















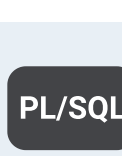

















-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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











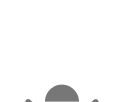


 Bug 45

 Security Hotspot 2

 Code Smell 137

Tags ▾

Search by name... 🔍

EXCEPTIONS"" clause	 Bug
Pipelined functions should have at least one "PIPE ROW" statement and not return an expression (PLS-00633)	 Bug
GOTO should not be used to jump backwards	 Code Smell
Quoted identifiers should not be used	 Code Smell
Loop start and end labels should match	 Code Smell
Block start and end labels should match	 Code Smell
Cipher algorithms should be robust	 Vulnerability
"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	

"FORALL" statements should use the "SAVE EXCEPTIONS" clause

Analyze your code

 Bug

 Blocker



When the FORALL statement is used without the SAVE EXCEPTIONS clause and an exception is raised by a DML query, the whole operation is rolled back and the exception goes unhandled. Instead of relying on this default behavior, it is better to always use the SAVE EXCEPTIONS clause and explicitly handle exceptions in a ORA-24381 handler.

Noncompliant Code Example

```
CREATE TABLE my_table(
  id NUMBER(10) NOT NULL
);

DECLARE
  TYPE my_table_id_type IS TABLE OF my_table.id%TYPE;
  my_table_ids my_table_id_type := my_table_id_type();
BEGIN
  FOR i IN 1 .. 10 LOOP
    my_table_ids.EXTEND;
    my_table_ids(my_table_ids.LAST) := i;
  END LOOP;

  -- Cause the failure
  my_table_ids(10) := NULL;

  FORALL i IN my_table_ids.FIRST .. my_table_ids.LAST -- Noncompliant
    INSERT INTO my_table
      VALUES (my_table_ids(i));
END;
/

SELECT COUNT(*) FROM my_table;

DROP TABLE my_table;
```

Compliant Solution

```
-- ...

DECLARE
  TYPE my_table_id_type IS TABLE OF my_table.id%TYPE;
  my_table_ids my_table_id_type := my_table_id_type();

  bulk_errors EXCEPTION;
  PRAGMA EXCEPTION_INIT(bulk_errors, -24381);
BEGIN
  FOR i IN 1 .. 10 LOOP
    my_table_ids.EXTEND;
    my_table_ids(my_table_ids.LAST) := i;
  END LOOP;

  -- Cause the failure
  my_table_ids(10) := NULL;

  FORALL i IN my_table_ids.FIRST .. my_table_ids.LAST SAVE EXCEPTIONS
    INSERT INTO my_table
      VALUES (my_table_ids(i));
EXCEPTION
  WHEN bulk_errors THEN
    -- Explicitly rollback the whole transaction,
    -- or handle each exception individually by looping over SQL%BULK_EXCEPTIONS
    ROLLBACK;
END;
/

-- ...
```

Available In:

sonarlint

|

sonarcloud

|

sonarqube

Developer Edition