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...

Code Smell

"FORALL" should be used
Code Smell

"FETCH ... BULK COLLECT INTO" should be used
Code Smell

Column aliases should be defined using "AS"
Code Smell

Procedures and functions should be encapsulated in packages
Code Smell

Procedures should have parameters
Code Smell

"EXECUTE IMMEDIATE" should be used instead of DBMS_SQL procedure calls
Code Smell

"NATURAL JOIN" queries should not be used
Code Smell

"END" statements of labeled loops should be labeled
Code Smell

In labeled loops "EXIT" should exit the label
Code Smell

"EXIT WHEN" should be used rather than "IF ... THEN EXIT; END IF;"
Code Smell

"FOR" loop end conditions should not be hard-coded
Code Smell

"FETCH ... BULK COLLECT INTO" should be used

Analyze your code

Code Smell | Major | performance

The `FETCH ... INTO` statement is inefficient when used in a loop (where many records are expected). It leads to many context-switches between the SQL and PL/SQL engines. Instead, the `FETCH ... BULK COLLECT INTO` statement will issue the SQL requests in bulk, minimizing context switches.

Noncompliant Code Example

```
SET SERVEROUTPUT ON

CREATE TABLE largeTable AS SELECT ROWNUM AS id FROM all_objects;

SET TIMING ON
DECLARE
  x PLS_INTEGER;
  CURSOR largeCursor IS SELECT ROWNUM FROM largeTable;
  largeTableRowId BINARY_INTEGER;
BEGIN
  OPEN largeCursor;

  x := 0;
  LOOP
    FETCH largeCursor INTO largeTableRowId; -- Noncompliant
    EXIT WHEN largeCursor%NOTFOUND;

    x := x + largeTableRowId;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Sum of rownums using alternative 1: ' || x);

  CLOSE largeCursor;
END;
/
SET TIMING OFF

DECLARE
  r largeTable%ROWTYPE;
  CURSOR myCursor IS SELECT * FROM largeTable;
BEGIN
  OPEN myCursor;
  FETCH myCursor INTO r; -- Compliant, outside of a loop
  CLOSE myCursor;
END;
/

DROP TABLE largeTable;
```

Compliant Solution

```
SET SERVEROUTPUT ON

CREATE TABLE largeTable AS SELECT ROWNUM AS id FROM all_objects;

SET TIMING ON
DECLARE
  x PLS_INTEGER;
  CURSOR largeCursor IS SELECT * FROM largeTable;
  TYPE largeTableRowIdArrayType IS TABLE OF BINARY_INTEGER INDEX BY BINARY_INTEGER;
  largeTableRowIdArray largeTableRowIdArrayType;
BEGIN
  OPEN largeCursor;

  x := 0;
  LOOP
    FETCH largeCursor BULK COLLECT INTO largeTableRowIdArray LIMIT 1000; -- Compliant

    FOR i IN largeTableRowIdArray.FIRST .. largeTableRowIdArray.LAST LOOP
      x := x + largeTableRowIdArray(i);
    END LOOP;

    EXIT WHEN largeCursor%NOTFOUND;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Sum of rownums using alternative 2: ' || x);

  CLOSE largeCursor;
END;
/
SET TIMING OFF

DECLARE
  r largeTable%ROWTYPE;
  CURSOR myCursor IS SELECT * FROM largeTable;
BEGIN
  OPEN myCursor;
  FETCH myCursor INTO r; -- Compliant, outside of a loop
  CLOSE myCursor;
END;
/

DROP TABLE largeTable;
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition