





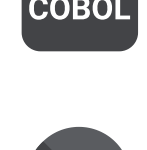











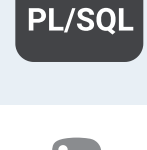


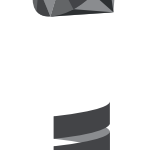













-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML


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... 

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"	 Code Smell
Object attributes should comply with a naming convention	 Code Smell
Record fields should comply with a naming convention	 Code Smell
Cursors should follow a naming convention	 Code Smell
Types should follow a naming convention	

Compound triggers should define at least two triggers

Analyze your code

 Code Smell

 Major 

 clumsy

Compound triggers were introduced to ease the implementation of multiple triggers which need to work in cooperation.

Typically, a `FOR EACH ROW` trigger accumulates facts, and an `AFTER STATEMENT` trigger performs the actual changes.

The compound trigger can hold a state common to all the triggers it defines, thereby removing the need to use package variables. This approach is sometimes the only possible one, as when avoiding a mutating table `ORA-04091` error, or it can be used to get better performance.

However, there is no point in defining a compound trigger which contains only a single trigger, since there is no state to be shared. In such cases, a simple trigger should be used instead.

Noncompliant Code Example

```
CREATE OR REPLACE TRIGGER my_trigger  -- Noncompliant; defines a single trigger
FOR INSERT ON my_table
COMPOUND TRIGGER

AFTER EACH ROW IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('New row inserted!');
END AFTER EACH ROW;

END;
/
```

Compliant Solution

```
CREATE OR REPLACE TRIGGER my_trigger

AFTER INSERT
ON my_table
FOR EACH ROW
BEGIN
    DBMS_OUTPUT.PUT_LINE('New row inserted!');
END;
/
```

Available In:

sonarlint 

sonarcloud 

sonarqube  Developer Edition