Programming with utPLSQL

Philipp Salvisberg
22nd November 2023

Philipp Salvisberg

Data Engineering Principal

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

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}))?'                        -- country
               || '([-. (]*(\d{3})[-. )]*)?'             -- 1st group
               || '((\d{3})[-. ]*'                       -- 2nd group, 1st sub
               || '(\d{2,4})'                            -- 2nd group, 2nd sub
               || '([-.x ]*(\d+))?)$'                    -- 2nd group, 3rd sub
          )
       )
);
```

insert into t
(phone_number)
values
('+41 79 558 35 22');

Source: https://regexr.com/38pvb

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

Package (Procedure, Function, Type)

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

exec etl.refresh_deptsal;

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

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

utPLSQL

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>[,...]
--%displayame(<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')
```

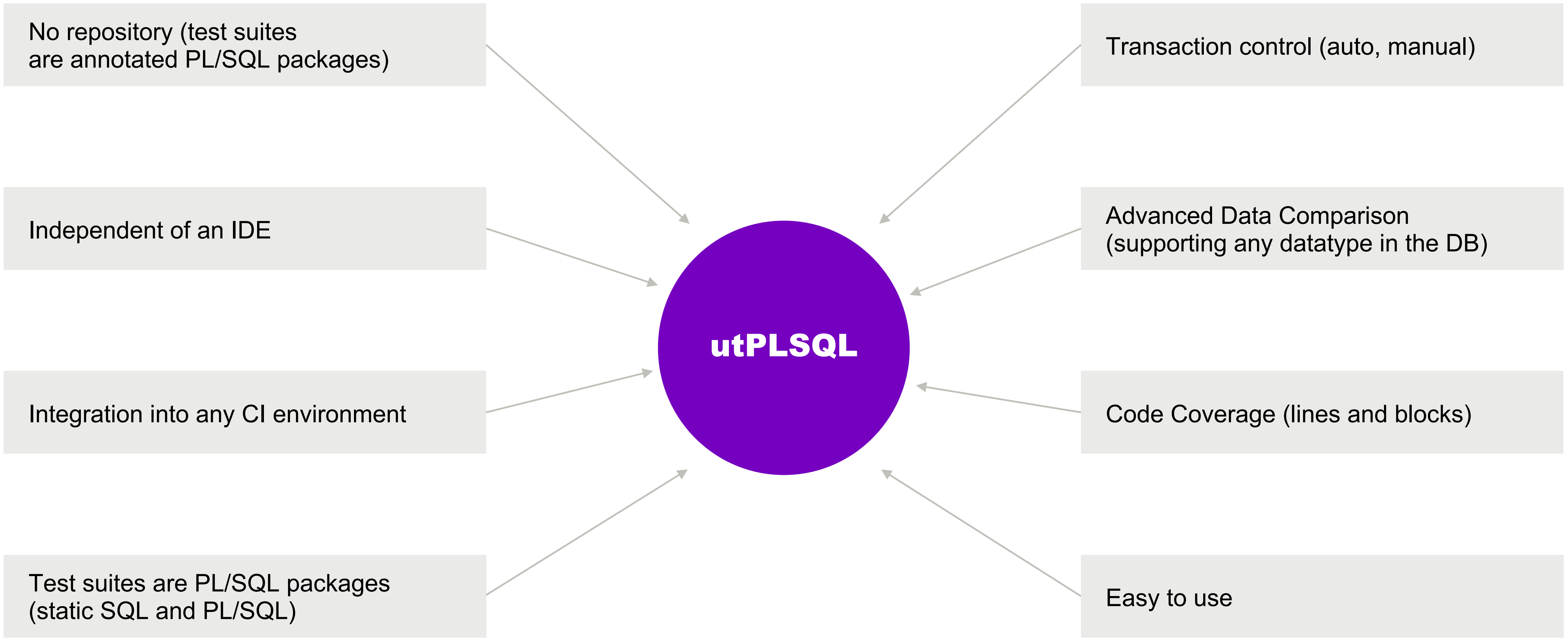
List of ...

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

Optionally extended by …
, a_tags => 'includeTag, -excludeTag[, ...]'

```
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)
```

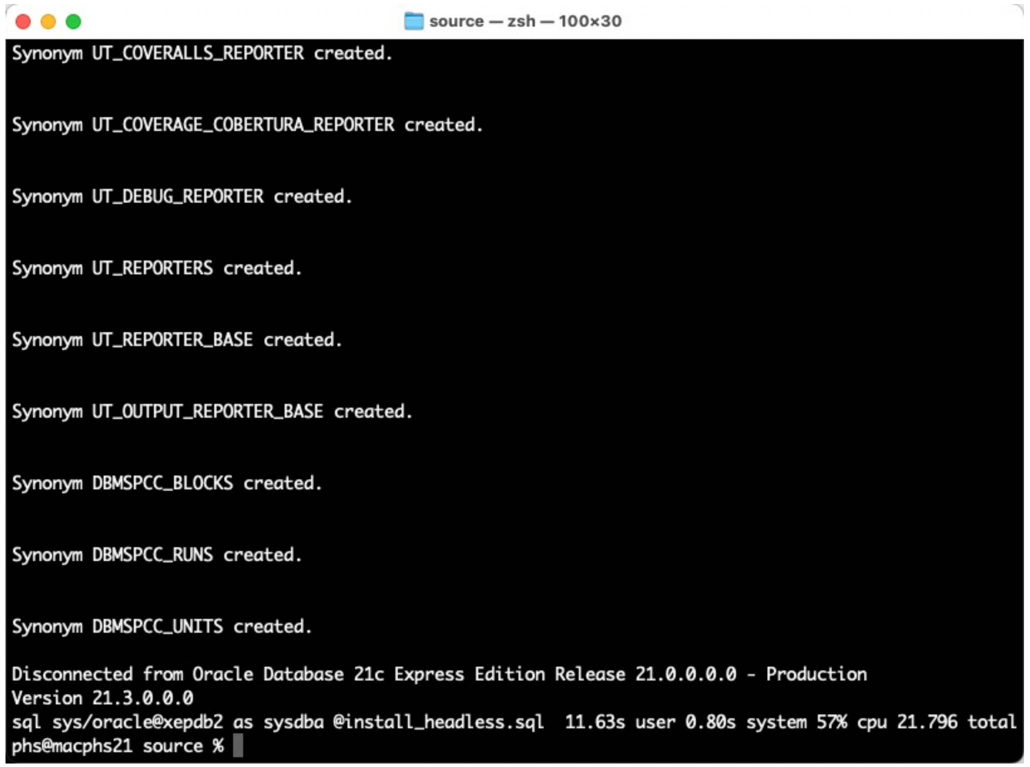
Why?



- No repository (test suites are annotated PL/SQL packages)
- Independent of an IDE
- Integration into any CI environment
- Test suites are PL/SQL packages (static SQL and PL/SQL)
- Transaction control (auto, manual)
- Advanced Data Comparison (supporting any datatype in the DB)
- Code Coverage (lines and blocks)
- Easy to use

Installation

Install utPLSQL Core Testing Framework

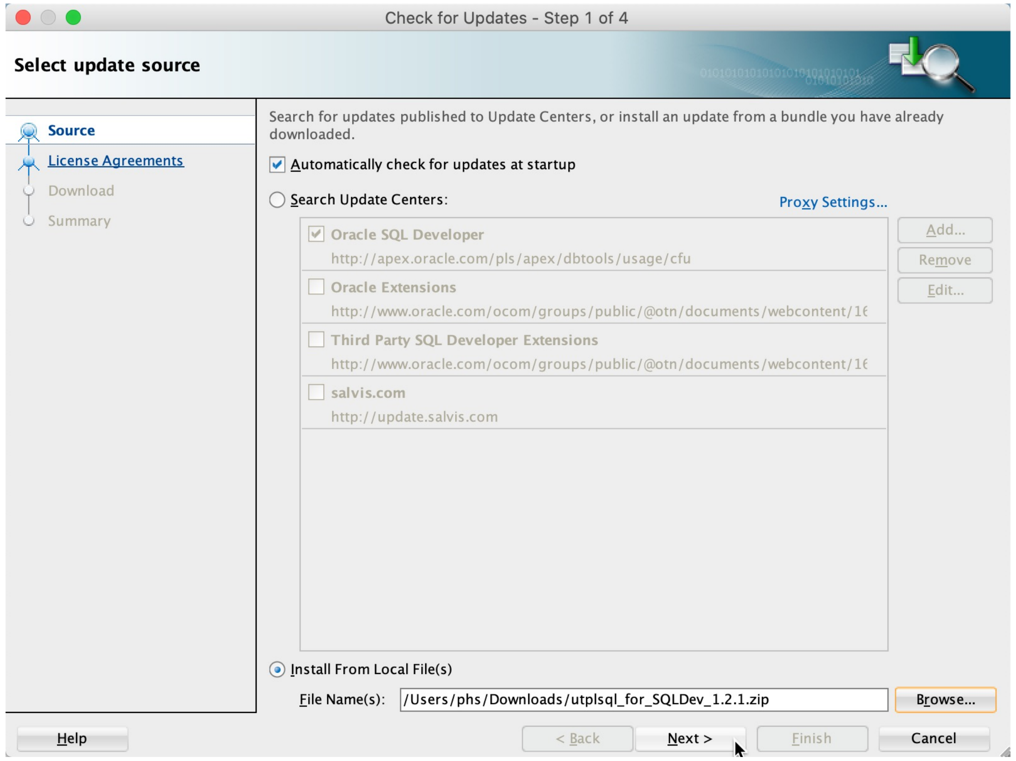

source — zsh — 100×30

```
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



Search for updates published to Update Centers, or install an update from a bundle you have already downloaded.

☑ Automatically check for updates at startup

○ Search Update Centers:

☑ Oracle SQL Developer http://apex.oracle.com/pls/apex/dbtools/usage/cfu

☐ Oracle Extensions http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/16

☐ Third Party SQL Developer Extensions http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/16

☐ salvis.com http://update.salvis.com

● Install From Local File(s)

File Name(s): /Users/phs/Downloads/utplsql_for_SQLDev_1.2.1.zip

Instructions

1. Download utplsql_for_SQLDev_*.zip
   from https://github.com/utPLSQL/utPLSQL-SQLDeveloper/releases

2. Start SQL Developer

3. Select “Check for Updates…” in the help menu

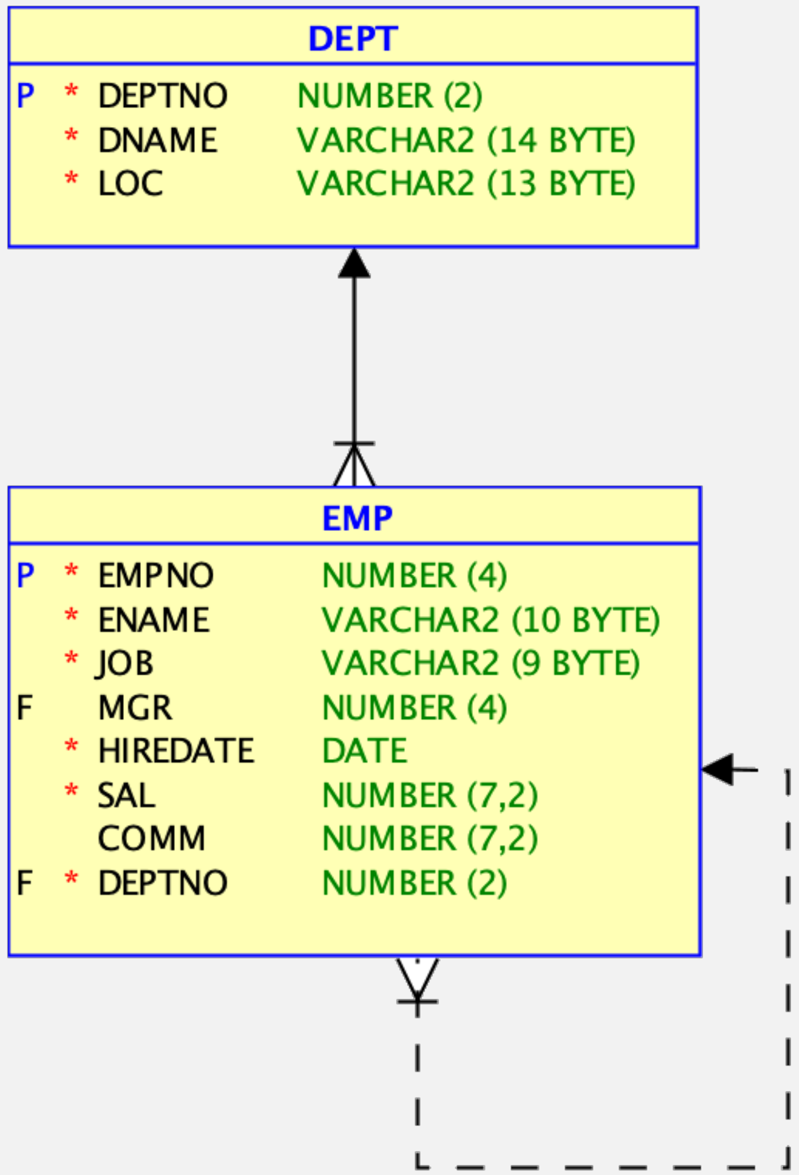
4. Use the “Install from Local File” option to install the previously downloaded “utplsql_for_SQLDev_*.zip” file

   - 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



DEPT			
P	*	DEPTNO	NUMBER (2)
	*	DNAME	VARCHAR2 (14 BYTE)
	*	LOC	VARCHAR2 (13 BYTE)

EMP			
P	*	EMPNO	NUMBER (4)
	*	ENAME	VARCHAR2 (10 BYTE)
	*	JOB	VARCHAR2 (9 BYTE)
F		MGR	NUMBER (4)
	*	HIREDATE	DATE
	*	SAL	NUMBER (7,2)
		COMM	NUMBER (7,2)
F	*	DEPTNO	NUMBER (2)

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.

Source: https://github.com/PhilippSalvisberg/utplsql-red-stack-demo

Review

Solution is more complex than necessary

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

We could use a view instead

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

Source: Heidi Moneymaker, https://www.instagram.com/p/BlLm4tCBzyJ/

- Dummies
- Stubs
- Spies
- Mocks
- Fakes

Might require a dedicated test schema

Missing frameworks

Good option for 3rd party system access

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

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;
/
```

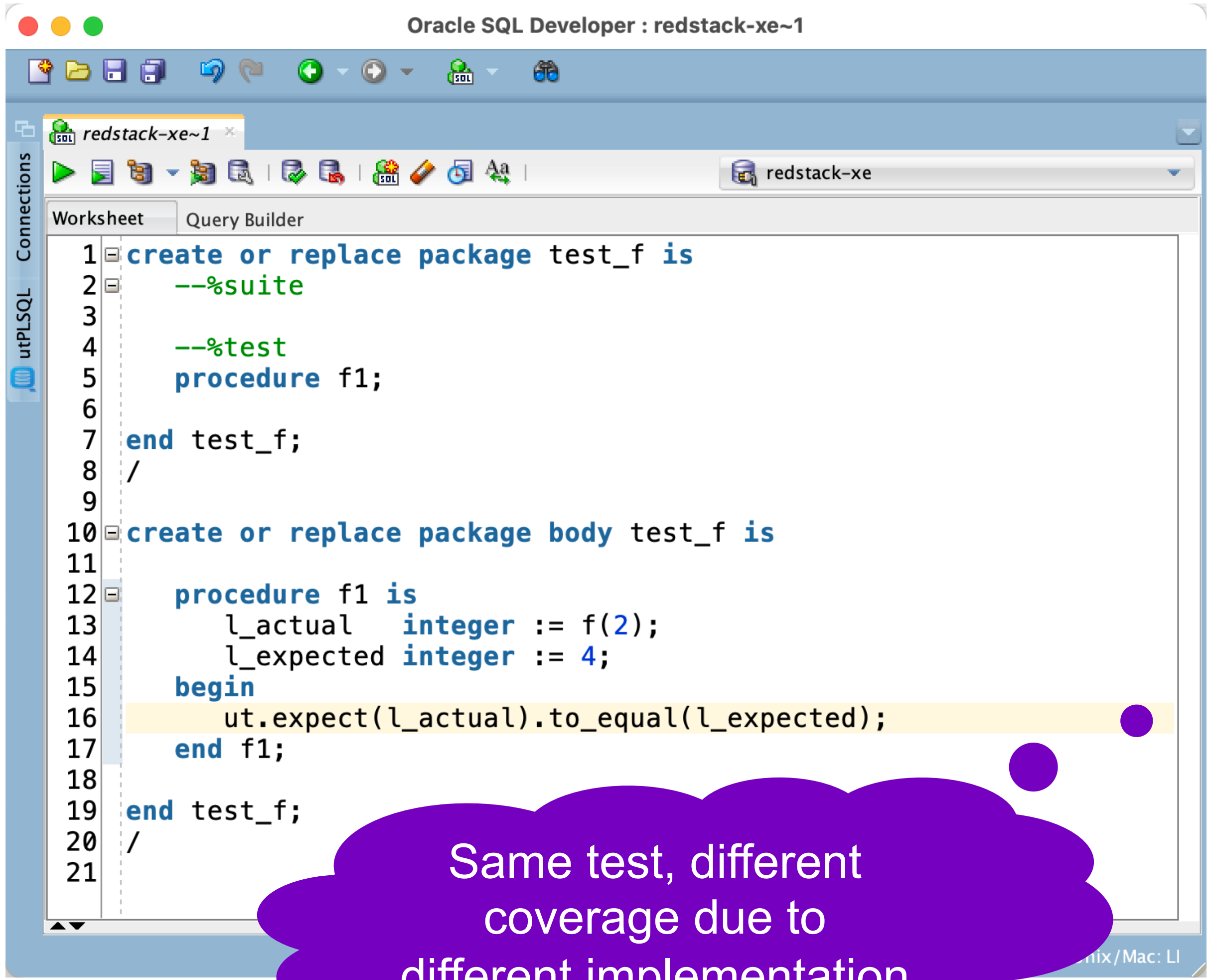
Two test cases for 100% coverage

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

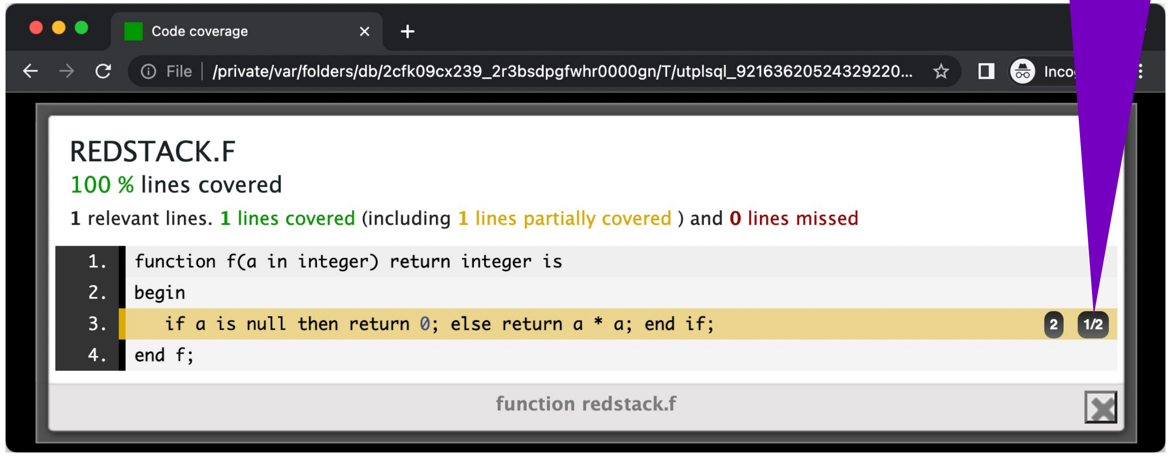
One test case for 100% coverage

utPLSQL – Line & Code Block Coverage

1 of 2
code blocks
covered

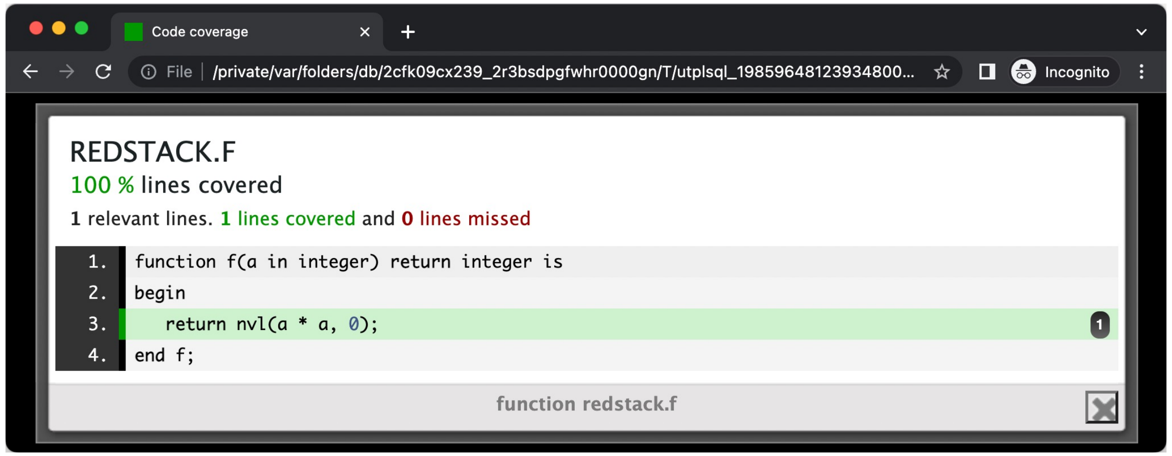


```
create or replace package test_f is
    --%suite

    --%test
    procedure f1;

end test_f;
/

create or replace package body test_f is

    procedure f1 is
        l_actual   integer := f(2);
        l_expected integer := 4;
    begin
        ut.expect(l_actual).to_equal(l_expected);
    end f1;

end test_f;
/
```


Code coverage
File | /private/var/folders/db/2cfk09cx239_2r3bsdpgfwhr0000gn/T/utplsql_92163620524329220...
function redstack.f

REDSTACK.F
100 % lines covered
1 relevant lines. 1 lines covered (including 1 lines partially covered) and 0 lines missed

```
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;
```


Code coverage
File | /private/var/folders/db/2cfk09cx239_2r3bsdpgfwhr0000gn/T/utplsql_19859648123934800...
Incognito
function redstack.f

REDSTACK.F
100 % lines covered
1 relevant lines. 1 lines covered and 0 lines missed

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

Same test, different coverage due to different implementation

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