Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

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...

Redundant pairs of parentheses should be removed
Code Smell

Functions and procedures should not have too many parameters
Code Smell

Collapsible "if" statements should be merged
Code Smell

Unused labels should be removed
Code Smell

Compound triggers should define at least two triggers
Code Smell

"EXIT" should not be used in loops
Code Smell

Jump statements should not be redundant
Code Smell

"EXCEPTION WHEN ... THEN" clauses should do more than "RAISE"
Code Smell

Single line comments should start with "--"
Code Smell

An "ORDER BY" direction should be specified explicitly
Code Smell

Oracle's join operator (+) should not be used
Code Smell

"cursor%NOTFOUND" should be used instead of "NOT cursor%FOUND"

Redundant pairs of parentheses should be removed

Analyze your code

Code Smell | Major | confusing

The use of parentheses, even those not required to enforce a desired order of operations, can clarify the intent behind a piece of code. But redundant pairs of parentheses could be misleading, and should be removed.

Noncompliant Code Example

```
x := (y / 2 + 1); -- Compliant even if the parentheses are ignored
IF (x > 0) AND ((x+y > 0)) THEN -- Noncompliant
  -- ...
END IF;
```

Compliant Solution

```
x := (y / 2 + 1);
IF (x > 0) AND (x+y > 0) THEN
  -- ...
END IF;
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition