


















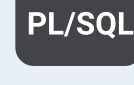














-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  Flex
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  **PL/SQL**
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML

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... 

"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

Large item lists should not be used with "IN" clauses

"FORALL" should be used

Analyze your code

 Code Smell  Major   performance

The performance of DML queries in loops can be improved by placing them in a `FORALL` statement. This way, queries will be sent in bulk, minimizing the number of context switches between PL/SQL and SQL.

### Noncompliant Code Example

```
SET SERVEROUTPUT ON

CREATE TABLE largeTable(
    foo VARCHAR2(42)
);

BEGIN
    FOR i IN 1 .. 100000 LOOP
        INSERT INTO largeTable VALUES('bar' || i); -- Non-compliant
    END LOOP;
END;
/

SET TIMING ON
DECLARE
    TYPE largeTableRowArrayType IS TABLE OF largeTable%ROWTYPE;
    largeTableRowArray largeTableRowArrayType;
BEGIN
    SELECT * BULK COLLECT INTO largeTableRowArray FROM largeTable;

    EXECUTE IMMEDIATE 'TRUNCATE TABLE largeTable';
    FOR i IN largeTableRowArray.FIRST .. largeTableRowArray.LAST LOOP
        INSERT INTO largeTable (foo) VALUES (largeTableRowArray(i).foo); -- Non-compliant
    END LOOP;
END;
/

SET TIMING OFF

DROP TABLE largeTable;
```

### Compliant Solution

```
SET SERVEROUTPUT ON

CREATE TABLE largeTable(
    foo VARCHAR2(42)
);

BEGIN
    FOR i IN 1 .. 100000 LOOP
        INSERT INTO largeTable VALUES('bar' || i); -- Non-compliant
    END LOOP;
END;
/

SET TIMING ON
DECLARE
    TYPE largeTableRowArrayType IS TABLE OF largeTable%ROWTYPE;
    largeTableRowArray largeTableRowArrayType;
BEGIN
    SELECT * BULK COLLECT INTO largeTableRowArray FROM largeTable;

    EXECUTE IMMEDIATE 'TRUNCATE TABLE largeTable';
    FORALL i IN largeTableRowArray.FIRST .. largeTableRowArray.LAST
        INSERT INTO largeTable (foo) VALUES (largeTableRowArray(i).foo); -- Compliant

    INSERT INTO largeTable (foo) VALUES ('baz'); -- Compliant, not in a loop
END;
/

SET TIMING OFF

DROP TABLE largeTable;
```

Available In:

**sonarlint**  | **sonarcloud**  | **sonarqube**  Developer Edition