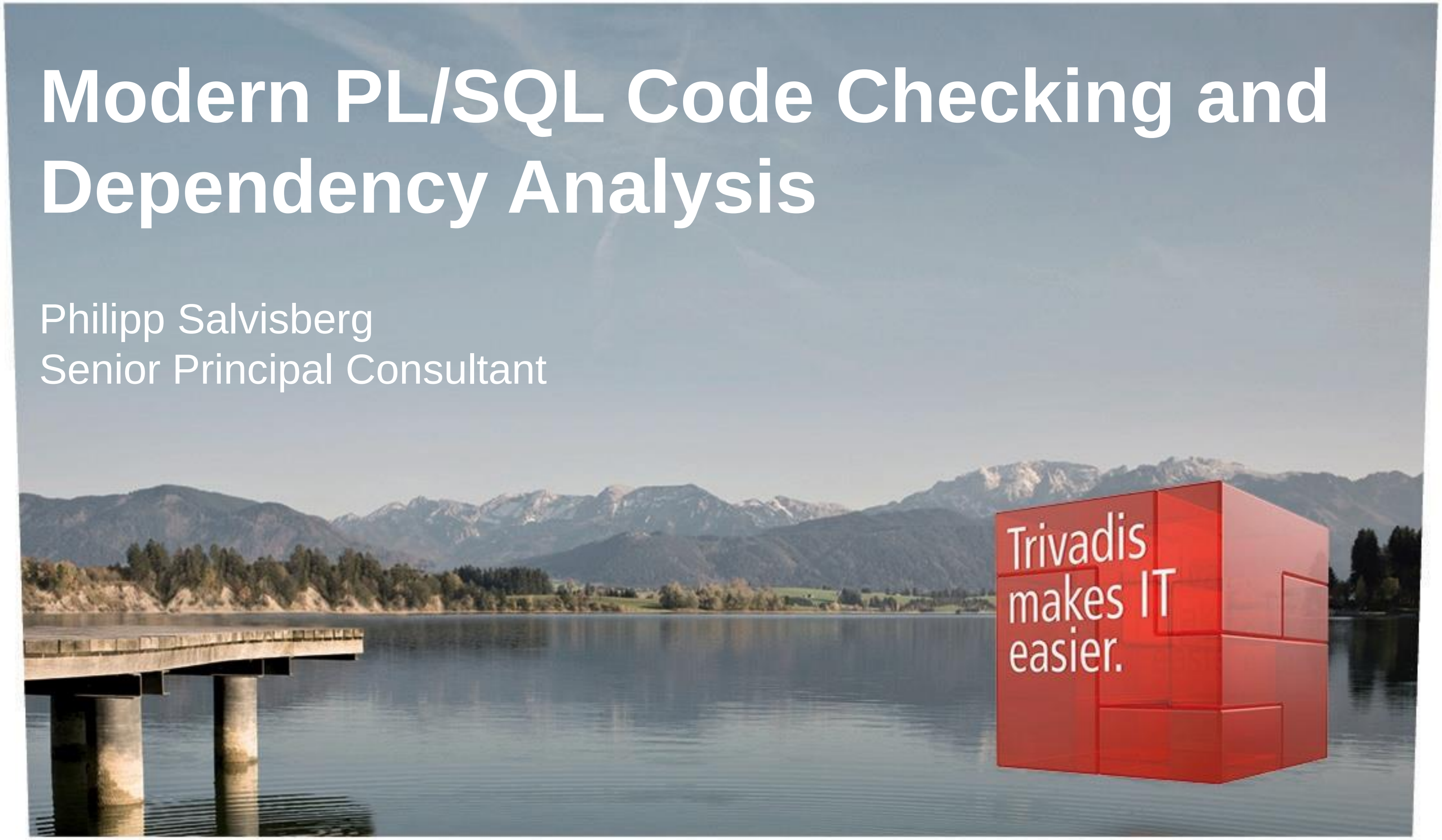


Modern PL/SQL Code Checking and Dependency Analysis

Philipp Salvisberg
Senior Principal Consultant

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA

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. Introduction
2. Grammar
3. Code Checking
4. Dependency Analysis
5. Core Messages

PL/SQL & SQL Coding Guidelines

Coding Guidelines are a crucial part of software development. It is a matter of fact, that code is more often read than written – therefore we should take efforts to ease the work of the reader, which is not necessarily the author.
I am convinced that this standard may be a good starting point for your own guidelines.

Roger Troller
Senior Consultant Trivadis

"Roger and his team have done an excellent job of providing a comprehensive set of clear standards that will undoubtedly improve the quality of your code. If you do not yet have standards in place, you should give strong consideration to using these as a starting point."

Steven Feuerstein

Steven Feuerstein
PL/SQL Evangelist

PL/SQL
& SQL

CODING GUIDELINES
VERSION 2.0

ORACLE Platinum Partner

trivadis
makes IT easier.

- Openly available since August 2009
- Download for free from www.trivadis.com

Coding Guideline #26

26. Always specify the target columns when executing an insert command.

- Reason
  - Data structures often change.
  - Having the target columns in your insert statements will lead to change-resistant code.
- Examples

```
-- Bad
INSERT INTO messages
     VALUES (l_mess_no, l_mess_typ, l_mess_text);
```

```
-- Good
INSERT INTO messages (mess_no, mess_typ, mess_text)
     VALUES (l_mess_no, l_mess_typ, l_mess_text);
```

PL/SQL Assessment

- Code Analysis based on Trivadis SQL & PL/SQL Guidelines
- Cookbook using e.g.
  - Quest CodeXpert
  - SQL Scripts using PL/Scope
  - SQL Scripts
  - Manual checks
  - Interviews
- Final Report
  - Results
  - Recommendations
- Fixed Price Offering

SWISS *IT* UP!

Trivadis AG
Europa-Strasse 5
CH-8152 Zürich (Glattbrugg)

www.trivadis.com

ORACLE Platinum Partner

EXPERIENCE *IT*

GARANTIERTE SCHWEIZER QUALITÄT –
EIN GUTES GEFÜHL!

Trivadis ist ein erfolgreiches Schweizer Unternehmen für IT-Lösungen und -Dienstleistungen mit über 15 Jahren Erfahrung und mehr als 550 Mitarbeitern in der Schweiz, Deutschland und Österreich!

Auszug aus unserer Kundenliste:
ASTRA – Bundesamt für Strassen, Bixi Systems, BMW AG, Boehringer Ingelheim, Deutsche Lufthansa, ETH Zürich, FIFA Zürich, Groupe Mutuel, Novartis Pharma, PEARL Agency GmbH, PostFinance AG, Raiffeisen, SWISS International Air Lines AG, Syngenta, UBS AG

trivadis
makes IT easier.

TRIVADIS PL/SQL ASSESSMENT

SWISS *IT* UP!

SWISS *IT* UP!

USE *IT*

UNSERE GUIDELINES – JETZT DOWNLOADEN.

Unsere «PL/SQL und SQL Coding Guidelines» ermöglichen Ihnen eine optimale und standardisierte Codierung von PL/SQL Anwendungen.

«Roger and his team have done an excellent job of providing a comprehensive set of clear standards that will undoubtedly improve the quality of your code. If you do not yet have standards in place, you should give strong consideration to using these as a starting point.»
Steven Feuerstein, PL/SQL Evangelist

Kostenloser Download unter:
www.trivadis.com/plsql

SWISS *IT* UP!

LEARN *IT*

VON DEN BESTEN LERNEN.

Profitieren Sie von unseren Best-Practice-Kursen!

Einführung in PL/SQL:
www.trivadis.com/o-pl/sql

PL/SQL für Fortgeschrittene:
www.trivadis.com/o-pl/sql-adv

Alle Kursbewertungen unserer Teilnehmer finden Sie direkt bei den Kursbeschreibungen.

SWISS *IT* UP!

KNOW *IT*

DAS GEBALLTE WISSEN UNSERER PL/SQL CRACKS.

1 Roger Troller, Senior Consultant im Oracle Umfeld, seit über 20 Jahren im IT-Business, Autor der «Trivadis PL/SQL und SQL Guidelines»

2 Perry Pakul, Technology Manager für Oracle Based Development, seit über 20 Jahren im IT-Business

3 Daniel Liebhart, Solution Manager Application Development, fokussiert auf den Bereich «Service Oriented Architecture (SOA)», seit über 25 Jahren im IT-Business

SWISS *IT* UP!

CHECK *IT* – PL/SQL

DAS ASSESSMENT FÜR IHRE PL/SQL ANWENDUNG!

Lassen Sie Ihre PL/SQL Anwendung auf Qualität, Wartbarkeit und Optimierungspotenzial checken – zum Fixpreis von CHF 5000.– / EUR 3000.–.

Unser umfassendes Assessment beruht auf unseren «PL/SQL und SQL Guidelines».

Mehr Infos unter:
www.trivadis.com/plsql

SWISS *IT* UP!

GET *IT*

PROFITIEREN SIE VON UNSEREM PL/SQL ASSESSMENT!

Ihre Vorteile:
- Klare Aussagen zur Qualität des Source Codes nach den «PL/SQL und SQL Guidelines» von Trivadis
- Klare Empfehlungen hinsichtlich qualitätssichernder Massnahmen
- Abschlussbericht mit Ergebnissen und Empfehlungen
- Deutlich einfachere Wartungs- und Weiterentwicklungsmöglichkeiten
- Steigerung der Code-Flexibilität
- Bessere Produktivität zur Bereitstellung datennaher Funktionen
- Mehr Transparenz in zentralen Applikationen

Fixpreis: CHF 5000.–/EUR 3000.–
(für maximal 10000 Lines of Code oder 3 Arbeitstage)

SWISS *IT* UP!

DO *IT*

LOS GEHT'S: NEHMEN SIE JETZT KONTAKT AUF!

www.trivadis.com/plsql

Direkter Kontakt:
Trivadis AG
Perry Pakull
assessment@trivadis.com
Tel. 0800 87 48 23 47
(Kostenfreie Rufnummer)

trivadis
makes IT easier.

Approach & Considerations

- Requirements
  - Parser to process SQL*Plus files
  - Code checking framework
- Options
  - SQL & PL/SQL Parser
    - Oracle JDeveloper Extensions (oracle.javatools.parser.plsql.PlsqlParser)
    - Free ANTLR based grammars (e.g. http://www.antlr3.org/grammar/list.html)
    - General SQL Parser
  - Sonar-Plugin
    - PL/SQL Plug-In with standard existing rules and ability for extension
  - Eclipse Xtext ✓
    - Framework for development of textual domain specific languages (DSL)
    - Uses ANTLR behind the scenes
    - Used successfully to generate database access layer for bitemporal tables

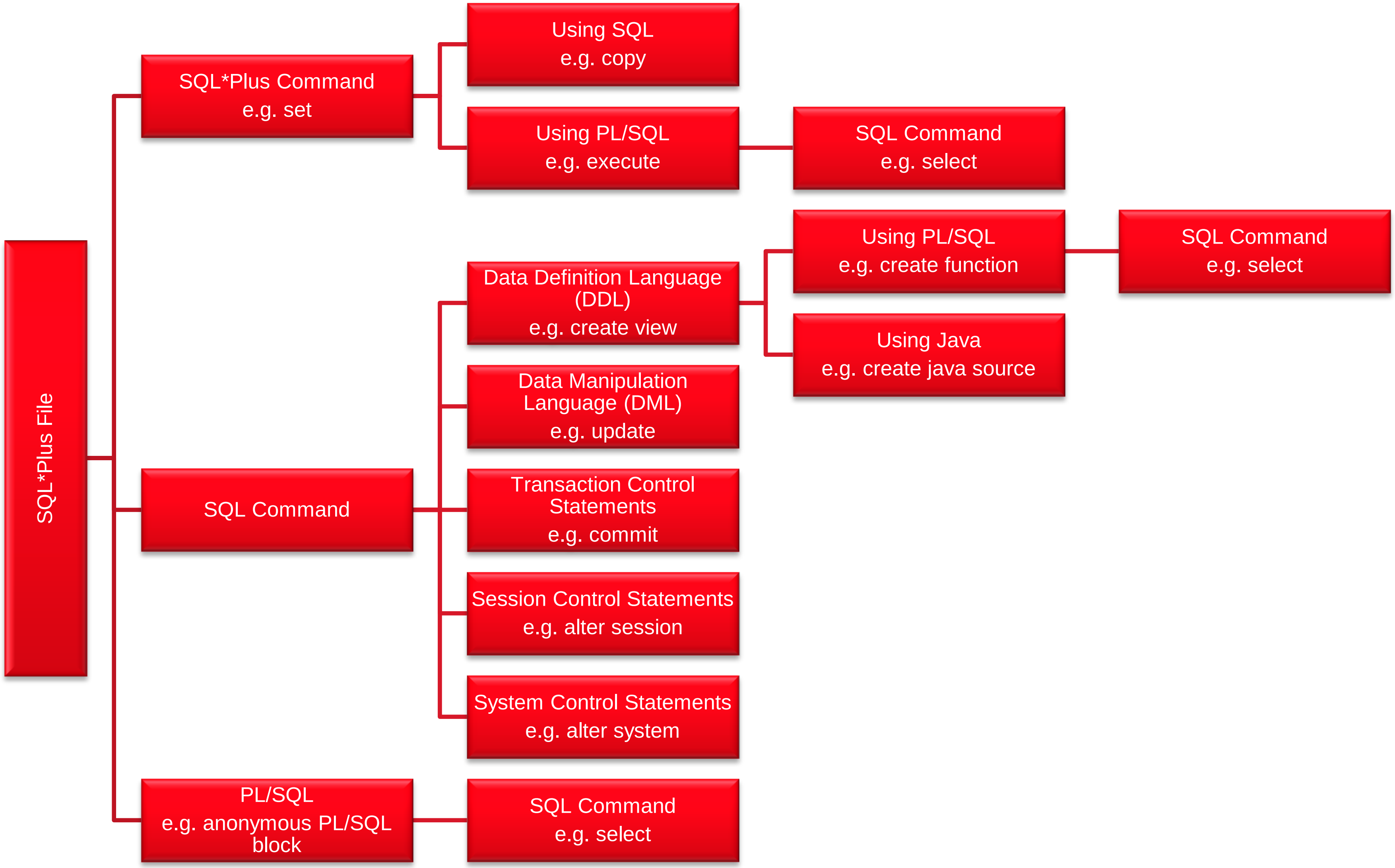
Xtext Features

- Eclipse-based Editors
  - Validation and Quick Fixes
  - Syntax Coloring
  - Code Completion
  - Outline View
  - Code Formatting
  - Bracket Matching
- Integration
  - Eclipse Modeling Framework (e.g. for graphical editors)
  - Eclipse Workbench (e.g. for list of problems/warnings)
  - Export into self-executing JAR (e.g. to build a command-line utility)

AGENDA

1. Introduction
2. Grammar
3. Code Checking
4. Dependency Analysis
5. Core Messages

Content of a SQL*Plus File


SQL*Plus File
SQL*Plus Command e.g. set
Using SQL e.g. copy
Using PL/SQL e.g. execute
SQL Command e.g. select
SQL Command
Data Definition Language (DDL) e.g. create view
Using PL/SQL e.g. create function
SQL Command e.g. select
Using Java e.g. create java source
Data Manipulation Language (DML) e.g. update
Transaction Control Statements e.g. commit
Session Control Statements e.g. alter session
System Control Statements e.g. alter system
PL/SQL e.g. anonymous PL/SQL block
SQL Command e.g. select

Complete Single Grammar Approach

- One, huge grammar (SQL*Plus, PL/SQL, SQL, Java)
- Conflicting keywords between SQL*PLUS and SQL, PL/SQL
  - "describe" is a SQL*Plus keyword, but not a reserved word in SQL (valid for table etc.)
  - Abbreviatory notation of SQL*Plus, e.g.
    - run command (r | ru | run)
    - accept command (a | ac | acc | acce | accep | accept)
- Grammar for a lot of complex commands which are not in focus for any analysis (e.g. CREATE DATABASE)
- Xtext and ANTLR cannot handle such a huge grammar
  - Maximum size of 64 KB for Java classes and methods
  - Maximum number of 65535 fields for Java classes

Reduced Single Grammar Approach

- One grammar, still huge
- Skeleton definition for less interesting commands
  - Swallow everything between start and end keywords

```
TtitleCommand: {TtitleCommand}
K_TTITLE3 text=GenericText? =>SqlPlusCmdEnd;
```

  - Necessary to avoid parse errors which would lead to incomplete analysis
- Complete definition of more interesting commands (e.g. SELECT)
- Not feasible before Xtext 2.0.1 because of generator limitations
- Still conflicting keywords between SQL*PLUS and SQL, PL/SQL

Multiple Grammar Approach

- Skeleton grammar for SQL*Plus files (SQL*Plus, SQL, PL/SQL, Java)
- Complete grammar for PL/SQL and more interesting SQL commands (e.g. CREATE VIEW)
- Chaining grammars
  - Parse SQL*Plus files using SQL*Plus parser
  - Parse PL/SQL and chosen SQL commands in SQL*Plus validator
  - Apply guidelines checks in PL/SQL validator
- No conflicting keywords between SQL*PLUS and SQL, PL/SQL

AGENDA

Excerpt of Grammar for Insert Statement

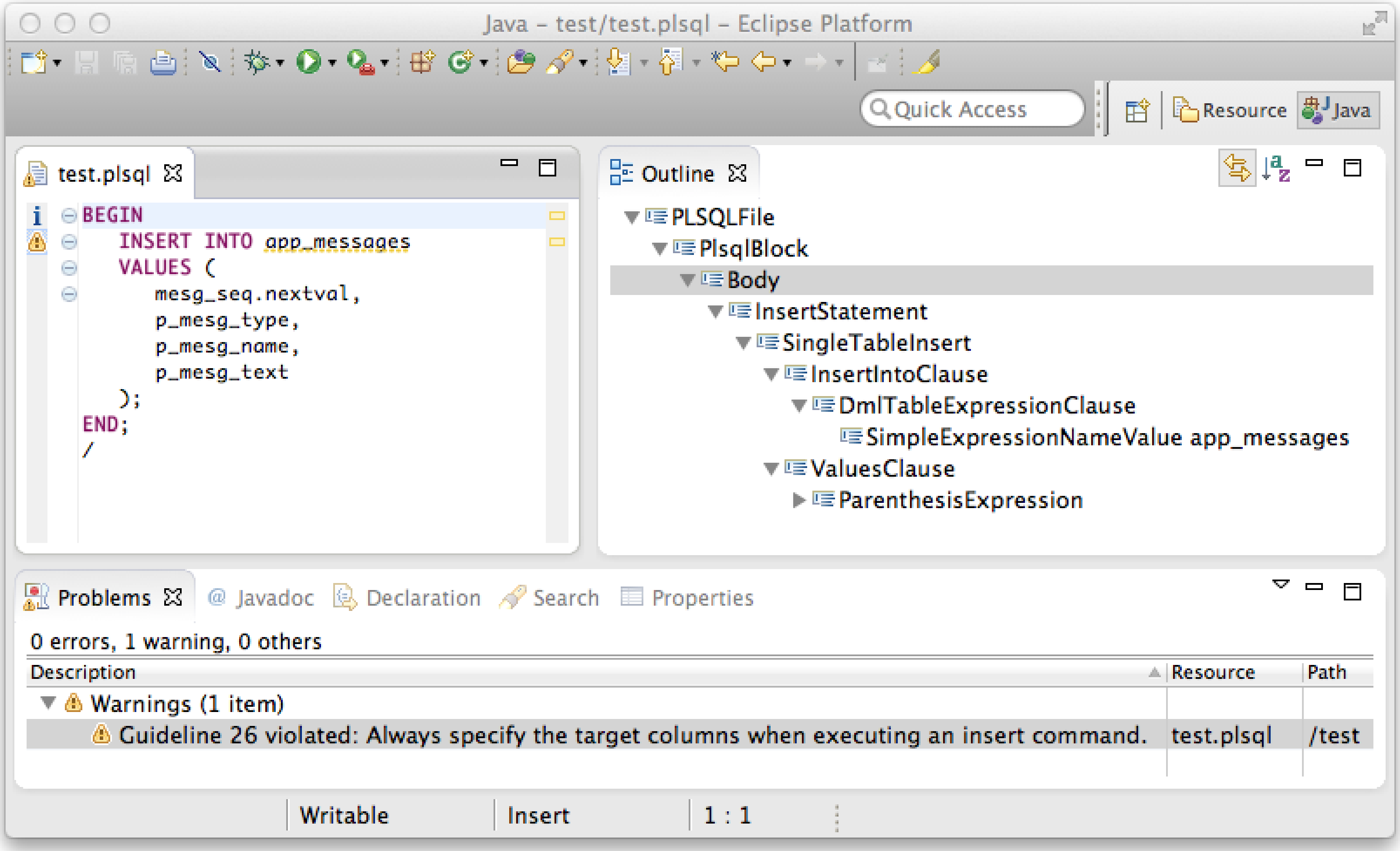
```
InsertStatement:
   InsertPlusHintsAndComments (
       singleTableInsert=SingleTableInsert
     | multiTableInsert=MultiTableInsert )
;

InsertPlusHintsAndComments returns InsertStatement hidden (WS):
   {InsertStatement} 'insert' (hints+=HintOrComment)*
;

SingleTableInsert:
   intoClause=InsertIntoClause (
       (valuesClause=ValuesClause returningClause=ReturningClause?)
     | (subquery=SelectStatement) ) errorLoggingClause=ErrorLoggingClause?
;

InsertIntoClause:
   'into' dmlExpressionClause=DmlTableExpressionClause alias=SqlNameExpression?
('(' columns+=QualifiedSqlNameExpression (',' columns+=QualifiedSqlNameExpression)* ')')?
;

ValuesClause:
   'values' expression=Expression
;
```

trivadis
makes IT easier.

Validator for Guideline #26

```
@Check
public void checkGuideline26(InsertIntoClause intoClause) {
   if (intoClause.getColumns().isEmpty()) {
      InsertStatement insert =
         EcoreUtil2.getContainerOfType(intoClause, InsertStatement.class);
      boolean ignore = false;
      SingleTableInsert singleTableInsert = insert.getSingleTableInsert();
      if (singleTableInsert != null) {
         ValuesClause valuesClause = singleTableInsert.getValuesClause();
         if (valuesClause != null) {
            Expression expr = valuesClause.getExpression();
            if (!(expr instanceof ParenthesisExpression)) {
               ignore = true; // record variable, column list not allowed!
            }
         }
      }
      if (!ignore) {
         warning(GUIDELINE_26_MSG, intoClause.getDmlExpressionClause(), null,
                 GUIDELINE_26, serialize(NodeModelUtils.getNode(insert).getParent()));
      }
   }
}
```

Eclipse Editor



```
BEGIN
    INSERT INTO app_messages
    VALUES (
        mesg_seq.nextval,
        p_mesg_type,
        p_mesg_name,
        p_mesg_text
    );
END;
/
```

```
▼ PLSQLFile
  ▼ PlsqlBlock
    ▼ Body
      ▼ InsertStatement
        ▼ SingleTableInsert
          ▼ InsertIntoClause
            ▼ DmlTableExpressionClause
                SimpleExpressionNameValue app_messages
          ▼ ValuesClause
            ▶ ParenthesisExpression
```

0 errors, 1 warning, 0 others

Description	Resource	Path
▼ Warnings (1 item)		
Guideline 26 violated: Always specify the target columns when executing an insert command.	test.plsql	/test

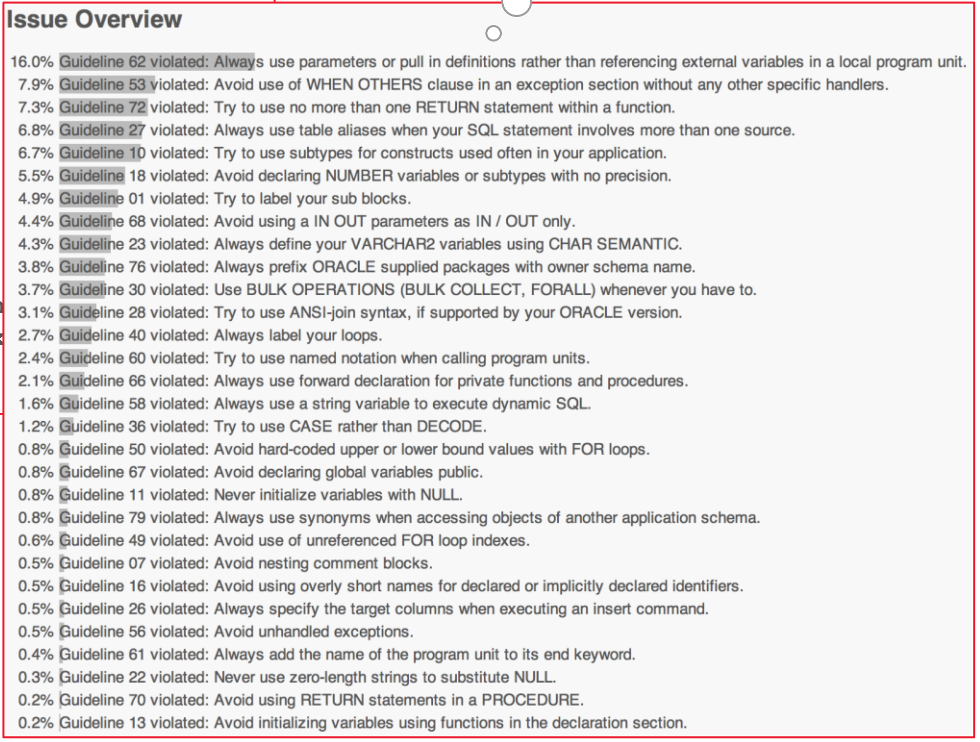
Writable | Insert | 1 : 1

TVDCC – Command Line Interface

```
processing file 'ESC_USER_ADMIN.pkb'... 346 issues found.
processing file 'ESC_USER_ADMIN.pks'... 129 issues found.
processing file 'ESC_USER_ADMIN_DEFAULT.pkb'... 30 issues found.
processing file 'ESC_USER_ADMIN_DEFAULT.pks'... 6 issues found.
processing file 'ESC_UTIL.pkb'... 193 issues found.
processing file 'ESC_UTIL.pks'... no issues found.

Summary:
- Total files: 42
- Total bytes: 1806872
- Total lines: 32062
- Total commands: 5322
- Total statements (PL/SQL): 5800
- Max. cyclomatic complexity: 140
- Total issues: 4668
- Total warnings: 4667
- Total errors: 1
- Total processing time in seconds: 23.496

transforming tvdcc_report.xml into tvdcc_report.h
transforming tvdcc_report.xml into tvdcc_report.x
cleanup completed.
```

XML, HTML, Excel Output

Console



Issue Overview

16.0% Guideline 62 violated: Always use parameters or pull in definitions rather than referencing external variables in a local program unit.
7.9% Guideline 53 violated: Avoid use of WHEN OTHERS clause in an exception section without any other specific handlers.
7.3% Guideline 72 violated: Try to use no more than one RETURN statement within a function.
6.8% Guideline 27 violated: Always use table aliases when your SQL statement involves more than one source.
6.7% Guideline 10 violated: Try to use subtypes for constructs used often in your application.
5.5% Guideline 18 violated: Avoid declaring NUMBER variables or subtypes with no precision.
4.9% Guideline 01 violated: Try to label your sub blocks.
4.4% Guideline 68 violated: Avoid using a IN OUT parameters as IN / OUT only.
4.3% Guideline 23 violated: Always define your VARCHAR2 variables using CHAR SEMANTIC.
3.8% Guideline 76 violated: Always prefix ORACLE supplied packages with owner schema name.
3.7% Guideline 30 violated: Use BULK OPERATIONS (BULK COLLECT, FORALL) whenever you have to.
3.1% Guideline 28 violated: Try to use ANSI-join syntax, if supported by your ORACLE version.
2.7% Guideline 40 violated: Always label your loops.
2.4% Guideline 60 violated: Try to use named notation when calling program units.
2.1% Guideline 66 violated: Always use forward declaration for private functions and procedures.
1.6% Guideline 58 violated: Always use a string variable to execute dynamic SQL.
1.2% Guideline 36 violated: Try to use CASE rather than DECODE.
0.8% Guideline 50 violated: Avoid hard-coded upper or lower bound values with FOR loops.
0.8% Guideline 67 violated: Avoid declaring global variables public.
0.8% Guideline 11 violated: Never initialize variables with NULL.
0.8% Guideline 79 violated: Always use synonyms when accessing objects of another application schema.
0.6% Guideline 49 violated: Avoid use of unreferenced FOR loop indexes.
0.5% Guideline 07 violated: Avoid nesting comment blocks.
0.5% Guideline 16 violated: Avoid using overly short names for declared or implicitly declared identifiers.
0.5% Guideline 26 violated: Always specify the target columns when executing an insert command.
0.5% Guideline 56 violated: Avoid unhandled exceptions.
0.4% Guideline 61 violated: Always add the name of the program unit to its end keyword.
0.3% Guideline 22 violated: Never use zero-length strings to substitute NULL.
0.2% Guideline 70 violated: Avoid using RETURN statements in a PROCEDURE.
0.2% Guideline 13 violated: Avoid initializing variables using functions in the declaration section.

trivadis makes IT easier.

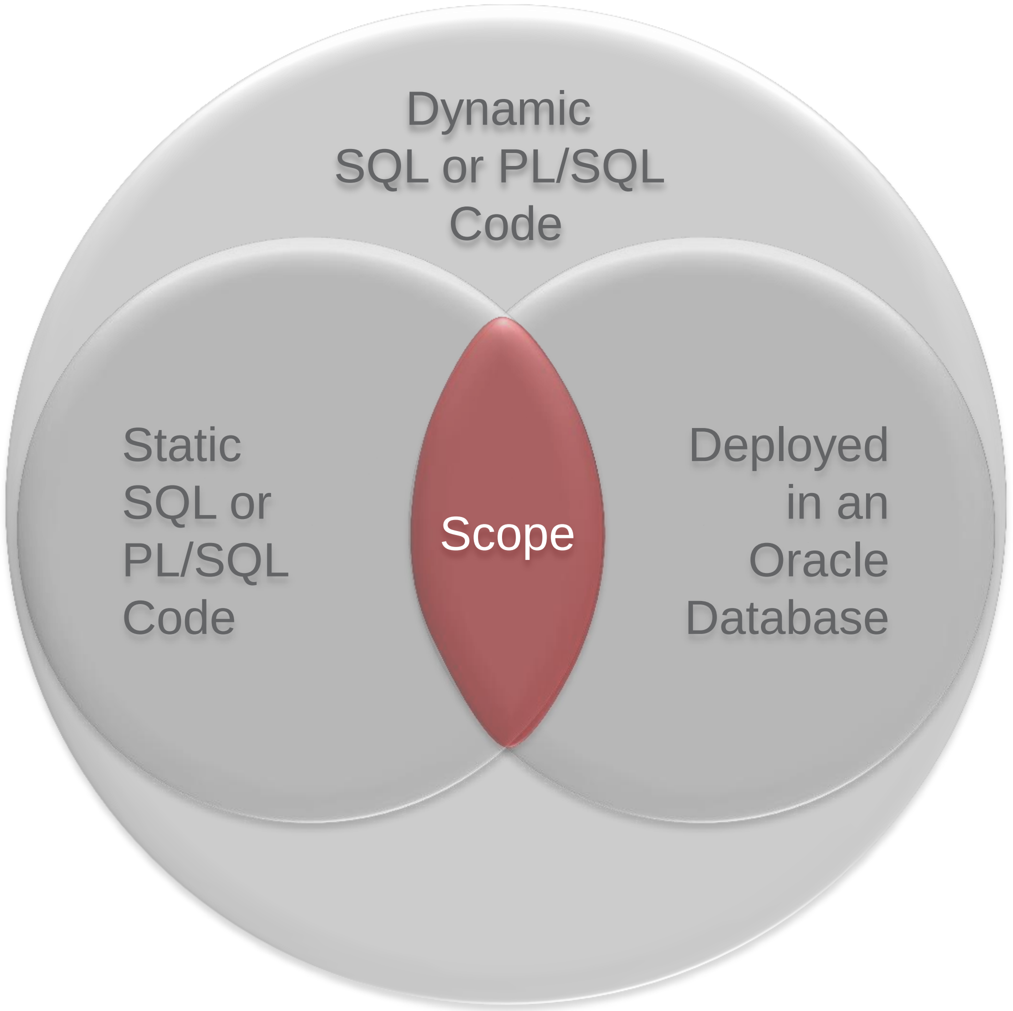
AGENDA

1. Introduction
2. Grammar
3. Code Checking
4. Dependency Analysis
5. Core Messages

Motivation

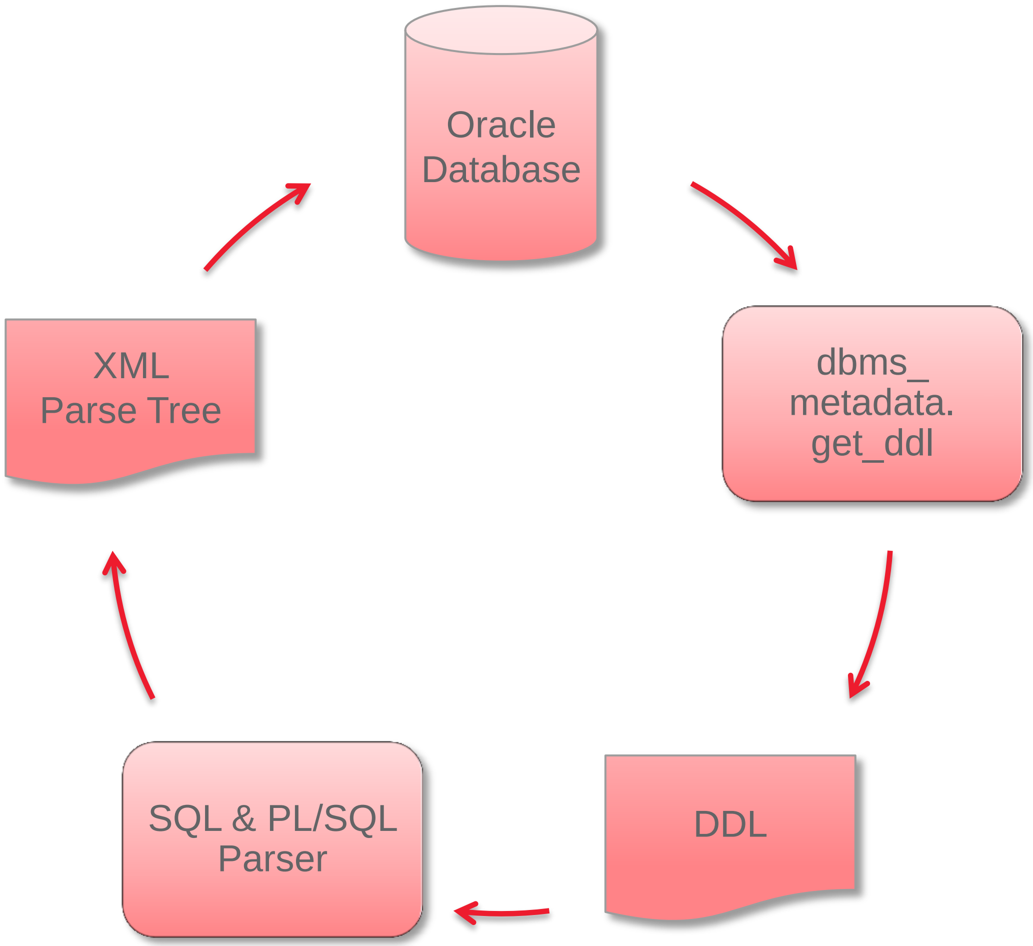
- Find Object/Subobject Usage
  - Table/view statements (e.g. Select, Insert, Update, Delete, Merge)
  - Package procedures/functions
  - Type methods
  - Table/view columns
- Manage Accessibility of Sensitive Columns
  - Client identifying data
  - Turnover, cost per order/customer, gross margin, discount
  - Nested views, named queries, subqueries – column name changes
- Estimate Impact of Software Design Changes
  - Aspect of design decisions
  - Identify impacted modules (e.g. for testing purposes)

Scope of Database Dependency Analysis


Dynamic SQL or PL/SQL Code
Static SQL or PL/SQL Code
Scope
Deployed in an Oracle Database

Extend the Oracle Data Dictionary

```
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
```



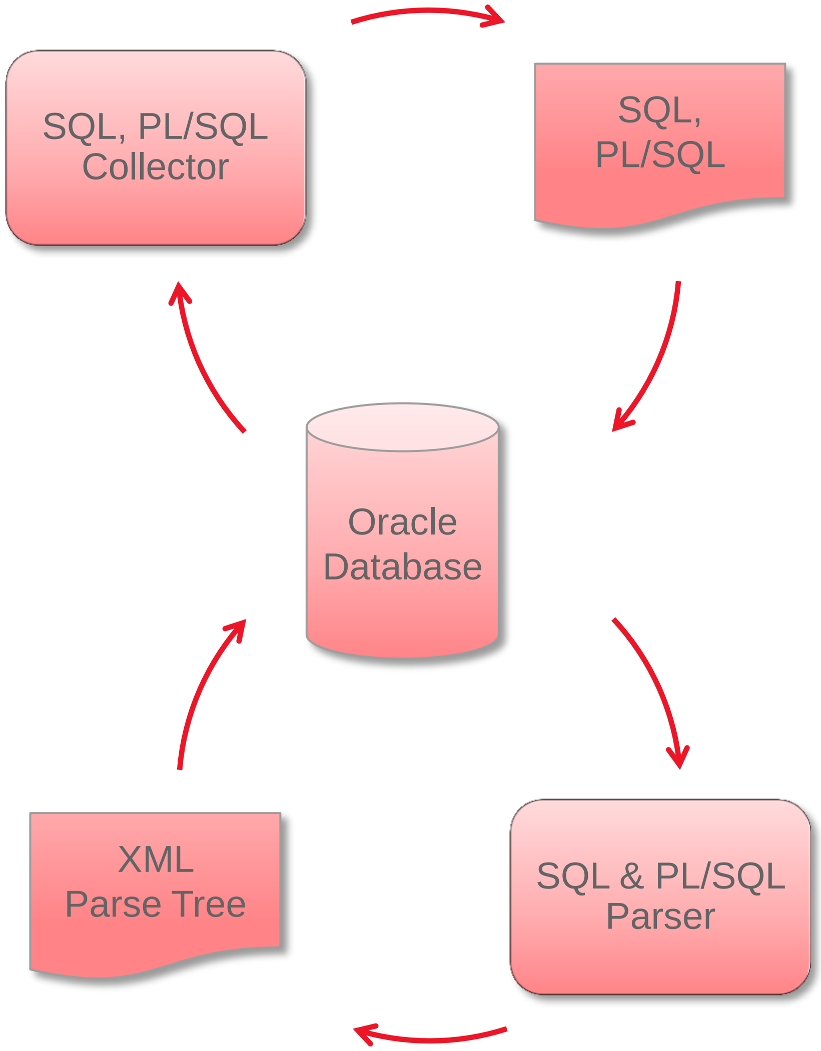
TVDCA – Tables, Views used in DML Statements

- Usage within Function, Procedure, Trigger, Package Spec/Body, Type Body
  - Consider Insert, Update, Delete, Merge statements
  - Procedure_name semantics according DBA_PROCEDURES
- Example

```
SQL> SELECT object_type, object_name, operation AS op, procedure_name,
  2         table_owner AS t_own, table_name
  3    FROM tvd_object_dml_usage_v t
  4   WHERE owner = 'TVDCC';

OBJECT_TYPE  OBJECT_NAME             OP     PROCEDURE_NAME       T_OWN TABLE_NAME
------------ ---------------------- ------ -------------------- ----- --------------------
PACKAGE BODY TVD_PARSED_OBJECTS_PKG INSERT REFRESH              TVDCC TVD_PARSED_OBJECTS_T
FUNCTION     TVD_SAMPLE_FUNCTION    INSERT INNER_PROCEDURE      TVDCC TVD_PARSED_OBJECTS_T
FUNCTION     TVD_SAMPLE_FUNCTION    INSERT INNER_FUNCTION       TVDCC TVD_PARSED_OBJECTS_T
FUNCTION     TVD_SAMPLE_FUNCTION    INSERT                      TVDCC TVD_PARSED_OBJECTS_T
PACKAGE BODY TVD_SAMPLE_PACKAGE     INSERT MOST_INNER_PROCEDURE TVDCC TVD_PARSED_OBJECTS_T
```

Extend the Scope of Dependency Analysis

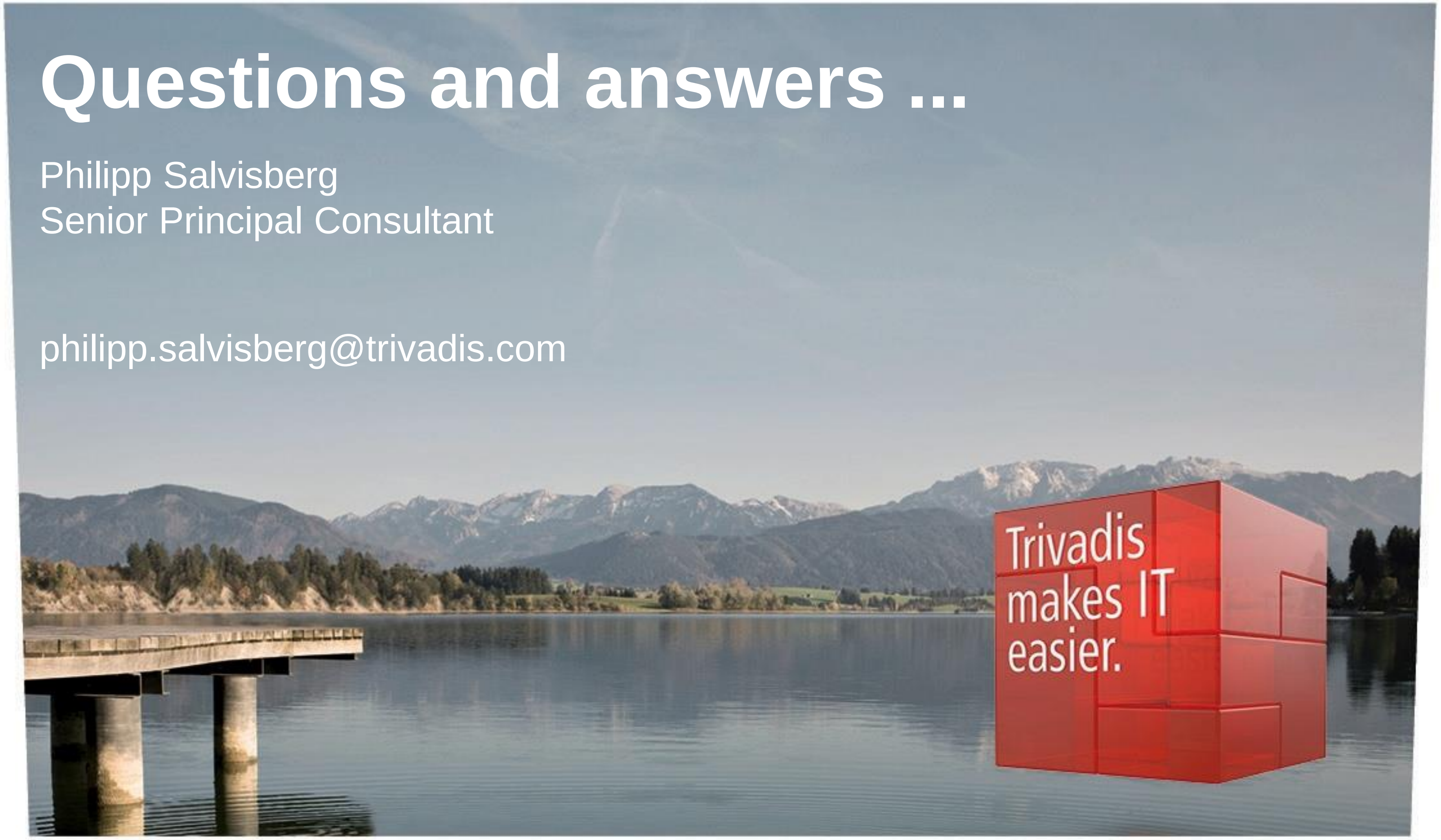
```
SQL> desc tvd_captured_sql_t

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
```



AGENDA

Core Messages

- Writing a SQL*Plus parser is laborious
- Writing a validator is simple
- Extend the Oracle Data Dictionary for Dependency analysis and consider code not deployed in the database
- Xtext is a complete DSL framework
  - More than just a parser generator
  - Separation of parser and validators
  - Promising for further applications like code fixing, code formatting, presenting graphical models, etc.



Questions and answers ...

Philipp Salvisberg
Senior Principal Consultant

philipp.salvisberg@trivadis.com

BASEL BERN BRUGG LAUSANNE ZÜRICH DÜSSELDORF FRANKFURT A.M. FREIBURG I.BR. HAMBURG MUNICH STUTTGART VIENNA