# WELCOME

# How to Find Unused Oracle Database Objects and Subprograms

Philipp Salvisberg

27th September 2013

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN







trivadis makes IT easier.

# About Me

- With Trivadis since April 2000
  - Senior Principal Consultant, Partner
  - Member of the Board of Directors
  - philipp.salvisberg@trivadis.com
  - www.salvis.com/blog
- Member of the trivadis performanceteam


- Main focus on database centric development with Oracle DB
  - Application Development
  - Business Intelligence
  - Application Performance Management
- Over 20 years experience in using Oracle products

# AGENDA

1. Background
2. Finding Unused Objects
3. Finding Unused Sub-Objects
4. Core Messages

# Advertising Business

- Posters, eBoards, ePanels
  - in streets
  - in train stations
  - at bus stops
  - in parking lots
  - in shopping centers
  - in tourist resorts
  - in airports
  - in and on vehicles
  - on buildings
- APG|SGA, End of 2012
  - Sales revenue in CH: MCHF 297.1
  - Net income:  MCHF 50.0
  - Employees: 652

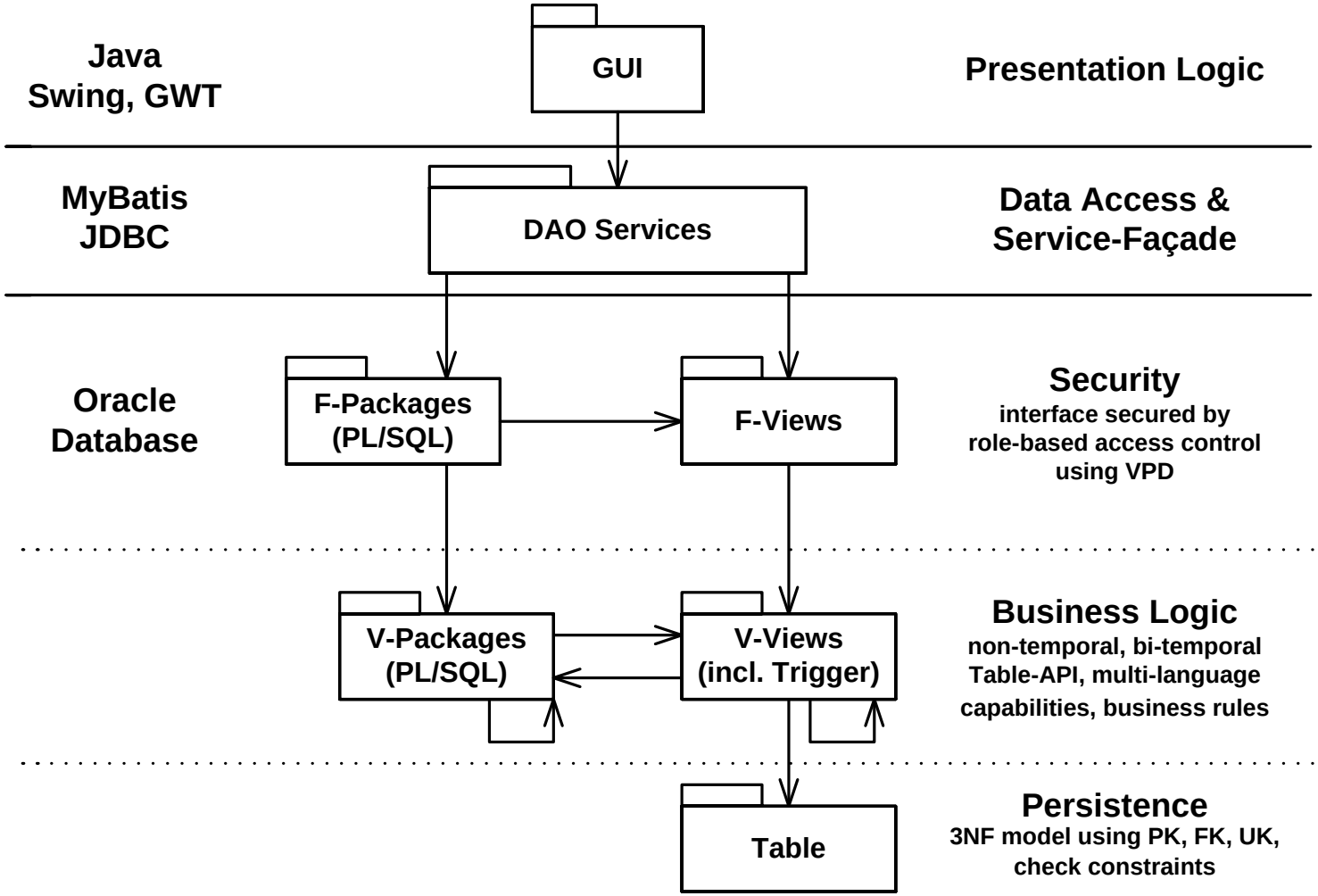







trivadis makes IT easier.

# Architecture



# Some Figures

- Initial release of "IT21" in 2001 to support all business processes
- Extended constantly due to new and revised demands
- Consists of 41 schemas consuming ~ 330 GB (on 22nd September 2013)

| Object Type | Count | Sub-objects | Lines of code | Code in KB |
|---|---|---|---|---|
| FUNCTION | 8 | - | 191 | 8 |
| PACKAGE | 1'405 | 15'223 | 164'348 | 6'797 |
| PACKAGE BODY | 1'391 | 21'209 | 1'068'633 | 41'728 |
| PROCEDURE | 6 | - | 281 | 9 |
| TABLE | 1'625 | - | - | - |
| TRIGGER | 1'203 | - | 63'820 | 3'213 |
| TYPE | 335 | 102 | 3'619 | 141 |
| TYPE BODY | 31 | 102 | 1'167 | 45 |
| VIEW | 4'619 | - | 107'699 | 5'119 |
| **Total** | **10'623** | **21'311** | **1'409'758** | **57'060** |

# Mandate

## Tasks

- Find unused objects
  - Functions
  - Packages
  - Procedures
  - Types
  - Views
- Find unused sub-objects
  - Package procedures
  - Package functions
  - Type methods

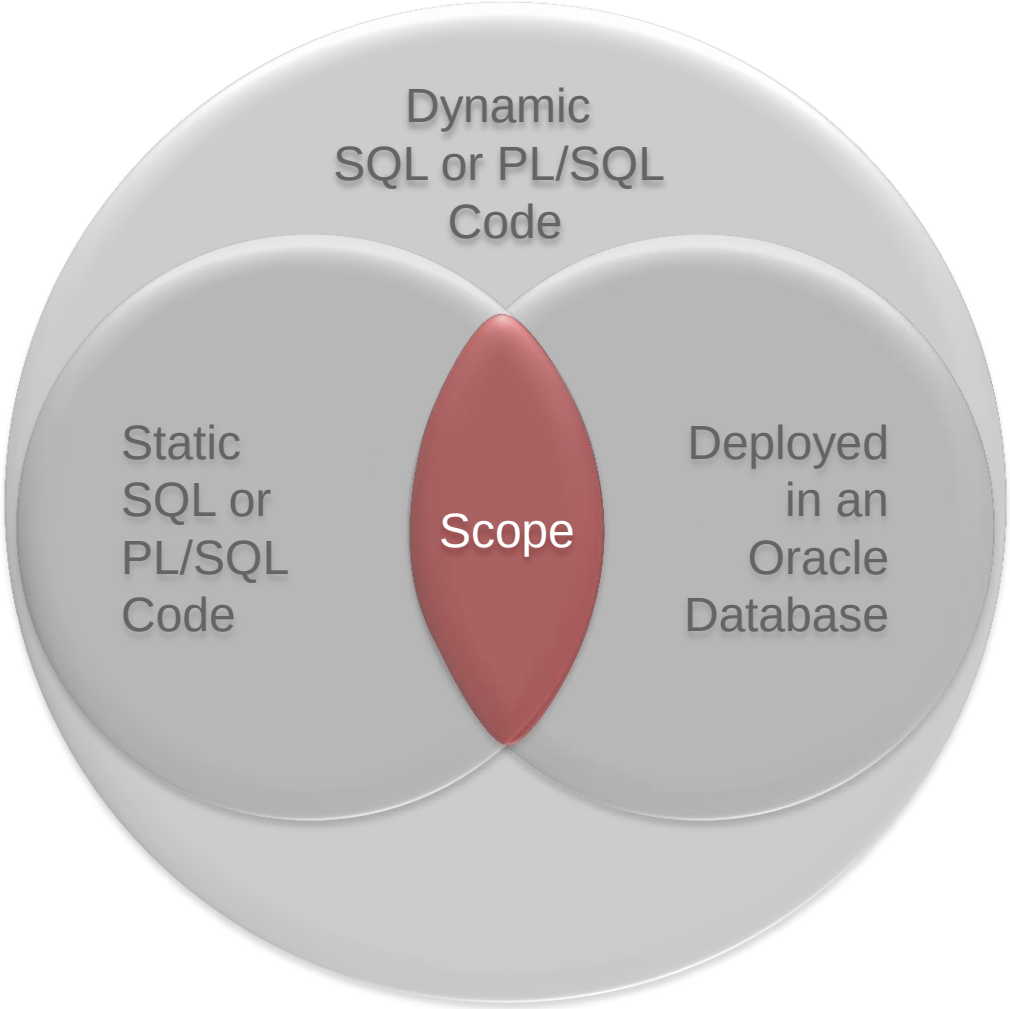
## Objectives

- Reduce maintenance base
- Avoid alignment of unused objects
- Simplify overall solution
- Increase development efficiency
- Reduce costs

# AGENDA

1. Background
2. Finding Unused Objects
3. Finding Unused Sub-Objects
4. Core Messages

# Scope of Database Dependency Analysis



# Query Unreferenced Objects (1)

```
WITH
   -- objects to be checked
   obj AS (
      SELECT /*+no_merge */ owner, object_type, object_name, status
        FROM dba_objects
       WHERE -- only grantable object types
             object_type IN ('FUNCTION', 'PROCEDURE', 'PACKAGE', 'TYPE', 'VIEW')
             -- only appl schemas (exclude Oracle schemas)
         AND owner NOT IN ('ANONYMOUS', 'APEX_PUBLIC_USER', 'CTXSYS', 'DBSNMP', 'DIP',
                           'EXFSYS', 'FLOWS_FILES', 'LBACSYS', 'MDDATA', 'MDSYS',
                           'MGMT_VIEW', 'OLAPSYS', 'ORACLE_OCM', 'ORDDATA',
                           'ORDPLUGINS', 'ORDSYS', 'OUTLN', 'OWBSYS',
                           'SI_INFORMTN_SCHEMA', 'SPATIAL_CSW_ADMIN_USR',
                           'SPATIAL_WFS_ADMIN_USR', 'SYS', 'SYSMAN', 'SYSTEM',
                           'WKPROXY', 'WKSYS', 'WK_TEST', 'WMSYS', 'XDB', 'XS$NULL')
   ), …
```

# Query Unreferenced Objects (2)

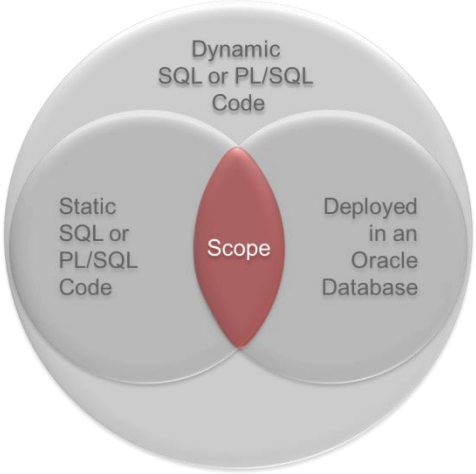
```
…  -- join grants and relevant usages
   dep AS (
      SELECT /*+use_hash (tp) use_hash(r) use_hash (d) */ obj.owner, obj.object_type,
             obj.object_name, obj.status, tp.grantee,
             SUM(CASE WHEN r.type IS NULL THEN 0 -- not referenced
                      WHEN r.type = 'SYNONYM' AND d.type IS NULL THEN 0 -- by syn. only
                      WHEN r.owner = obj.owner AND r.type = 'PACKAGE BODY' AND
                           obj.object_type = 'PACKAGE' AND r.name = obj.object_name THEN
                        0 -- referenced by own package body only
                      ELSE 1
                  END) AS ref_count
        FROM obj
        LEFT JOIN dba_tab_privs tp -- privileges of the object
          ON tp.owner = obj.owner
             AND tp.table_name = obj.object_name
        LEFT JOIN dba_dependencies r -- dependencies of the object
          ON r.referenced_name = obj.object_name
             AND r.referenced_owner = obj.owner
             AND r.referenced_type = obj.object_type
        LEFT JOIN dba_dependencies d -- relevant for synonym dependencies only
          ON d.referenced_name = r.name
             AND d.referenced_owner = r.owner
             AND d.referenced_type = r.type
       GROUP BY obj.owner, obj.object_type, obj.object_name, obj.status, tp.grantee
   ), …
```

# Query Unreferenced Objects (3)

```
…  -- build grantee_list
   glist AS (
      SELECT owner,
             object_type,
             object_name,
             status,
             SUM(ref_count) AS ref_count,
             listagg(grantee, ',') within GROUP(ORDER BY grantee) AS grantee_list
        FROM dep
       GROUP BY owner, object_type, object_name, status
   )
-- main
SELECT owner, object_type, object_name, status, grantee_list
  FROM glist
 WHERE ref_count = 0
 ORDER BY owner, object_type, object_name;
```

```
OWNER      OBJECT_TYP OBJECT_NAME                    STATUS GRANTEE_LIST
---------- ---------- ------------------------------ ------ --------------------
AC_READ    VIEW       MITARBEITER_STAMMDATEN_V       VALID
AENV       FUNCTION   APPL_PARAMETER_STRING_EVALUATE VALID
…
ALOG       VIEW       APPL_LOG_F                     VALID  BASISROLLE,GRUNDDATEN
…
2419 rows selected.
```

# False Positives




- Usage outside of the database
  - Java GUI using generated DAO
  - Java Services using plain JDBC
  - Excel ODBC queries optionally through VBA
  - Legacy Oracle Forms/Reports
  - Scripts (Installation, Test, Configuration Management, Operation)
- Dynamic SQL & PL/SQL within the database
  - Job scheduler
  - VPD policies
  - String fragments within code (execute immediate, DBMS_SQL)
  - Application tables containing SQL or PL/SQL fragments
- Generated Code
  - Table API (Views, Trigger, PL/SQL Packages)
  - Enumerations/Domains (Views)
  - AQ-Views
- New Code (incomplete, currently in developed for new release)

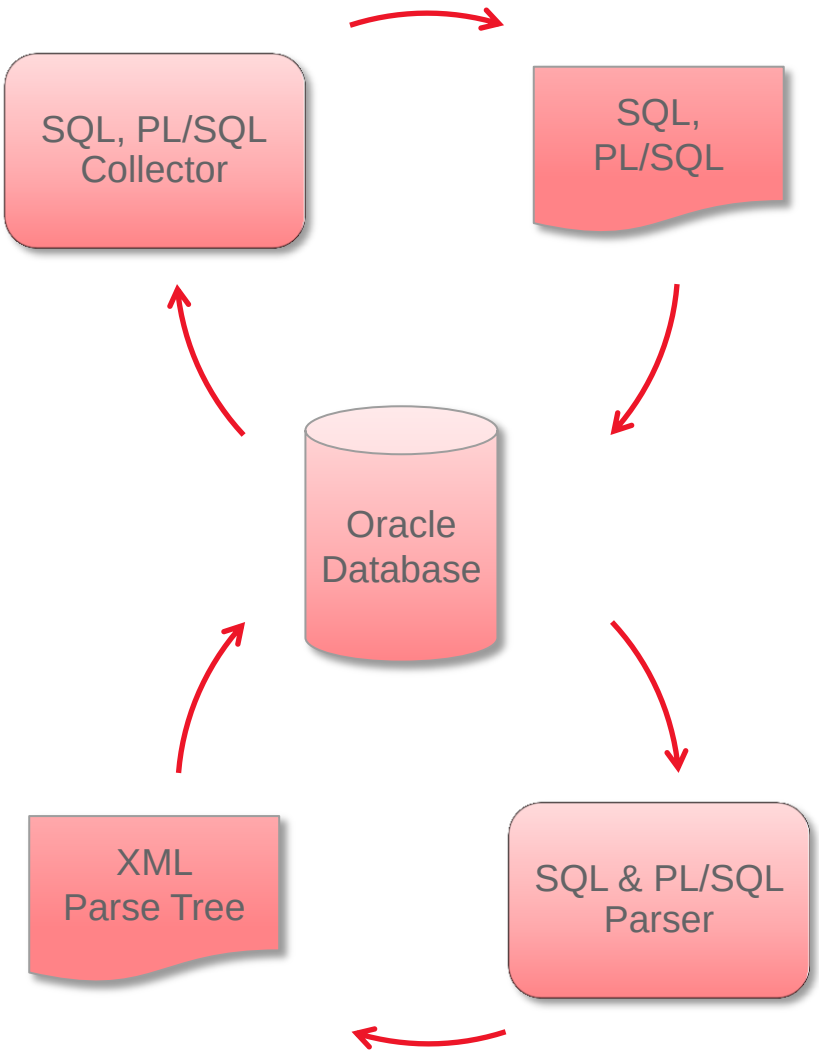
# TVDCA – Trivadis PL/SQL & SQL CodeAnalyzer

- Command line utility to parse PL/SQL and SQL source code within an Oracle database and store the resulting XML parse-trees in dedicated relational tables – as an extension to the Oracle Data Dictionary – for further analysis
- Analysis is supported for static and dynamic code
- Free download from http://www.salvis.com/blog/

# Extend the Oracle Data Dictionary with Captured SQL

```
SQL> desc tvd_captured_sql_t

Name            Type
-------------   -----------------
CAP_ID          NUMBER
CAP_SOURCE      CLOB
SQL_ID          VARCHAR2(13)
USER_NAME       VARCHAR2(30)
SCHEMA_NAME     VARCHAR2(30)
MODULE          VARCHAR2(64)
ACTION          VARCHAR2(64)
LAST_LOAD_TIME  DATE
…
PARSE_TREE      XMLTYPE
```



# Related TVDCA Views

| View name | Shows usage of | Scope | Important columns |
|---|---|---|---|
| tvd_sql_usage_v | Tables, views | Select, insert, update, delete and merge statements in captured SQL | table_owner, table_name |
| tvd_sql_paren_ident_v | Parenthesis expressions | All identifiers in captured SQL, PL/SQL | name (e.g function or type), context (e.g. package or schema) |
| tvd_sql_dot_ident_v | Dot expressions | All identifiers in captured SQL, PL/SQL | left_value, right_value |

# False Negatives

- Ambiguous naming
  - Different object types with same name
  - Same object names used in different schemas
- Examples
  - View ABGRENZUNGSBUCHUNGEN_V exists in schema FM, VK
  - Package GP_UTIL_PA exists in schema GP, VK

```
SELECT object_name,
       COUNT(DISTINCT owner) AS count_owner,
       COUNT(DISTINCT object_type)
  FROM dba_objects
 WHERE object_type IN ('FUNCTION', 'PROCEDURE', 'PACKAGE', 'TYPE', 'VIEW')
       AND owner NOT IN ('ZX', 'APCDWH')
 GROUP BY object_name
HAVING COUNT(DISTINCT owner) > 1 OR COUNT(DISTINCT object_type) > 1;
```

# Automate Most – But Not All – 100% is not Feasible

Captured

- Java DAO usage
- Job network (PL/SQL)
- Excel ODBC access
- Quality queries
- Workflow calls (PL/SQL)
- Search utility Interface (Views)
- Managed VPD predicates

Excluded

- Dynamic Code within database
- Java Services using non-standard access paths (e.g. plain JDBC)
- Legacy Oracle Forms/Reports
- Scripts

# Identified Unused Objects

| What | Quantity |
|---|---|
| Unreferenced objects | 2'419 |
| ./. Generated F-Views | 1'468 |
| ./. Packages with grants | 56 |
| ./. Generated domain packages (enumerations) | 215 |
| ./. Generated table API (temporal views) | 223 |
| ./. Objects in irrelevant schemas (generated, operations, samples) | 44 |
| ./. Manually managed false positives (apg_ignore_objects_v) | 54 |
| ./. Exclusive dynamic usage identified through TVDCA | 69 |
| **= Total** | **290** |

# AGENDA

1. Background
2. Finding Unused Objects
3. Finding Unused Sub-Objects
4. Core Messages

# Oracle Data Dictionary Dependency Granularity

DBA_DEPENDENCIES

- Object (Table, View, Package, Package Body, …)
- Internally there's more, but not exposed,
  see Rob van Wijk's post about DBA_DEPENDENCY_COLUMNS
  http://rwijk.blogspot.com/2008/10/dbadependencycolumns.html

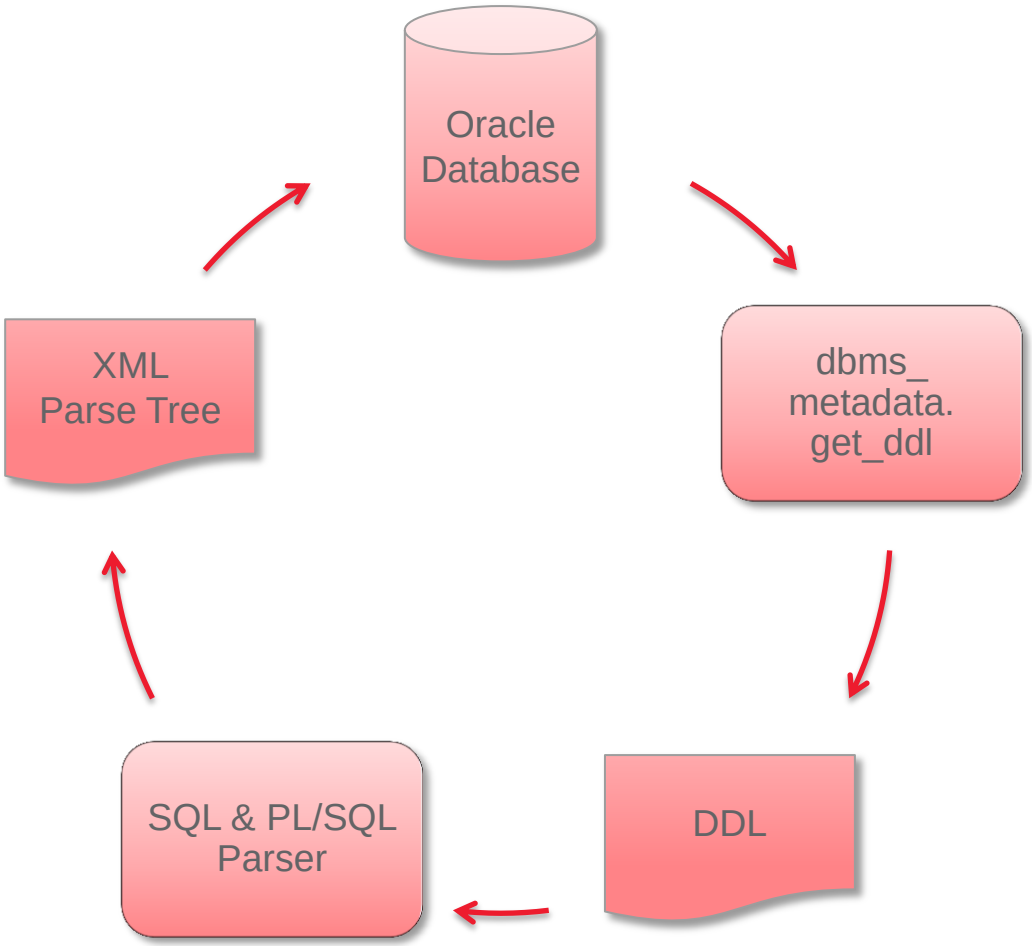
DBA_IDENTIFIERS

- PL/SQL identifier (variable, function, …)
  with usage (call, reference, …)
- Context as hierarchy of usage_id (usage_context_id = parent)
- But no support for SQL (Select, Insert, Update, Delete, Merge)

# Extend the Oracle Data Dictionary with Parsed Objects

```
SQL> desc tvd_parsed_objects_t

Name            Type
-------------- ------------------
OBJECT_ID       NUMBER
OWNER           VARCHAR2(30)
OBJECT_NAME     VARCHAR2(128)
OBJECT_TYPE     VARCHAR2(30)
LAST_DDL_TIME   DATE
DDL_SOURCE      CLOB
PARSE_TREE      XMLTYPE
```



# Related TVDCA Views

| View name | Shows usage of | Scope | Important columns |
|---|---|---|---|
| tvd_sql_paren_ident_v | Parenthesis expressions | All identifiers in captured SQL, PL/SQL | name (e.g function or type), context (e.g. package or schema) |
| tvd_sql_dot_ident_v | Dot expressions | All identifiers in captured SQL, PL/SQL | left_value, right_value |
| tvd_object_paren_ident_v | Parenthesis expressions | All identifiers in SQL, PL/SQL in stored objects | name (e.g function or type), context (e.g. package or schema) |
| tvd_object_dot_ident_v | Dot expressions | All identifiers in SQL, PL/SQL in stored objects | left_value, right_value |
| tvd_object_proc_call_v | Procedures | All PL/SQL proc./func. calls in stored objects | called_proc_name |
| tvd_object_string_usage_v | Strings | All strings in stored objects | string_value (CLOB) |

# False Negatives

- Overloading

```
PROCEDURE schreiben(p_text IN VARCHAR2);
PROCEDURE schreiben(p_log_typ IN log_type, p_text IN VARCHAR2);
```

```
FUNCTION to_integer (in_boolean IN BOOLEAN) RETURN PLS_INTEGER;
FUNCTION to_integer (in_varchar2 IN VARCHAR2) RETURN PLS_INTEGER;
```

- Ambiguous Naming
  - Different object types with same name
  - Same object names used in different contexts (e.g. same method name across multiple types)

# Query, Result and Next Steps

- Query Approach
  - Materialize TVDCA views
  - Use DBA_PROCEDURES as starting point (exclude irrelevant schemas and objects)
  - LEFT Join with TVDCA views and other references (e.g. DBA_POLICIES)
  - If non of the left joined data is found, the procedure/function is unused
  - Aggregate result on package procedure/function level
- Result
  - ~ 1'200 unused package procedures/functions found
  - False positives based on "some" dynamic SQL expected
  - Deletion is in progress by responsible teams (if no usage is found in suspected areas by the topic specialists)
  - Refine query criteria based on patterns identified by manual checks

# AGENDA

1. Background
2. Finding Unused Objects
3. Finding Unused Sub-Objects
4. Core Messages

# Core Messages

- Consider usage outside of the database when looking for unused database objects
- Establish an "ETL" process to store all information in a single repository to be combined with the Oracle Data Dictionary
- Use own and 3rd party parsers to increase accuracy of results
- Capturing runtime SQL is in most cases not feasible
- Make it a part of your development process – results are becoming better with ever iteration

# THANK YOU.

Trivadis AG

Philipp Salvisberg

Europastrasse 5
CH-8152 Glattbrugg (Zürich)

Phone +41-58-459 55 55
Fax +41-58-459 55 95

philipp.salvisberg@trivadis.com
www.trivadis.com

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MÜNCHEN STUTTGART WIEN







trivadis
makes IT easier.