


































-  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... 

"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

 Code Smell

"GOTO" should not be used within loops

 Code Smell

"FULL OUTER JOINS" should be used with caution

 Code Smell

"CASE" should be used rather than "DECODE"

 Code Smell

"CROSS JOIN" queries should not be used

 Code Smell

"END" statements of labeled blocks should be labeled

## "EXECUTE IMMEDIATE" should be used instead of DBMS\_SQL procedure calls

Analyze your code

 Code Smell  Major  clumsy

EXECUTE IMMEDIATE is easier to use and understand than the DBMS\_SQL package's procedures. It should therefore be preferred, when possible.

### Noncompliant Code Example

```
SET SERVEROUTPUT ON

CREATE TABLE myTable(
    foo VARCHAR2(42)
);

CREATE PROCEDURE drop_table(tableName VARCHAR2) AS
    cursorIdentifier INTEGER;
BEGIN
    cursorIdentifier := DBMS_SQL.OPEN_CURSOR; -- Compliant; this is not a procedure call
    DBMS_SQL.PARSE(cursorIdentifier, 'DROP TABLE ' || tableName, DBMS_SQL.NATIVE); -- Noncompliant
    DBMS_SQL.CLOSE_CURSOR(cursorIdentifier); -- Noncompliant

    DBMS_OUTPUT.PUT_LINE('Table ' || tableName || ' dropped.');
```

```
EXCEPTION
    WHEN OTHERS THEN
        DBMS_SQL.CLOSE_CURSOR(cursorIdentifier); -- Noncompliant
END;
/

BEGIN
    drop_table('myTable');
END;
/

DROP PROCEDURE drop_table;
```

### Compliant Solution

```
SET SERVEROUTPUT ON

CREATE TABLE myTable(
    foo VARCHAR2(42)
);

CREATE PROCEDURE drop_table(tableName VARCHAR2) AS
    cursorIdentifier INTEGER;
BEGIN
    EXECUTE IMMEDIATE 'DROP TABLE ' || tableName;
    DBMS_OUTPUT.PUT_LINE('Table ' || tableName || ' dropped.');
```

```
END;
/

BEGIN
    drop_table('myTable');
END;
/

DROP PROCEDURE drop_table;
```

Available In:

 |  |  Developer Edition