

































-  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...



"COMMIT" should not be used inside a loop

 Bug

Individual "WHERE" clause conditions should not be unconditionally true or false

 Bug

Nullable subqueries should not be used in "NOT IN" conditions

 Bug

"COMMIT" and "ROLLBACK" should not be called from non-autonomous transaction triggers

 Bug

Positional and named arguments should not be mixed in invocations

 Bug

Functions should end with "RETURN" statements

 Bug

Collections should not be iterated in "FOR" loops

 Bug

Using weak hashing algorithms is security-sensitive

 Security Hotspot

Dynamically executing code is security-sensitive

 Security Hotspot

SQL "JOIN" conditions should involve all joined tables

 Code Smell

"SELECT" statements used as argument of "EXISTS" statements should be selective

 Code Smell

"LIKE" clauses should not be used without wildcards

 Code Smell

Weak "REF CURSOR" types should not be used

 Code Smell

"COMMIT" should not be used inside a loop

Analyze your code

 Bug  Critical 

Frequent commits are widely understood to negatively impact performance. Thus, committing inside a loop (even when only executed conditionally once every n iterations) is highly likely to cause unwanted performance impacts.

Further, in general use `COMMIT` should only be used at the end of a transaction. Code that is not structured to have one transaction per loop iteration could yield unexpected results if `COMMIT` is nonetheless used inside the loop. Code that is structured to have one transaction per loop iteration should probably be reconsidered.

Note that when dealing with very large data sets, a `COMMIT` may be required every n iterations, but the goal should be to avoid `COMMIT`s inside loops.

Noncompliant Code Example

```
FOR item IN itemlist
LOOP
  -- ...
  COMMIT;  -- Noncompliant
END LOOP;
```

Compliant Solution

```
FOR item IN itemlist
LOOP
  -- ...
END LOOP;
COMMIT;
```

Available In:

sonarlint

| **sonarcloud**

| **sonarqube**

Developer Edition