FIGHTING BAD PL/SQL

Philipp Salvisberg
20th October 2021

trivadis
Part of Accenture

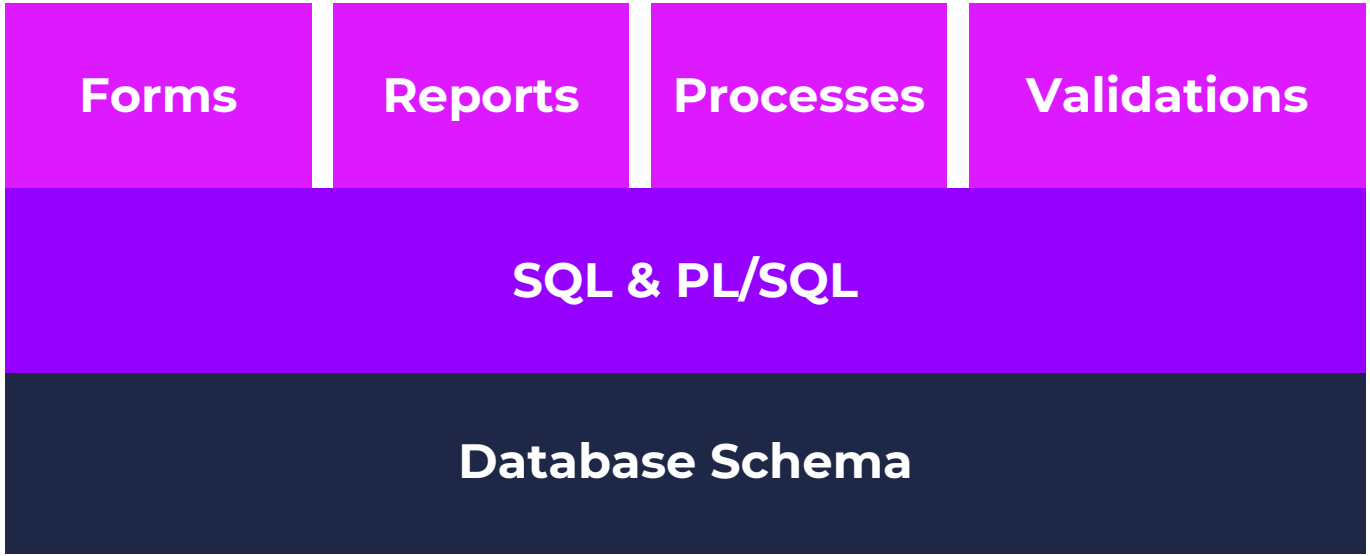
PHILIPP SALVISBERG

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

WHERE IS MY CODE?

4 APPLICATION (APEX, ...)



5 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
```

6 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;
```

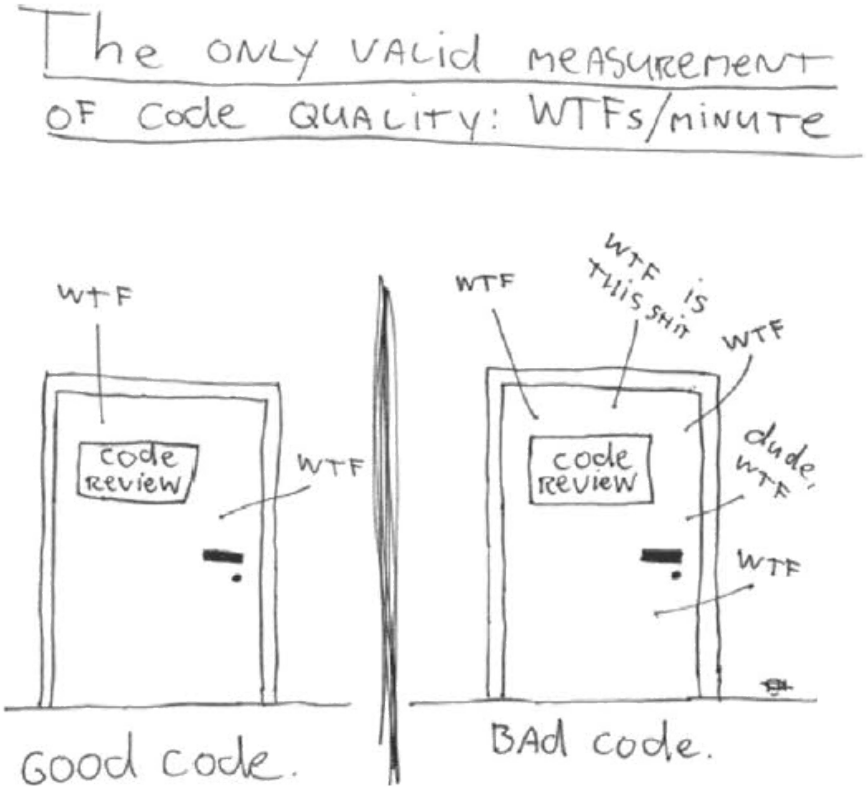
Simplified Process in APEX

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

trivadis
Part of Accenture

SOFTWARE QUALITY

8 CODE REVIEWS

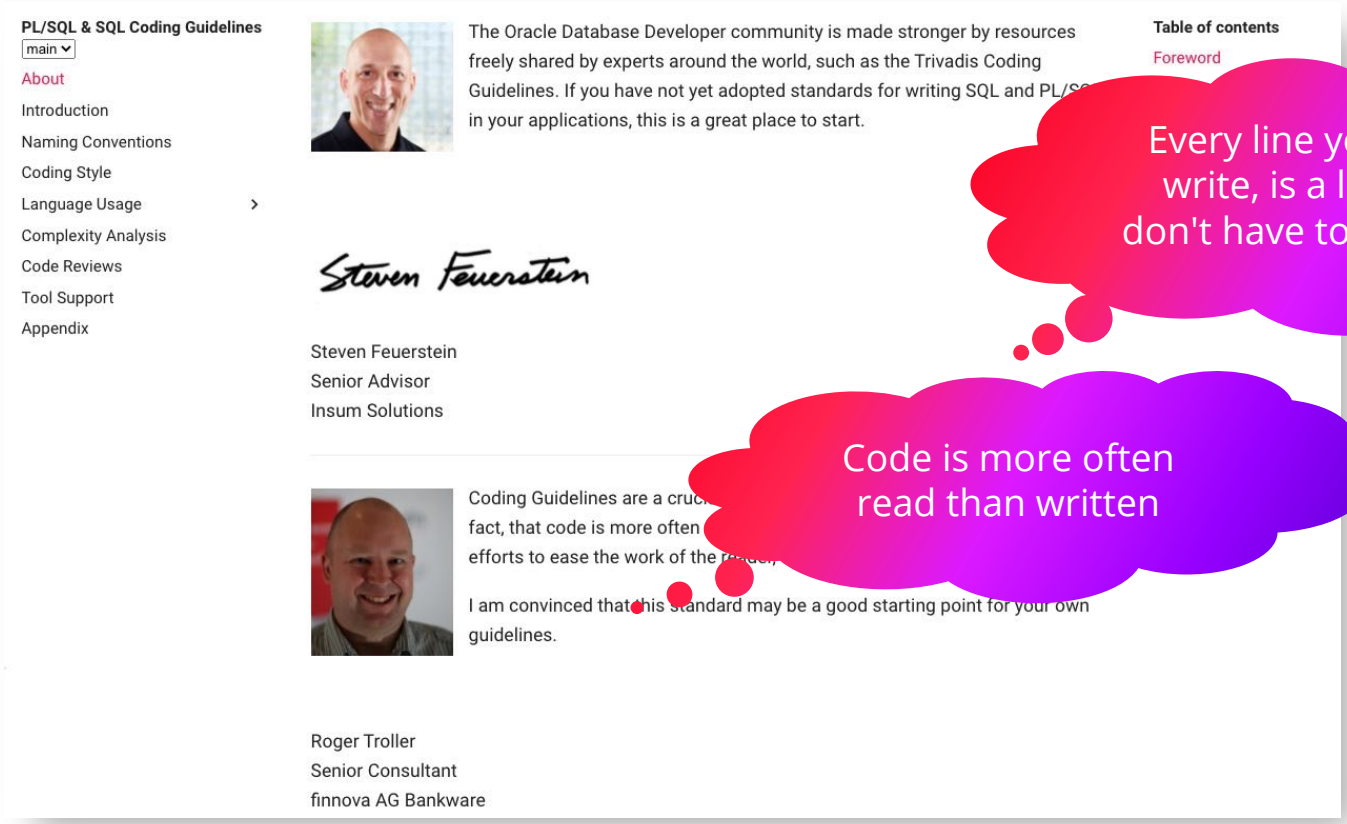


Source: http://www.osnews.com/story/19266/WTFs_m

9 TIPS

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

10 PL/SQL & SQL CODING GUIDELINES



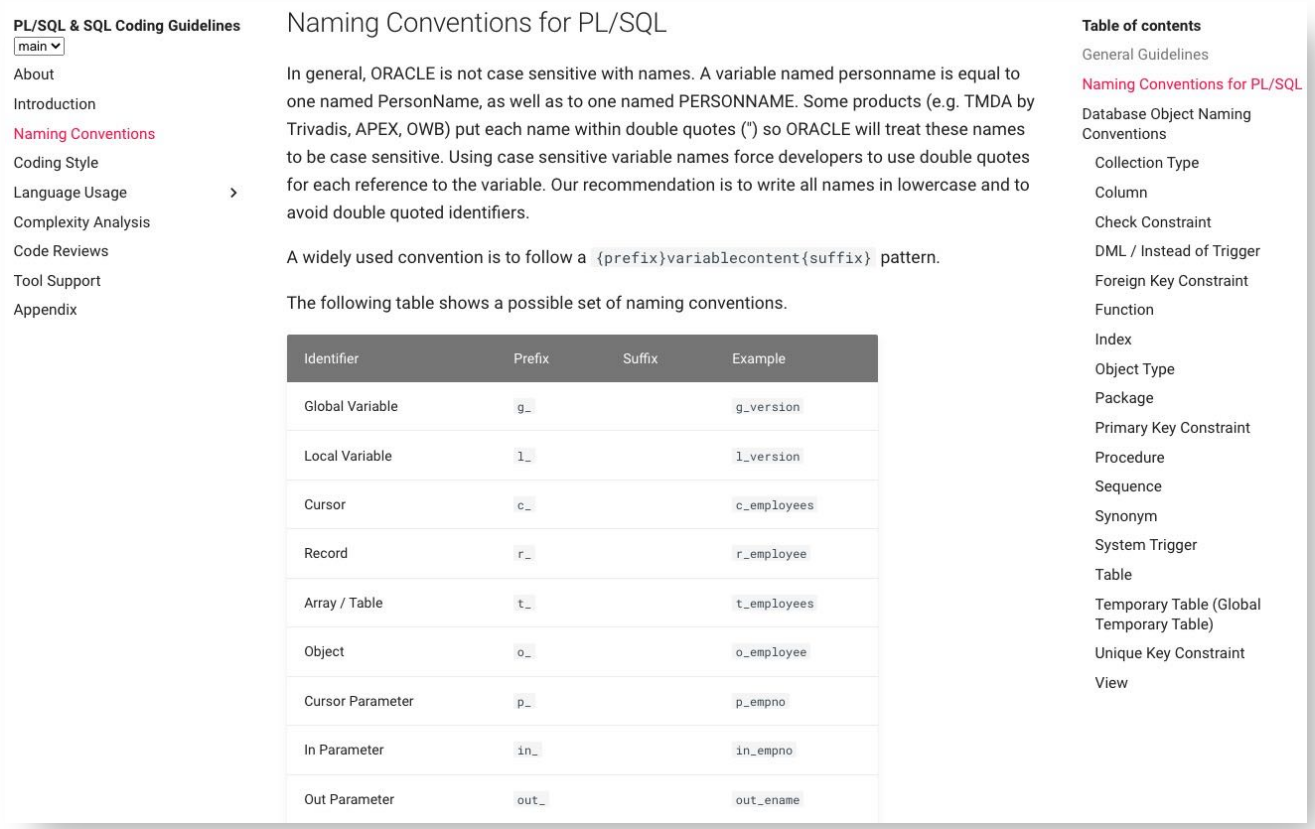
Every line you don't write, is a line you don't have to maintain

Code is more often read than written

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

trivadis
Part of Accenture

11 NAMING CONVENTIONS



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

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

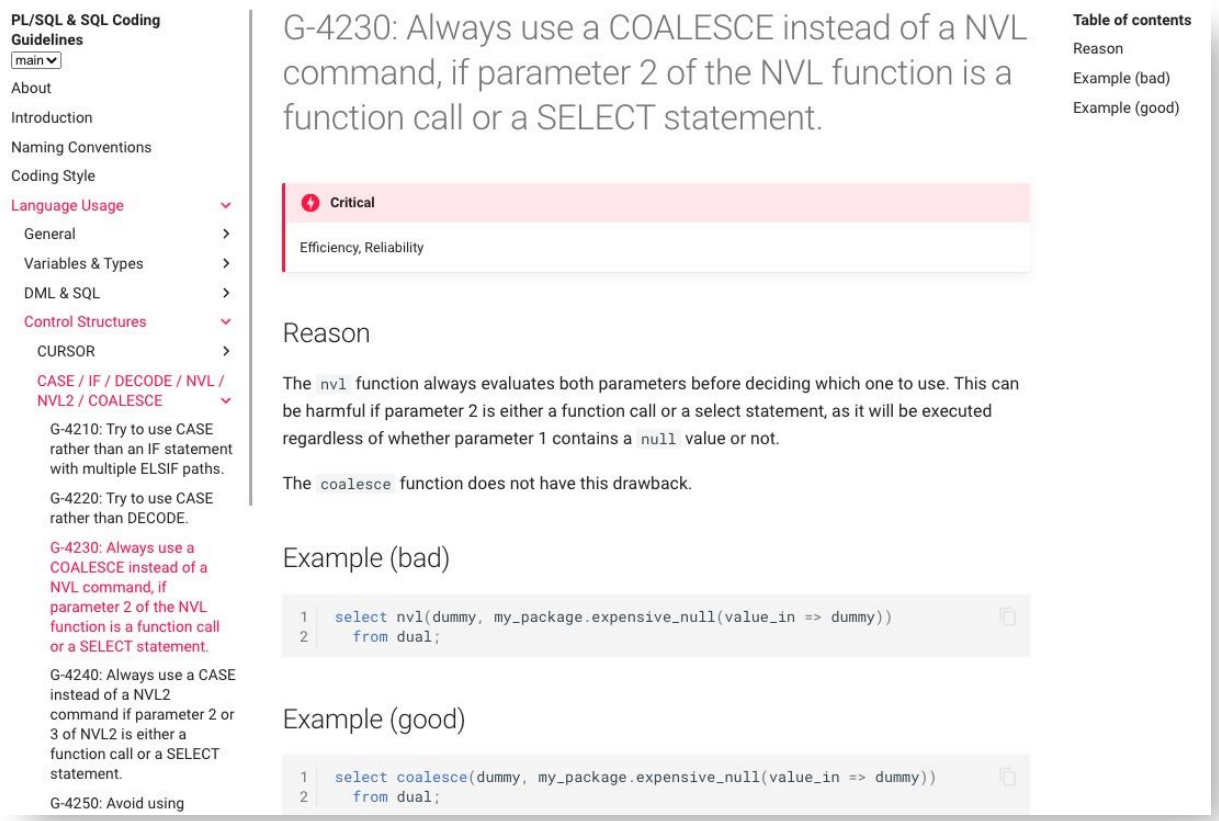
12 CODE STYLE

```
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;
/
```

trivadis
Part of Accenture

13 GUIDELINES



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

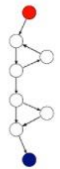
trivadis
Part of Accenture

14 COMPLEXITY ANALYSIS

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

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 /
```


Source: https://trivadis.github.io/plsql-and-sql-coding-guidelines/main/5-complexity-analysis/complexity-analysis/

db* CODECOP

CODING TO GO

A PRODUCT BY trivadis
Part of Accenture

FUNCTIONALITIES AND BENEFITS

db* CODECOP

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

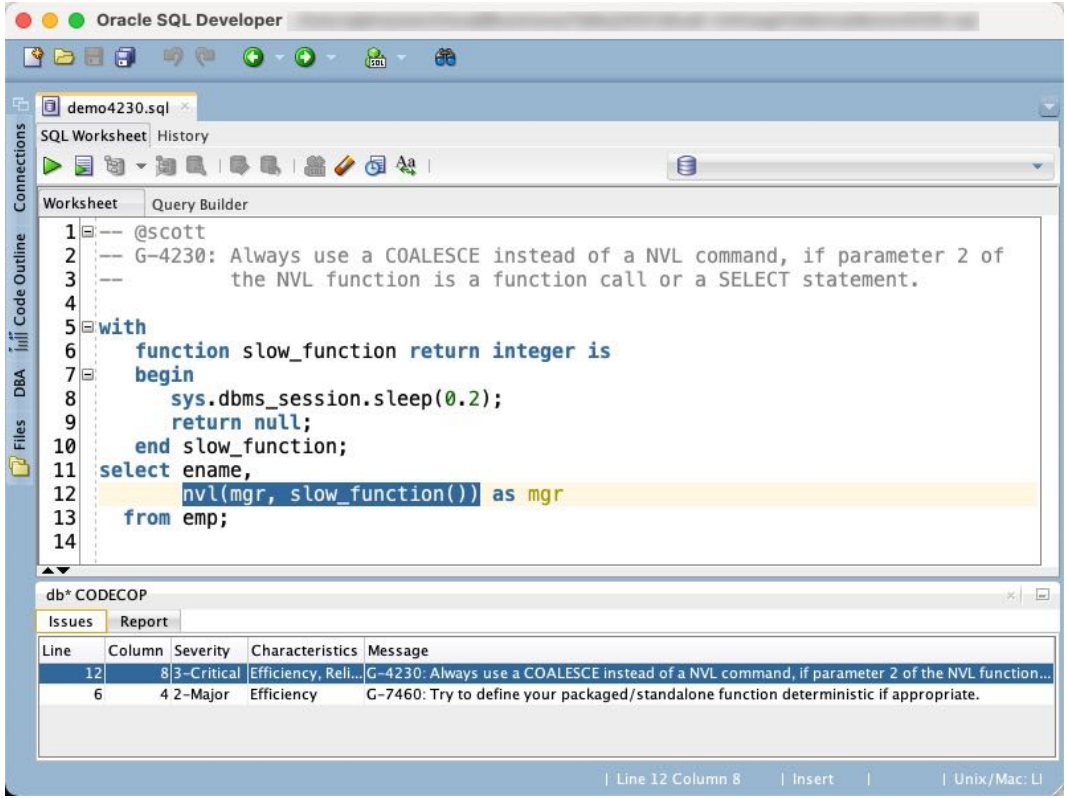
SQL Developer Extension and SonarQubeTM plugin

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

17 db* CODECOP FOR SQL DEVELOPER - DEMO



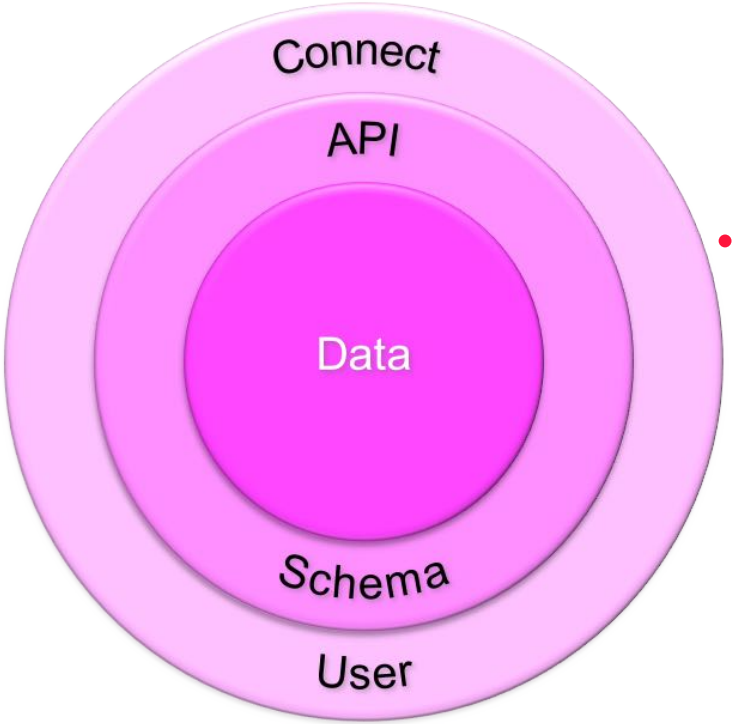
- Format (G-1050 / G-4320)
- G-1080
- G-2150
- G-3185
- G-4230 (G-7460)
- G-4250
- G-5080
- G-7810
- G-9010
- G-9501
- G-9600-9603

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

trivadis
Part of Accenture

WHERE IS MY CODE AGAIN?

19 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

trivadis
Part of Accenture