# How to Develop an MLE Module with Oracle Database 23ai?

Philipp Salvisberg
19th November 2024

GRISSELBAV
simple & accurate IT solutions.

# Welcome

**Philipp Salvisberg**
Founder, Owner and CEO of Grisselbav GmbH

- Database-centric Development
- Model-driven Software Development
- Open-Source Development

philipp.salvisberg@grisselbav.com

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

https://github.com/PhilippSalvisberg/js23c/tree/main/sandbox3

# Using Third Party Library

# Check Validity of E-Mail Addresses

```
select e_mail, validator_api.is_email(e_mail) as is_valid
  from (values
          ('esther.muster@example.com'),
          ('Esther Muster <esther.muster@example.com>'),
          ('esther.muster@dubious.com')
       ) test_data (e_mail);
```

```
E_MAIL                                      IS_VALID
------------------------------------------- ----------
esther.muster@example.com                            1
Esther Muster <esther.muster@example.com>            0
esther.muster@dubious.com                            0
```

# Install MLE Module

```
script https://raw.githubusercontent.com/PhilippSalvisberg/mle-sqlcl/main/mle.js
install validator_mod https://esm.run/validator@13.12.0 13.12.0

select version, language_name, length(module)
  from user_mle_modules
 where module_name = 'VALIDATOR_MOD';
```

```
MLE module VALIDATOR_MOD compiled

VERSION    LANGUAGE_NAME       LENGTH(MODULE)
---------- ---------------- ----------------
13.12.0    JAVASCRIPT                 123260
```

# MLE Module Wrapper

PL/SQL Package Specification

```
create or replace package validator_api is
   function is_email(in_email in varchar2)
      return boolean deterministic;
end validator_api;
/
```

Hidden MLE Call Spec in the package body!

PL/SQL Package Body

```
create or replace package body validator_api is
   function is_email_internal(
      in_email   in varchar2,
      in_options in json
   ) return boolean deterministic as mle module
   validator_mod signature 'default.isEmail(string, any)';

   function is_email(in_email in varchar2)
      return boolean deterministic is
   begin
      return is_email_internal(
                 in_email   => in_email,
                 in_options => json('
                    {"allow_display_name": false,
                     "allow_undescores": false,
                     "require_display_name": false,
                     "allow_utf8_local_part": true,
                     "require_tld": true,
                     "allow_ip_domain": false,
                     "domain_specific_validation": false,
                     "blacklisted_chars": "",
                     "ignore_max_length": false,
                     "host_blacklist": ["dubious.com"],
                     "host_whitelist": []}'));
   end is_email;
end validator_api;
```

# Check Validity of E-Mail Addresses (2)

```
select e_mail, validator_api.is_email(e_mail) as is_valid
  from (values
          ('esther.muster@example.com'),
          ('esther.muster@my_example.com'),
          ('esther_muster@example.com')
       ) test_data (e_mail);
```

```
E_MAIL                          IS_VALID
------------------------------ ----------
esther.muster@example.com               1
esther.muster@my_example.com            0
esther_muster@example.com               1
```

# Use Case of Own Library

# Create Emp/Dept in Current Schema

```
exec demo.create_tabs;
select listagg(deptno, ', ') from dept;
select listagg(ename, ', ') from emp;
```

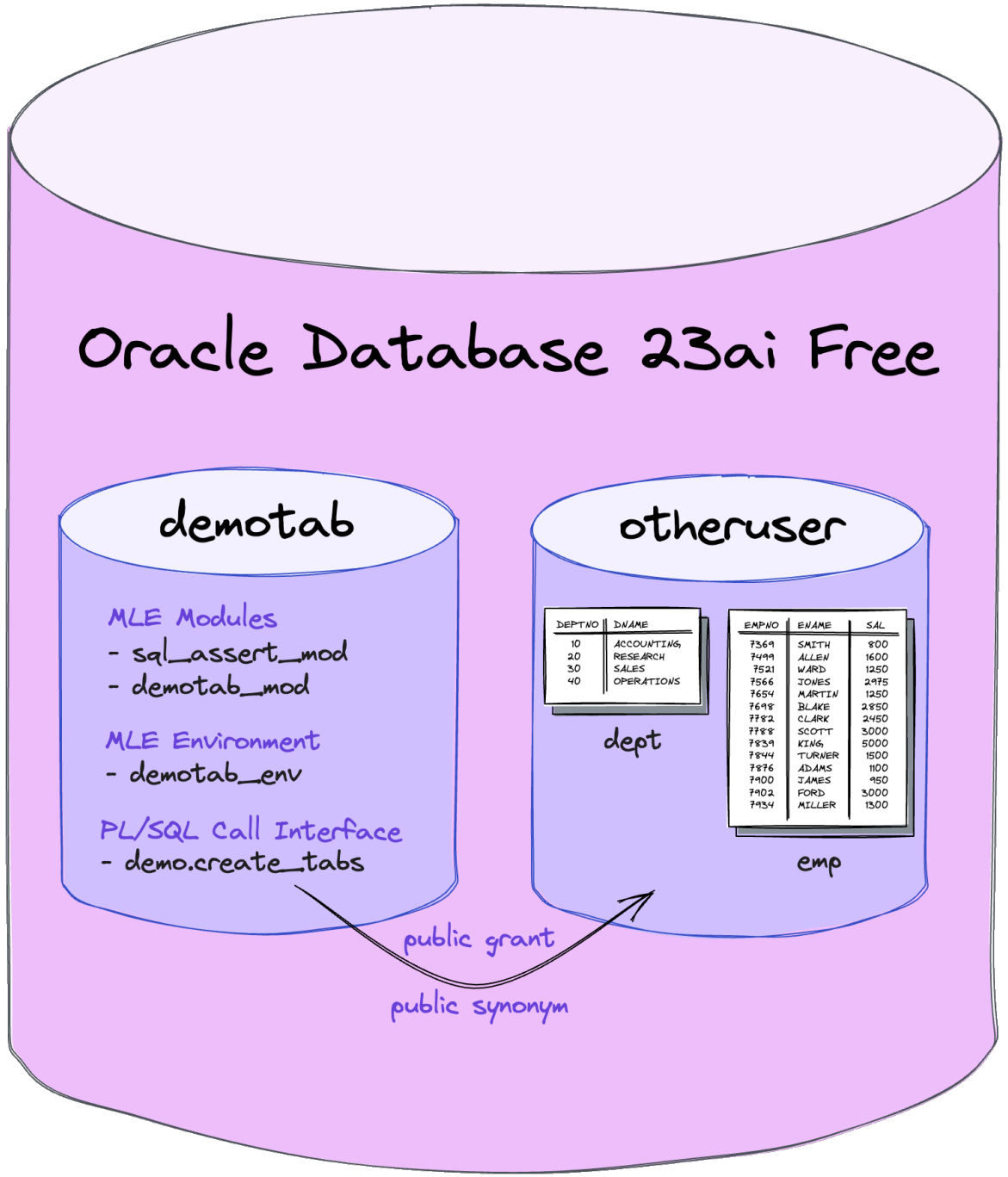
```
PL/SQL procedure successfully completed.

LISTAGG(DEPTNO,',')
--------------------
10, 20, 30, 40

LISTAGG(ENAME,',')
--------------------------------------------------------------------------------
CLARK, SCOTT, BLAKE, ADAMS, WARD, JONES, FORD, ALLEN, JAMES,
MILLER, SMITH, MARTIN, KING, TURNER
```

# More Requirements

- Call to "demo.create_tab" is idempotent
- Data correspond to SCOTT schema
- Can pass an alternative table names
- Error when using invalid SQL names (avoid SQL injection)
- Error when using quoted names
- Create referential integrity constraints
- Add missing rows
- Reset changed rows
- Keep rows with unknown primary key

# Design & Development Approach

# Design


Oracle Database 23ai Free
demotab
MLE Modules
- sql_assert_mod
- demotab_mod
MLE Environment
- demotab_env
PL/SQL Call Interface
- demo.create_tabs
otheruser
DEPTNO DNAME
10 ACCOUNTING
20 RESEARCH
30 SALES
40 OPERATIONS
dept
EMPNO ENAME SAL
7369 SMITH 800
7499 ALLEN 1600
7521 WARD 1250
7566 JONES 2975
7654 MARTIN 1250
7698 BLAKE 2850
7782 CLARK 2450
7788 SCOTT 3000
7839 KING 5000
7844 TURNER 1500
7876 ADAMS 1100
7900 JAMES 950
7902 FORD 3000
7934 MILLER 1300
emp
public grant
public synonym

# Development Approach

## Development Outside of DB with Visual Studio Code

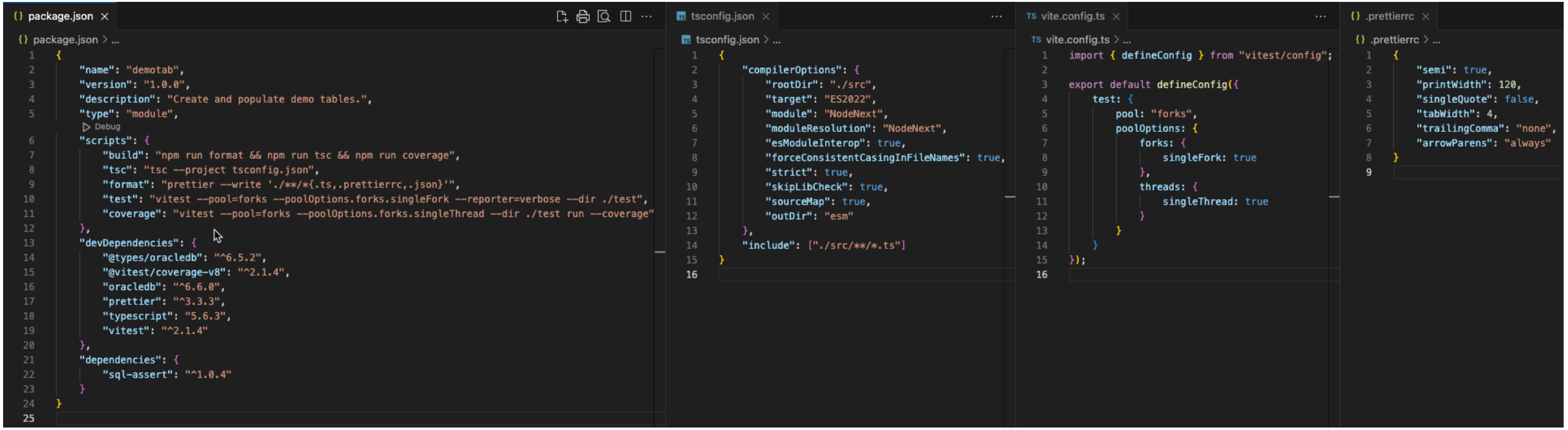
- TypeScript
- Node.js
- node-oracledb
- sql-assert
- Vitest

## Deployment into DB with SQLcl

- JavaScript MLE modules
- MLE environment
- PL/SQL Call Specification

# Demo 1: Project Setup

# Configuration



package.json

```
{
    "name": "demotab",
    "version": "1.0.0",
    "description": "Create and populate demo tables.",
    "type": "module",
    "scripts": {
        "build": "npm run format && npm run tsc && npm run coverage",
        "tsc": "tsc --project tsconfig.json",
        "format": "prettier --write './**/*{.ts,.prettierrc,.json}'",
        "test": "vitest --pool=forks --poolOptions.forks.singleFork --reporter=verbose --dir ./test",
        "coverage": "vitest --pool=forks --poolOptions.forks.singleThread --dir ./test run --coverage"
    },
    "devDependencies": {
        "@types/oracledb": "^6.5.2",
        "@vitest/coverage-v8": "^2.1.4",
        "oracledb": "^6.6.0",
        "prettier": "^3.3.3",
        "typescript": "5.6.3",
        "vitest": "^2.1.4"
    },
    "dependencies": {
        "sql-assert": "^1.0.4"
    }
}
```

tsconfig.json

```
{
    "compilerOptions": {
        "rootDir": "./src",
        "target": "ES2022",
        "module": "NodeNext",
        "moduleResolution": "NodeNext",
        "esModuleInterop": true,
        "forceConsistentCasingInFileNames": true,
        "strict": true,
        "skipLibCheck": true,
        "sourceMap": true,
        "outDir": "esm"
    },
    "include": ["./src/**/*.ts"]
}
```

vite.config.ts

```
import { defineConfig } from "vitest/config";

export default defineConfig({
    test: {
        pool: "forks",
        poolOptions: {
            forks: {
                singleFork: true
            },
            threads: {
                singleThread: true
            }
        }
    }
});
```

.prettierrc

```
{
    "semi": true,
    "printWidth": 120,
    "singleQuote": false,
    "tabWidth": 4,
    "trailingComma": "none",
    "arrowParens": "always"
}
```

# Install NPM Modules

```
npm install
```

```
Debugger listening on ws://127.0.0.1:52457/b5a359a2-0cc6-45fd-babc-74abe3d36ffb
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
(node:89799) ExperimentalWarning: CommonJS module
/opt/homebrew/lib/node_modules/npm/node_modules/debug/src/node.js is loading ES Module
/opt/homebrew/lib/node_modules/npm/node_modules/supports-color/index.js using require().
Support for loading ES Module in require() is an experimental feature and might change at
any time
(Use `node --trace-warnings ...` to show where the warning was created)

added 113 packages, and audited 114 packages in 4s

28 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Waiting for the debugger to disconnect...
```

# Demo 2: Develop

# demotabs.ts – Async, Global "session"

TS demotab.ts U

src > TS demotab.ts > ...

```
1   import { simpleSqlName } from "sql-assert";
2   import oracledb from "oracledb";
3
4   // global variable for default connection in the database
5   declare const session: oracledb.Connection;
6
7   /**…
18  export async function create(deptName: string = "dept", empName: string = "emp"): Promise<void> {
19      const dept = simpleSqlName(deptName);
20      const emp = simpleSqlName(empName);
21      await session.execute(`…
27      `);
28      await session.execute(`…
44      `);
45      await session.execute(`…
56      `);
57      await session.execute(`create index if not exists ${emp}_mgr_fk_i on ${emp} (mgr)`);
58      await session.execute(`create index if not exists ${emp}_deptno_fk_i on ${emp} (deptno)`);
59      await session.execute(`alter table ${emp} disable constraint ${emp}_mgr_fk`);
60      await session.execute(`…
91      `);
92      await session.execute(`alter table ${emp} enable constraint ${emp}_mgr_fk`);
93  }
94
```

# Compile TypeScript to JavaScript

```
npm run tsc

Debugger listening on ws://127.0.0.1:60284/be6d64ff-26a6-4807-a0ba-6d39b7de21b2
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

> demotab@1.0.0 tsc
> tsc --project tsconfig.json

Debugger listening on ws://127.0.0.1:60289/8df9f64e-d24b-4ff1-9edc-2607af6682c0
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Waiting for the debugger to disconnect...
Waiting for the debugger to disconnect...
```

# demotab.js – Without Type Definitions

JS demotab.js

esm > JS demotab.js > ...

```
1   import { simpleSqlName } from "sql-assert";
2   /**…
13  export async function create(deptName = "dept", empName = "emp") {
14      const dept = simpleSqlName(deptName);
15      const emp = simpleSqlName(empName);
16      await session.execute(`…
22      `);
23      await session.execute(`…
39      `);
40      await session.execute(`…
51      `);
52      await session.execute(`create index if not exists ${emp}_mgr_fk_i on ${emp} (mgr)`);
53      await session.execute(`create index if not exists ${emp}_deptno_fk_i on ${emp} (deptno)`);
54      await session.execute(`alter table ${emp} disable constraint ${emp}_mgr_fk`);
55      await session.execute(`…
86      `);
87      await session.execute(`alter table ${emp} enable constraint ${emp}_mgr_fk`);
88  }
89  //# sourceMappingURL=demotab.js.map
```

# Create User, Session for Tests

TS dbconfig.ts ×

test > TS dbconfig.ts > ...

```
1   import oracledb from "oracledb";
2
3   let sysSession: oracledb.Connection;
4   export let demotabSession: oracledb.Connection;
5
6   const connectString = "127.0.0.1:1522/freepdb1";
7
8 > const sysConfig: oracledb.ConnectionAttributes = {…
13  };
14
15  export const demotabConfig: oracledb.ConnectionAttributes = {
16      user: "demotab",
17      password: "demotab",
18      connectString: connectString
19  };
20
21  export async function createSessions(): Promise<void> {
22      sysSession = await oracledb.getConnection(sysConfig);
23      await createUser(demotabConfig);
24      sysSession.close();
25      demotabSession = await oracledb.getConnection(demotabConfig);
26  }
27
28 > async function createUser(config: oracledb.ConnectionAttributes): Promise<void> {…
37  }
38
39 > export async function closeSessions(): Promise<void> {…
41  }
```

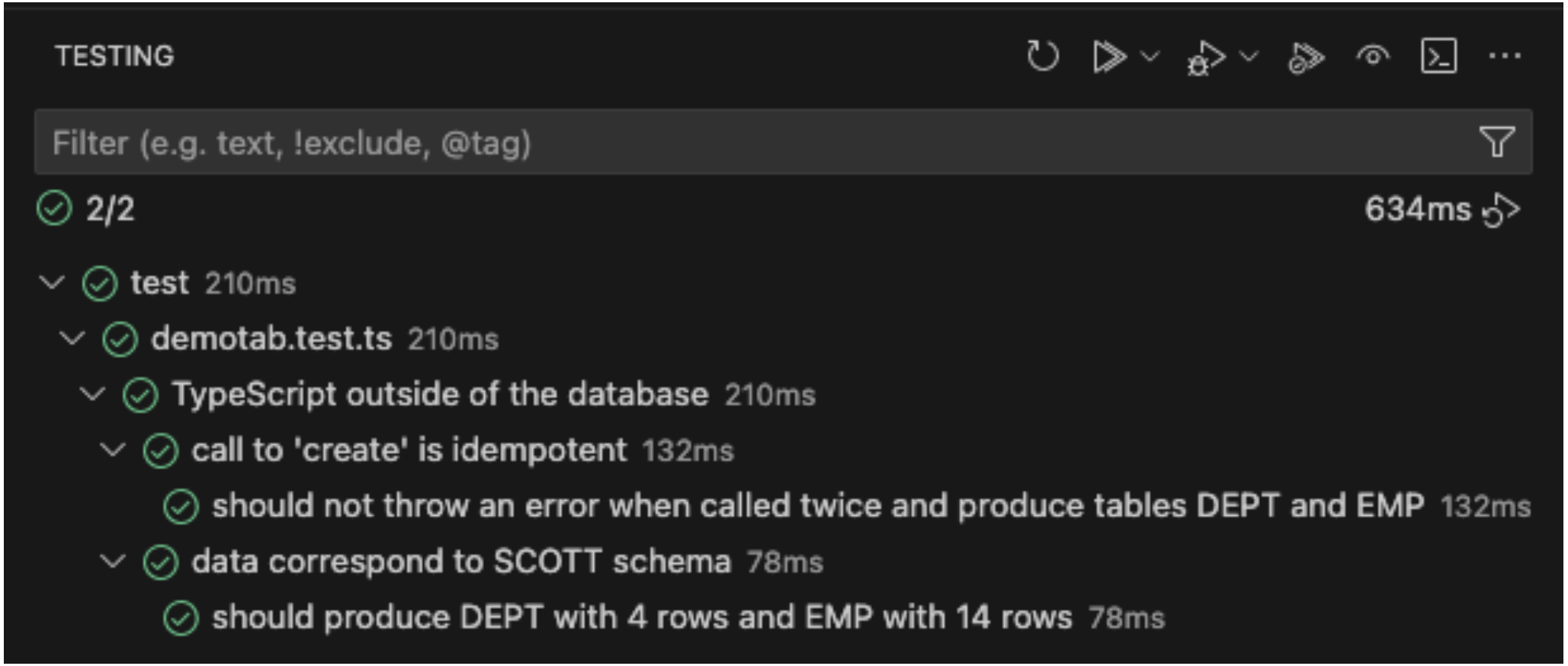
# Tests

TS demotab.test.ts

test > TS demotab.test.ts > ...

```
1   import { beforeAll, beforeEach, afterAll, describe, it, expect } from "vitest";
2   import oracledb from "oracledb";
3   import { createSessions, closeSessions, demotabSession } from "./dbconfig";
4   import { create } from "../src/demotab";
5
6   describe("TypeScript outside of the database", () => {
7       beforeAll(async () => {
8           await createSessions();
9           global.session = demotabSession;
10      });
11
12  >   beforeEach(async () => {…
22      });
23
24  >   afterAll(async () => {…
26      });
27
28      describe("call to 'create' is idempotent", () => {
29          it("should not throw an error when called twice and produce tables DEPT and EMP", async () => {
30              await create();
31              await create();
32              const tabs = await demotabSession.execute("select table_name from user_tables order by table_name");
33              expect(tabs.rows).toEqual([["DEPT"], ["EMP"]]);
34          });
35      });
36
37  >   describe("data correspond to SCOTT schema", () => {…
69      });
70  });
```

# Run Tests


TESTING
Filter (e.g. text, !exclude, @tag)
2/2
634ms
test 210ms
demotab.test.ts 210ms
TypeScript outside of the database 210ms
call to 'create' is idempotent 132ms
should not throw an error when called twice and produce tables DEPT and EMP 132ms
data correspond to SCOTT schema 78ms
should produce DEPT with 4 rows and EMP with 14 rows 78ms

# Run Tests With Code Coverage

```
phs@macphs21 demo % npm run coverage
Debugger listening on ws://127.0.0.1:54575/5571610a-bab7-4f00-94a7-e1fb4fd6788d
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

> demotab@1.0.0 coverage
> vitest --pool=forks --poolOptions.forks.singleThread --dir ./test run --coverage

Debugger listening on ws://127.0.0.1:54600/d8bd94b1-fa7b-4101-8575-5bccc1a563db
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.

 RUN  v2.1.4 /Users/phs/SynologyDrive/MyDrive/Business/Talks/2024/DOAG2024/how-to-develop-an-mle-module-with-oracle-database-23ai/demo
      Coverage enabled with v8

Debugger listening on ws://127.0.0.1:54634/15bfab89-e239-4038-bf44-2d4d1d88cd90
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
 ✓ test/demotab.test.ts (2) 403ms
   ✓ TypeScript outside of the database (2) 403ms
 ✓ test/demotab.test.ts (2) 403ms
   ✓ TypeScript outside of the database (2) 403ms
     ✓ call to 'create' is idempotent (1)
       ✓ should not throw an error when called twice and produce tables DEPT and EMP
     ✓ data correspond to SCOTT schema (1)
       ✓ should produce DEPT with 4 rows and EMP with 14 rows

 Test Files  1 passed (1)
      Tests  2 passed (2)
   Start at  12:42:20
   Duration  830ms (transform 27ms, setup 0ms, collect 69ms, tests 403ms, environment 0ms, prepare 55ms)

 % Coverage report from v8
------------|---------|----------|---------|---------|-------------------
File        | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------|---------|----------|---------|---------|-------------------
All files   |     100 |      100 |     100 |     100 |
 demotab.ts |     100 |      100 |     100 |     100 |
------------|---------|----------|---------|---------|-------------------
Waiting for the debugger to disconnect...
Waiting for the debugger to disconnect...
```

# Code Coverage Report

**All files** demotab.ts

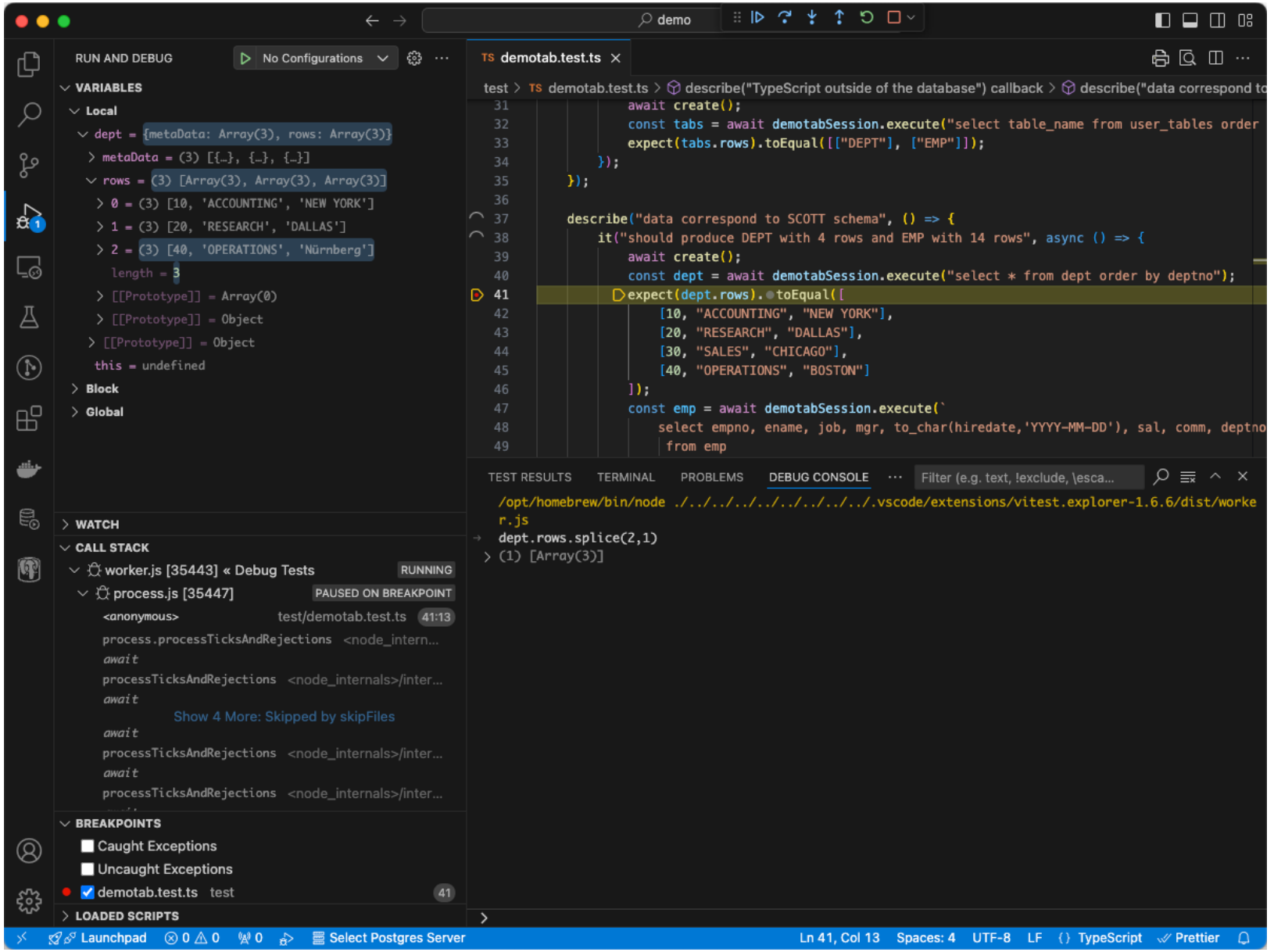
**100%** Statements 25/25 **100%** Branches 1/1 **100%** Functions 1/1 **100%** Lines 25/25

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

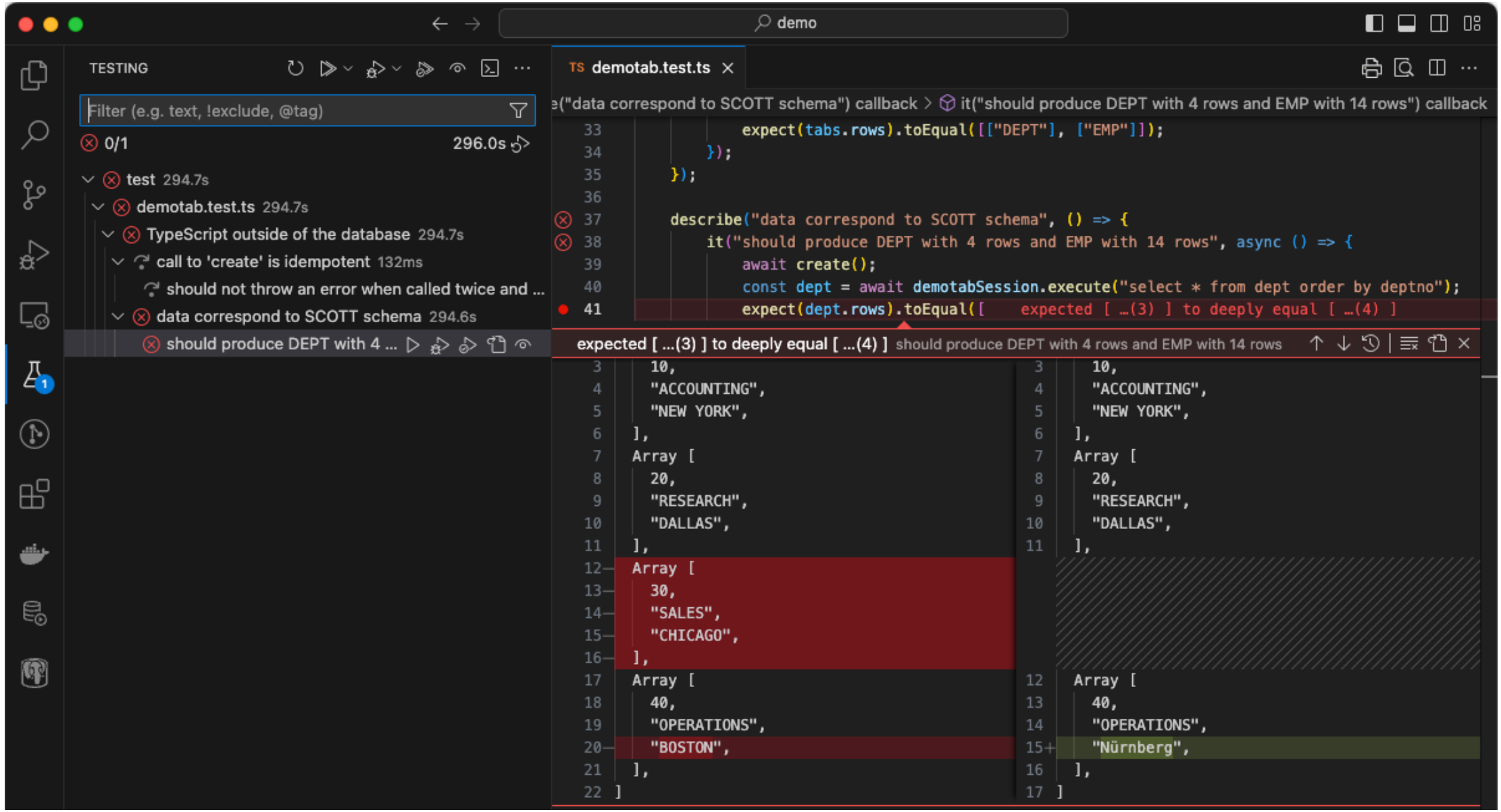
```
1x  import { simpleSqlName } from "sql-assert";
    import oracledb from "oracledb";

    // global variable for default connection in the database
    declare const session: oracledb.Connection;

    /**
     * Creates demo tables with initial data for the well-known tables `dept` and `emp`.
     * Alternative table names can be passed to this function. The tables are not re-created
     * if they already exist. However, the rows for the 4 departments and the 14 employees
     * should be reset to their initial state while other rows are left unchanged.
     * Problems are reported via exceptions.
     *
     * @param [deptName="dept"] name of the dept table.
     * @param [empName="emp"] name of the emp table.
     * @returns {Promise<void>}.
     */
3x  export async function create(deptName: string = "dept", empName: string = "emp"): Promise<void> {
3x      const dept = simpleSqlName(deptName);
3x      const emp = simpleSqlName(empName);
3x      await session.execute(`
3x          create table if not exists ${dept} (
3x             deptno number(2, 0)      not null constraint ${dept}_pk primary key,
                dname  varchar2(14 char) not null,
                loc    varchar2(13 char) not null
            )
3x      `);
3x      await session.execute(`
3x          merge into ${dept} t
            using (values
                     (10, 'ACCOUNTING', 'NEW YORK'),
                     (20, 'RESEARCH',   'DALLAS'),
                     (30, 'SALES',      'CHICAGO'),
                     (40, 'OPERATIONS', 'BOSTON')
                  ) s (deptno, dname, loc)
               on (t.deptno = s.deptno)
             when matched then
                  update
                     set t.dname = s.dname,
                         t.loc = s.loc
             when not matched then
                  insert (t.deptno, t.dname, t.loc)
                  values (s.deptno, s.dname, s.loc)
3x      `);
3x      await session.execute(`
3x          create table if not exists ${emp} (
3x              empno    number(4, 0)      not null  constraint ${emp}_pk primary key,
```

```
                insert (t.deptno, t.dname, t.loc)
                values (s.deptno, s.dname, s.loc)
3x      `);
3x      await session.execute(`
3x          create table if not exists ${emp} (
3x              empno    number(4, 0)      not null  constraint ${emp}_pk primary key,
                ename    varchar2(10 char) not null,
                job      varchar2(9 char)  not null,
3x              mgr      number(4, 0)                constraint ${emp}_mgr_fk references ${emp},
                hiredate date              not null,
                sal      number(7, 2),
                comm     number(7, 2),
3x              deptno   number(2, 0)      not null  constraint ${emp}_deptno_fk references ${dept}
            )
3x      `);
3x      await session.execute(`create index if not exists ${emp}_mgr_fk_i on ${emp} (mgr)`);
3x      await session.execute(`create index if not exists ${emp}_deptno_fk_i on ${emp} (deptno)`);
3x      await session.execute(`alter table ${emp} disable constraint ${emp}_mgr_fk`);
3x      await session.execute(`
3x          merge into ${emp} t
            using (values
                     (7839, 'KING',   'PRESIDENT', null, date '1981-11-17', 5000, null, 10),
                     (7566, 'JONES',  'MANAGER',   7839, date '1981-04-02', 2975, null, 20),
                     (7698, 'BLAKE',  'MANAGER',   7839, date '1981-05-01', 2850, null, 30),
                     (7782, 'CLARK',  'MANAGER',   7839, date '1981-06-09', 2450, null, 10),
                     (7788, 'SCOTT',  'ANALYST',   7566, date '1987-04-19', 3000, null, 20),
                     (7902, 'FORD',   'ANALYST',   7566, date '1981-12-03', 3000, null, 20),
                     (7499, 'ALLEN',  'SALESMAN',  7698, date '1981-02-20', 1600,  300, 30),
                     (7521, 'WARD',   'SALESMAN',  7698, date '1981-02-22', 1250,  500, 30),
                     (7654, 'MARTIN', 'SALESMAN',  7698, date '1981-09-28', 1250, 1400, 30),
                     (7844, 'TURNER', 'SALESMAN',  7698, date '1981-09-08', 1500,    0, 30),
                     (7900, 'JAMES',  'CLERK',     7698, date '1981-12-03',  950, null, 30),
                     (7934, 'MILLER', 'CLERK',     7782, date '1982-01-23', 1300, null, 10),
                     (7369, 'SMITH',  'CLERK',     7902, date '1980-12-17',  800, null, 20),
                     (7876, 'ADAMS',  'CLERK',     7788, date '1987-05-23', 1100, null, 20)
                  ) s (empno, ename, job, mgr, hiredate, sal, comm, deptno)
               on (t.empno = s.empno)
             when matched then
                  update
                     set t.ename = s.ename,
                         t.job = s.job,
                         t.mgr = s.mgr,
                         t.hiredate = s.hiredate,
                         t.sal = s.sal,
                         t.comm = s.comm,
                         t.deptno = s.deptno
             when not matched then
                  insert (t.empno, t.ename, t.job, t.mgr, t.hiredate, t.sal, t.comm, t.deptno)
                  values (s.empno, s.ename, s.job, s.mgr, s.hiredate, s.sal, s.comm, s.deptno)
3x      `);
3x      await session.execute(`alter table ${emp} enable constraint ${emp}_mgr_fk`);
3x  }
```

# Debug


RUN AND DEBUG
No Configurations
VARIABLES
Local
dept = {metaData: Array(3), rows: Array(3)}
metaData = (3) [{…}, {…}, {…}]
rows = (3) [Array(3), Array(3), Array(3)]
0 = (3) [10, 'ACCOUNTING', 'NEW YORK']
1 = (3) [20, 'RESEARCH', 'DALLAS']
2 = (3) [40, 'OPERATIONS', 'Nürnberg']
length = 3
[[Prototype]] = Array(0)
[[Prototype]] = Object
[[Prototype]] = Object
this = undefined
Block
Global
WATCH
CALL STACK
worker.js [35443] « Debug Tests
RUNNING
process.js [35447]
PAUSED ON BREAKPOINT
<anonymous> test/demotab.test.ts 41:13
process.processTicksAndRejections <node_intern...
await
processTicksAndRejections <node_internals>/inter...
await
Show 4 More: Skipped by skipFiles
await
processTicksAndRejections <node_internals>/inter...
await
processTicksAndRejections <node_internals>/inter...
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
demotab.test.ts test 41
LOADED SCRIPTS
demotab.test.ts
test > demotab.test.ts > describe("TypeScript outside of the database") callback > describe("data correspond to
31 await create();
32 const tabs = await demotabSession.execute("select table_name from user_tables order
33 expect(tabs.rows).toEqual([["DEPT"], ["EMP"]]);
34 });
35 });
36
37 describe("data correspond to SCOTT schema", () => {
38 it("should produce DEPT with 4 rows and EMP with 14 rows", async () => {
39 await create();
40 const dept = await demotabSession.execute("select * from dept order by deptno");
41 expect(dept.rows).toEqual([
42 [10, "ACCOUNTING", "NEW YORK"],
43 [20, "RESEARCH", "DALLAS"],
44 [30, "SALES", "CHICAGO"],
45 [40, "OPERATIONS", "BOSTON"]
46 ]);
47 const emp = await demotabSession.execute(`
48 select empno, ename, job, mgr, to_char(hiredate,'YYYY-MM-DD'), sal, comm, deptno
49 from emp
TEST RESULTS
TERMINAL
PROBLEMS
DEBUG CONSOLE
Filter (e.g. text, !exclude, \esca...
/opt/homebrew/bin/node ./../../../../../../../../.vscode/extensions/vitest.explorer-1.6.6/dist/worke
r.js
dept.rows.splice(2,1)
(1) [Array(3)]
Launchpad
0 0
0
Select Postgres Server
Ln 41, Col 13
Spaces: 4
UTF-8
LF
TypeScript
Prettier

# Unexpected Test Results



# Demo 3: Deploy

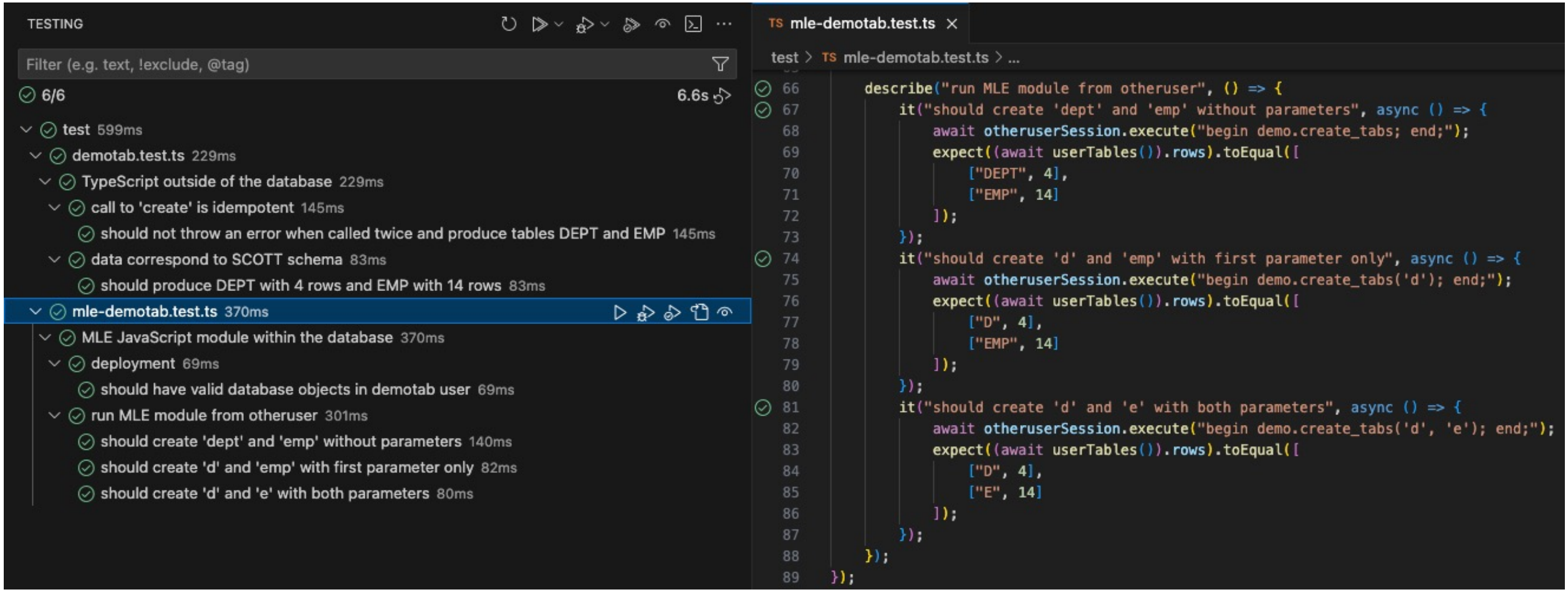
deploy.sql

```
mle install sql_assert_mod https://esm.run/sql-assert@1.0.4 1.0.4
mle install demotab_mod ./esm/demotab.js 1.0.0

create or replace mle env demotab_env
   imports('sql-assert' module sql_assert_mod)
   language options 'js.strict=true, js.console=false, js.polyglot-builtin=true';

create or replace package demo authid current_user is
   procedure create_tabs as
   mle module demotab_mod env demotab_env signature 'create()';

   procedure create_tabs(
      in_dept_table_name in varchar2
   ) as mle module demotab_mod env demotab_env signature 'create(string)';

   procedure create_tabs(
      in_dept_table_name in varchar2,
      in_emp_table_name  in varchar2
   ) as mle module demotab_mod env demotab_env signature 'create(string, string)';
end demo;
/

grant execute on demo to public;
create or replace public synonym demo for demotab.demo;
```

# Tests

```
describe("MLE JavaScript module within the database", () => {
    const timeout = 15000;

    async function userTables(): Promise<oracledb.Result<unknown>> {…
    }

    beforeAll(async () => {
        await createSessions();
        const execAsync = util.promisify(exec);
        await execAsync(
            `sql -S ${demotabConfig.user}/${demotabConfig.password}@${demotabConfig.connectString} @deploy.sql`
        );
    }, timeout);

    beforeEach(async () => {…
    });

    afterAll(async () => {…
    });

    describe("deployment", () => {
        it("should have valid database objects in demotab user", async () => {
            const mods = await demotabSession.execute(`
                select object_type, object_name, status
                  from user_objects
                 order by object_type, object_name
            `);
            expect(mods.rows).toEqual([
                ["MLE ENVIRONMENT", "DEMOTAB_ENV", "VALID"],
                ["MLE MODULE", "DEMOTAB_MOD", "VALID"],
                ["MLE MODULE", "SQL_ASSERT_MOD", "VALID"],
                ["PACKAGE", "DEMO", "VALID"]
            ]);
        });
    });

    describe("run MLE module from otheruser", () => {…
    });
});
```

# Run All Tests



# Add Tests to Cover All Requirements

```
Debugger listening on ws://127.0.0.1:60278/59009ae8-5221-4ecf-b797-1cbffcbe21e6
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
 ✓ test/demotab.test.ts (16) 1586ms
   ✓ TypeScript outside of the database (16) 1586ms
     ✓ call to 'create' is idempotent (1)
       ✓ should not throw an error when called twice and produce tables DEPT and EMP
     ✓ data correspond to SCOTT schema (1)
       ✓ should produce DEPT with 4 rows and EMP with 14 rows
     ✓ can pass an alternative table names (3)
       ✓ should produce table D and EMP
       ✓ should produce table DEPT and E
       ✓ should produce table D and E
     ✓ error when using invalid SQL names (avoid SQL injection) (2)
       ✓ should throw an error for invalid dept name
       ✓ should throw an error for invalid emp name
     ✓ error when using quoted names (2)
       ✓ should throw an error for quoted dept name
       ✓ should throw an error for quoted emp name
     ✓ create referential integrity constraints' (1)
       ✓ should produce referential integrity constraints
     ✓ add missing rows (2)
       ✓ should add deleted rows in DEPT table
       ✓ should add deleted rows in EMP table
     ✓ reset changed rows (2)
       ✓ should reset changed rows in DEPT table
       ✓ should reset changed rows in EMP table
     ✓ keep rows with unknown primary key (2)
       ✓ should keep new rows in DEPT table
       ✓ should add deleted rows in EMP table
 ✓ test/mle-demotab.test.ts (4) 6025ms
   ✓ MLE JavaScript module within the database (4) 6025ms
     ✓ deployment (1)
       ✓ should have valid database objects in demotab user
     ✓ run MLE module from otheruser (3) 311ms
       ✓ should create 'dept' and 'emp' without parameters
       ✓ should create 'd' and 'emp' with first parameter only
       ✓ should create 'd' and 'e' with both parameters

 Test Files  2 passed (2)
      Tests  20 passed (20)
   Start at  15:22:59
   Duration  8.01s (transform 37ms, setup 0ms, collect 90ms, tests 7.61s, environment 0ms, prepare 49ms)
```

# Core Messages

# Develop MLE Modules Outside of the DB

- VS Code has everything you need
  - Editor, Formatter, Linter
  - TypeScript
  - Test Integration
  - Debugger
- Deploy unaltered and tested JavaScript code with SQLcl
- Enables integration into a CI/CD pipeline
- Suitable for code with/without SQL

# Thank you!

GRISSELBAV

simple & accurate IT solutions.