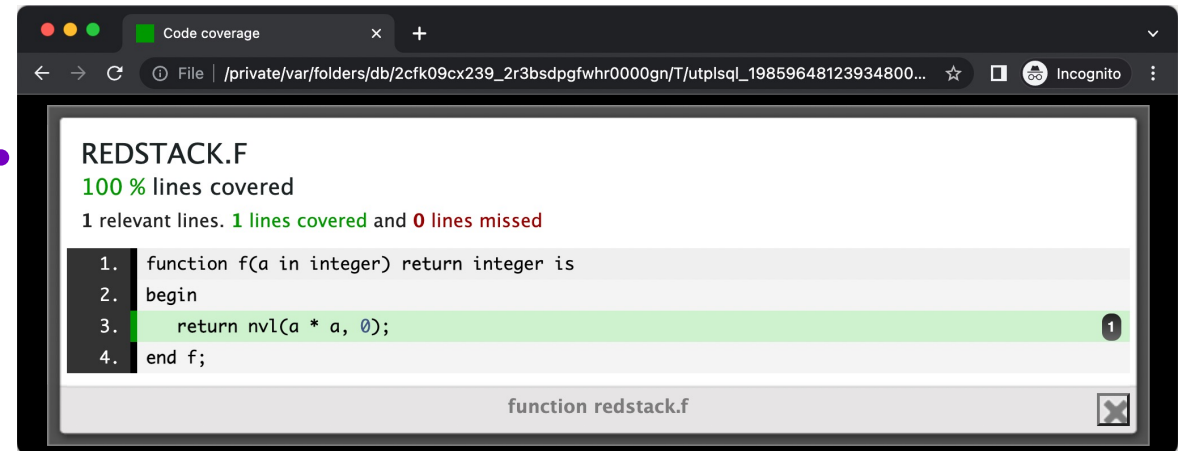
REDSTACK.F

100 % lines covered

1 relevant lines. 1 lines covered (including 1 lines partially covered) and 0 lines missed

Line	Code	Coverage
1.	function f(a in integer) return integer is	
2.	begin	
3.	if a is null then return 0; else return a * a; end if;	2 1/2
4.	end f;	

function redstack.f



Code coverage

REDSTACK.F

100 % lines covered

1 relevant lines. 1 lines covered and 0 lines missed

Line	Code	Coverage
1.	function f(a in integer) return integer is	
2.	begin	
3.	return nvl(a * a, 0);	1
4.	end f;	

function redstack.f

# Key Messages



# Programming with utPLSQL – This Is the Way

- **Set up a test-friendly environment**
  - Install utPLSQL core testing framework
  - Install SQL Developer for utPLSQL
- **Start with tests**
  - To reproduce bugs
  - For new requirements
- **utPLSQL will change how you code**
  - Write smaller units
  - Isolate code that is difficult to test





**Thank You**

The Educational Conference For Oracle Technology Users

ODTUG  
Kscope23  
aurora, co    june 25-29



Interested in volunteering?  
Scan the code to sign up



Don't Forget To  
Fill Out Your Evals