












PL/SQL

PL/SQL static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your PL/SQL code

- All rules** 188 Vulnerability 4 Bug 45 Security Hotspot 2 Code Smell 137

Tags Search by name...

"GOTO" should not be used within loops
 Code Smell
"FULL OUTER JOINS" should be used with caution
 Code Smell
"CASE" should be used rather than "DECODE"
 Code Smell
"CROSS JOIN" queries should not be used
 Code Smell
"END" statements of labeled blocks should be labeled
 Code Smell
Two branches in a conditional structure should not have exactly the same implementation
 Code Smell
Unused assignments should be removed
 Code Smell
"LIKE" clauses should not start with wildcard characters
 Code Smell
Column names should be used in a SQL "ORDER BY" clause
 Code Smell
Procedures and functions should be documented
 Code Smell
SQL statements should not join too many tables
 Code Smell
Function and procedure names should comply with a naming convention

"GOTO" should not be used within loops

Analyze your code

- Code Smell
 Major

 pitfall

The use of `GOTO` in general is arguable. However, when used within loops, `GOTO` statements are even more evil, and they can often be replaced by other constructs.

Noncompliant Code Example

```
DECLARE
    i PLS_INTEGER := 0;
BEGIN
    LOOP
        IF i = 3 THEN
            GOTO loopEnd; -- Noncompliant
        END IF;

        DBMS_OUTPUT.PUT_LINE('i = ' || i);

        i := i + 1;
    END LOOP;

    <<loopEnd>>
    DBMS_OUTPUT.PUT_LINE('Loop end');
END;
/
```

Compliant Solution

```
DECLARE
    i PLS_INTEGER := 0;
BEGIN
    LOOP
        EXIT WHEN i = 3; -- Compliant

        DBMS_OUTPUT.PUT_LINE('i = ' || i);

        i := i + 1;
    END LOOP;

    DBMS_OUTPUT.PUT_LINE('Loop end');
END;
/
```

Available In:

