

# FIGHTING BAD PL/SQL

Philipp Salvisberg  
18<sup>th</sup> November 2021

## 2 WELCOME

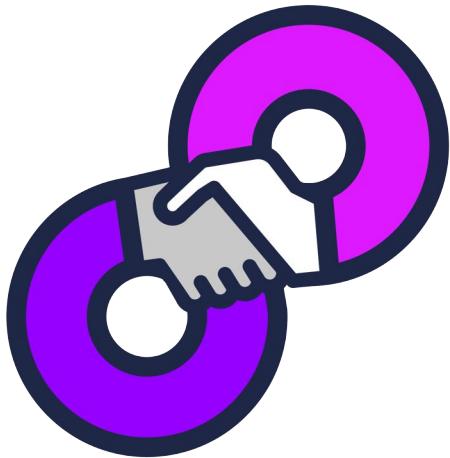


# PHILIPP SALVISBERG

## SENIOR PRINCIPAL CONSULTANT

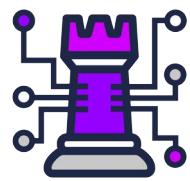
- Database centric development
- Model Driven Software Development
- Author of free SQL Developer Extensions  
PL/SQL Unwrapper, db\* CODECOP, utPLSQL,  
plscope-utils, oddgen and Bitemp Remodeler

### 3 TRIVADIS & ACCENTURE: #1 FOR DATA & AI

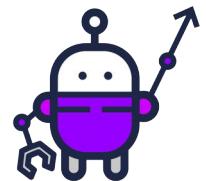


- Together we are 1500 specialists at 34 locations in Switzerland, Germany and Austria with a focus on **Data & Applied Intelligence**.
- Together we support you in the **intelligent end-to-end use of your data**.
- We cover the entire spectrum: from the **development and operation of data platforms and solutions**, to the refinement of data as well as **consulting and training**.
- We achieve this through the unique combination of Trivadis' **technological expertise** and Accenture's **strategic know-how** in the field of data.

## 4 OUR RANGE OF SERVICE



BUSINESS  
ANALYTICS



BUSINESS  
AUTOMATION



MODERN DATA  
PLATFORMS



MODERN CLOUD  
INFRASTRUCTURE

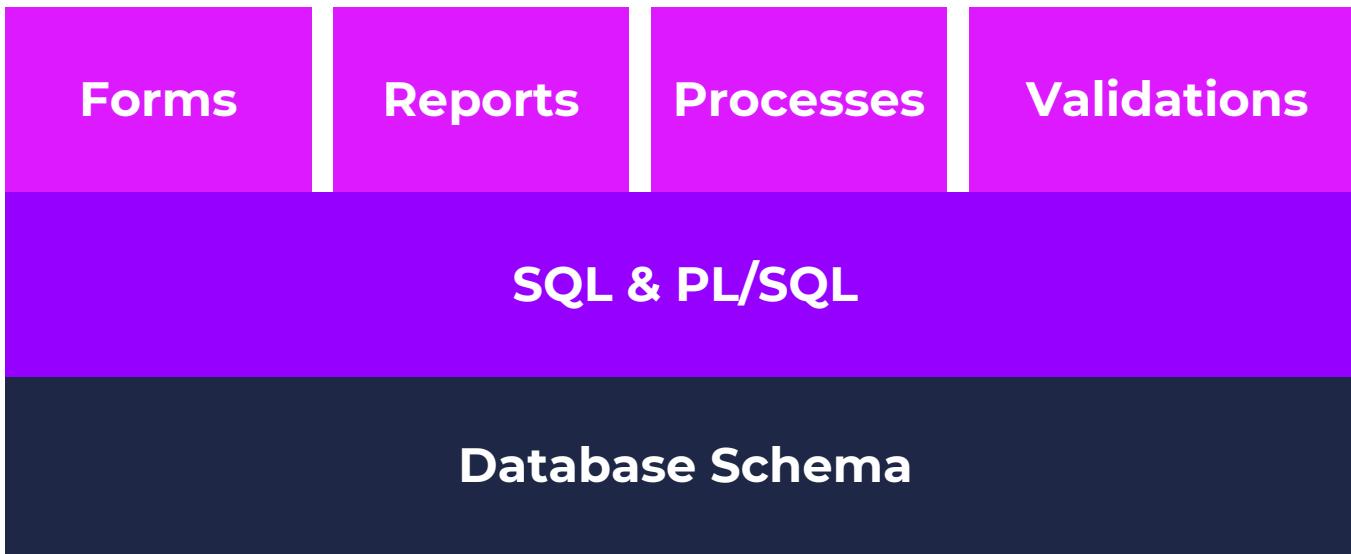
## 5 VISIT OUR BOOTH IN GATHER.TOWN - STAGE 1

- Find us on stage 1 to the right of the escalator
- Play our game «ALPAKA»
- Meet our experts at the booth
- Attend the interview about the joint future of Trivadis and Accenture on Thursday, 18.11.2021, 11:50h (DOAG Studio)



# WHERE IS MY CODE?

## 7 APPLICATION (APEX, ...)



# 8 SQL

## Original Report in APEX

```
select d.deptno, d.dname, ...
  from dept d
  join emp e
    on e.deptno = d.deptno
  join salgrade s ...
  join ...
  join ...
where d.deptno = :p10_deptno
  and ...
```



## Relational View in the Database

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

## Simplified Report in APEX

```
select d.deptno, d.dname, ...
  from dept_emp_v
where d.deptno = :p10_deptno
```

# 9 PL/SQL

## Original Process in APEX

```
declare
    l_comm emphist.comm%type;
    ...
begin
    select h.comm into l_comm
        from ...
        where e.id = :p20_id;
    if l_comm > 4000 then
        ...
    else
        case ...
            end case;
    end if;
    :p20_comm := l_comm;
end;
```



## PL/SQL Package in the Database

```
create package emp_mgmt is
    function comm(in_id in integer)
    return number is ...
end;
create package body emp_mgmt is
    function comm(in_id in integer)
    return number is ...
    begin
        ...
    end; ...
end;
```

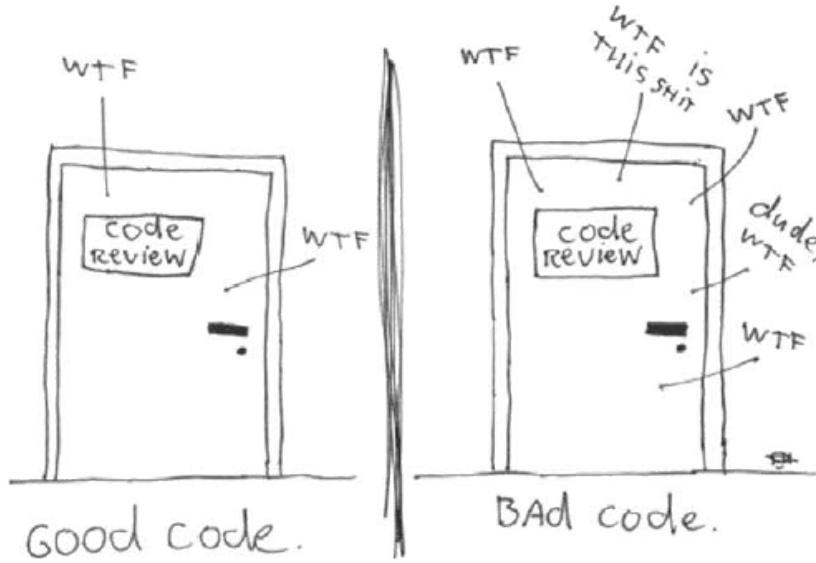
## Simplified Process in APEX

```
:p20_comm := emp_mgmt.comm(:p20_id);
```

# SOFTWARE QUALITY

## 11 CODE REVIEWS

The ONLY VALID MEASUREMENT  
OF Code QUALITY: WTFs/minute



Source: [http://www.osnews.com/story/19266/WTFs\\_m](http://www.osnews.com/story/19266/WTFs_m)

# 12 TIPS



Source: <https://www.youtube.com/watch?v=IhE4pTprQ4E>

# 13 PL/SQL & SQL CODING GUIDELINES

## PL/SQL & SQL Coding Guidelines

[main](#) ▾

[About](#)

[Introduction](#)

[Naming Conventions](#)

[Coding Style](#)

[Language Usage](#)

[Complexity Analysis](#)

[Code Reviews](#)

[Tool Support](#)

[Appendix](#)



The Oracle Database Developer community is made stronger by resources freely shared by experts around the world, such as the Trivadis Coding Guidelines. If you have not yet adopted standards for writing SQL and PL/SQL in your applications, this is a great place to start.

*Steven Feuerstein*

Steven Feuerstein  
Senior Advisor  
Insum Solutions



Coding Guidelines are a crucial part of any developer's toolkit. In fact, that code is more often read than written, it makes sense to put efforts to ease the work of the reader.

I am convinced that this standard may be a good starting point for your own guidelines.

Roger Troller  
Senior Consultant  
finnova AG Bankware

## Table of contents

[Foreword](#)

Every line you don't write, is a line you don't have to maintain

Code is more often read than written

# 14 NAMING CONVENTIONS

## PL/SQL & SQL Coding Guidelines

main ▾

About

Introduction

Naming Conventions

Coding Style

Language Usage

## 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

Source: <https://trivadis.github.io/plsql-and-sql-coding-guidelines/main/2-naming-conventions/naming-conventions/>

# 15 CODE STYLE

```
1 begin
2   for rec in (
3     select r.country_region as region,
4           p.prod_category,
5           sum (s.amount_sold) as amount_sold
6     from sales s
7     join products p
8       on p.prod_id = s.prod_id
9     join customers cust
10      on cust.cust_id = s.cust_id
11    join times t
12      on t.time_id = s.time_id
13    join countries r
14      on r.country_id = cust.country_id
15    where calendar_year = 2000
16    group by r.country_region,
17             p.prod_category
18    order by r.country_region, p.prod_category
19  )
20  loop
21    if rec.region = 'Asia' then
22      if rec.prod_category = 'Hardware' then /* print only one line for demo purposes */
23        sys.dbms_output.put_line('Amount: ' || rec.amount_sold);
24      end if;
25    end if;
26  end loop;
27 end;
28 /
```

```
1 begin for rec in (select r.country_region as region, p.prod_category,
2 sum (s.amount_sold) as amount_sold from sales s join products p on p.
3 prod_id = s .prod_id join customers cust on cust .cust_id = s.cust_id
4 join times t on t.time_id= s.time_id join countries r on r.country_id
5 = cust .country_id where calendar_year=2000 group by r.country_region
6 , p . prod_category order by r . country_region , p . prod_category )
7 loop if rec.region = 'Asia' then if rec . prod_category = 'Hardware'
8 then /* print only one line for demo purposes */ sys . dbms_output .
9 put_line('Amount: '||rec.amount_sold );end if; end if; end loop; end;
10 /
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

# 16 GUIDELINES

## 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.



Critical

Efficiency, Reliability

## Table of contents

Reason

Example (bad)

Example (good)

## 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;
```

Source: <https://trivadis.github.io/plsql-and-sql-coding-guidelines/main/.../g-4230/>

# 17 COMPLEXITY ANALYSIS

## PL/SQL & SQL Coding Guidelines

main ✓

About

Introduction

Naming Conventions

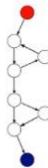
Coding Style

Language Usage

where

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

- $M$  = 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 /
```

## Table of contents

Halstead Metrics

Calculation

McCabe's Cyclomatic Complexity

Description

Calculation



# db\* CODECOP

CODING TO GO

A PRODUCT BY

**trivadis**  
Part of Accenture

# FUNCTIONALITIES AND BENEFITS



**Automated analysis** of PL/SQL code



**Determining essential software metrics**  
(like McCabe's cyclomatic complexity or the Halstead metric)



**Quality reports** of source codes incl. recommendations for action



**SQL Developer Extension** and SonarQube™ plugin

## BENEFITS

- Early detection of errors and undesirable developments
- Shorten development cycles
- Seamless integration into SQL Developer, into existing environments for static code analysis and into build management tools
- Execution outside the development environment and integration into automated processes possible via command line

A PRODUCT BY

**trivadis**

# 20 db\* CODECOP FOR SQL DEVELOPER - DEMO

The screenshot shows the Oracle SQL Developer interface. In the central workspace, there is a SQL Worksheet containing the following PL/SQL code:

```
1 -- @scott
2 -- G-4230: Always use a COALESCE instead of a NVL command, if parameter 2 of
3 -- the NVL function is a function call or a SELECT statement.
4
5 with
6   function slow_function return integer is
7   begin
8     sys.dbms_session.sleep(0.2);
9     return null;
10    end slow_function;
11  select ename,
12    nvl(mgr, slow_function()) as mgr
13  from emp;
14
```

In the bottom right corner of the code editor, the word "mgr" is highlighted in yellow, indicating it is being analyzed by the tool.

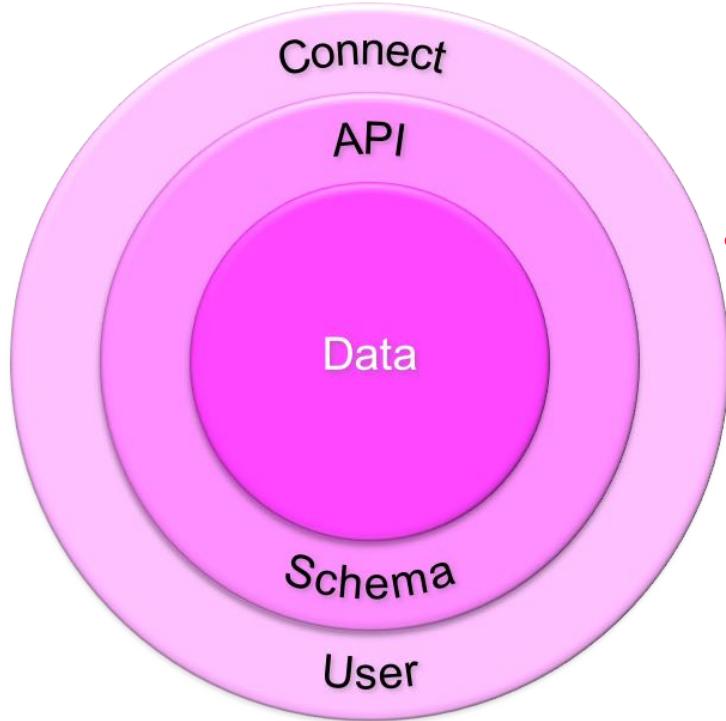
Below the workspace is the "db\* CODECOP" interface. The "Issues" tab is selected, displaying the following results:

Line	Column	Severity	Characteristics	Message
12	8	3-Critical	Efficiency, Reli...	G-4230: Always use a COALESCE instead of a NVL command, if parameter 2 of the NVL function...
6	4	2-Major	Efficiency	G-7460: Try to define your packaged/standalone function deterministic if appropriate.

- [Format \(G-1050 / G-4320\)](#)
- [G-1080](#)
- [G-2150](#)
- [G-3185](#)
- [G-4230 \(G-7460\)](#)
- [G-4250](#)
- [G-5080](#)
- [G-7810](#)
- [G-9010](#)
- [G-9501](#)
- [G-9600-9603](#)

# WHERE IS MY CODE AGAIN?

## 22 SmartDB & PinkDB – TWO OF A KIND



Data is more often  
read than written

Consistent data  
simplifies the work of  
consumers

More information: <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**