WELCOME

SAN ANTONIO, TEXAS ✱ JUNE 24-28

Modern PL/SQL Code Checking and Dependency Analysis

Philipp Salvisberg

25th July 2012

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

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.trivadis.com
- Member of the trivadis performanceteam

Hamburg
Dusseldorf
~200 employees
Frankfurt
Stuttgart
Vienna
Freiburg
Munich
Basel
Zurich
~30 employees
Bern
Lausanne
~380 employees

- 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. Xtext Live – Parsing & Validating
3. Finalizing Grammar, Checks and Tooling
4. Dependency Analysis
5. Challenges
6. Conclusion

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

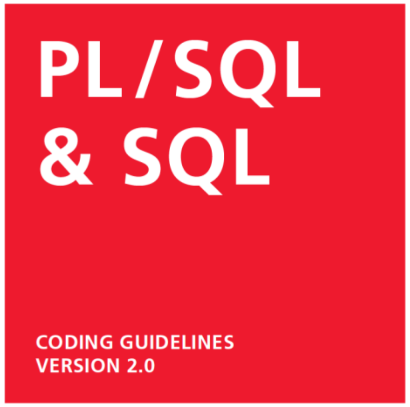
Steven Feuerstein
PL/SQL Evangelist

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

PL/SQL & SQL
CODING GUIDELINES VERSION 2.0
ORACLE Platinum Partner
trivadis makes IT easier.

See http://www.trivadis.com/technologie/oracle/oracle-application-development/oracle-sql-und-plsql.html

Trivadis PL/SQL & SQL Guideline #26


PL/SQL
& SQL

CODING GUIDELINES
VERSION 2.0

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.

Example:

```
-- 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!

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

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.

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

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

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)

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)

Shortcoming of PL/SQL Assessment

- Some guidelines check scripts need manual post-processing
- Some guidelines checks are not automated at all
- One snapshot – Assessment of a defined release
- Repetitive execution is time-consuming, expensive, not feasible
- Not part of an automated, continuous integration strategy

Goal

- Fully automated code checking
- Considering the Trivadis PL/SQL & SQL Guidelines
- Extendable and adaptable to suit customer needs
- Part of an automated build process

Approach & Considerations

- Requirements
  - Parser to process SQL*Plus files
  - Code checking framework
- Options
  - SQL & PL/SQL grammar as part of Oracle JDeveloper Extensions
    - http://www.oracle.com/technetwork/developer-tools/jdev/index-099997.html, see class oracle.javatools.parser.plsql.PlsqlParser
    - Required libraries (javatools-nodeps.jar) are part of SQL Developer
  - ANTLR
    - Several SQL & PL/SQL grammars on http://www.antlr.org/grammar/list
  - Eclipse Xtext
    - Framework for development of textual domain specific languages (DSL)
    - Used successfully to generate database access layer for bitemporal tables
    - Uses ANTLR behind the scenes

Xtext Features

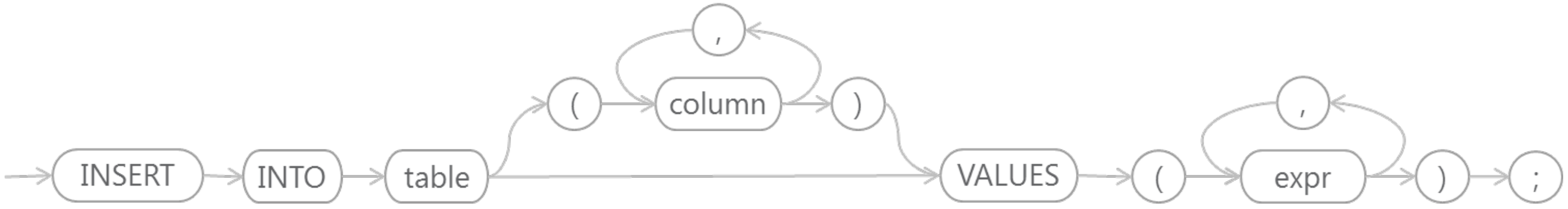
- Eclipse-based Editors
  - Validation and Quick Fixes
  - Syntax Coloring
  - Code Completion
  - Outline View
  - Code Formatting
  - Bracket Matching
- Integration
  - Eclipse Modeling Framework (e.g. for graphical editors)
  - Eclipse Workbench (e.g. for list of problems/warnings)
  - Export into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

AGENDA

1. Introduction
2. Xtext Live – Parsing & Validating
3. Finalizing Grammar, Checks and Tooling
4. Dependency Analysis
5. Challenges
6. Conclusion

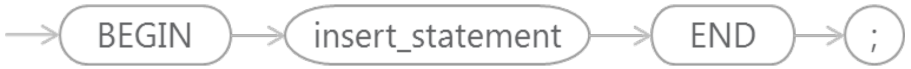
Demo Grammar (BNF)

insert_statement::=



```
INSERT INTO table [( column [, column ]... )]
VALUES ( expr [, expr ]... ) ;
```

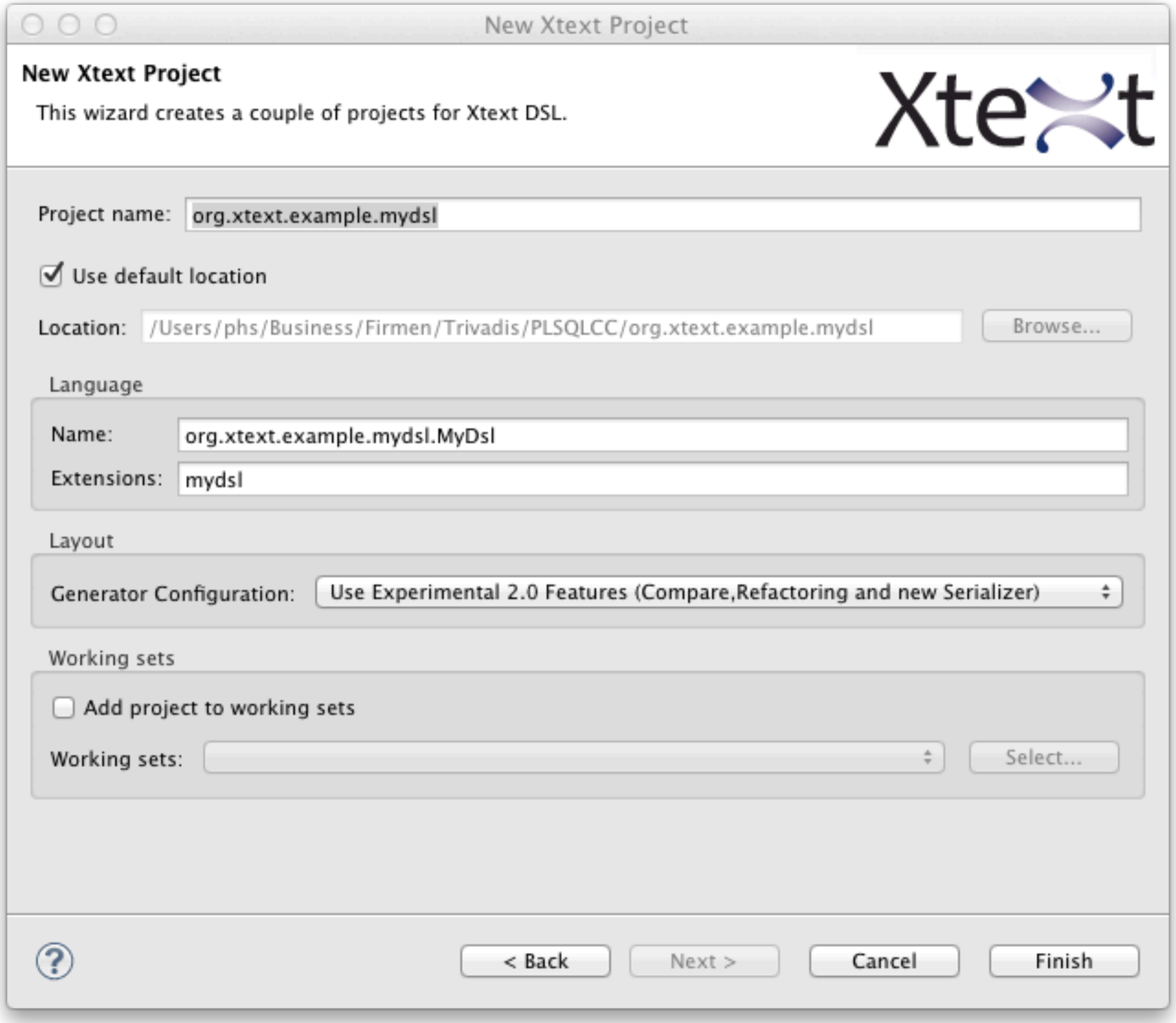
plsql_unit::=



```
BEGIN insert_statement END ;
```

Default Xtext Project

DEMO


New Xtext Project

New Xtext Project

This wizard creates a couple of projects for Xtext DSL.

Project name: org.xtext.example.mydsl

☑ Use default location

Location: /Users/phs/Business/Firmen/Trivadis/PLSQLCC/org.xtext.example.mydsl

Language

Name: org.xtext.example.mydsl.MyDsl

Extensions: mydsl

Layout

Generator Configuration: Use Experimental 2.0 Features (Compare,Refactoring and new Serializer)

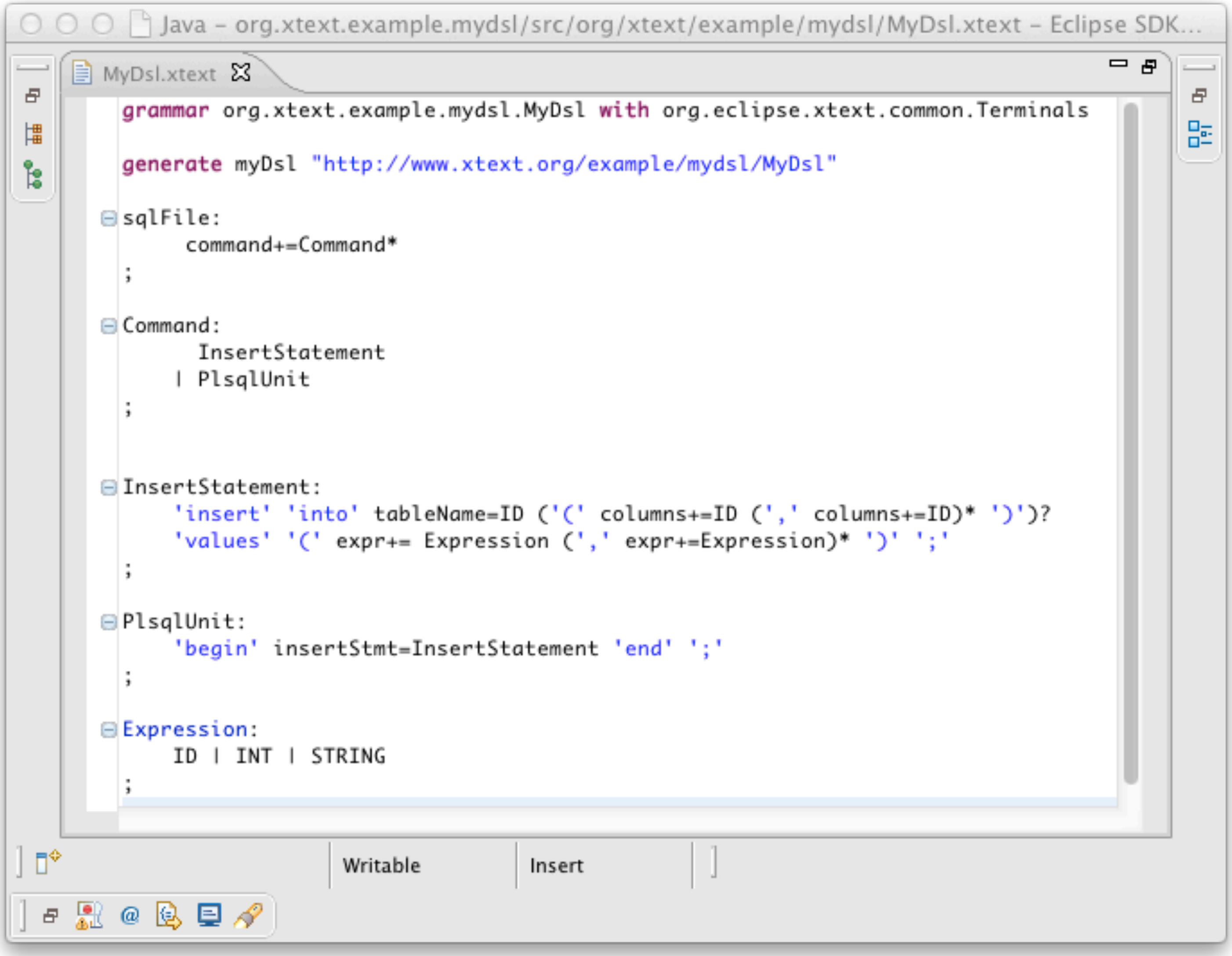
Working sets

☐ Add project to working sets

Working sets:

Demo Grammar (Xtext)

DEMO

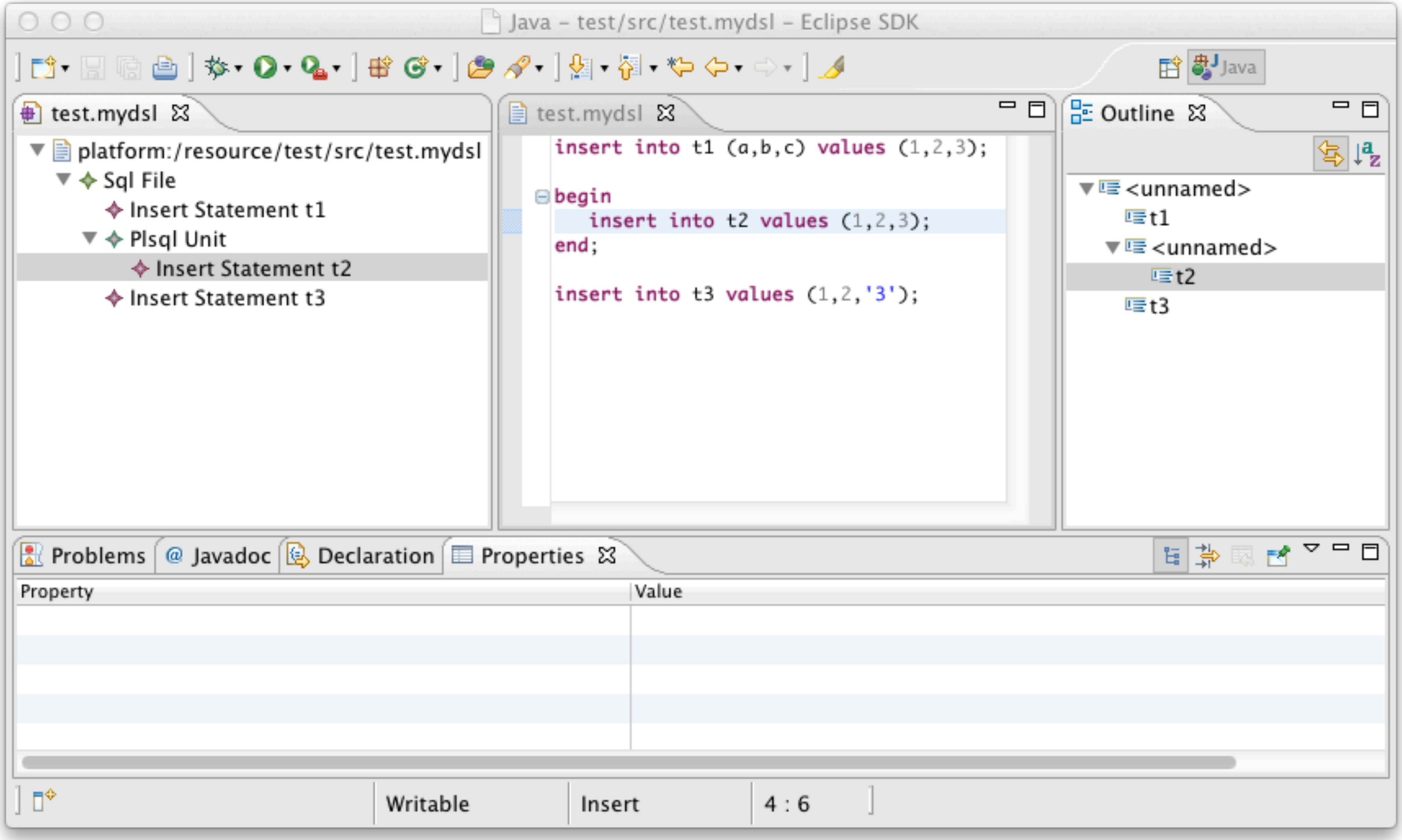

Java – org.xtext.example.mydsl/src/org/xtext/example/mydsl/MyDsl.xtext – Eclipse SDK...
MyDsl.xtext
Writable
Insert

```
grammar org.xtext.example.mydsl.MyDsl with org.eclipse.xtext.common.Terminals

generate myDsl "http://www.xtext.org/example/mydsl/MyDsl"

sqlFile:
     command+=Command*
;

Command:
      InsertStatement
    | PlsqlUnit
;


InsertStatement:
    'insert' 'into' tableName=ID ('(' columns+=ID (',' columns+=ID)* ')')?
    'values' '(' expr+= Expression (',' expr+=Expression)* ')' ';'
;

PlsqlUnit:
    'begin' insertStmt=InsertStatement 'end' ';'
;

Expression:
    ID | INT | STRING
;
```

ODTUG Kscope12 trivadis makes IT easier.

Eclipse Editors

DEMO

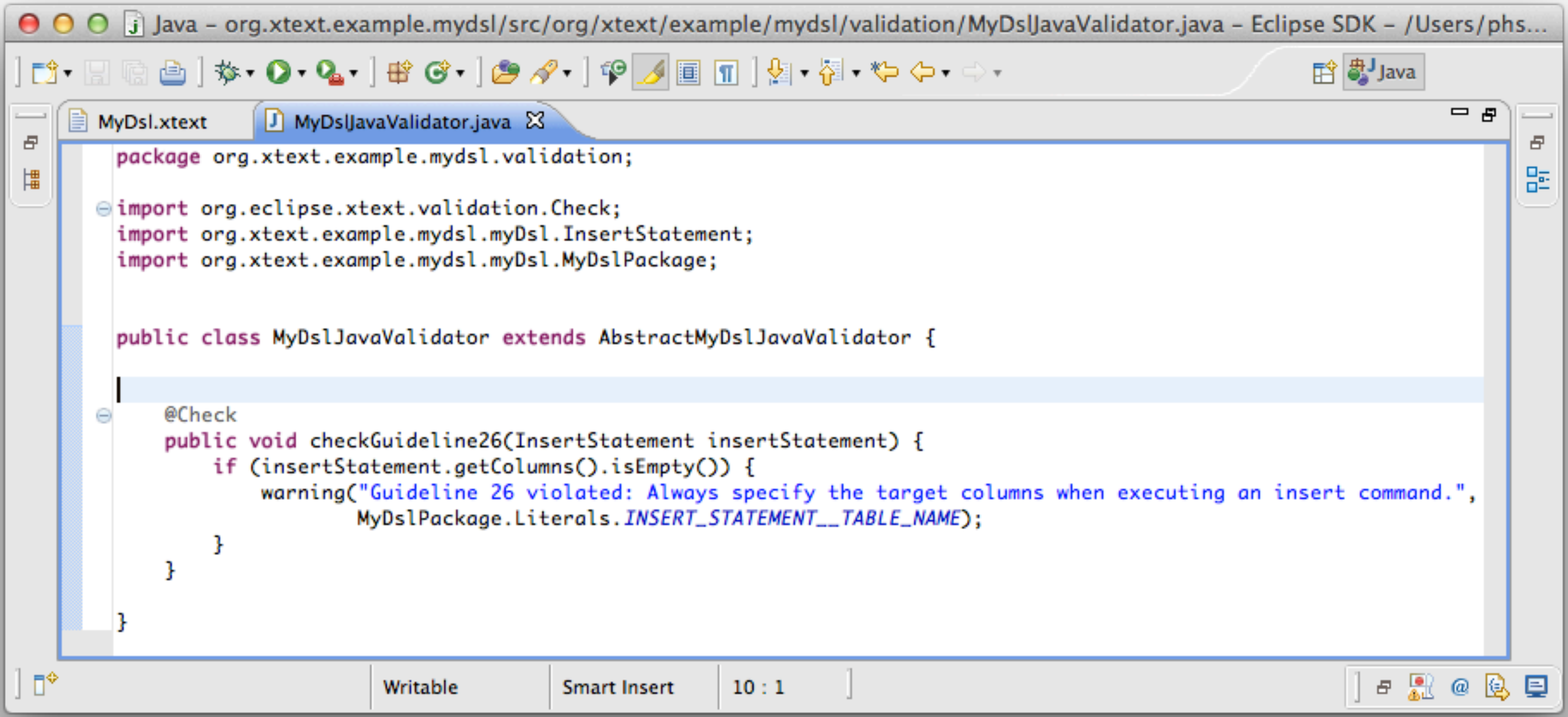

Java – test/src/test.mydsl – Eclipse SDK

```
insert into t1 (a,b,c) values (1,2,3);

begin
   insert into t2 values (1,2,3);
end;

insert into t3 values (1,2,'3');
```

ODTUG Kscope12 trivadis makes IT easier.

Validator for Guideline #26

DEMO


Java – org.xtext.example.mydsl/src/org/xtext/example/mydsl/validation/MyDslJavaValidator.java – Eclipse SDK – /Users/phs...
Java
MyDsl.xtext
MyDslJavaValidator.java
Writable
Smart Insert
10 : 1

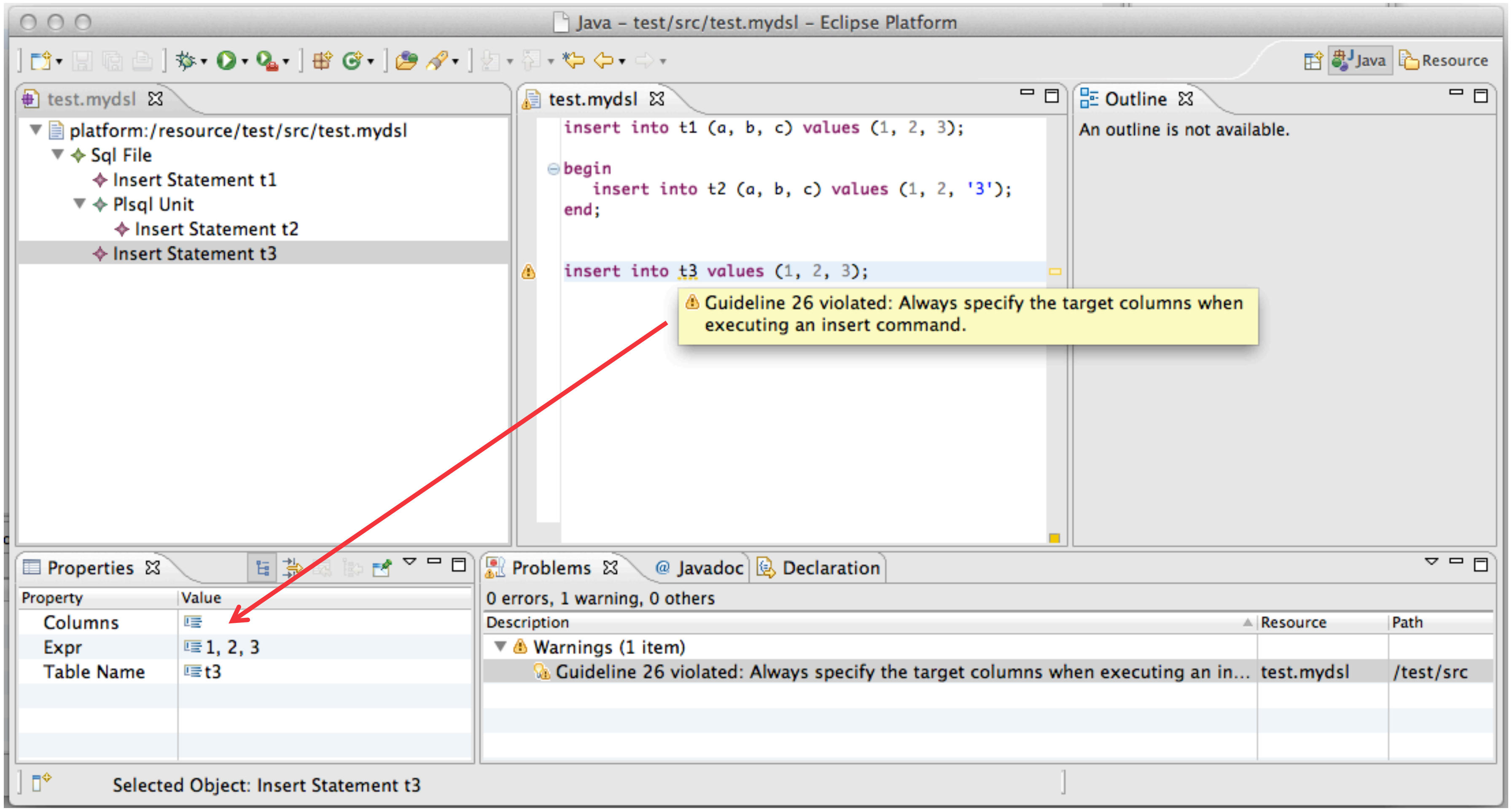
```java
package org.xtext.example.mydsl.validation;

import org.eclipse.xtext.validation.Check;
import org.xtext.example.mydsl.myDsl.InsertStatement;
import org.xtext.example.mydsl.myDsl.MyDslPackage;


public class MyDslJavaValidator extends AbstractMyDslJavaValidator {

    @Check
    public void checkGuideline26(InsertStatement insertStatement) {
        if (insertStatement.getColumns().isEmpty()) {
            warning("Guideline 26 violated: Always specify the target columns when executing an insert command.",
                    MyDslPackage.Literals.INSERT_STATEMENT__TABLE_NAME);
        }
    }

}
```

ODTUG Kscope12

trivadis makes IT easier.

Validator in Action

DEMO



Java – test/src/test.mydsl – Eclipse Platform

test.mydsl

- platform:/resource/test/src/test.mydsl
  - Sql File
    - Insert Statement t1
    - Plsql Unit
      - Insert Statement t2
    - Insert Statement t3

```
insert into t1 (a, b, c) values (1, 2, 3);

begin
   insert into t2 (a, b, c) values (1, 2, '3');
end;


insert into t3 values (1, 2, 3);
```

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

Outline

An outline is not available.

Properties

Property	Value
Columns	
Expr	1, 2, 3
Table Name	t3

Problems | Javadoc | Declaration

0 errors, 1 warning, 0 others

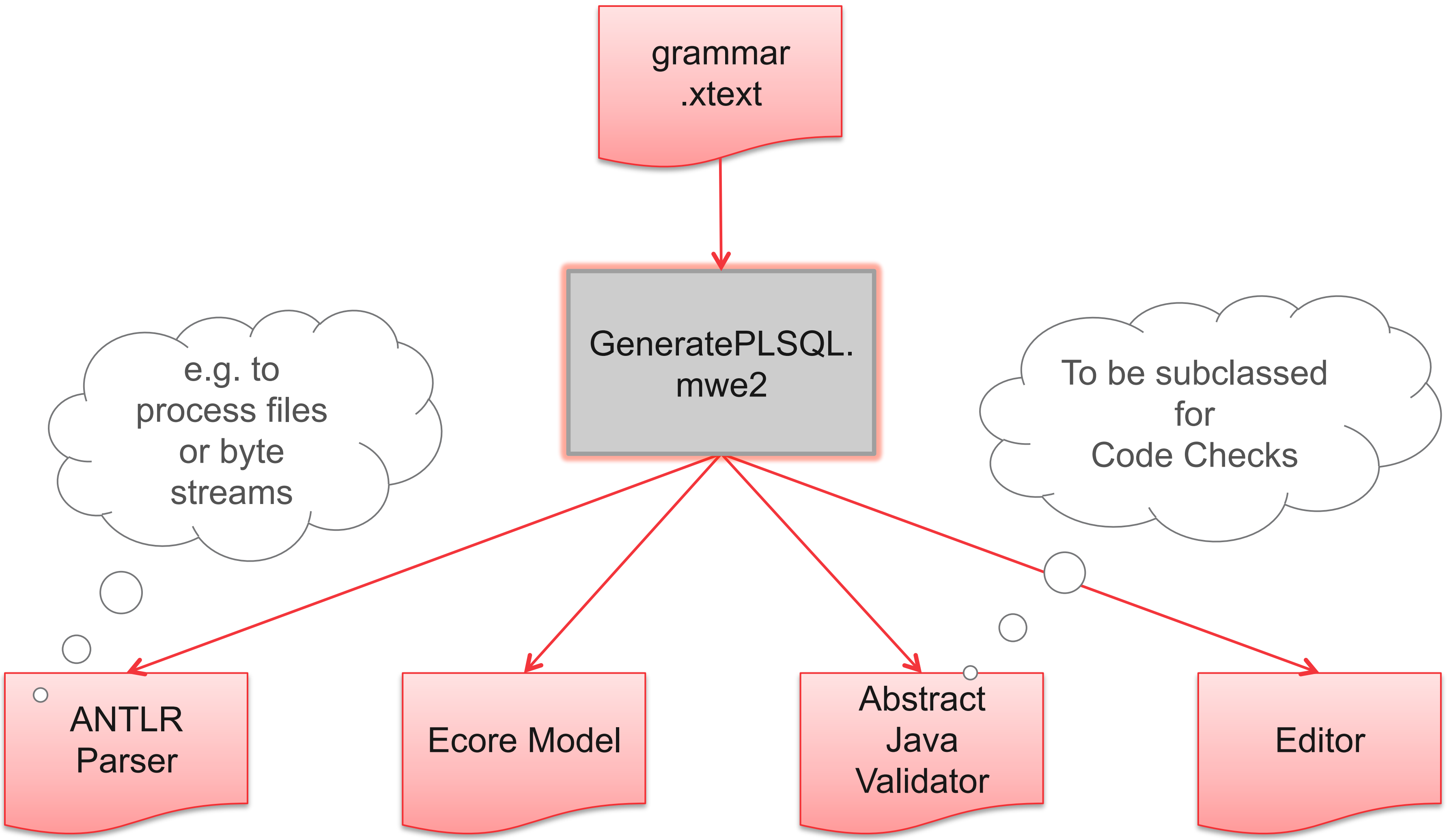
Description	Resource	Path
Warnings (1 item)		
Guideline 26 violated: Always specify the target columns when executing an in...	test.mydsl	/test/src

Selected Object: Insert Statement t3

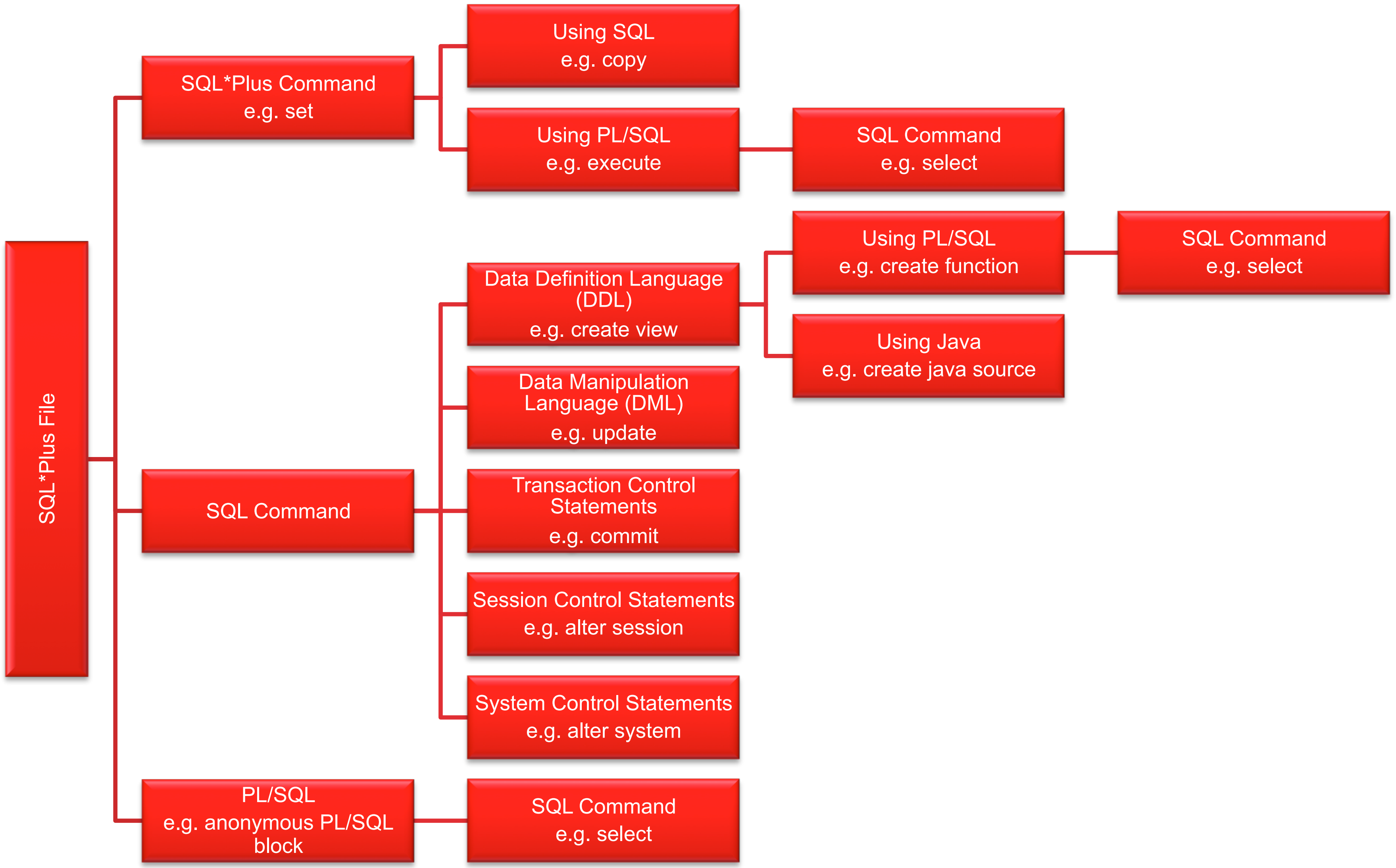
ODTUG Kscope12 trivadis makes IT easier.

AGENDA

Generate Grammar via Xtext



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

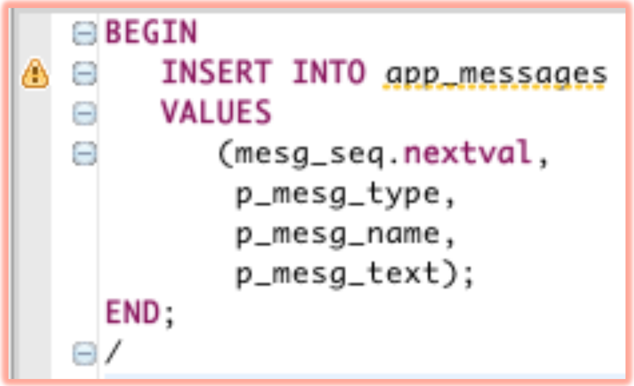
```
TtitleCommand: {TtitleCommand}
K_TTITLE3 text=GenericText? =>SqlPlusCmdEnd;
```

  - 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

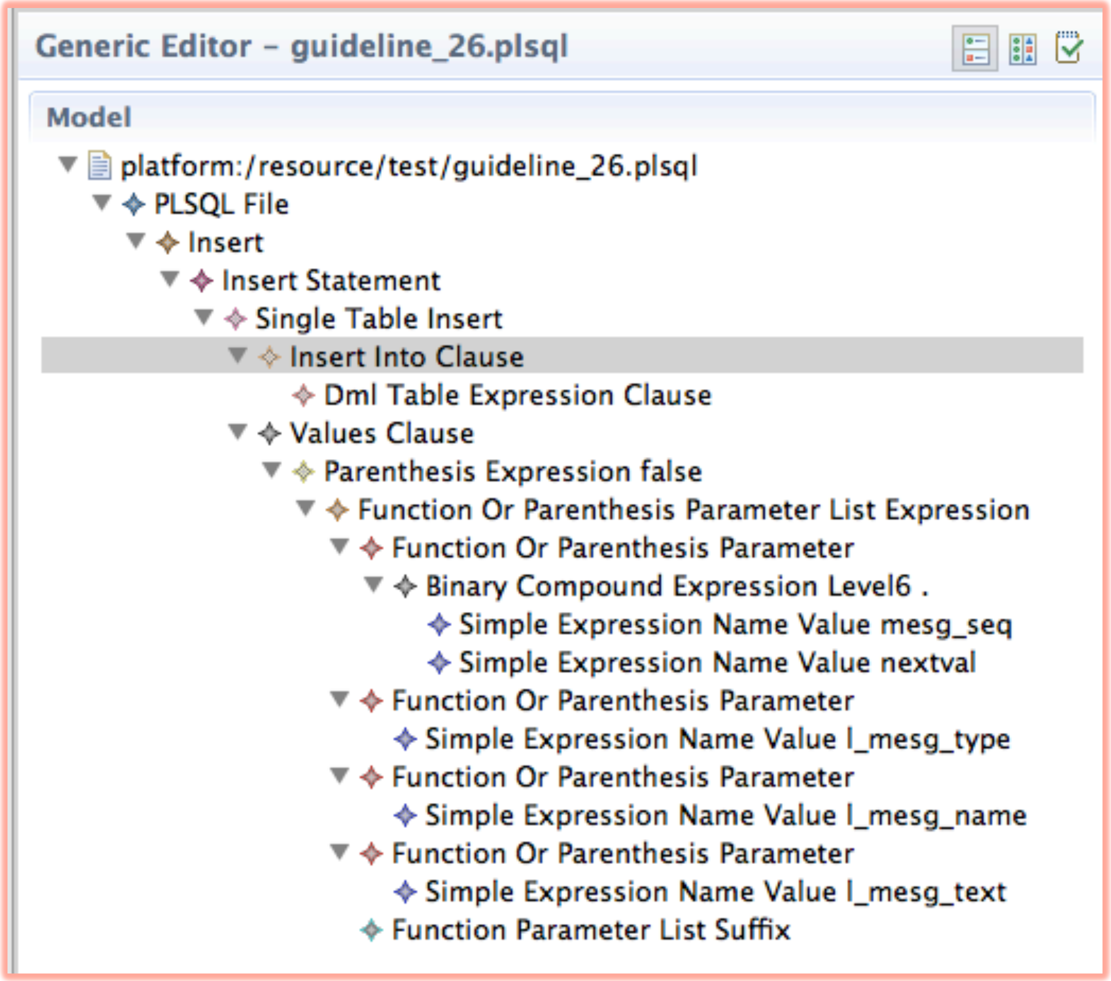
- Skeleton grammar for SQL*Plus files (SQL*Plus, SQL, PL/SQL, Java)
- Complete grammar for PL/SQL and more interesting SQL commands (e.g. CREATE VIEW)
- Chaining grammars
  - Parse SQL*Plus files using SQL*Plus parser
  - Parse PL/SQL and chosen SQL commands in SQL*Plus validator
  - Apply guidelines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

Source, Model & Warning for Guideline #26



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

line 2 - Guideline 26 violated: Always specify the target columns when executing an insert command.

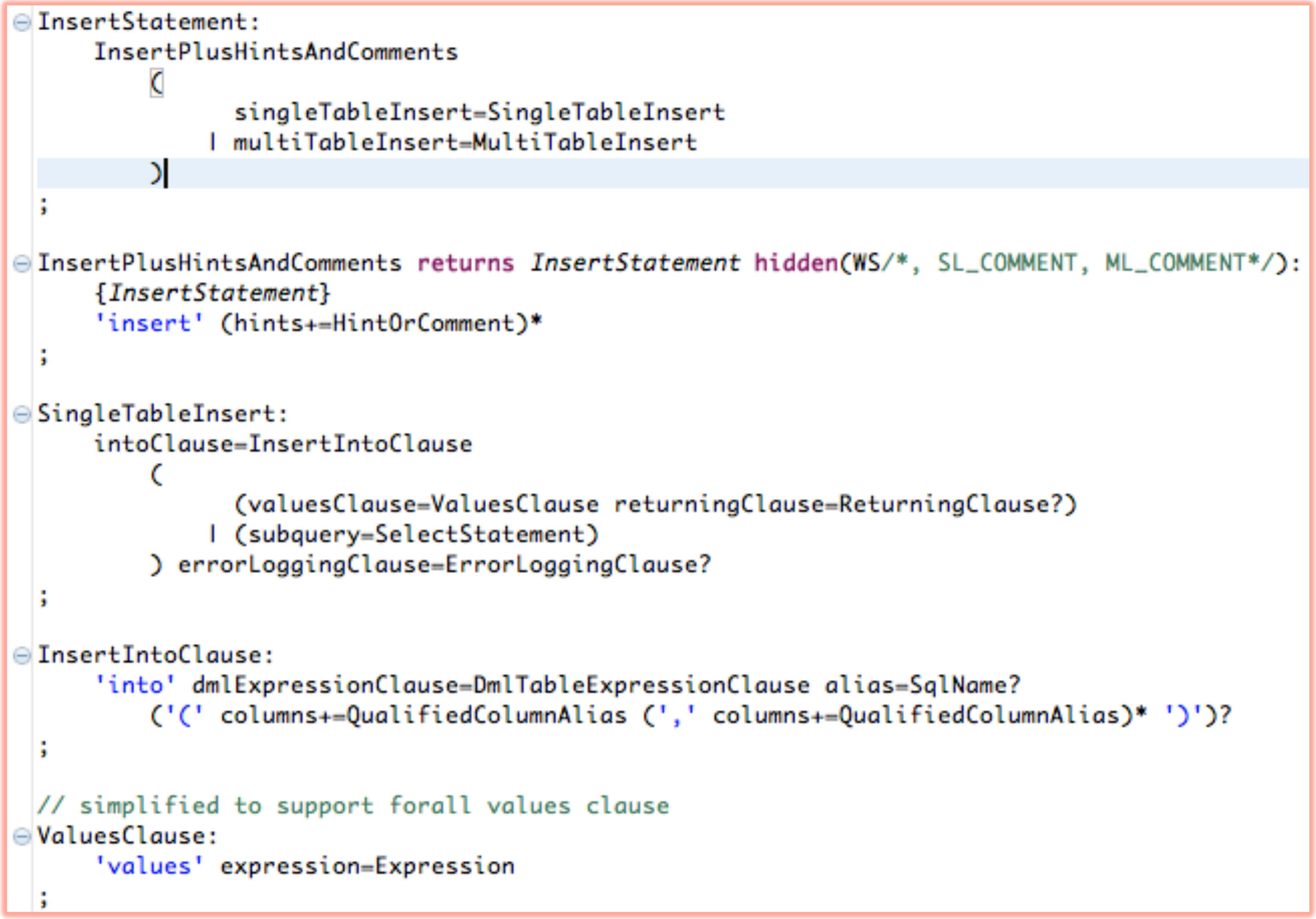


Generic Editor - guideline_26.plsql

Model

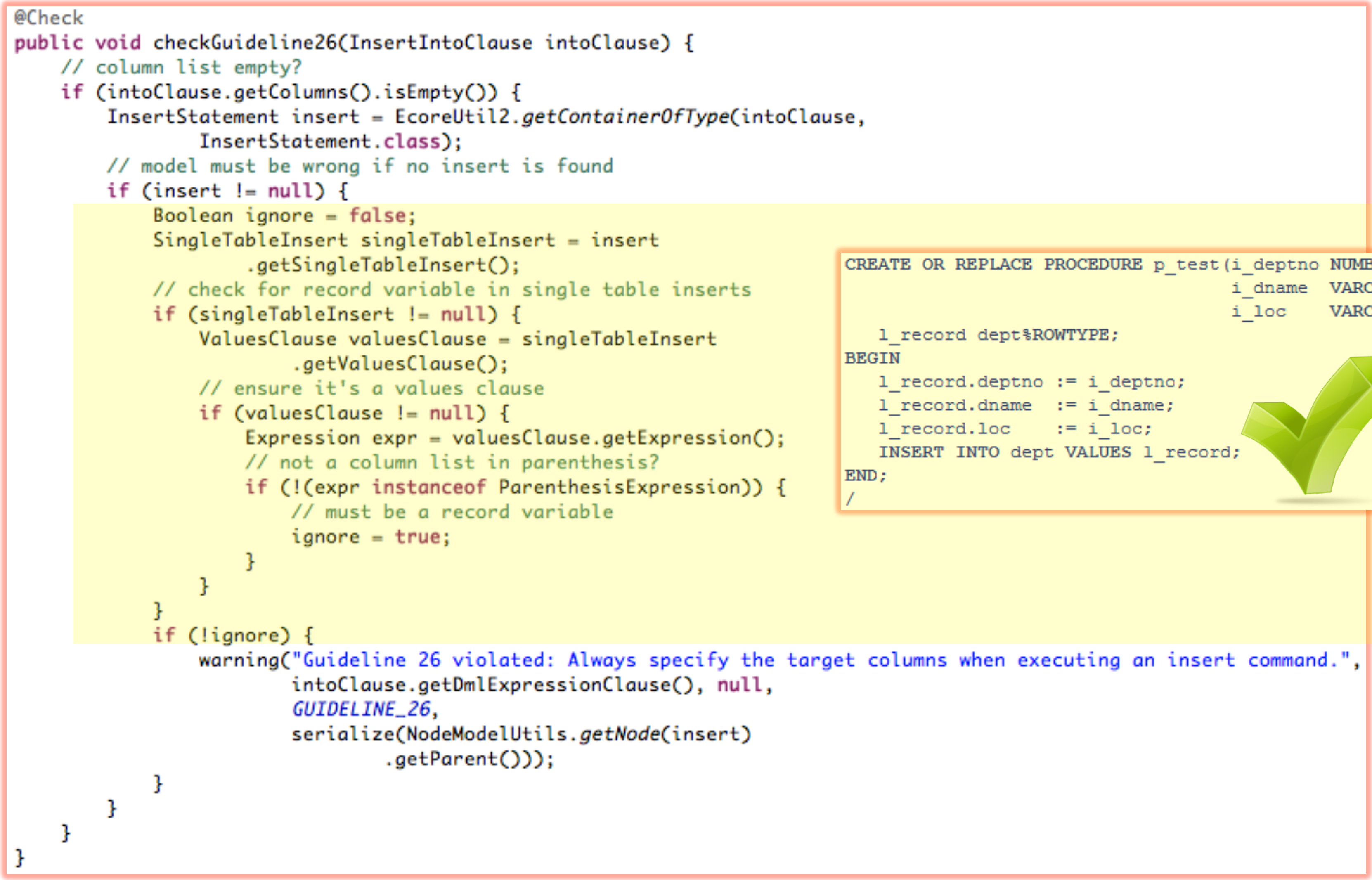
```
▼ platform:/resource/test/guideline_26.plsql
   ▼ PLSQL File
      ▼ Insert
         ▼ Insert Statement
            ▼ Single Table Insert
               ▼ Insert Into Clause
                    Dml Table Expression Clause
               ▼ Values Clause
                  ▼ Parenthesis Expression false
                     ▼ Function Or Parenthesis Parameter List Expression
                        ▼ Function Or Parenthesis Parameter
                           ▼ Binary Compound Expression Level6 .
                                Simple Expression Name Value mesg_seq
                                Simple Expression Name Value nextval
                        ▼ Function Or Parenthesis Parameter
                             Simple Expression Name Value l_mesg_type
                        ▼ Function Or Parenthesis Parameter
                             Simple Expression Name Value l_mesg_name
                        ▼ Function Or Parenthesis Parameter
                             Simple Expression Name Value l_mesg_text
                          Function Parameter List Suffix
```

Excerpt of Grammar for Insert Statement

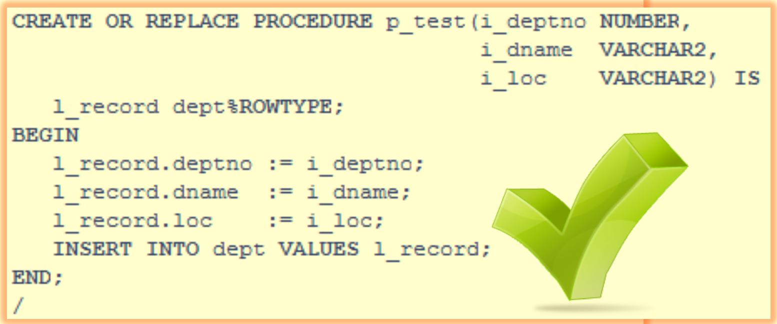


```
InsertStatement:
    InsertPlusHintsAndComments
        (
              singleTableInsert=SingleTableInsert
            | multiTableInsert=MultiTableInsert
        )
;

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

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

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

// simplified to support forall values clause
ValuesClause:
    'values' expression=Expression
;
```

Validator for Guideline #26

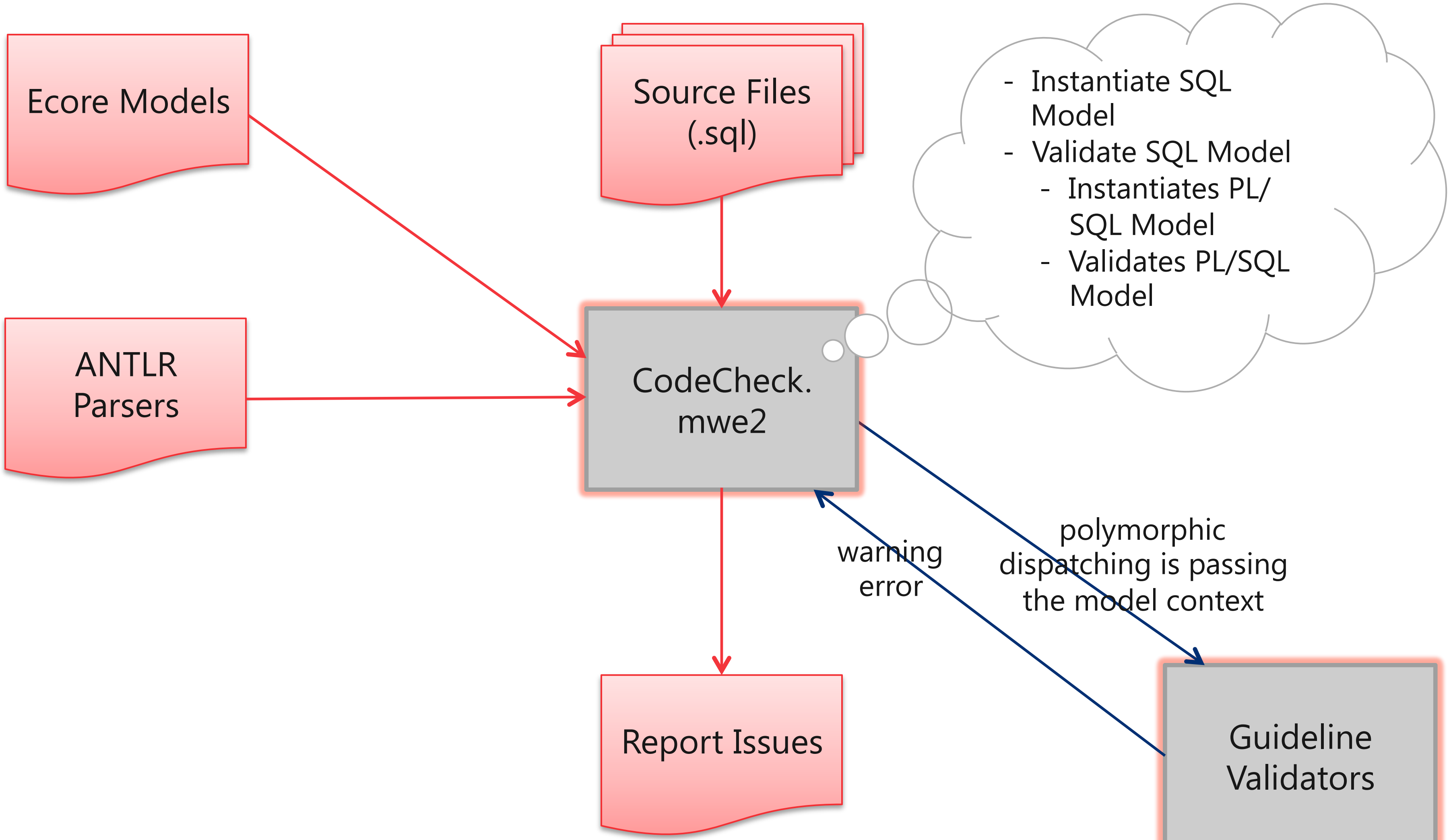


```
@Check
public void checkGuideline26(InsertIntoClause intoClause) {
    // column list empty?
    if (intoClause.getColumns().isEmpty()) {
        InsertStatement insert = EcoreUtil2.getContainerOfType(intoClause,
                InsertStatement.class);
        // model must be wrong if no insert is found
        if (insert != null) {
            Boolean ignore = false;
            SingleTableInsert singleTableInsert = insert
                    .getSingleTableInsert();
            // check for record variable in single table inserts
            if (singleTableInsert != null) {
                ValuesClause valuesClause = singleTableInsert
                        .getValuesClause();
                // ensure it's a values clause
                if (valuesClause != null) {
                    Expression expr = valuesClause.getExpression();
                    // not a column list in parenthesis?
                    if (!(expr instanceof ParenthesisExpression)) {
                        // must be a record variable
                        ignore = true;
                    }
                }
            }
            if (!ignore) {
                warning("Guideline 26 violated: Always specify the target columns when executing an insert command.",
                        intoClause.getDmlExpressionClause(), null,
                        GUIDELINE_26,
                        serialize(NodeModelUtils.getNode(insert)
                                .getParent()));
            }
        }
    }
}
```

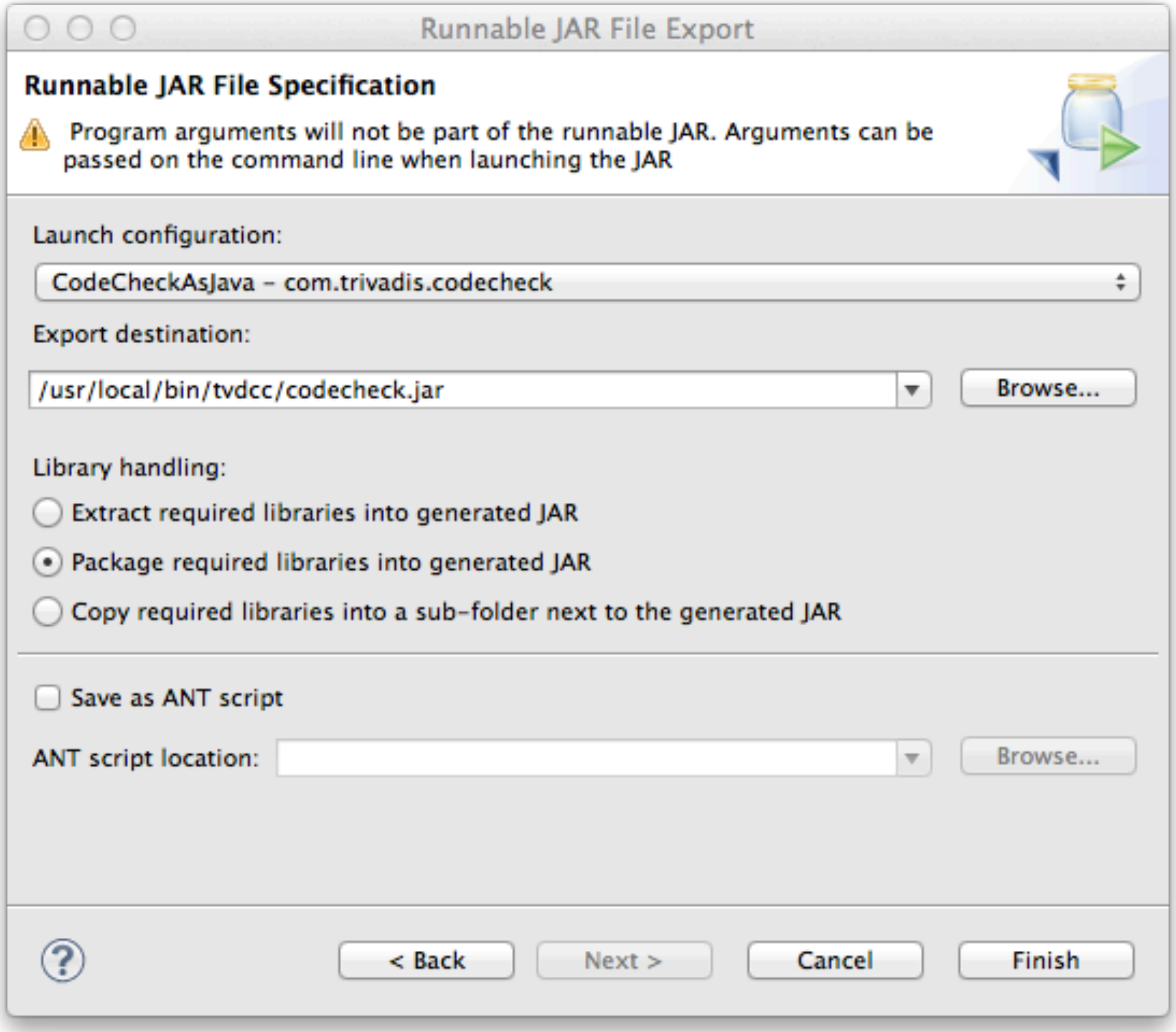


```
CREATE OR REPLACE PROCEDURE p_test(i_deptno NUMBER,
                                   i_dname  VARCHAR2,
                                   i_loc    VARCHAR2) IS
   l_record dept%ROWTYPE;
BEGIN
   l_record.deptno := i_deptno;
   l_record.dname  := i_dname;
   l_record.loc    := i_loc;
   INSERT INTO dept VALUES l_record;
END;
/
```

Apply Code Checks (via Command Line)



Build Runnable JAR


Runnable JAR File Export

Runnable JAR File Specification

⚠ Program arguments will not be part of the runnable JAR. Arguments can be passed on the command line when launching the JAR

Launch configuration:

CodeCheckAsJava – com.trivadis.codecheck

Export destination:

/usr/local/bin/tvdcc/codecheck.jar

Library handling:

○ Extract required libraries into generated JAR

● Package required libraries into generated JAR

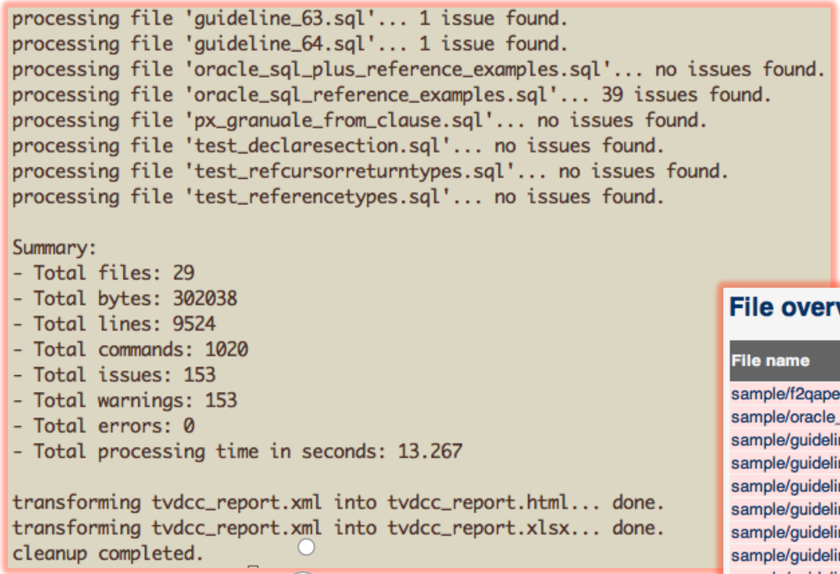
○ Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location:

Command Line Interface

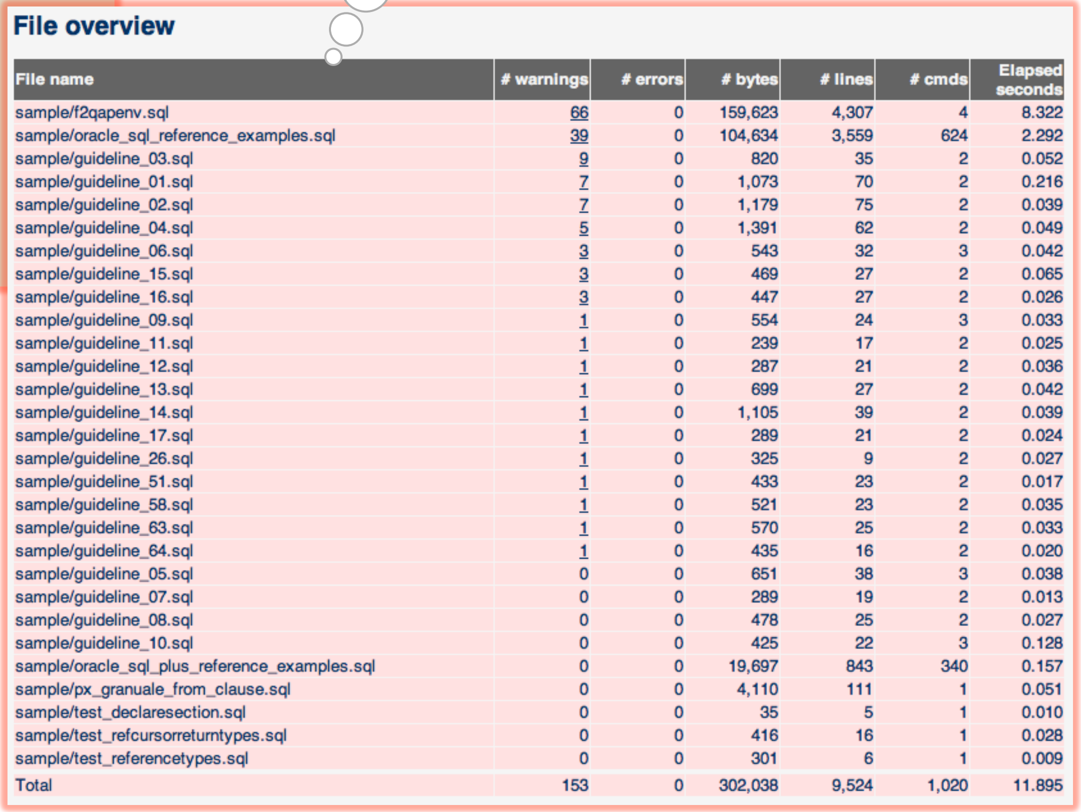
DEMO



```
processing file 'guideline_63.sql'... 1 issue found.
processing file 'guideline_64.sql'... 1 issue found.
processing file 'oracle_sql_plus_reference_examples.sql'... no issues found.
processing file 'oracle_sql_reference_examples.sql'... 39 issues found.
processing file 'px_granuale_from_clause.sql'... no issues found.
processing file 'test_declaresection.sql'... no issues found.
processing file 'test_refcursorreturntypes.sql'... no issues found.
processing file 'test_referencetypes.sql'... no issues found.

Summary:
- Total files: 29
- Total bytes: 302038
- Total lines: 9524
- Total commands: 1020
- Total issues: 153
- Total warnings: 153
- Total errors: 0
- Total processing time in seconds: 13.267

transforming tvdcc_report.xml into tvdcc_report.html... done.
transforming tvdcc_report.xml into tvdcc_report.xlsx... done.
cleanup completed.
```

XML, HTML,
Excel
Strategies

Console
Strategy



File overview

File name	# warnings	# errors	# bytes	# lines	# cmds	Elapsed seconds
sample/f2qapenv.sql	66	0	159,623	4,307	4	8.322
sample/oracle_sql_reference_examples.sql	39	0	104,634	3,559	624	2.292
sample/guideline_03.sql	9	0	820	35	2	0.052
sample/guideline_01.sql	7	0	1,073	70	2	0.216
sample/guideline_02.sql	7	0	1,179	75	2	0.039
sample/guideline_04.sql	5	0	1,391	62	2	0.049
sample/guideline_06.sql	3	0	543	32	3	0.042
sample/guideline_15.sql	3	0	469	27	2	0.065
sample/guideline_16.sql	3	0	447	27	2	0.026
sample/guideline_09.sql	1	0	554	24	3	0.033
sample/guideline_11.sql	1	0	239	17	2	0.025
sample/guideline_12.sql	1	0	287	21	2	0.036
sample/guideline_13.sql	1	0	699	27	2	0.042
sample/guideline_14.sql	1	0	1,105	39	2	0.039
sample/guideline_17.sql	1	0	289	21	2	0.024
sample/guideline_26.sql	1	0	325	9	2	0.027
sample/guideline_51.sql	1	0	433	23	2	0.017
sample/guideline_58.sql	1	0	521	23	2	0.035
sample/guideline_63.sql	1	0	570	25	2	0.033
sample/guideline_64.sql	1	0	435	16	2	0.020
sample/guideline_05.sql	0	0	651	38	3	0.038
sample/guideline_07.sql	0	0	289	19	2	0.013
sample/guideline_08.sql	0	0	478	25	2	0.027
sample/guideline_10.sql	0	0	425	22	3	0.128
sample/oracle_sql_plus_reference_examples.sql	0	0	19,697	843	340	0.157
sample/px_granuale_from_clause.sql	0	0	4,110	111	1	0.051
sample/test_declaresection.sql	0	0	35	5	1	0.010
sample/test_refcursorreturntypes.sql	0	0	416	16	1	0.028
sample/test_referencetypes.sql	0	0	301	6	1	0.009
Total	153	0	302,038	9,524	1,020	11.895

AGENDA

Customer Use Case

- Starting Position
  - Database centric application environment (business logic within the database)
  - Views are granted to roles and additionally protected by FGAC policies
  - Views are accessible via GUI and 3rd party products
  - Some columns contain sensitive data (e.g. turnover, margins, costs per order/customer)
- Questions to be answered
  - Which views present sensitive data as columns?
  - Who may access these data?

Sample – View SH.PROFITS (existing)

Which view columns use COSTS.UNIT_COST?

```
CREATE OR REPLACE VIEW PROFITS AS
SELECT s.channel_id,
       s.cust_id,
       s.prod_id,
       s.promo_id,
       s.time_id,
       c.unit_cost,
       c.unit_price,
       s.amount_sold,
       s.quantity_sold,
       c.unit_cost * s.quantity_sold TOTAL_COST
 FROM  costs c, sales s
 WHERE c.prod_id = s.prod_id
   AND c.time_id = s.time_id
   AND c.channel_id = s.channel_id
   AND c.promo_id = s.promo_id;
```

Sample – View SH.GROSS_MARGINS (new)

Which view columns use PROFITS.UNIT_COST, PROFITS.TOTAL_COST?

```
CREATE OR REPLACE VIEW GROSS_MARGINS AS
WITH gm AS
 (SELECT time_id, revenue, revenue - cost AS gross_margin
    FROM (SELECT time_id,
                 unit_price * quantity_sold AS revenue,
                 total_cost AS cost
            FROM profits))
SELECT t.fiscal_year,
       SUM(revenue) AS revenue,
       SUM(gross_margin) AS gross_margin,
       round(100 * SUM(gross_margin) / SUM(revenue), 2)
            AS gross_margin_percent
  FROM gm
 INNER JOIN times t ON t.time_id = gm.time_id
 GROUP BY t.fiscal_year
 ORDER BY t.fiscal_year;
```

Sample – View SH.REVENUES (new)

Which view columns use GROSS_MARGINS.GROSS_MARGIN, GROSS_MARGINS.GROSS_MARGIN_PERCENT?

```
CREATE OR REPLACE VIEW REVENUES AS
SELECT fiscal_year, revenue
  FROM gross_margins;
```

no columns:
table used but no
sensitive column

Sample – View SH.SALES_ORDERED_BY_GM (new)

Which view columns use PROFITS.UNIT_COST, PROFITS.TOTAL_COST?

```
CREATE OR REPLACE VIEW SALES_ORDERED_BY_GM AS
SELECT channel_id,
       cust_id,
       prod_id,
       promo_id,
       time_id,
       amount_sold,
       quantity_sold
  FROM profits
 ORDER BY (unit_price - unit_cost) DESC;
```

no columns:
usage outside of
column list

Approach

- Use PL/Scope (DBA_IDENTIFIERS)
  - Not applicable, PL/Scope collects data for PL/SQL source data only
- Query the Oracle data dictionary (DBA_DEPENDENCIES)
  - No column dependencies
- Create own Oracle data dictionary view with column dependencies (which are internally available since 11gR1)
  - See Rob van Wijk's post about DBA_DEPENDENCY_COLUMNS
  - No usage context
    (part of column expression, part of where clause, part of order by clause?)
  - No relation to affected view columns
- Use a PL/SQL parser in conjunction with data dictionary queries
  - Query Oracle dictionary to get dependent views and DDLs
  - Parse DDLs to get affected view columns

Which Parser? How to Use?

- Use Oracle parser if applicable
  - E.g. UTL_XML.PARSEQUERY, see http://www.salvis.com/blog/?p=117
- Use own or 3rd party parser in other cases
  - Provide parse tree as XML, e.g. via (web) service
  - Provide specific PL/SQL functions for analysis purposes (to keep interface stable, even if parse tree changes deeply)

```
SQL> SELECT *
  2  FROM TABLE(coldep_pkg.get_dep('sh', 'costs', 'unit_cost'));

SCHEMA_NAME VIEW_NAME     COLUMN_NAME
----------- ------------- ---------------------
SH          PROFITS       UNIT_COST
SH          PROFITS       TOTAL_COST
SH          GROSS_MARGINS GROSS_MARGIN
SH          GROSS_MARGINS GROSS_MARGIN_PERCENT
```

Produce XML Parse Tree via Web Service – How-Tos

- Create Web Service in Eclipse using Axis2
  - See http://www.softwareagility.gr/index.php?q=node/29
- Serialize Ecore model to XML
  - See http://www.eclipse.org/Xtext/documentation/2_1_0/100-serialization.php

```
URI fileURI = URI.createFileURI("example.xml");
Resource res = new XMLResourceFactoryImpl().createResource(fileURI);
ByteOutputStream os = new ByteOutputStream();
res.getContents().add(model);
res.save(os, null);
String xml = String(os.getBytes());
```

- Call Web Service from Oracle Database using UTL_DBWS
  - See http://docs.oracle.com/cd/E11882_01/java.112/e10587/intro.htm#JJPUB24035
  - See http://www.oracle-base.com/articles/10g/utl_dbws-10g.php
  - See https://forums.oracle.com/forums/thread.jspa?threadID=2162997

AGENDA

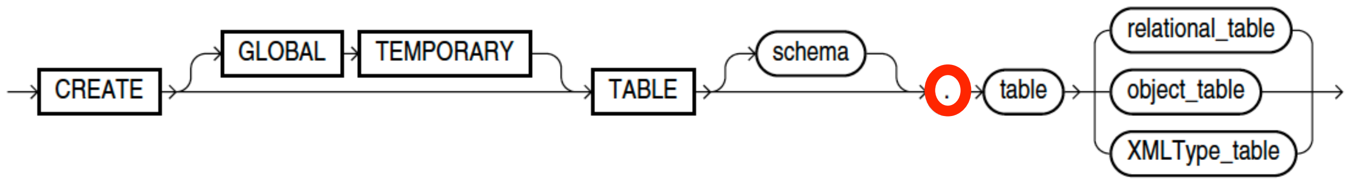
1. Introduction
2. Xtext Live – Parsing & Validating
3. Finalize Grammar, Checks and Tooling
4. Dependency Analysis
5. Challenges
6. Conclusion

Xtext

- One grammar, one Parser
  - The workflow GeneratePLSQL.mwe2 needs 4 minutes to complete
  - Bug 256403 - Multiple Grammar Mixin / Grammars as Library
- Maximum size of 64 KB for Java classes and methods
  - Use Xtext 2.0.1 and later to address "... is exceeding 65535 bytes ..." errors
- Output of underlying parser generator is passed 1:1 to the user
  - Fundamental knowledge of ANTLR is mandatory
  - Ability to distinguish between ANTLR and Xtext artifacts is necessary
- Convention over configuration
  - The first DSL incl. editors are created very fast using Xtext
  - Typically it's working but you easily do not know why and how
  - Usually things may be amended very elegantly and with just a few lines of code (e.g. outline, validators, formatter)
  - However, to find out what to do could take a serious time for an inexperienced fellow

Grammar

- Undocumented, old or incorrect grammar may break the parser
  - "timestamp" clause for packages, procedures and functions
  - Use of "id" or "oid" instead of "identifier" for object views
- Documentation bugs may lead to wrong grammar



- User defined operators lead to ambiguous grammar
  - Probably solvable by refactoring the Expression and Condition parser rules
  - The workaround is, to simply add the customer's operators when needed

SQL*Plus – CodeChecker Limitations

- The block terminator character '.' is not supported (nor configurable)
- The command separator character ';' is not supported (nor configurable)
- The SQLTerminator is not configurable, the default ';' is supported
- The line continuation character '-' does not support tailing whitespaces
- REMARK and PROMPT must not contain unterminated single/double quotes, single line or multi line comments (these commands cannot be defined as terminals because of conflicts with other parser rules – mainly identifiers)

AGENDA

Conclusion

PL/SQL & SQL

Tooling

- The grammar to parse SQL*Plus files is huge
  - Chaining multiple parsers is the way to go
- Xtext is a complete DSL framework
  - More than just a parser generator
  - Separation of parser and validators
  - Promising for further applications like code fixing, presenting graphical models, calculating complexity, etc.
- Even if a significant subset of the SQL*Plus, SQL, PL/SQL grammar needs to be maintained continuously, Xtext is a good choice to implement future code checking and dependency analysis requirements

THANK YOU.

Trivadis AG

Philipp Salvisberg

Europastrasse 5
8152 Glattbrugg (Zürich)

Tel. +41-44-808 70 20
Fax +41-44-808 70 21

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

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