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

"%TYPE" and "%ROWTYPE" should not be used in package specification
Code Smell

Functions and procedures should not be too complex
Code Smell

"IF ... ELSEIF" constructs should end with "ELSE" clauses
Code Smell

"WHEN OTHERS" clauses should be used for exception handling
Code Smell

"CREATE_TIMER" should not be used
Code Smell

"TO_DATE" and "TO_TIMESTAMP" should be used with a datetime format model
Code Smell

Features deprecated in Oracle 12 should not be used
Code Smell

Tables should be aliased
Code Smell

"SIMPLE_INTEGER" should be used instead of "PLS_INTEGER"
Code Smell

Queries should not "SELECT" too many columns
Code Smell

"ROWID" and "UROWID" data types should not be used

"%TYPE" and "%ROWTYPE" should not be used in package specification

Analyze your code

Code Smell | Critical

`%TYPE` and `%ROWTYPE` seem like an easy way to make sure that a variable's type always matches the type of the relevant column in an existing table. If the column type changes, then the variable changes with it.

However, Oracle Forms compiled against a procedure using either of these two symbols won't get the benefit of that flexibility. Instead, at compile time, the relevant type is looked up from the underlying database and used in the form. If the column type changes later or the form is running against a database with different length semantics, attempting to use the form results in an "ORA-04062: Signature of package has been changed" error on the package in question. And the form needs to be recompiled on exactly the same database environment where it will run to avoid the error.

Note that `%TYPE` and `%ROWTYPE` can be used in a package's private procedures and functions, private package variables, and local variables without issue, but not in the package specification.

Noncompliant Code Example

```
CREATE OR REPLACE PACKAGE PACK IS
  TYPE mytype IS RECORD (
    var1 mytable.mycolumn%TYPE -- Noncompliant
  );

  FUNCTION MY_FUNC(param1 IN mytable.mycolumn%TYPE) RETURN VARCHAR2; -- Noncompliant

  FUNCTION MY_FUNC2(param1 IN mytable%ROWTYPE) RETURN VARCHAR2; -- Noncompliant

END;
```

Compliant Solution

```
CREATE OR REPLACE PACKAGE PACK IS
  TYPE mytype IS RECORD (
    var1 VARCHAR2(100) -- Compliant
  );

  FUNCTION MY_FUNC(param1 IN VARCHAR2) RETURN VARCHAR2; -- Compliant

  TYPE myrowtype IS RECORD (
    col1 NUMBER,
    col2 VARCHAR2(30)
  );
  FUNCTION MY_FUNC2(param1 IN myrowtype) RETURN VARCHAR2; -- Compliant

END;
```

Available In:

sonarlint | sonarcloud | sonarqube Developer Edition