APEX & db* CODECOP – CODEANALYSE LEICHT GEMACHT!

Michael Schmid & Philipp Salvisberg
13. Oktober 2021

trivadis
Part of Accenture

MICHAEL SCHMID

PRINCIPAL CONSULTANT

- APEX, SQL- und PL/SQL
- Datenbankdesign und Webtechnologien
- Referent für APEX Kurse
- Autor der Open Source Lösung
  JR PrintServer für APEX und PL/SQL

HOI ZÄME

PHILIPP SALVISBERG

SENIOR PRINCIPAL CONSULTANT

- Datenbanknahe Entwicklung
- Model Driven Software Development
- Autor kostenloser SQL Developer Extensions PL/SQL Unwrapper, db* CODECOP, utPLSQL, plscope-utils, oddgen and Bitemp Remodeler

APEX, SQL UND PL/SQL

trivadis
Part of Accenture

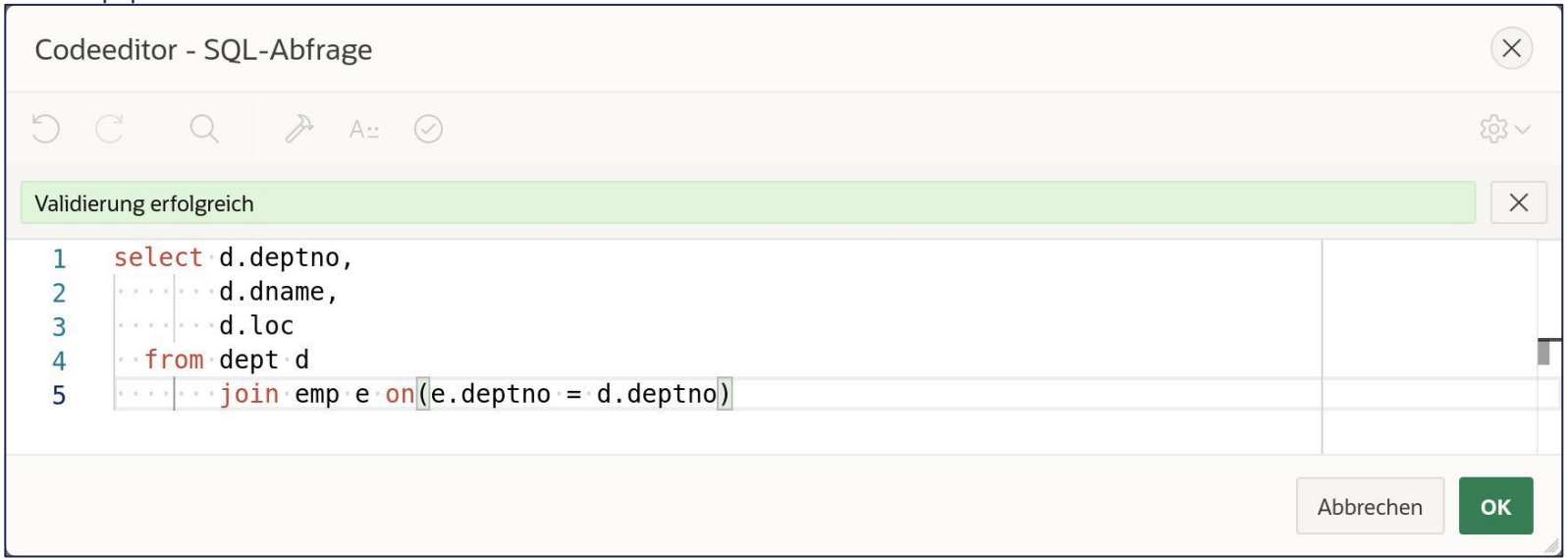
5 APEX

- SQL und PL/SQL bilden den (daten-technischen) Kern einer APEX-Applikation:

- Die meisten APEX-Komponenten basieren auf SQL- bzw. PL/SQL-Code.

6 CODEEDITOR

- APEX bietet inzwischen die Möglichkeit SQL- und PL/SQL-Code komfortabel im App Builder zu editieren:



- Undo, Redo, Suche, Syntaxhervorhebung, Codevervollständigung, Validierung: Alle Features eines guten Editors sind vorhanden.

7 CODE IN APEX – VORTEILE

- Wenn wir unseren SQL und PL/SQL-Code in der APEX-Applikation ablegen bieten sich uns folgende Vorteile:
  - Einfach: alles in einem Platz
  - Komfortabel: leistungsfähige Editoren
  - Übersichtlich: nur die Applikation muss eingespielt werden.

- Also alles klar…? Nicht ganz…
- Ein Blick auf die Alternative ist sehr lohnenswert.

SQL IN DER DATENBANK

- Die Alternative: Auslagern der SQL- und PL/SQL-Komplexität in das Datenbank-Schema
- Komplexes SQL mit Joins, Standard-Filtern, Aggregierungen kann meist in Views ausgelagert werden:

Bericht

```
select d.deptno, d.dname, ...
  from dept d
       join emp e
       on(e.deptno = d.deptno)
       join salgrade s ...
       ...
       ...
       ...
 where d.deptno = :P10_DEPTNO
```

View

```
create view dept_emp_v as
select d.deptno, d.dname, ...
  from dept d
   ...
```

Bericht

```
select d.deptno, d.dname, ...
  from dept_emp_v
 where d.deptno = :P10_DEPTNO
```

PL/SQL IN DER DATENBANK

- PL/SQL-Code lässt nahezu immer sehr gut in Prozeduren und Funktionen (in Packages) auslagern:

Prozess

```
declare
  l_comm emp.comm%type;
  ...
begin
  select comm into l_comm
    from ...
   where x = :P20_ID;
  if l_comm > 4000 then
    ...
  else
    case when ...
    end case;
  end if;
  :P20_COMM := l_comm;
end;
```

Package

```
create package emp_maintenance as
  function get_comm(p_id number) return number;
  ...
end;
create package body emp_maintrenance as
  function get_comm(p_id number) return number
  as
  begin
  ...
  end;
end;
```

Prozess

```
:P20_COMM :=
  emp_maintenance.get_comm(:P20_ID);
```

10 CODE IN DER DATENBANK – VORTEILE (1)

- OK, das Auslagern von SQL- und PL/SQL-Komplexität in das Datenbank-Schema ist möglich. Welche Vorteile bringt es?
- **Code-Verwaltung:**
  Es liegen richtige, „klassische“ SQL- bzw. PL/SQL-Quellcode-Dateien vor, die sich in *datei-orientierte* Versionsverwaltungssysteme einchecken, committen, diffen, mergen etc. lassen.
- **Code-Wiederverwendung:**
  Der PL/SQL-Code in der DB kann nicht nur von APEX, sondern auch von anderen Aufrufern genutzt werden, wie z.B. REST, Java etc.
- **DevOps:**
  Der SQL- bzw. PL/SQL-Code kann unabhängig von der APEX-Applikation verändert/eingespielt werden (solange das Interface gleichbleibt).

es geht weiter...

CODE IN DER DATENBANK – VORTEILE (2)

- **Performance:**
  - Der SQL- und PL/SQL-Code liegt in der Applikation nur als Zeichenkette vor, der bei jedem Aufruf geparst werden muss.
  - Bei der Auslagerung in das Schema, muss deutlich einfacherer SQL- bzw. PL/SQL-Code (im Ideal-Fall nur ein Call) geparst (und immer wieder geparst) werden.
  - Rechenintensiver PL/SQL-Code könnte zudem ggf. noch nativ-kompiliert werden.
- **Performance-Tuning:**
  Die üblichen Tools (z.B. DBMS_PROFILER) können einfach genutzt werden.
- **Transparenz:**
  Das Dependency-Tracking der Datenbank (user_dependencies) kann genutzt werden, um Fragen zu klären wie: „In welchem Code wird die Tabelle XYZ verwendet?“

es geht weiter...

12 CODE IN DER DATENBANK – VORTEILE (3)

- **Organisation:**
  Nach Definition der Interfaces, können PL/SQL- und APEX-Entwickler *parallel* und weitgehend unabhängig voneinander arbeiten.
- **Coding:**
  Alternative Editoren wie der Oracle SQL Developer, die noch leistungsfähiger und komfortabler sind als die APEX-Editoren, können verwendet werden.
- **Qualität:**
  - Das Wichtigste zuerst:
    Datenbank-Code wird im Falle eines Problem zur *Compile-Zeit* ungültig. Bei Code innerhalb der APEX-Applikation „kracht“ es erst zur *Laufzeit*!
  - Testen der PL/SQL-Logik mit automatisierten Tests wird möglich.

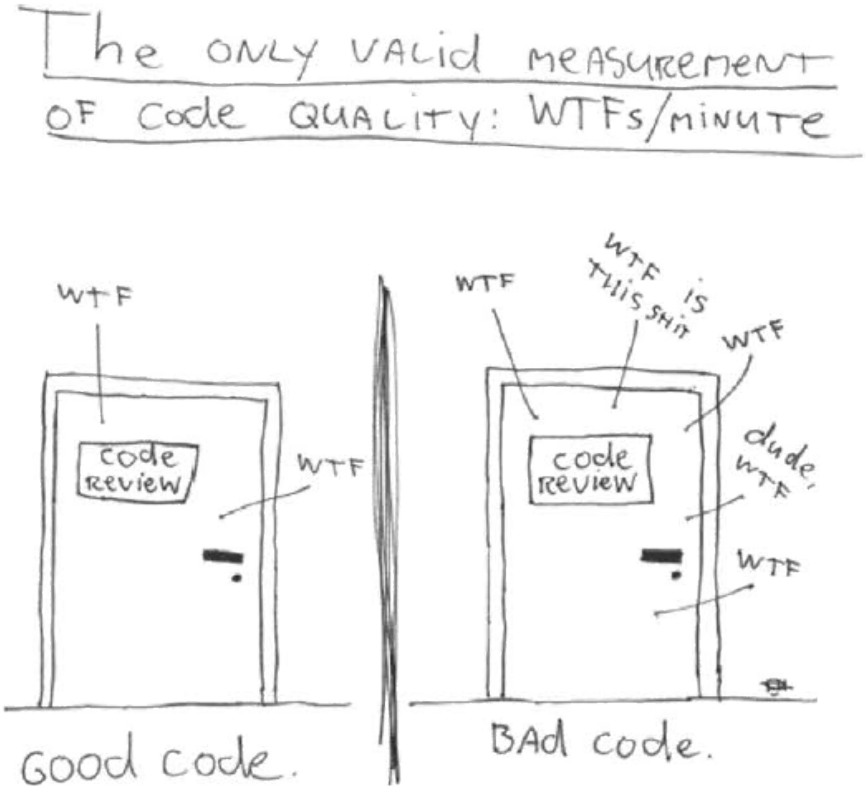
es geht weiter...

CODE IN DER DATENBANK – VORTEILE (4)

- Statische Codeanalyse → Qualität, Performance, Wartbarkeit:
  Durch das Vorliegen des SQL- bzw. PL/SQL als „purer“ Code ist eine statische Codeanalyse mit dem **Trivadis db* CODECOP** möglich!

SOFTWARE QUALITÄT

CODE REVIEWS

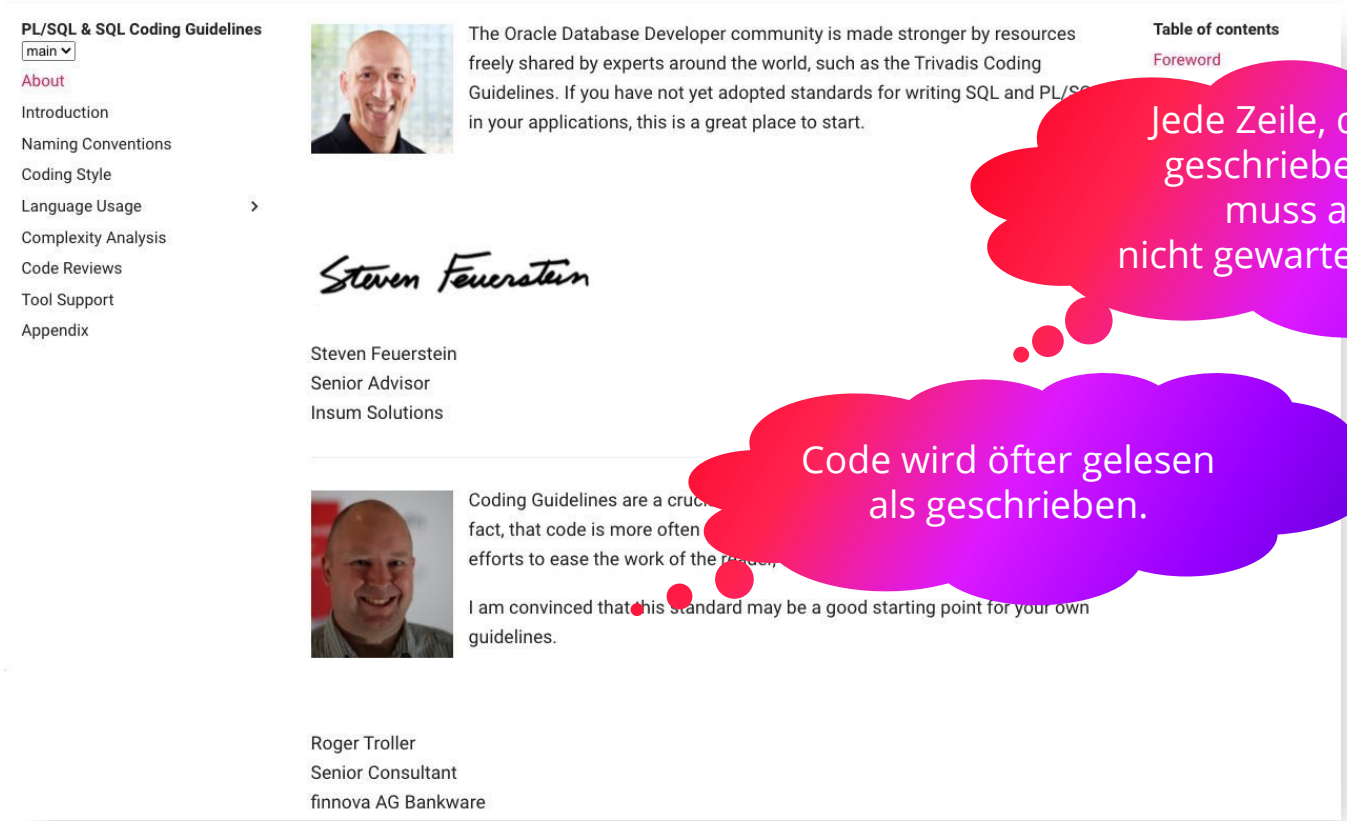


Quelle: http://www.osnews.com/story/19266/WTFs_m

16 TIPPS

Quelle: https://www.youtube.com/watch?v=IhE4pTprQ4E

PL/SQL & SQL CODING GUIDELINES

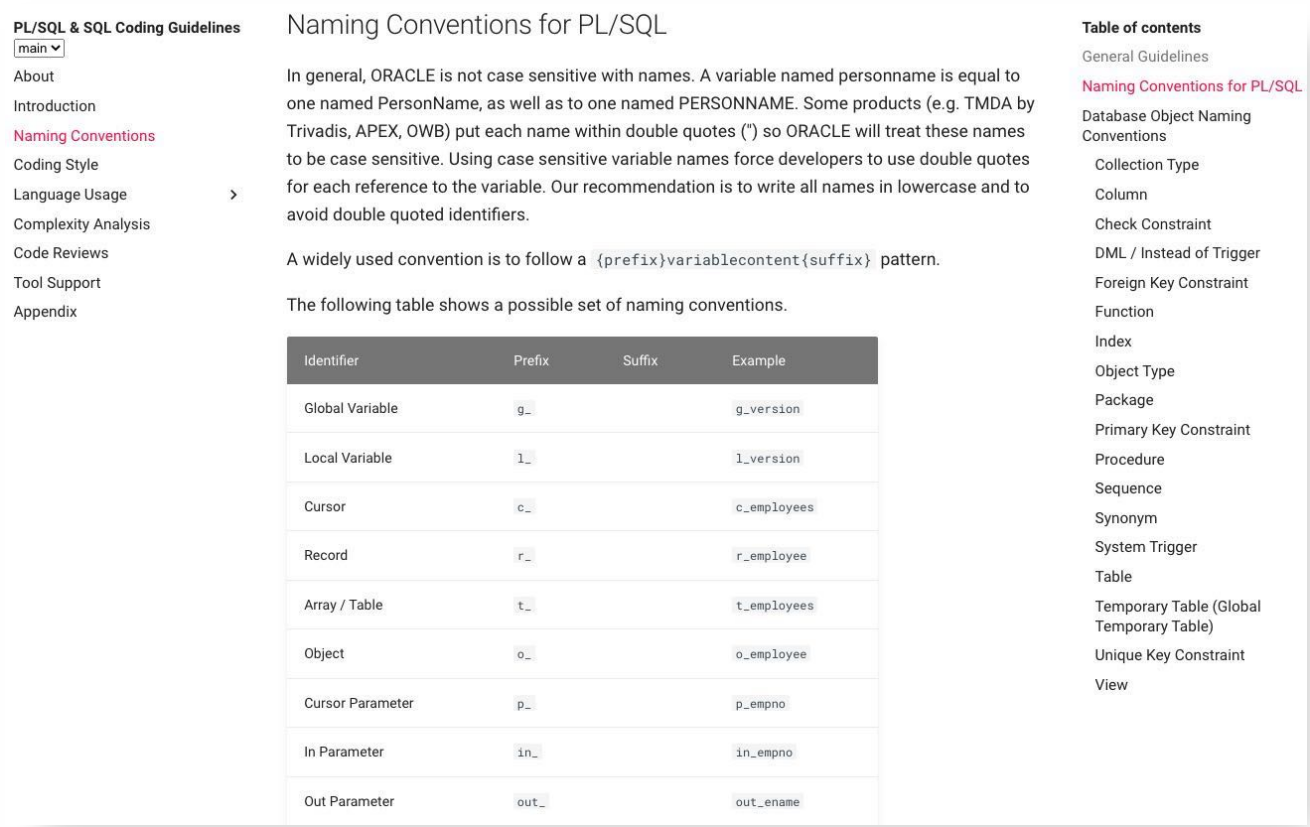


Jede Zeile, die nicht geschrieben wird, muss auch nicht gewartet werden.

Code wird öfter gelesen als geschrieben.

Quelle: https://trivadis.github.io/plsql-and-sql-coding-guidelines/

18 NAMENSKONVENTIONEN



PL/SQL & SQL Coding Guidelines

main

About
Introduction
Naming Conventions
Coding Style
Language Usage
Complexity Analysis
Code Reviews
Tool Support
Appendix

Naming Conventions for PL/SQL

In general, ORACLE is not case sensitive with names. A variable named personname is equal to one named PersonName, as well as to one named PERSONNAME. Some products (e.g. TMDA by Trivadis, APEX, OWB) put each name within double quotes (") so ORACLE will treat these names to be case sensitive. Using case sensitive variable names force developers to use double quotes for each reference to the variable. Our recommendation is to write all names in lowercase and to avoid double quoted identifiers.

A widely used convention is to follow a `{prefix}variablecontent{suffix}` pattern.

The following table shows a possible set of naming conventions.

Identifier	Prefix	Suffix	Example
Global Variable	`g_`		`g_version`
Local Variable	`l_`		`l_version`
Cursor	`c_`		`c_employees`
Record	`r_`		`r_employee`
Array / Table	`t_`		`t_employees`
Object	`o_`		`o_employee`
Cursor Parameter	`p_`		`p_empno`
In Parameter	`in_`		`in_empno`
Out Parameter	`out_`		`out_ename`

Table of contents

General Guidelines
Naming Conventions for PL/SQL
Database Object Naming Conventions
Collection Type
Column
Check Constraint
DML / Instead of Trigger
Foreign Key Constraint
Function
Index
Object Type
Package
Primary Key Constraint
Procedure
Sequence
Synonym
System Trigger
Table
Temporary Table (Global Temporary Table)
Unique Key Constraint
View

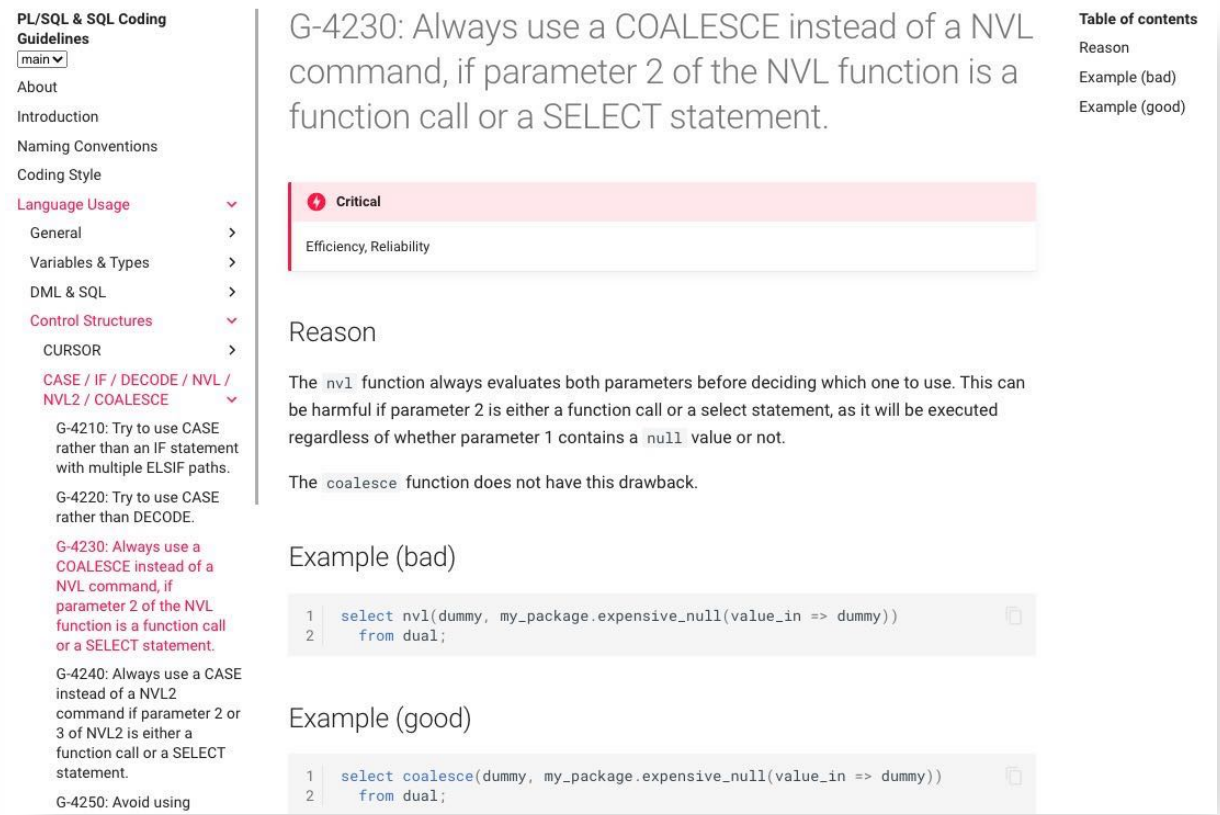
Quelle: https://trivadis.github.io/plsql-and-sql-coding-guidelines/v4.0/2-naming-conventions/naming-conventions/

19 FORMATIERUNG

```
begin
  for rec in (
    select r.country_region as region,
           p.prod_category,
           sum (s.amount_sold) as amount_sold
      from sales s
      join products p
        on p.prod_id = s.prod_id
      join customers cust
        on cust.cust_id = s.cust_id
      join times t
        on t.time_id = s.time_id
      join countries r
        on r.country_id = cust.country_id
     where calendar_year = 2000
     group by r.country_region,
           p.prod_category
     order by r.country_region, p.prod_category
  )
  loop
    if rec.region = 'Asia' then
      if rec.prod_category = 'Hardware' then /* print only one line for demo purposes */
        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
      end if;
    end if;
  end loop;
end;
/
```

```
begin for rec in (select r.country_region as region, p.prod_category,
sum (s.amount_sold) as amount_sold from sales s join products p on p.
prod_id = s .prod_id join customers cust on cust .cust_id = s.cust_id
join times t on t.time_id= s.time_id join countries r on r.country_id
= cust .country_id where calendar_year=2000 group by r.country_region
, p . prod_category order by r . country_region , p . prod_category )
loop if rec.region = 'Asia'  then if rec . prod_category = 'Hardware'
then /* print only one line for demo purposes */  sys . dbms_output .
put_line('Amount: '||rec.amount_sold );end if; end if; end loop; end;
/
```

PROGRAMMIERRICHTLINIEN



PL/SQL & SQL Coding Guidelines

main

About
Introduction
Naming Conventions
Coding Style
Language Usage
General
Variables & Types
DML & SQL
Control Structures
CURSOR
CASE / IF / DECODE / NVL / NVL2 / COALESCE
G-4210: Try to use CASE rather than an IF statement with multiple ELSIF paths.
G-4220: Try to use CASE rather than DECODE.
G-4230: Always use a COALESCE instead of a NVL command, if parameter 2 of the NVL function is a function call or a SELECT statement.
G-4240: Always use a CASE instead of a NVL2 command if parameter 2 or 3 of NVL2 is either a function call or a SELECT statement.
G-4250: Avoid using

G-4230: Always use a COALESCE instead of a NVL command, if parameter 2 of the NVL function is a function call or a SELECT statement.

Table of contents
Reason
Example (bad)
Example (good)

Critical

Efficiency, Reliability

Reason

The `nvl` function always evaluates both parameters before deciding which one to use. This can be harmful if parameter 2 is either a function call or a select statement, as it will be executed regardless of whether parameter 1 contains a `null` value or not.

The `coalesce` function does not have this drawback.

Example (bad)

```
1  select nvl(dummy, my_package.expensive_null(value_in => dummy))
2    from dual;
```

Example (good)

```
1  select coalesce(dummy, my_package.expensive_null(value_in => dummy))
2    from dual;
```

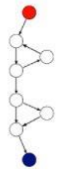
Quelle: https://trivadis.github.io/.../2-case-if-decode-nvl-nvl2-coalesce/g-4230/

21 KOMPLEXITÄTSANALYSE

$$M = E - N + 2P$$

where

- M = cyclomatic complexity
- E = the number of edges of the graph
- N = the number of nodes of the graph
- P = the number of connected components.



Take, for example, a control flow graph of a simple program. The program begins executing at the red node, then enters a loop (group of three nodes immediately below the red node). On exiting the loop, there is a conditional statement (group below the loop), and finally the program exits at the blue node. For this graph, $E = 9, N = 8$ and $P = 1$, so the cyclomatic complexity of the program is 3.

```
1  begin
2     for i in 1..3
3     loop
4        dbms_output.put_line('in loop');
5     end loop;
6     --
7     if 1 = 1
8     then
9        dbms_output.put_line('yes');
10    end if;
11    --
12    dbms_output.put_line('end');
13 end;
14 /
```


Quelle: https://trivadis.github.io/plsql-and-sql-coding-guidelines/v4.0/5-complexity-analysis/complexity-analysis/

db* CODECOP

CODING TO GO

A PRODUCT BY trivadis
Part of Accenture

FUNKTIONALITÄTEN UND VORTEILE

db* CODECOP

Automatisierte Analyse
von PL/SQL-Code

Ermittlung wesentlicher Software-Metriken
wie die zyklomatische Komplexität nach McCabe oder die Halstead-Metrik

Reports zur Qualität von Source-Codes inkl. Handlungsempfehlungen

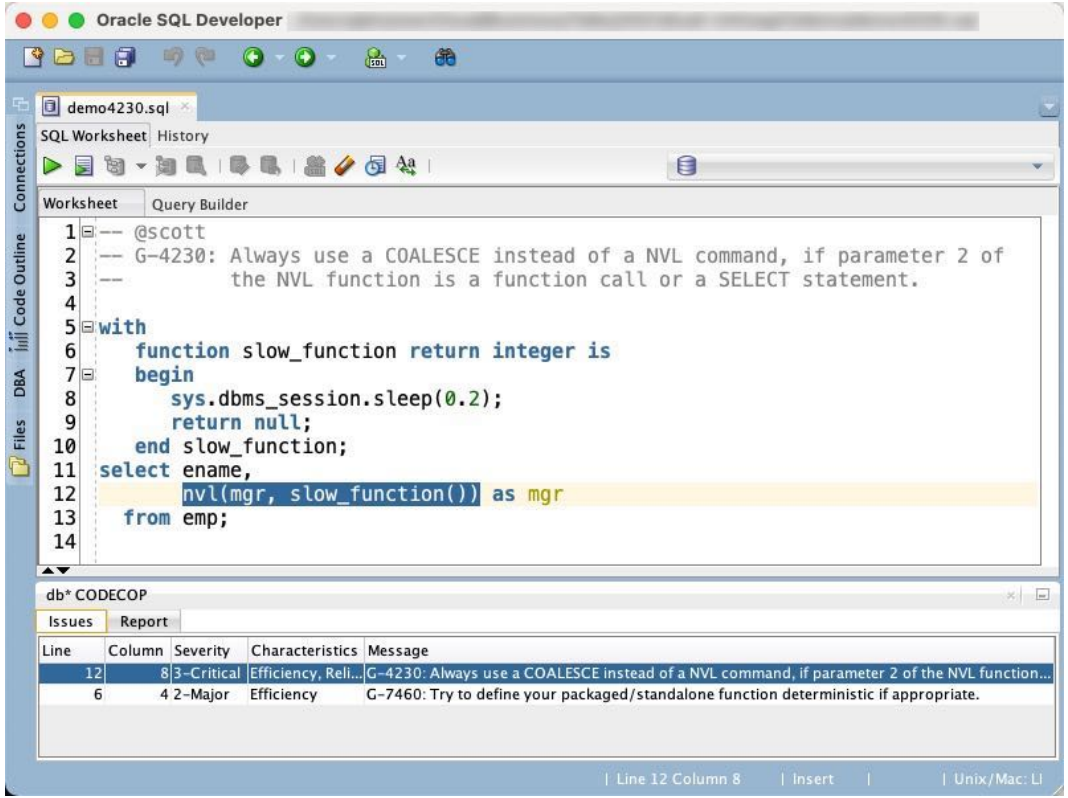
SQL-Developer-Extension
und SonarQube™-Plugin

VORTEILE

- Frühzeitige Erkennung von Fehlern und Fehlentwicklungen
- Verkürzung von Entwicklungszyklen
- Nahtlose Integration in SQL Developer, in bestehende Umgebungen zur statischen Code-Analyse und in Build-Management-Werkzeuge
- Via Kommandozeile auch Ausführung ausserhalb der Entwicklungsumgebung und Integration in automatisierte Prozesse möglich

A PRODUCT BY trivadis
Part of Accenture

24 db* CODECOP FOR SQL DEVELOPER - DEMO



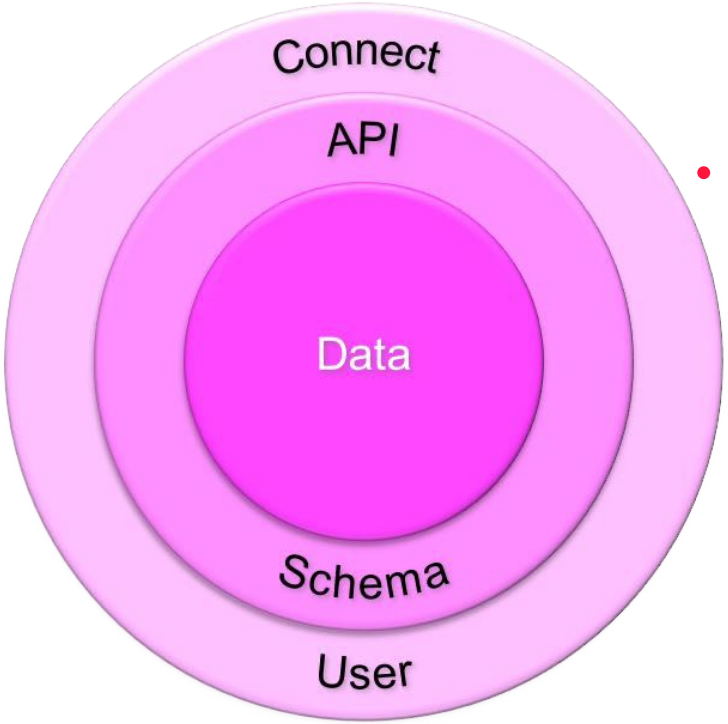
- Format (G-1050 / G-4320)
- G-1080
- G-2150
- G-3185
- G-4230 (G-7460)
- G-4250
- G-7810
- G-9010
- G-9501

Download & Installationsanleitung: https://github.com/Trivadis/plsql-cop-sqldev

trivadis
Part of Accenture

SCHLUSSBEMERKUNGEN

SMARTDB UND PINKDB



Daten werden öfters gelesen als geschrieben.

Konsistente Daten vereinfachen die Arbeit für die Konsumenten.

Weitere Informationen: https://www.salvis.com/blog/2018/07/18/the-pink-database-paradigm-pinkdb/

TOGETHER WE ARE #1 PARTNER FOR BUSINESSES TO HARNESS THE POWER OF DATA FOR A SMARTER LIFE

trivadis
Part of Accenture