

The Trivadis PL/SQL & SQL Coding Guidelines Are Dead – What Now?

Philipp Salvisberg
14th May 2025



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/>

Introduction (History)

Trivadis PL/SQL & SQL Coding Guidelines



v1.2
August 2009
52 pages
Word



v2.0
August 2011
57 pages
Word



v3.0
January 2016
149 pages
Word



v3.3
July 2018
141 pages
MkDocs



v4.4
March 2024
208 pages
MkDocs

Tooling in 2009/2010?

TRIVADIS PL/SQL ASSESSMENT

SWISS IT UP!

EXPERIENCE IT
GARANTIERTE SOFTWARE-QUALITÄT – EIN GUTES GEFÜHLE!

Trivadis ist ein erfolgreiches Schweizer Unternehmen für Trainings und Dienstleistungen mit über 15 Jahren Erfahrung und mehr als 550 Mitarbeitern in der Schweiz, Deutschland und Österreich!

Ausgangspunkt unserer Kundenliste:
AS TRAC – Bundesamt für Steuern, BLS Systems, BMW AG, Brechtler Ingenieurbüro, Deutsche Leihhaus, EBN Zürich, FIBA Zürich, Groupa Mutual, Novartis Pharma, P&A, Laury Group, PostFinance AG, Swisscom, SWISS-Internet und A1 Linien AG, Springer, UBS AG

trivadis
makes IT easier.

USE IT
UNSERE GUIDELINES – JETZT DOWNLOADEN!

Unsere «PL/SQL und SQL Coding Guidelines» ermöglichen Ihnen eine optimale und standardisierte Codierung von PL/SQL Anwendungen.

«Roger and his team have done an excellent job of providing a comprehensive set of clear standards that will undoubtedly improve the quality of your code. If you do not yet have standards in place, you should give strong consideration to using these as a starting point.»
Steven Feuerstein, PL/SQL Evangelist

Kostenlos Download unter:
www.trivadis.com/plsql

LEARN IT
VON DEN BESTEN LERNEN.

Profitieren Sie von unseren Best-Practice-Kursen!

Einführung in PL/SQL:
www.trivadis.com/o-pl/sql/

PL/SQL für Fortgeschrittene:
www.trivadis.com/o-pl/sql-adv/

Alle Kursbewertungen unserer Teilnehmer finden Sie direkt bei den Kursbeschreibungen.

KNOW IT
DAS GEBALLTE WISSEN UNSERER PL/SQL CRACKS.

- 1 Roger Troller, Senior Consultant im Oracle Umfeld, seit über 20 Jahren im IT-Business, Autor der «Trivadis PL/SQL und SQL Guidelines»
- 2 Perry Pakul, Technology Manager für Oracle Based Development, seit über 20 Jahren im IT-Business
- 3 Daniel Liebhart, Solution Manager Application Development, fokussiert auf den Bereich «Service Oriented Architecture (SOA)», seit über 25 Jahren im IT-Business

CHECK IT – PL/SQL
DAS ASSESSMENT FÜR IHRE PL/SQL ANWENDUNG!

Lesen Sie Ihre PL/SQL Anwendung auf Qualität, Wartbarkeit und Optimierungspotenzial checken – zum Eigenen von CHF 5000.- / EUR 3000.-

Unser umfassendes Assessment beruht auf unseren «PL/SQL und SQL Guidelines».

Mehr Infos unter:
www.trivadis.com/plsql

GET IT
PROFITIEREN SIE VON UNSEREM PL/SQL ASSESSMENT!

Ihre Vorteile:

- Klare Aussagen zur Qualität des Source Codes nach den «PL/SQL und SQL Guidelines» von Trivadis
- Klare Empfehlungen hinsichtlich qualitätssichernder Massnahmen
- Abschreibbarkeit mit Ergebnissen und Empfehlungen
- Deutlich einfachere Wartungs- und Weiterentwicklungsmöglichkeiten
- Steigerung der Code-Flexibilität
- Bessere Produktivität zur Bereitstellung datenbanknahen Funktionen
- Mehr Transparenz in zentralen Applikationen

Preiswerte: CHF 5000.- / EUR 3000.-
Für max. 10000 Lines of Code oder 3 Anträge

DO IT
LOS GEHT'S: NEHMEN SIE JETZT KONTAKT AUF!

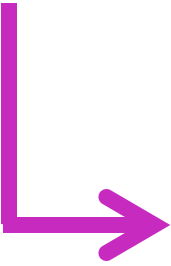
www.trivadis.com/plsql/

Direkter Kontakt:
Trivadis AG
Perry Pakul
mcassment@trivadis.com
Tel. 0800 87 48 23 47
(kostenlos Rufnummer)

No custom rules
Sonar used also
CodeXpert back then

- Cook-Book
 - Quest CodeXpert
 - SQL scripts with/without PL/Scope
 - Manual code reviews
 - Interviews

renamed to
PL/SQL Cop,
db* CODECOP



Code Checker Prototype
April 2010
G-3110, G-5020, G-6010



Requirements (My Whish List)

Guidelines



Similar rules as in the Trivadis Guidelines



Configurable



Extensible

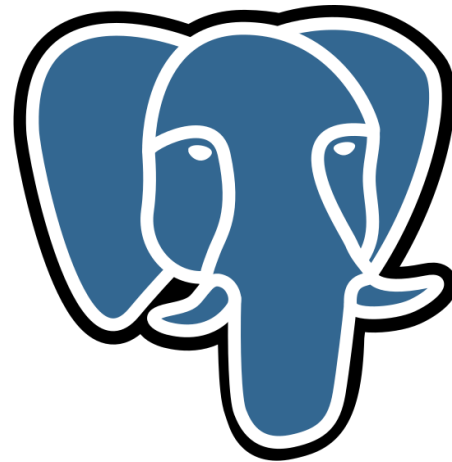


Regular updates

Grammar for Static Code Analysis

- Oracle Database
 - SQL*Plus
 - SQLcl
 - SQL
 - PL/SQL

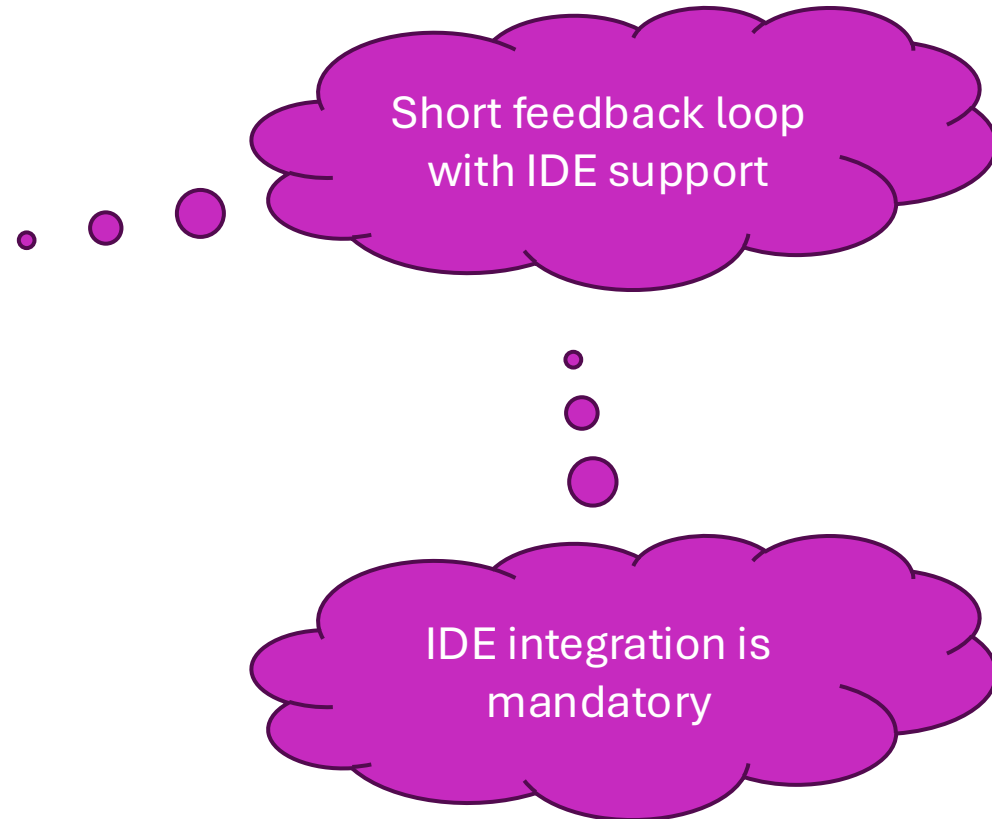
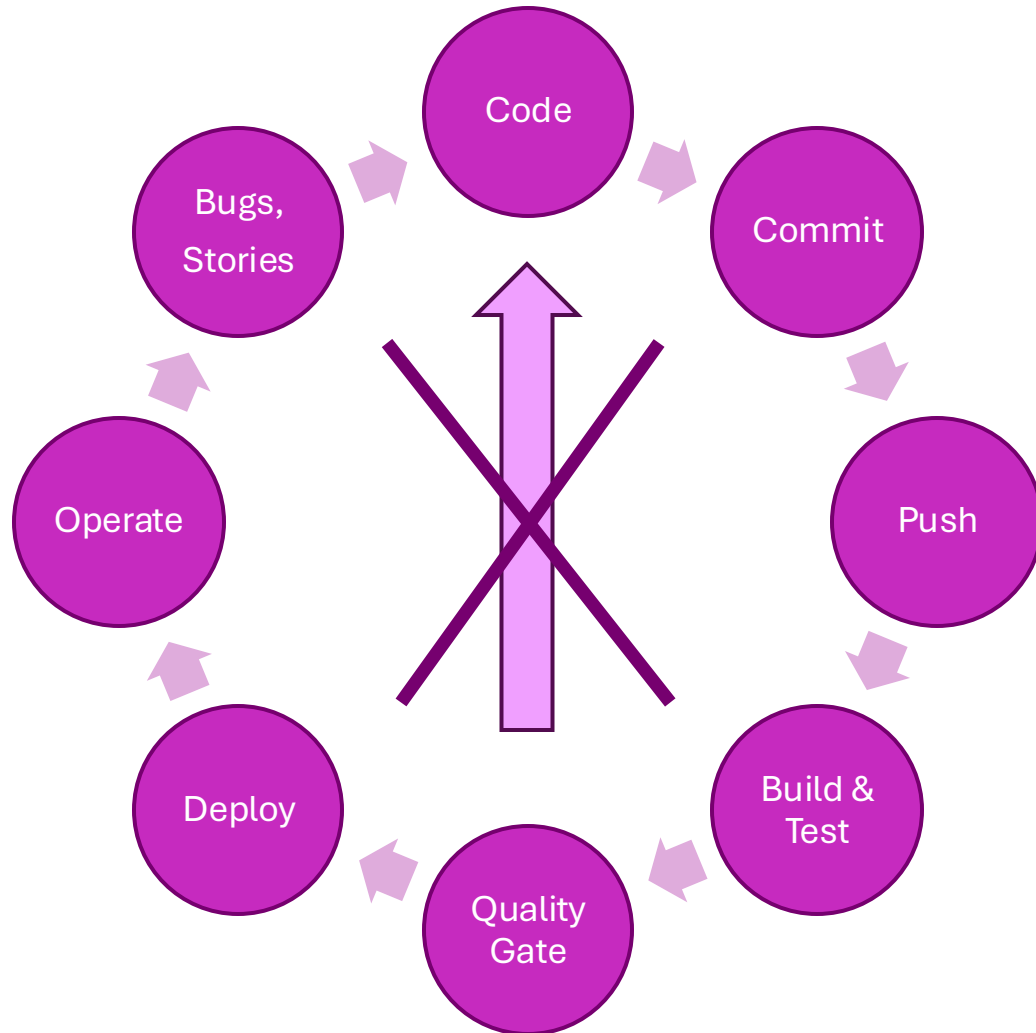
ORACLE Database 23^{ai}



PostgreSQL 17

- PostgreSQL
 - psql
 - SQL
 - PL/pgSQL
- Regular updates

DevOps Lifecycle – Focus Development



Tooling


- Integration
 - IDEs (VS Code, SQL Developer, PL/SQL Developer, TOAD, Data Grip, DBeaver, ...)
 - Build Pipelines (Jenkins, Bamboo, GitHub, GitLab, Bitbucket, OCI, Azure, AWS, Google Cloud, ...)
 - Code Quality Repositories (SonarQube Community Build/Server/Cloud, Codacy, ...)
- Static code analysis
 - Report issues with clear messages and quick fixes, if applicable
 - Configurable rules (parameters and SQL lookups)
 - Custom rules
- SQL-based Test
 - Implement rules via SQL queries returning issues and migration scripts if applicable
- Shared configuration

Guidelines (for Static Code Analysis)

Trivadis PL/SQL & SQL Coding Guidelines

- 124 rules
 - PL/SQL
 - Oracle SQL
- Apache-2.0 license
- Fork on [GitHub](#)
- SQLDev extension; CLI; SonarQube plugin

G-1080: Avoid using the same expression on both sides of a relational comparison operator or a logical operator.

 Blocker

Maintainability, Efficiency, Testability

Reason

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code and should be simplified.

This rule ignores operators `+`, `*` and `||`, and expressions: `1=1`, `1<>1`, `1!=1`, `1~=1` and `1^=1`.

Example (bad)

```
1 declare
2   co_max_salary constant emp.salary%type := 3000;
3 begin
4   select emp.first_name
5          ,emp.last_name
6          ,emp.salary
7          ,emp.hire_date
8   from employees emp
9   where emp.salary > co_max_salary
10        or emp.salary > co_max_salary
11   order by emp.last_name,emp.first_name;
12 end;
13 /
```

Example (good)

```
1 declare
2   co_max_salary constant emp.salary%type := 3000;
3 begin
4   select emp.first_name
5          ,emp.last_name
6          ,emp.salary
7          ,emp.hire_date
8   from employees emp
9   where emp.salary > co_max_salary
10   order by emp.last_name,emp.first_name;
11 end;
12 /
```

SQLcl Codescan

- 106 Rules
 - 94 Trivadis Coding Guidelines
 - 12 PSR rules
 - Implemented as Arbori Queries
 - No list of supported rules
- [NFTC-license](#)
- CLI



```
sql -nolog <<EOF
codescan -path d2-1080-nc.sql -format json -ignore G-1030,G-5010
exit
EOF
```

```
SQLcl: Release 25.1 Production on Sun May 04 19:42:42 2025
```

```
Copyright (c) 1982, 2025, Oracle. All rights reserved.
```

```
1 files, 1 total distinct warnings
```

```
[
{"file": "d2-1080-nc.sql", "issues": [
  { "line": 7, "col": 7, "ruleNo": "G-1080", "msg": "Avoid using the same
expression on both sides of a relational comparison operator or a
logical operator"}
]}
]
```

PMD

- 21 rules
 - PL/SQL
- BSD-style license
- Fork on [GitHub](#)
- CLI

TomKytesDespair

Since: PMD 5.1

Priority: Medium (3)

"WHEN OTHERS THEN NULL" hides all errors - (Re)RAISE an exception or call RAISE_APPLICATION_ERROR

This rule is defined by the following XPath expression:

```
//ExceptionHandler[QualifiedName/@Image='OTHERS']  
  [count(Statement)=1]  
  [Statement/UnlabelledStatement/Expression/PrimaryPrefix/Literal/NullLiteral]
```

Example(s):

```
CREATE OR REPLACE PACKAGE BODY update_planned_hrs  
IS  
  
PROCEDURE set_new_planned (p_emp_id IN NUMBER, p_project_id IN NUMBER, p_hours IN NUMBER)  
IS  
BEGIN  
  UPDATE employee_on_activity ea  
  SET ea.ea_planned_hours = p_hours  
  WHERE  
    ea.ea_emp_id = p_emp_id  
    AND ea.ea_proj_id = p_project_id;  
  
EXCEPTION  
  WHEN NO_DATA_FOUND THEN  
    RAISE_APPLICATION_ERROR (-20100, 'No such employee or project');  
  
END set_new_planned;  
  
FUNCTION existing_planned (p_emp_id IN NUMBER, p_project_id IN NUMBER) RETURN NUMBER  
IS  
  
existing_hours NUMBER(4);  
  
BEGIN  
  SELECT ea.ea_planned_hours INTO existing_hours  
  FROM employee_on_activity ea  
  WHERE  
    ea.ea_emp_id = p_emp_id  
    AND ea.ea_proj_id = p_project_id;  
  
  RETURN (existing_hours);  
  
EXCEPTION  
  WHEN OTHERS THEN NULL;  
  
END existing_planned;  
  
END update_planned_hrs;  
/
```

Use this rule by referencing it:

```
<rule ref="category/plsql/bestpractices.xml/TomKytesDespair" />
```

SonarQube

- 189 rules (\geq Developer Edition)
 - PL/SQL
 - Oracle SQL
- 89 rules (\geq Developer Edition)
 - T-SQL
 - SQL Sever SQL
- LGPL-3.0 license
(only for parts of Community Build)
- XPath 1.0 based custom rules
- VS Code extension, DataGrip plugin;
CLI; SonarQube plugin

Identical expressions should not be used on both sides of a binary operator

Analyze your code

Intentionality - Logical Reliability

Bug suspicious

Why is this an issue?

More Info

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified.

This rule ignores operators +, * and |, and expressions: 1=1, 1<>1, 1!=1, 1-=1 and 1^=1.

Noncompliant code example

```
SELECT code
FROM Person
WHERE first_name IS NULL OR first_name IS NULL; -- Noncompliant

SELECT * FROM Users
INNER JOIN Clients ON Clients.id = Clients.id; -- Noncompliant
```

Compliant solution

```
SELECT code
FROM Person
WHERE first_name IS NULL OR last_name IS NULL;

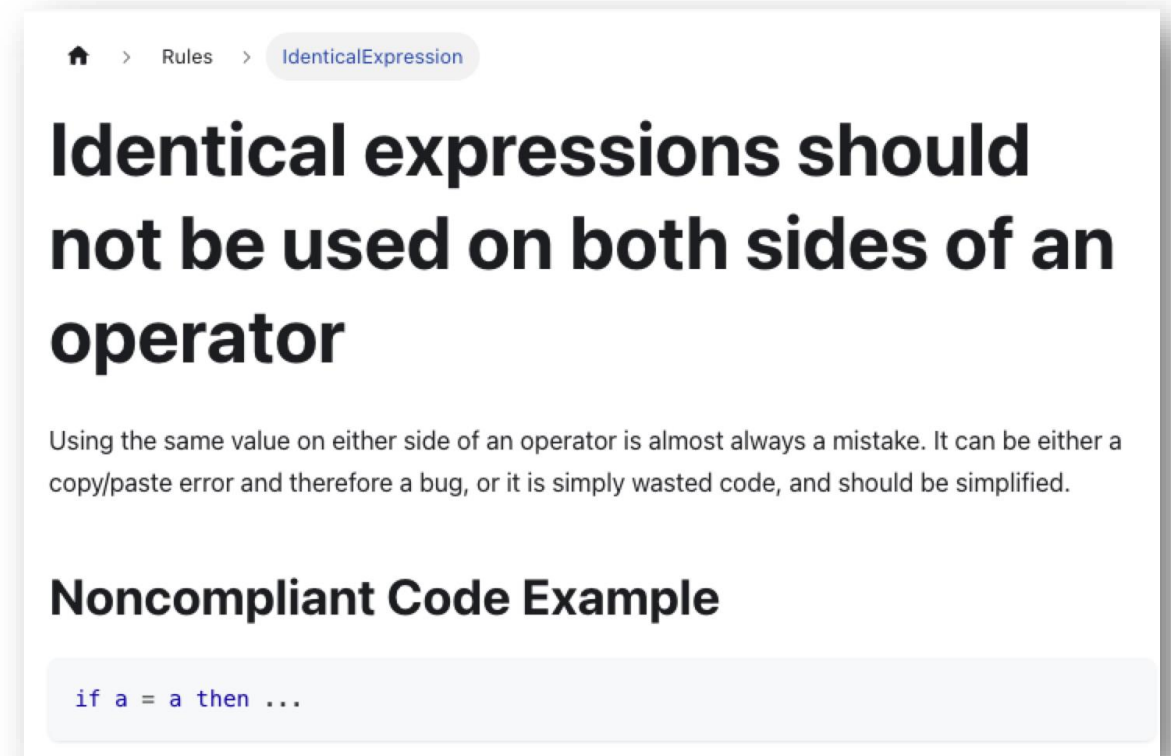
SELECT * FROM Users
INNER JOIN Clients ON Clients.id = Users.id;
```

Exceptions

This rule ignores *, +, and =.

ZPA

- 54 rules (all versions of SonarQube Server)
 - PL/SQL
 - Oracle SQL
- LGPL-3.0 license
- Fork on [GitHub](#)
- XPath 1.0 based custom rules
- CLI; SonarQube plugin



The screenshot shows a web page for a SonarQube rule. At the top, there is a breadcrumb navigation: a home icon, a right arrow, the word 'Rules', another right arrow, and a blue pill-shaped button containing the text 'IdenticalExpression'. Below this is the main title of the rule: 'Identical expressions should not be used on both sides of an operator'. Underneath the title is a descriptive paragraph: 'Using the same value on either side of an operator is almost always a mistake. It can be either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified.' Below the paragraph is a section header 'Noncompliant Code Example'. Under this header is a code block with a light blue background containing the text 'if a = a then ...'.

SQLFluff

- 67 rules
 - SQL (ANSI, Athena, BigQuery, Clickhouse, Databricks, Db2, DuckDB, Exasol, Greenplum, Hive, Impala, MariaDB, Materialize, MySQL, Oracle, PostgreSQL, Redshift, Snowflake, SOQL, Spark SQL, SQLite, StarRocks, Teradata, Trino, T-SQL, Vertica)
- MIT license
- Fork on [GitHub](#)
- VS Code extension; CLI

Code **ST10**

Rule **structure.constant_expression**

Redundant constant expression.

Including an expression that always evaluates to either `TRUE` or `FALSE` regardless of the input columns is unnecessary and makes statements harder to read and understand.

Constant conditions are sometimes mistakes (by mistyping the column name intended), and sometimes the result of incorrect information that they are necessary in some circumstances. In the former case, they can sometimes result in a cartesian join if it was supposed to be a join condition. Given the ambiguity of intent, this rule does not suggest an automatic fix, and instead invites the user to resolve the problem manually.

Name: `structure.constant_expression`

Groups: `all, structure`

Anti-pattern

```
SELECT *
FROM my_table
-- This following WHERE clause is redundant.
WHERE my_table.col = my_table.col
```

Best practice

```
SELECT *
FROM my_table
-- Replace with a condition that includes meaningful logic,
-- or remove the condition entirely.
WHERE my_table.col > 3
```

dbLinter

- 183 rules
 - Oracle Database (≥ 7.3.4): SQL, PL/SQL, SQL*Plus, SQLcl
 - PostgreSQL (≥ 7.0.3): SQL, PL/pgSQL, psql
- IslandSQL parser (Apache-2.0 license)
- EULA
- Joint Venture between Grisselbav & United Codes
- Free plans
- Custom rules with professional subscription
- VS Code extension, CLI, SonarQube plugin

G-1080 Free Error

Avoid using the same expression on both sides of a relational comparison operator or a logical operator.

General

Rule Details

Severity Level	Quality Area	Fixing times
Blocker	Maintainability	5 mins
		Quick Fix Local-medium

Database Support

OracleDB (v7.3.4 +)

Reason

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code and should be simplified.

This rule ignores operators `+`, `*` and `||`, and expressions: `1=1`, `1<>1`, `1!=1`, `1<=1` and `1*=1`.

Example

✗ Non-Compliant Example	✓ Compliant Solution - ★★★★★
--------------------------------------	---

```
1 begin
2   select emp.first_name
3         ,emp.last_name
4         ,emp.salary
5         ,emp.hire_date
6   from employees emp
7  where emp.salary > 3000
8         or emp.salary > 3000
9  order by emp.last_name,emp.first_name;
10 end;
11 /
```

```
1 begin
2   select emp.first_name
3         ,emp.last_name
4         ,emp.salary
5         ,emp.hire_date
6   from employees emp
7  where emp.salary > 3000
8  order by emp.last_name,emp.first_name;
9  end;
10 /
```

Parameters

Use parameters to customize the rule to your needs.

Parameter	Description	Default Value
IgnoreConstantConditions	Comma-separated list of conditions (without whitespace) that are either always true or always false.	1=1, 1!=1, 1<>1, 1=<=1, 1*=1

References

- same as [Trivadis G-1080](#)
- same as [plsqlopen:IdenticalExpression](#)
- same as [plsql:S1764](#)
- same as [SQLFluff structure.constant_expression](#)

dbLinter Rules RoadMap

Status	Trivadis	PMD	SonarQube	ZPA	SQLFluff
Included	124	4	88	24	8
To include	-	1	27	17	16
Do not include	-	16	74	13	43
Total	124	21	189	54	67

Examples of rules not considered:

- PMD: [AvoidTabCharacter](#) (code style)
- SonarQube: [Constant declarations should contain initialization assignments](#) (copying the compiler)

Examples of rules unlikely to be considered:

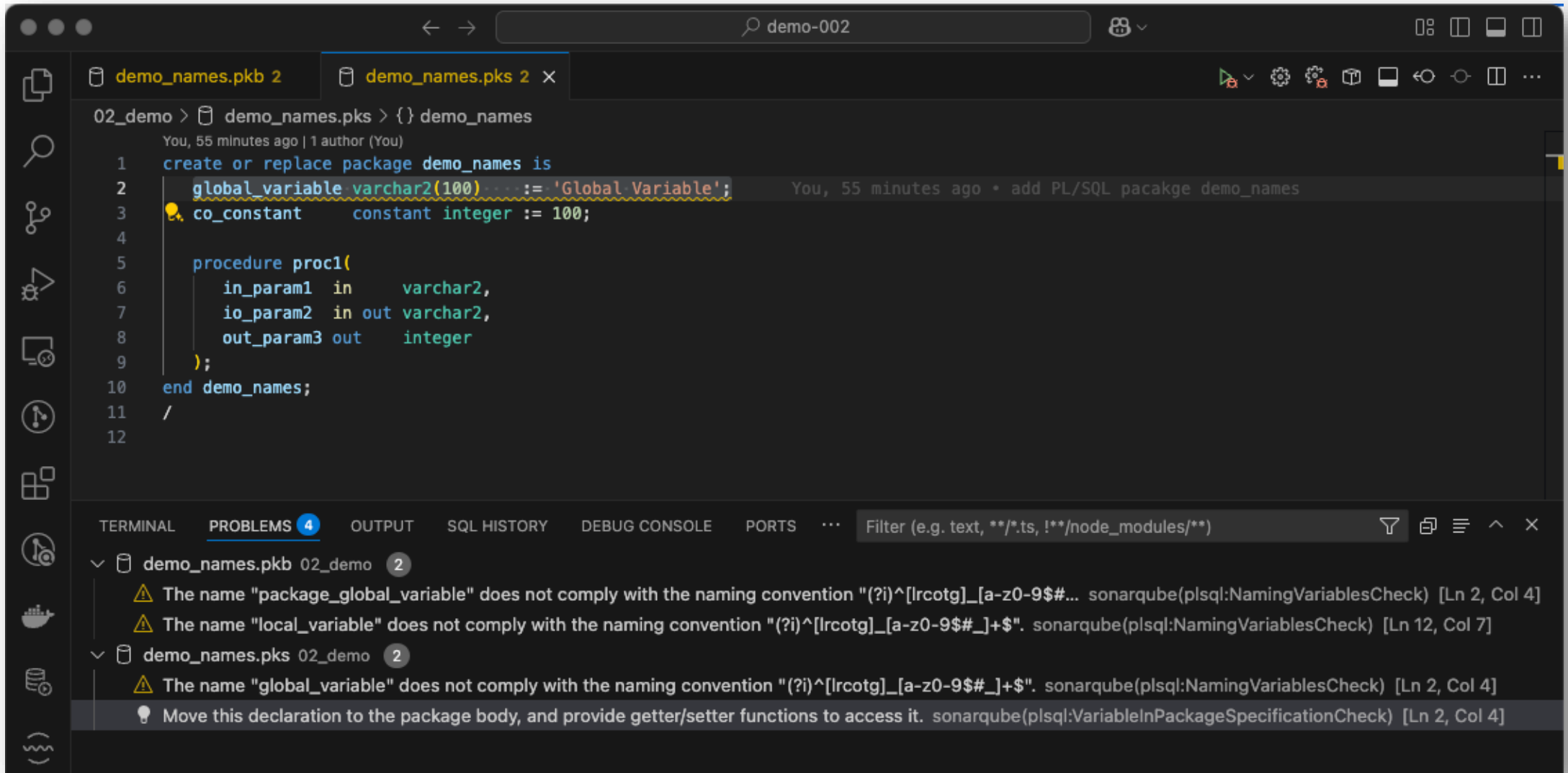
- SonarQube: [An "ORDER BY" direction should be specified explicitly](#) (ASC or DESC)
- ZPA: [Add parentheses in nested expression](#) (regardless of default precedence)
- SQLFluff: [Implicit/explicit aliasing of columns](#) (AS keyword)

Static Code Analysis Demo

PL/SQL Naming Conventions

Identifier Type	dbLinter Rule	dbLinter Regex	SonarQube Rule	SonarQube Regex
Global Variable	G-9101	(?i)^ g _[a-z0-9\$#_]+\$	plsql:NamingVariablesCheck	(?i)^[lrcot g]_[a-z0-9\$#_]+\$
Local Variable	G-9102	(?i)^ l _[a-z0-9\$#_]+\$	plsql:NamingVariablesCheck	(?i)^[lrcot g]_[a-z0-9\$#_]+\$
Cursor Variable	G-9103	(?i)^ c _[a-z0-9\$#_]+\$	plsql:NamingCursorsCheck	(?i)^ c _[a-z0-9\$#_]+\$
Record Variable	G-9104	(?i)^ r _[a-z0-9\$#_]+\$	plsql:NamingVariablesCheck	(?i)^[lrcot g]_[a-z0-9\$#_]+\$
Collection Variable	G-9105	(?i)^ t _[a-z0-9\$#_]+\$	plsql:NamingVariablesCheck	(?i)^[lrcot g]_[a-z0-9\$#_]+\$
Object Variable	G-9106	(?i)^ o _[a-z0-9\$#_]+\$	plsql:NamingVariablesCheck	(?i)^[lrcot g]_[a-z0-9\$#_]+\$
Cursor Parameter	G-9107	(?i)^ p _[a-z0-9\$#_]+\$	plsql:NamingCursorParametersCheck	(?i)^ p _[a-z0-9\$#_]+\$
IN Parameter	G-9108	(?i)^ in _[a-z0-9\$#_]+\$	plsql:NamingFunctionAndProcedureParametersCheck	(?i)^(in out io)_[a-z0-9\$#_]+\$
OUT Parameter	G-9109	(?i)^ out _[a-z0-9\$#_]+\$	plsql:NamingFunctionAndProcedureParametersCheck	(?i)^(in out io)_[a-z0-9\$#_]+\$
IN/OUT Parameter	G-9110	(?i)^ io _[a-z0-9\$#_]+\$	plsql:NamingFunctionAndProcedureParametersCheck	(?i)^(in out io)_[a-z0-9\$#_]+\$
Record Type Definition	G-9111	(?i)^ r _[a-z0-9\$#_]+_type\$	plsql:NamingTypesCheck	(?i)^([rt])?[a-z0-9\$#_]+_(ot ct type)\$
Collection Type Definition	G-9112	(?i)^ t _[a-z0-9\$#_]+_type\$	plsql:NamingTypesCheck	(?i)^([rt])?[a-z0-9\$#_]+_(ot ct type)\$
Exception	G-9113	(?i)^ e _[a-z0-9\$#_]+\$	plsql:NamingExceptionsCheck	(?i)^ e _[a-z0-9\$#_]+\$
Constant	G-9114	(?i)^ co _[a-z0-9\$#_]+\$	plsql:NamingConstantsCheck	(?i)^ co _[a-z0-9\$#_]+\$
Subtype	G-9115	(?i)^[a-z][a-z0-9\$#_]*_type\$	plsql:NamingTypesCheck	(?i)^([rt])?[a-z0-9\$#_]+_(ot ct type)\$
Record Field	G-9116	(?i)^[a-z0-9\$#_]+\$	plsql:NamingRecordField	(?i)^[a-z0-9\$#_]+\$

SonarQube for IDE (VS Code)



The screenshot shows the VS Code IDE interface with a PL/SQL package definition in the editor and associated linting errors in the Problems panel.

```
02_demo > demo_names.pks > {} demo_names
You, 55 minutes ago | 1 author (You)
1 create or replace package demo_names is
2   global_variable varchar2(100) := 'Global Variable';
3   co_constant     constant integer := 100;
4
5   procedure proc1(
6     in_param1 in      varchar2,
7     io_param2 in out  varchar2,
8     out_param3 out   integer
9   );
10 end demo_names;
11 /
12
```

The Problems panel shows the following errors:

- demo_names.pkb 02_demo** (2 errors):
 - ⚠ The name "package_global_variable" does not comply with the naming convention "(?i)^[lrcotg]_[a-z0-9\$#... sonarqube(plsql:NamingVariablesCheck) [Ln 2, Col 4]
 - ⚠ The name "local_variable" does not comply with the naming convention "(?i)^[lrcotg]_[a-z0-9\$#_]+\$". sonarqube(plsql:NamingVariablesCheck) [Ln 12, Col 7]
- demo_names.pks 02_demo** (2 errors):
 - ⚠ The name "global_variable" does not comply with the naming convention "(?i)^[lrcotg]_[a-z0-9\$#_]+\$". sonarqube(plsql:NamingVariablesCheck) [Ln 2, Col 4]
 - 💡 Move this declaration to the package body, and provide getter/setter functions to access it. sonarqube(plsql:VariableInPackageSpecificationCheck) [Ln 2, Col 4]

dbLinter for VS Code

The screenshot shows the VS Code editor with a PL/SQL package named 'demo_names.pks'. The code includes a global variable, a constant, and a procedure. The dbLinter extension has identified several errors in the 'PROBLEMS' panel:

```
02_demo > demo_names.pks > {} demo_names > global_variable varchar2(100) := 'Global Variable'
You, 60 minutes ago | 1 author (You)
1 create or replace package demo_names is
2   global_variable varchar2(100) := 'Global Variable';
3   co_constant constant integer := 100;
4
5   procedure proc1(
6     in_param1 in varchar2,
7     io_param2 in out varchar2,
8     out_param3 out integer
9   );
10 end demo_names;
11 /
12
```

PROBLEMS 7

- demo_names.pkb 02_demo 5
 - Global variable package_global_variable does not match '(?i)^g_[a-z0-9\$#_]+\$'. dbLinter Core(G-9101: Always follow naming conventions for global variables.) [Ln 2, Col 4]
 - Local variable local_variable does not match '(?i)^l_[a-z0-9\$#_]+\$'. dbLinter Core(G-9102: Always follow naming conventions for local variables.) [Ln 12, Col 7]
 - Cursor variable l_cur2 does not match '(?i)^c_[a-z0-9\$#_]+\$'. dbLinter Core(G-9103: Always follow naming conventions for cursor variables.) [Ln 16, Col 7]
 - Record variable l_dept1 does not match '(?i)^r_[a-z0-9\$#_]+\$'. dbLinter Core(G-9104: Always follow naming conventions for record variables.) [Ln 27, Col 7]
 - Collection variable l_dept2 does not match '(?i)^t_[a-z0-9\$#_]+\$'. dbLinter Core(G-9105: Always follow naming conventions for collection variables (arrays/tables).) [Ln 28, Col 7]
- demo_names.pks 02_demo 2
 - Global variable global_variable is public. dbLinter Core(G-7230: Avoid declaring global variables public.) [Ln 2, Col 4]
 - Global variable global_variable does not match '(?i)^g_[a-z0-9\$#_]+\$'. dbLinter Core(G-9101: Always follow naming conventions for global variables.) [Ln 2, Col 4]

Tools for SQL-based Tests

utPLSQL

- Compare actual and expected SQL cursors
- Perfect for hard rules without exceptions
- Not suited for warnings
- No concept to share common tests across projects
- Output in UI or in JUnit XML format from CLI
- Apache-2.0 license
- Source of Core, CLI, IDE plugins on [GitHub](#)
- See [documentation](#)



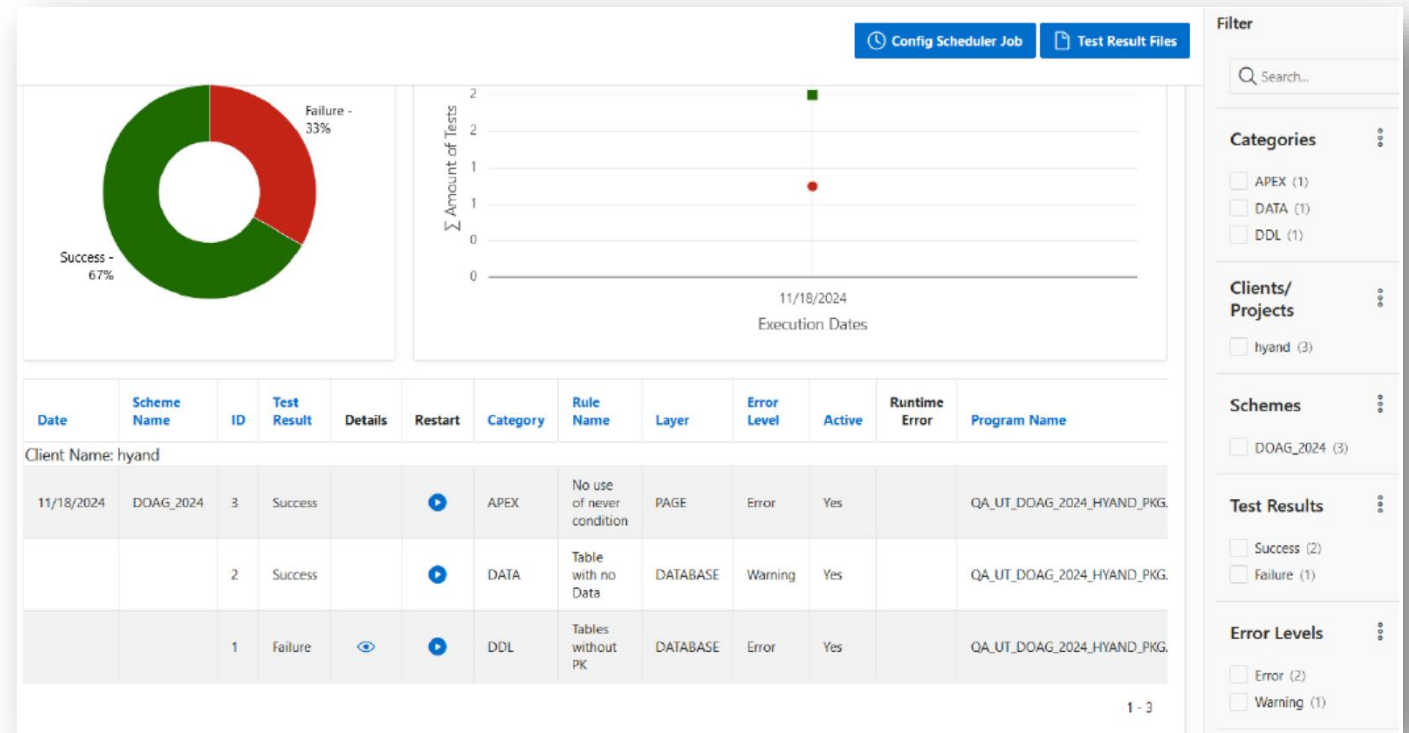
ora* CODECOP

- Implements rules with SQL
- Rules in DB
- Results in DB
- MIT license
- Fork on [GitHub](https://github.com/yerberba1704/occ)
- Contributed by Markus Schulze, Helaba Invest Kapitalanlagegesellschaft mbH
- Presented at [DOAG Conference + Exhibition 2023](#)

Rule ID ↑	Code	Title	Rule Object	Characteristic	Severity	Tags
OCC-30010	DESCRIPTION_FOR_ALL_DATA_OBJECTS	All data objects must have a description.	SQL Data Object	Maintainability	Minor	cio
OCC-40010	PARAMETER_NAMING_RULE	IN parameters must start with Prefix P.	PL/SQL Unit	Maintainability	Major	
OCC-40020	MAX_LINE_SIZE	Code should not exceed 160 characters per line.	PL/SQL Unit	Maintainability	Minor	
OCC-79010	TABLE_COMMENT	All tables must have a description.	Tables	Maintainability	Minor	
OCC-80010	READONLY_CONSTRAINTS_FOR_VIEWS	All views should have readonly constraints.	Views	Security	Major	cio,demo

QUASTO

- Implements rules with SQL
- Rules in DB
- Results in DB of multiple schemas
- utPLSQL integration
- MIT license
- Fork on [GitHub](#)
- Product Lead is Oliver Lemm, Hyand Solutions GmbH
- Presented at [DOAG Conference + Exhibition 2024](#)



dbLinter

- Implements rules with SQL or static code analysis
- Runs tests from client
- Ignore chosen findings
- Apply migration scripts (fixes)
- Output in UI or in JUnit XML format from CLI
- See rules with tests in dblinters.com

G-1240 FREE

Warning

Try to index foreign key columns.

Modeling Performance

▼ Test SQL query

```
1 with
2 function create_fk_index_stmt (
3     in_owner      in varchar2,
4     in_table_name in varchar2,
5     in_constraint_name in varchar2
6 ) return clob is
7 co_tmpl constant varchar2(200 char) := 'create index #IF_NOT_EXISTS# #OWNER#. #INDEX_NAME# on #OWNER#. #TABLE_NAME# (#COLS#)';
8 l_stmt varchar2(4000 byte) := co_tmpl;
9 l_cols varchar2(4000 byte);
10 begin
11 if dbms_db_version.version >= 23 then
12     l_stmt := replace(l_stmt, '#IF_NOT_EXISTS#', 'if not exists');
13 else
14     l_stmt := replace(l_stmt, '#IF_NOT_EXISTS#', null);
15 end if;
16 l_stmt := replace(l_stmt, '#OWNER#', in_owner);
17 l_stmt := replace(l_stmt, '#INDEX_NAME#', in_constraint_name || '_I');
18 l_stmt := replace(l_stmt, '#TABLE_NAME#', in_table_name);
19 select listagg(column_name, ', ' within group (order by position)
20     into l_cols
21     from dba_cons_columns
22     where owner = in_owner
23     and table_name = in_table_name
24     and constraint_name = in_constraint_name;
25 l_stmt := replace(l_stmt, '#COLS#', l_cols);
26 return l_stmt;
27 end create_fk_index_stmt;
28 select c.owner || '.' || c.constraint_name || '.' || cc.column_name as identifier,
29     'Column ' || cc.column_name || ' in the foreign key constraint ' || c.constraint_name
30     || ' of table ' || c.owner || '.' || c.table_name || ' is not indexed in the order of the constraint.' as message,
31     create_fk_index_stmt(c.owner, c.table_name, c.constraint_name) as migration
32 from dba_constraints c
33 join dba_cons_columns cc
34 on cc.owner = c.owner
35    and cc.constraint_name = c.constraint_name
36    and cc.table_name = c.table_name
37 left join dba_ind_columns ic
38 on ic.table_owner = c.owner
39    and ic.table_name = c.table_name
40    and ic.column_name = cc.column_name
41    and ic.column_position = coalesce(cc.position, 1)
42 where c.constraint_type = 'R'
43    and c.status = 'ENABLED'
44    and ic.column_name is null
45    and c.owner in (#SchemaNames#)
46 order by c.owner, c.constraint_name, c.table_name, cc.position, cc.column_name
```

Test results

Identifier	Message	Migration
DBL_OWNER.CHILD_PARENT_FK.PARENT_ID	Column PARENT_ID in the foreign key constraint CHILD_PARENT_FK of table DBL_OWNER.CHILD is not indexed in the order of the constraint.	create index if not exists DBL_OWNER.CHILD_PARENT_FK_I on DBL_OWNER.CHILD(PARENT_ID);

Parameters

Use parameters to customize the rule to your needs.

Parameter	Description	Default Value
SchemaNames	Comma-separated list of database schemas owning the database objects of an application.	dbl_owner

SQL-based Tests Demo

dbLinter for VS Code

The screenshot displays the VS Code interface with the dbLinter extension. The left sidebar shows a list of rules under the 'DBLINTER' section. Two rules are highlighted with a warning icon (triangle):

- Column DEPTNO in the foreign key constraint EMP_DEPTNO_FK of table DEMO_APP.EMP is not indexed in the order of the constraint.
- Column MGR in the foreign key constraint EMP_MGR_FK of table DEMO_APP.EMP is not indexed in the order of the constraint.

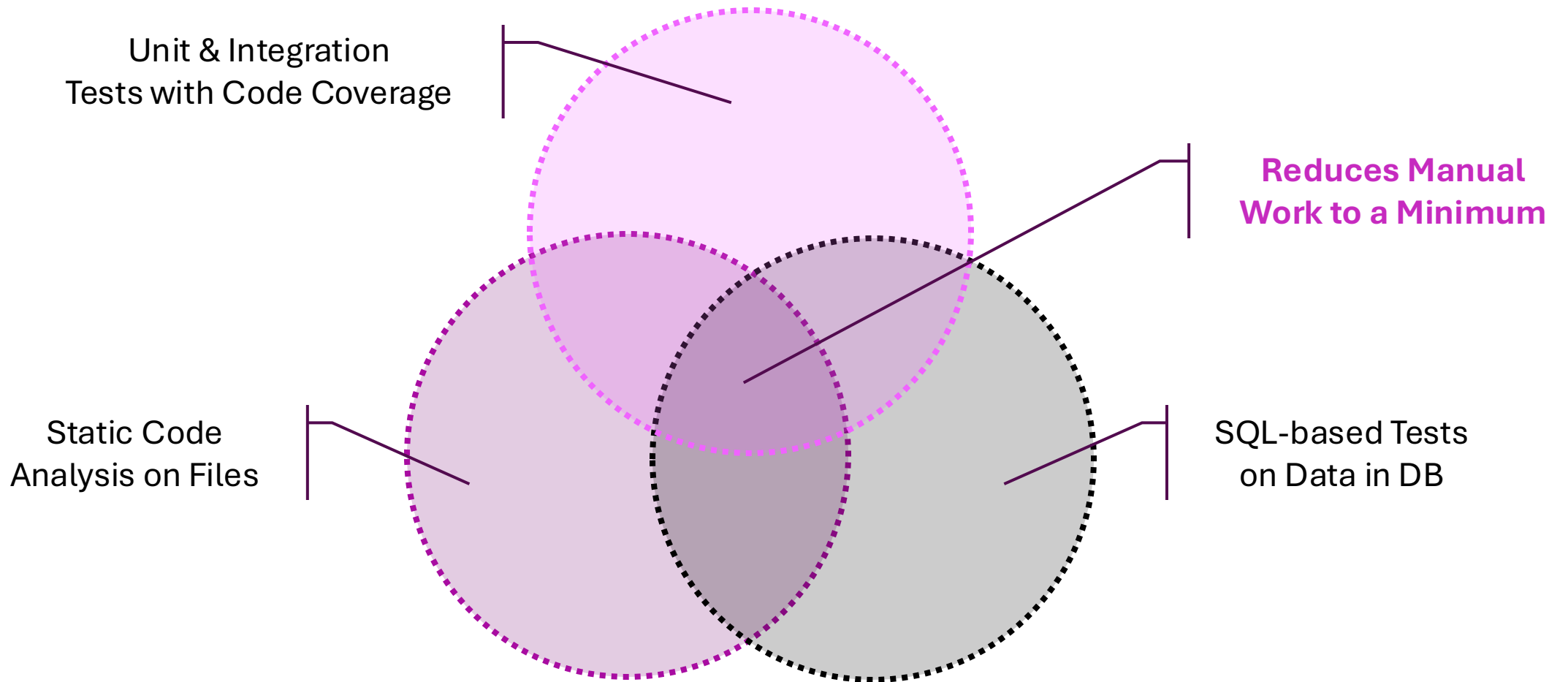
A context menu is open over the second rule, with the 'Show Migration Script' option selected. The main editor shows the generated SQL script:

```
-- Script generated by dbLinter, please review it before applying it.
1
2
3 create index if not exists DEMO_APP.EMP_DEPTNO_FK_I on DEMO_APP.EMP(DEPTNO);
4
5 create index if not exists DEMO_APP.EMP_MGR_FK_I on DEMO_APP.EMP(MGR);
6
```

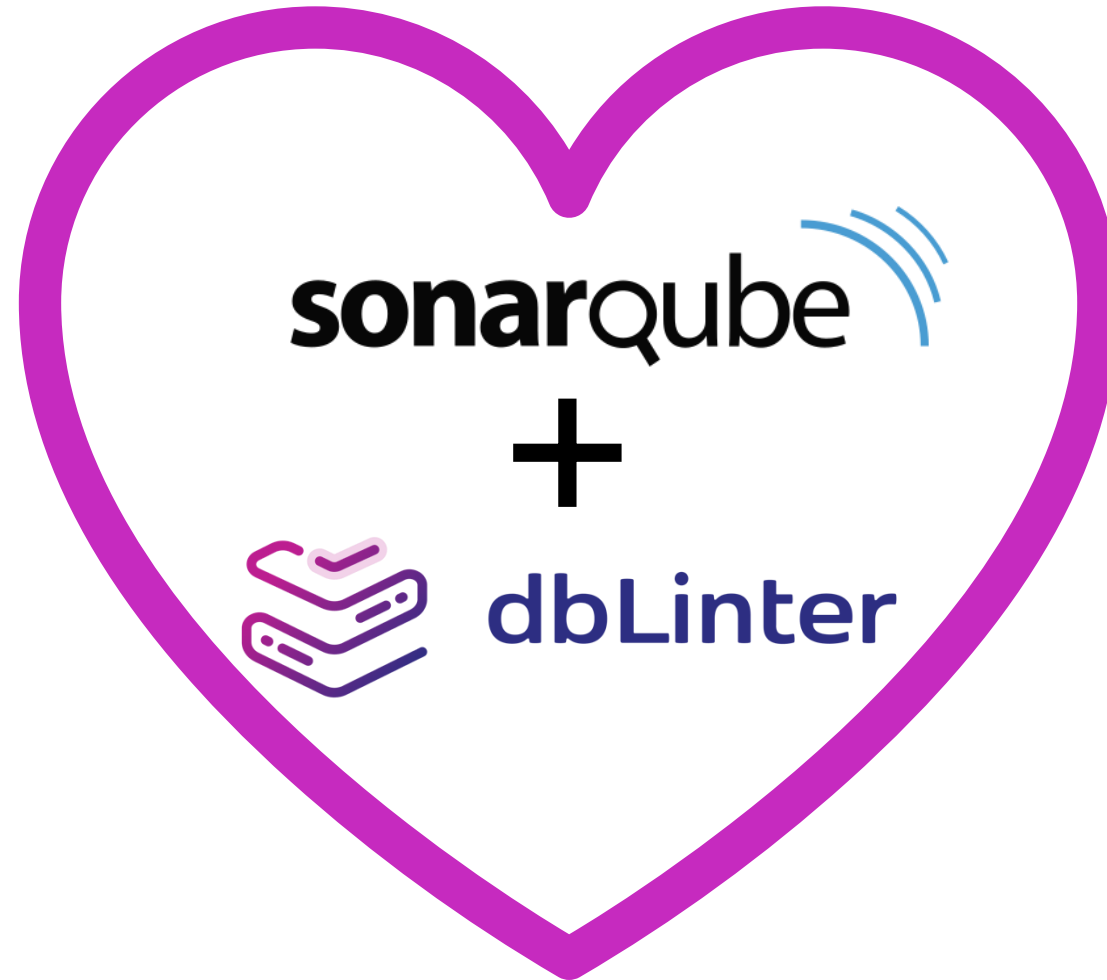
The bottom status bar indicates 'Ln 6, Col 1 Spaces: 3 UTF-8 {} PL/SQL' and a red error message: 'No connection attached'. The 'PROBLEMS' panel at the bottom shows 'No problems have been detected in the workspace.'

Core Messages

Quality Assessment of Code in the DB



Good Fit in DevOps Lifecycle



IDE Integration for PL/SQL & SQL

SonarQube for IDE

- ✗ Undefined grammar scope
- 😞 Own set of rules
- 😞 Disable check as quick fix
- ✗ No SQL-based Tests
- 😞 Custom rules with Xpath
- ✗ No custom quick fixes
- ✅ Mature solution

dbLint for VS Code

- ✅ Recent grammar versions
- ✅ Rules based on Trivadis Guidelines
- ✅ Quick fixes
- ✅ SQL-based Tests with migrations
- ✅ Custom rules with SQL and Java
- ✅ Custom quick fixes
- 😞 Preview, GA in 2025

Thank you!

